Algoritmoen analisia:

sarreraren tamaina,
denbora-ekuazioak eta -ordenak,
erreminta matematikoak,
errekurtsio ekuazioak.

R. Arruabarrena
LSI - UPV/EHU

Algoritmoen analisia

- Ideia nagusiak:
  - algoritmoen eraginkortasuna
  - oinarrizko eragiketak eta eragiketa kritikoa
  - sarreraren tamaina
  - **M**emoria **E**spazio **E**stra (MEE)
  - $\mathbf{T_t}$ (n), T_o(n), T_b (n)
  - T(n) eta inbariantza printzipioa
  - funtzioak taldekatzen: ordena
  - idazkerak: O,Ω,Θ,o eta ω. Propietateak
  - erregela mnemoteknikoak
  - T(n)a ez-errekurtsiboa bilakatzen
  - formula matematiko erabilienak
  - Errekurtsio ekuazioak ebazten: hedapena, ekuazio karakteristikoa, aldagai aldaketa

irakasgai **osoan zehar**

behin eta berriro erabiliak

1 Algoritmoen eraginkortasuna

Eraginkortasuna:

baliabideen zentzuzko/egokizko erabilpena

Helburua: problemak ebaztea

(a) algoritmoa idatzi

(b) algoritmo desberdin ezagunak

- Zein aukeratu?
  - zenbat aldiz erabiliko da?
  - zein sarrerako instantziekin erabiliko da?
  - eraginkorra izanik, zein argia/irakurgarria da?
  - zenbat memoria behar du?

- Eraginkortasuna behar duten algoritmoak
  - Bilaketa
  - Ordenazioa: (fitxategiak, taulak, egiturak)
  - Optimizazioa
  - Denbora-errealeko sistemak: (uholde, trafiko, zentral elektriko eta lurrikaren kudeaketan...)
  - DBen atzipena: (galderen optimizazioan, galdera-lengoaiak...)
  - Sistema Eragileak: (baliabideen kudeaketa...)
  - Lehentasun dinamikoko sistemak (hilaren kudeaketa, prozesuen kudeaketa...)
  - Lengoaia interpretatuak

- Zentzurik du eraginkortasunean inbertitzeak?

Demagun Alg1-ek Konp1-ean $10^{-4}\ 2^n$ s behar duela

Sarreraren tamaina (osagai kopurua)	Exekuzio denbora
n = 10	$10^{-4}\ 2^{10}$ s
n = 20	
n= 30	
n= 38	

∃ Konp2: 100 aldiz azkarragoa Konp1 baino
Zenbat denbora behar du Alg1-ek Konp2-an?

Alg1 Konp2 makinan urtebetez egikaritzen utziez gero, gehienez zer tamainako sarreraren instantzia ebatzi ahal izango dugu?

✍ Alg1-ek Konp1-ean: $T_{k1}(n) = 10^{-4}\ 2^n$ s

N	Exek. T.
10	**$10^{-4}\ 2^{10}$ s = 0.1024 s » 0.1”**
20	**$10^{-4}\ 2^{20}$ s =104.8576s =1.74’ > 1’**
30	**107374.1824s=1e 5h 49’ > 1 e**
38	**2748779.690s= 318e 3h = 0,87 u**
39	**10-4 239 s = 1u 271 e =1,74 u**

✍ Konp2 100 aldiz azkarragoa da Konp1 baino. Zenbat denbora behar du Alg1-ek Konp2-an?

$T_{k2}(n) = T_{k1}(n)/100 = 10^{-6}\ 2^n$ s

✍ Alg1 Konp2 makinan urtebetez utziz, n handiena:

$10^{-6}\ 2^n$ = 365*24*60*60 s = 31536000s

n=44,8421;

Eraginkortasuna kalkulatzeko estrategiak:

Enpirikoa (a posteriori):

algoritmo desberdinak programatu eta ondoren, probak egin sarrerako tamaina desberdinak dituzten instantziekin

Teorikoa (a priori):

sarreraren tamaina kontuan hartuz aldez aurretik behar diren baliabideak matematikoki kalkulatu (exekuzio-denbora, memoria-espazioa...)

Hibridoa

Estrategia teorikoa:

Abantailak

1. ez dago konputagailu, programazio-lengoaia ez eta programatzailearen trebetasunaren menpe

2. algoritmo ez-eraginkorren programazio- eta exekuzio-denbora aurrezten du

3. edozein tamainako instantzia azter dezake

Desabantailak

1. baliabide matematikoen erabileran trebea izan

- Algoritmoen baliabidea kontsumoa
  - memoria espazioa
    - gehitzeak exekuzio-denbora jaistea dakar maiz
  - exekuzio-denbora: faktore desberdinen menpe
    - algoritmoaren sarrera
    - algoritmoaren konplexutasuna denborarekiko
    - programak erabiltzen duen espazioa
    - konpilatzaileak sortutako kodearen kalitatea
    - erabilitako makinaren aginduen izaera eta abiadura

2 Sarreraren tamaina

- Parametro errealen “tamaina”: algoritmoaren egikaritzapenean lan gehiago eginaraziko duena

- fitxategien kasuan → osagai kopurua
  osagai baten bilaketa
- tamaina handiko znbk→ adierazpideak behar duen bit kopurua
  biderkaketa bi zenbaki artean
- zenbaki bakuna → zenbakiaren balioa
  zenbaki baten faktoriala
- taulak → osagai kopurua
  ordenazioa
- matrize karratua → dimentsio baten tamaina
  determinantearen balioa kalkulatzea

3 Memoria Espazio Estra

MEE

Exekuzio garaian,
algoritmoa exekutagarria izan dadin
hark beharko duen
gehienezko memoria espazioa da

Nork ematen dute kontsumo hori?

- parametro errealez at, algoritmoaren exekuzioan aldagai lokalei esleitutako memoria espazioak

- dei errekurtsiboek kontsumatzen duten pila espazioa

4 Lanaren neurketa: exekuzio-denbora

- Sarreraren tamainaren funtzio bezala neurtuko da

T(n)-ren bidez, *n* tamainako sarrera izanik, algoritmoak behar duen exekuzio-denboraren neurketa bat adieraziko du (lan kopurua)

Sarreraren tamaina *n* izanik, neurketa desberdinak egin litezke:

T(n)= **# dei errekurtsiboen**

T(n)= **# konparaketa**

T(n)= **# egikaritzen diren eragiketa**

....

Momentu oro zer neurtzen ari garen **argitu behar dugu**

5 Oinarrizko eragiketa

T(n) = Algoritmoak zenbat eragiketa egingo ditu n tamaina duen sarrerarentzat?

Oinarrizko eragiketa

bere exekuzio-denbora konstante batez mugatua duena.

Konstante hori erabilitako inplementazio konkretuaren menpe dago soilik (makina, programazio lengoaia...)

Dira:

- R gaineko eragiketak: +, -, * y /
- koma mugikorrezko zenbakien gainekoak
- =,>,<,..., :=, prozeduren deiak

```
X:= X+Y;
Y:= 3 * Y+ 8;
```

```
for I in 1 .. N loop
    X:= X+Y;
end loop;
```

```
for I in 1 .. N loop
    for J in 1 .. N loop
        X:= X+Y;
    end loop;
end loop;
```

(1) $T_1(n)$ = exekuzio-denbora = (i oinarrizko eragiketa egikaritzen den aldi kopurua) * (i oinarrizko eragiketaren exekuzioak hartzen duen denbora)

$T_1(n)= k1 + k2 = k3$ **$T_1(n)= k1 * n$** **$T_1(n)= k1 * n^2$**

(2) T_2 (n)= egiten diren batuketa kopurua

$T_2(n)= 2$ **$T_2(n)= 1 * n$** **$T_2(n)= 1 * n^2$**

Zein T(n) da egokiena? Txarrik badago?

Dena batu: T(n)a beti zuzena da

Σ konplexuak

Helburu hurbila:

Eragiketa kritikoak (**adierazgarriak**) soilik batu

```
X:= X+Y;          for I in 1 .. N  loop    for I in 1 .. N  loop
Y:= 3 * Y+ 8;        X:= X+Y;                 for J in 1 .. N loop
                   end loop;                     X:= X+Y;
                                               end loop;
                                             end loop;
```

(1) $T_1(n)$ = exekuzio-denbora = (i oinarrizko eragiketa egikaritzen den aldi kopurua) * (i oinarrizko eragiketaren exekuzioak hartzen duen denbora)

$T_1(n)= k1 + k2 = k3$ **$T_1(n)= k1 * n$** **$T_1(n)= k1 * n^2$**

(2) T_2 (n)= egiten diren batuketa kopurua

$T_2(n)= 2$ **$T_2(n)= 1 * n$** **$T_2(n)= 1 * n^2$**

Zein T(n) da egokiena? Txarrik badago?

Dena batu: T(n)a beti zuzena da

Σ konplexuak

Helburu hurbila:

Eragiketa kritikoak (**adierazgarriak**) soilik batu

6 Inbariantza printzipioa

Algoritmo beraren
bi inplementazio desberdinen
eraginkortasunen arteko diferentzia
konstante biderkatzaile batean datza

$$t1(n) \leq \mathbf{k}\ t2(n)$$

n handia denean
(neurri asintotikoa)

Ondorioak:

- edozein konputagailu eta programatzaileren trebetasunetarako balio du
- eraginkortasunaren hobekuntza algoritmoaren aldaketa baten bidez bakarrik lortuko da
- algoritmoaren eraginkortasun teorikoak **ez** du **unitaterik**, konstante biderkatzaile baten funtziopean ematen baita.

6 Inbariantza printzipioa

Algoritmo beraren
bi inplementazio desberdinen
eraginkortasunen arteko diferentzia
konstante biderkatzaile batean datza

$$t1(n) \leq \mathbf{k}\ t2(n)$$

n handia denean
(neurri asintotikoa)

Ondorioak:

- edozein konputagailu eta programatzaileren trebetasunetarako balio du
- eraginkortasunaren hobekuntza algoritmoaren aldaketa baten bidez bakarrik lortuko da
- algoritmoaren eraginkortasun teorikoak **ez** du **unitaterik**, konstante biderkatzaile baten funtziopean ematen baita.

7 $T_o(n)$, $T_t(n)$ eta $T_b(n)$

Normalki:

- algoritmoek exekuzio-denbora desberdina hartzen dute tamaina desberdineko instatziak egikaritzeko,

- baina, tamaina berdineko instantzientzat gerta liteke exekuzio-denbora desberdina hartu behar izatea

algoritmoa sentikorra

da sarreraren egoerarekiko

✍ Txanpon faltsua mugatzen

✍ Bilaketa lineala: ordenatua ala ez

Kasu onena: $T_o(n)$

n tamainako sarrera guztien artetik denbora minimoa behar duen haren denbora

$\Rightarrow$ gutxienez itxaron beharko dugun denbora

Kasu txarrena: $T_t(n)$

n tamainako sarrera guztien artetik denbora maximoa behar duen haren denbora

$\Rightarrow$ erantzun denbora kritikoa denean

Batez besteko kasua: $T_b(n)$

n tamainako sarrera guztien exekuzio-denboren batez besteko denbora

$\Rightarrow$ exekuzio asko sarrera oso desberdinekin

✍ Txanpon faltsua mugatzen

T(n)= pisatze kopurua

alg1

$T_o(n)=1$ $T_t(n)=n-2$

$T_b(n)=1/n\ (1+(n-2)+(1+2+...+n-2))=n/2-1/2$

$T_b(10)=4.5$ $T_b(32)=15.5$

alg2

$T_o(n)=1$ $T_t(n)=n/2$

$T_b(n)=2/n\ (1+2+...+n/2)=n/4+1/2$

$T_b(10)=3$ $T_b(32)=8.5$

alg3 suposatuz: $n=2^k$

$T_o(n)=k$ $T_t(n)=k$

$T_b(n)=1/n\ (k+k+...+k)=1/n\ (n\ k) = k$

$T_b(32)=5$

✍ Bilaketa T(n)= konparazio kop.

ez-ord

$T_o(n)=1$ $T_t(n)=n$

$T_b(n)=p/n\ (1+2+ \ldots + n)+(1-p)\ n$

$= p/2\ (1+n) +(1-p)\ n$

p=0,5-> $T_b(16)=12.25$

Ord+lineala

$T_o(n)=1$ $T_t(n)=n$

$T_b(n)=p/n\ (1+2+ \ldots + n)+(1-p)/(n+1)\ (1+\ldots+n+n)$

$= p/2\ (1+n) +(1-p)/2\ n +(1-p)\ n/(n+1)$

p=0,5-> $T_b(16)=8.72$

Ord+dikotomikoa

$T_o(n)=1$ $T_t(n)=lg_2\ n+1$

$T_t(16)=5$

8 Exekuzio-denboraren ordena

Algoritmoak hartzen duen *denbora*
f(n)-ren ordenakoa dela esango da
baldin eta soilik baldin
k konstanteak existitzen badu,
non instantzia bakoitzeko problema *k* f(n)* segundotan (μsegundotan,...),
gehienez ebazten baita

k konstanteari konstante biderkatzailea deritzo

8 Exekuzio-denboraren ordena (2)

- Konstante biderkatzaileak funtzioek ordena berdina dutenean bakarrik kontuan hartuko ditugu: inplementazio txikikerietan sartzea ekiditen du

- Ordenaren erabilera:

  problema bera ebazten duten eta sarreraren tamaina berdina duten algoritmoen exekuzio-denborak konpara ahal izateko.

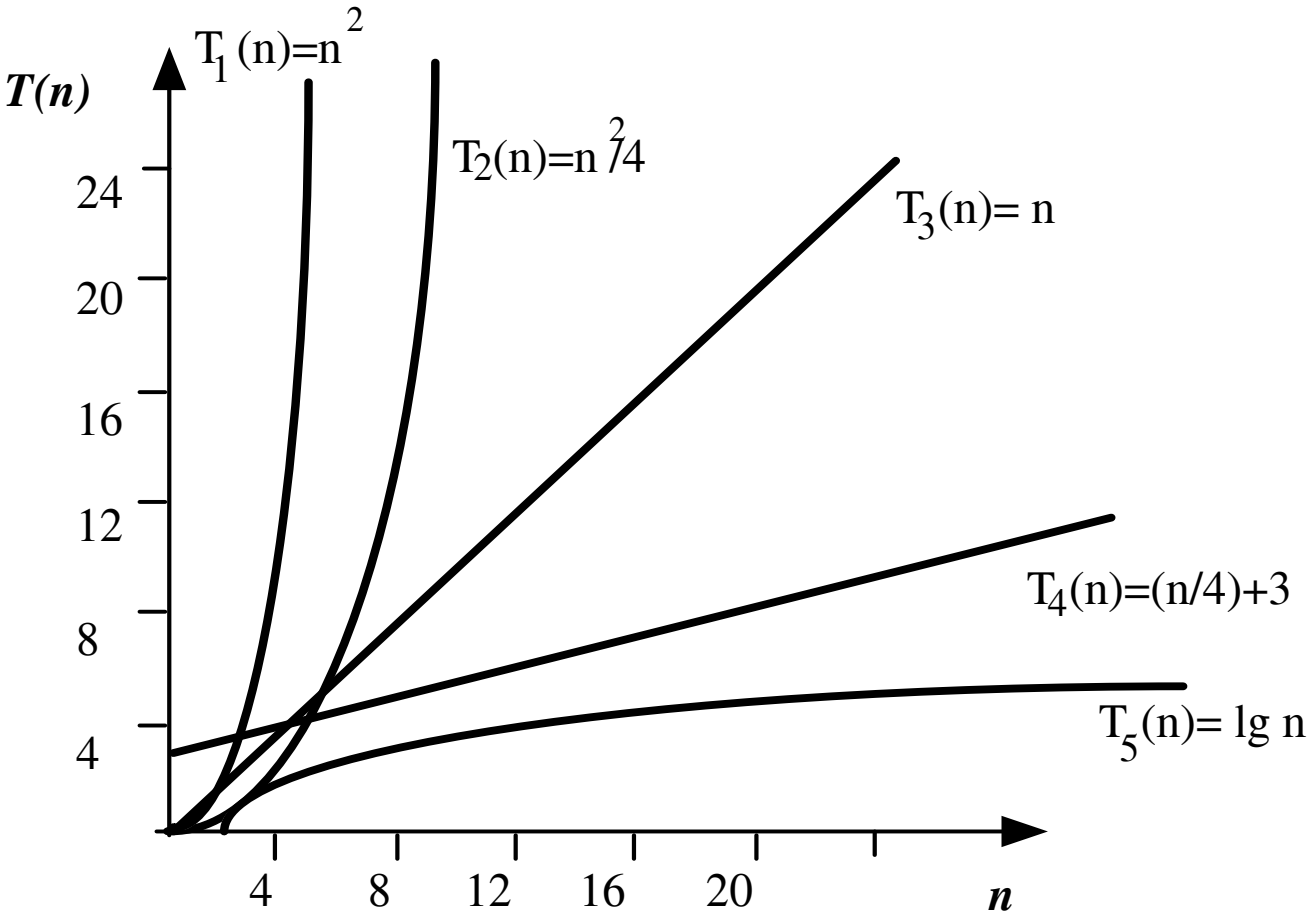
  - Ordena bereko soluzioak, soluzio parekoak kontsideratuko ditugu.
  - Ordena txikiagoko algoritmoak, azkarragoak dira.

- Maiz azaltzen diren ordenak

n-ren ordenakoa → algoritmoa *lineala* da

n^2	*koadratikoa*
n^3	*kubikoa*
n^k	*polinomiala*
a^n	*esponentziala*
lg n	*logaritmikoa*

(k eta a konstante egokiak dira)



9 Hurbilketa asintotikoak

N-ren balio handietarako

- f(n) funtzioa t(n)-ren bornea da
- borne sinpleenak kalkulatuko ditugu
- funtzioak (exekuzio denborak guretzat) elkarrekin alderatzeko, taldekatzeko eta eraginkorrenak mugatzeko

O larria (gehienez edo berdin)

$$O(f(n)) = \{ t:N \to R^* \mid \exists c\ \exists n_0\ ((c \in R^+) \wedge (n_0 \in N) \wedge \forall n\ (n \geq n_0 \Rightarrow t(n) \leq c\ f(n))) \}$$

- $O(f(n))$ irakurriko da *f(n) ordena*

Omega larria : Ω (gutxienez edo berdin)

$$\Omega(f(n)) = \{ t:N \to R^* \mid \exists d\ \exists n_0\ ((d \in R^+) \wedge (n_0 \in N) \wedge \forall n\ (n \geq n_0 \Rightarrow t(n) \geq d\ f(n))) \}$$

- $\Omega(f(n))$ irakurriko *f(n) omega ordena*

Teta larria: Θ (zehazki)

$$\Theta(f(n)) = \{ t:N\rightarrow R^* \mid \exists c,d\ \exists n_0\ ((c,d \in R^+) \wedge (n_0 \in N) \wedge \forall n\ (\mathbf{n} \geq n_0 \Rightarrow d\ f(n) \leq t(n) \leq c\ f(n))) \}$$

- $\Theta(f(n))$ irakurriko dugu *f(n) ordena zehatza*

o xehea: o (gehienez eta ≠)

$$o(f(n)) = \{ t:N\rightarrow R^* \mid \exists c\ \exists n_0\ ((c \in R^+) \wedge (n_0 \in N) \wedge \forall n\ (\mathbf{n} \geq n_0 \Rightarrow t(n) < c\ f(n))) \}$$

Omega xehea: ω (gutxienez eta ≠)

$$\omega(f(n)) = \{ t:N\rightarrow R^* \mid \exists d\ \exists n_0\ ((d \in R^+) \wedge (n_0 \in N) \wedge \forall n\ (\mathbf{n} \geq n_0 \Rightarrow t(n) > d\ f(n))) \}$$

Hurbilketa asintotikoak. Aplikazioak: T(n)-en klasifikazioa

- Definizio bidez

  $2^n \in O(3^n)$? *Bai*

  $\exists$ d=1, no=1 $\forall$n>no $\Rightarrow 2^n \leq d\ (3^n)$

  $3^n \in O(2^n)$? *Ez*

  $\neg\exists$ d,no, $\forall$n>no $\Rightarrow$ 3n $\leq$ d (2n) ; (3/2)n $\leq$ d

- Edo limiteak erabiliz:

  $L_{n\to\infty}\ 2^n/3^n = L_{n\to\infty}\ (2/3)^n = L_{n\to\infty}\ (0,6)^n = 0$

$2^n \in O(3^n)$; $O(2^n) \subset O(3^n)$; $2^n \notin \Theta(3^n)$

- ½ $n^2+4n+5 \in \Theta(n^2)$? *Bai*

  $\exists$ c= ½, d=1, no=10 $\forall$n>no

  $\Rightarrow c\ n^2 \leq$ ½ $n^2+4n+5 \leq d\ n^2$

Θ (½ n^2+4n+5)=$\Theta(n^2)$

- L'Hopital

  $T_1(n) = a_1 \sqrt{n} + a_2$ $\qquad$ $T_2(n) = b_1 \lg_2 n + b_2$

$$L_{n\to\infty} \frac{b_1 \lg_2 n + b_2}{a_1\sqrt{n} + a_2} =$$

$$= L_{n\to\infty} \frac{b_1 / \ln 2 \ln n + b_2}{a_1\sqrt{n} + a_2} \quad =_{LH} \quad L_{x\to\infty} \frac{b_1 / \ln 2 \cdot 1/x \cdot 1}{a_1 \cdot 1/2\, x^{-1/2}}$$

$$= C \cdot L_{x\to\infty} \frac{1}{x^{1/2}} = C \cdot L_{x\to\infty} \frac{1}{\sqrt{x}} = 0$$

$\Theta(b_1 \lg_2 n + b_2) = \Theta(\lg_2 n)$

$\Theta(a_1\sqrt{n} + a_2) = \Theta(\sqrt{n})$

$O(\lg_2 n) \subset O(\sqrt{n})$

10 Algoritmo iteratiboen T(n)a eta ordena

- Erregela mnemoteknikoak erabiliz T(n)-a idatzi
  1. parametroek ® sarreraren tamaina
  2. :=, >,<,<=,...,I/O,... ÎO(1)
     Salbuespenak: array-ekin adi!
  3. agindu-sekuentziak: T_1+...+ T_k , baturaren er.
  4. if-then: T_{bald}+ T_{then}
     if-then-else: T_{bald}+ max {T_{then} ,T_{else}}
  5. Begiztek : ($T_{gorputza}$+ T_{bald}) x bira-kopuru
  6. Azpiprograma ez-errekurtsiboen deiek:
     $T_{azpiprog}$+T_{deia}+$T_{parametropasatze}$
- Exekuzio-denboraren ekuazioa sinplifikatu
- Ordena mugatu

Baturaren erregela

P1 $T_1(n) \in \Theta(f(n))$

\+

P2 $T_2(n) \in \Theta(g(n))$

$T_1(n)+T_2(n) \in \Theta(f(n)+g(n)) = \Theta(\max(f(n)+g(n))$

$$\Theta(f(n)+g(n)) = \Theta(\max(f(n),g(n)))$$
$$O(f(n)+g(n)) = O(\max(f(n),g(n)))$$
$$\Omega(f(n)+g(n)) = \Omega(\max(f(n),g(n)))$$

Biderkaketaren erregela

P1

P2

P1 → $T_1(n) \in \Theta(f(n))$ P2 → $T_2(n) \in \Theta(g(n))$

P2 programa P1tik T1(n) aldiz dei egiten bazaio, programa osoak hartzen duen denbora

$T_1(n)*T_2(n) \in \Theta(f(n)*g(n))$

DEA II:

Erreminta matematikoak

Esponentzialak eta logaritmoak: $\forall a, b, c, n > 0$. Bestalde, logaritmoen oinarriak 1 baino handiagoak dira, eta ezer aipatzen ez bada $\lg a = \log_2 a$.

$\log_a 1 = 0$	$\log_a b^c = c \cdot \log_a b$	$a^{\log_a b} = \log_a a^b = b$
$\log_a(b \cdot c) = \log_a b + \log_a c$	$\log_a b = \frac{\log_c b}{\log_c a}$	$a^{\log_b c} = c^{\log_b a}$
$\log_a \frac{b}{c} = log_a b - \log_a c$	$\log_a b = \frac{1}{log_b a}$	$a^0 = 1$
$\ln 2 = 0.7$, $\lg e = 1.44$ eta $\lg 10 = 3.32$	$\log_a \frac{1}{c} = -\log_a c$	$a^{-1} = \frac{1}{a}$
$\lg \lg a = \lg(\lg a)$	$\frac{n}{2} < 2^{\lfloor \lg n \rfloor} <= n$	$a^b \cdot a^c = a^{b+c}$
$\lg^a b = (\lg b)^a$	$n <= 2^{\lceil \lg n \rceil} < 2n$	$(a^b)^c = (a^c)^b = (a^{b \cdot c})$

Batukariak: $\forall a, b, c, n > 0$, $a <= b$

$$\sum_{i=a}^{b} i = \frac{(a+b)(b-a+1)}{2} \qquad \sum_{i=1}^{n} i = \frac{(1+n)n}{2}$$

$$\sum_{i=a}^{b} c^i = \frac{c^{b+1}-c^a}{c-1} \qquad \sum_{i=a}^{b} \frac{1}{c^i} = \frac{1/c^{b+1}-1/c^a}{(1/c)-1}$$

$$\sum_{i=a}^{b} i^2 = \frac{b^3-a^3}{3} + \frac{b^2-a^2}{2} + \frac{b-a}{6} \qquad \sum_{i=1}^{n} i^2 = \frac{2n^3+3n^2+n}{6}$$

$$\sum_{i=a}^{b} i \cdot c^i = \frac{b \cdot c^{b+1} - a \cdot c^a}{c-1} + \frac{c^{a+1}-c^{b+1}}{(c-1)^2} \qquad \sum_{i=1}^{n} i \cdot 2^i = (n-1)2^{n+1} + 2$$

Funtzio deribatuak, integralak eta batuketak integral bidez bornatzen a edozein konstantea da, x-k aldagaia adierazten du eta u eta v x aldagai gainek funtzioak (positiboak, bornaketen kasuan) dira.

$a' = 0$	$x' = 1$	$(u+v)' = u' + v'$
$(-u)' = -(u')$	$(a \cdot v)' = a \cdot v'$	$(u \cdot v)' = u' \cdot v + u \cdot v'$
$(u^a)' = a \cdot u^{a-1} \cdot u'$	$(a^u)' = a^u \cdot u' \cdot \ln a$	$\sqrt{x} = \frac{1}{2\sqrt{x}}$
$(\log_a u)' = \frac{u'}{u \cdot \ln a}$	$(\ln u)' = \frac{u'}{u}$	

$$\int a \cdot u \cdot dx = a \cdot \int u \cdot dx \qquad \int -u \cdot dx = -\int u \cdot dx$$
$$\int (u+v) \cdot dx = \int u \cdot dx + \int v \cdot dx \qquad \int u \cdot dv = u \cdot v - \int v \cdot du$$

f jarraia eta gorakorra denean:

$$\int_{a-1}^{b} f(x) \cdot dx <= \sum_{i=a}^{b} f(i) <= \int_{a}^{b+1} f(x) \cdot dx$$

f jarraia eta beherakorra denean:

$$\int_{a}^{b+1} f(x) \cdot dx <= \sum_{i=a}^{b} f(i) <= \int_{a-1}^{b} f(x) \cdot dx$$

Exekuzio-denbora funtzio desberdinak konparatzeko erremintak:
Demagun problema bat ebazteko algoritmo bi lortu ditugula non $f(n)$ eta $g(n)$ funtzioek haien exekuzio denborak adierazten duten. Soluzio eraginkorrenarekin geratzeko exekuzio-denbora funtzioak alderatu behar ditugu. Aukera desberdinak ditugu hori egiteko: $f, g : \mathbb{N} \to \mathbb{R}$

1. $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \Rightarrow \Theta(f(n)) \subset \Theta(g(n))$ edo $f(n) \in o(g(n))$
2. $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \Rightarrow f(n) \in O(g(n))$ eta $g(n) \notin O(f(n))$
3. $\lim_{n\to\infty} \frac{f(n)}{g(n)} = K \in \mathbb{R}^+ \Rightarrow \Theta(f(n)) = \Theta(g(n))$ edo $f(n) \in \Theta(g(n))$
4. **L'Hopitalen erregela**: betetzen badira
   (a) $\lim_{n\to\infty} f(n) = \lim_{n\to\infty} g(n) = 0$, edo $\lim_{n\to\infty} f(n) = \lim_{n\to\infty} g(n) = \infty$,
   (b) $f(n)$ eta $g(n)$ funtzioak arruntetatik $f^\star(x)$ eta $g^\star(x)$ errealetara edagarriak badira
   (c) $f^{\star'}(x)$ eta $g^{\star'}(x)$(funtzio hedatuak deribagarriak badira) eta
   (d) $\exists \lim_{x\to\infty} \frac{f^{\star'}(x)}{g^{\star'}(x)}$

   orduan

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{x\to\infty} \frac{f^{\star'}(x)}{g^{\star'}(x)}$$

Azkuntza abiadura: exekuzio-denboren ordenen klasifikazioa. $\forall a, k$ konstanteak eta $a > 0$

$$O(1) \subset O(\lg n) \subset O(\sqrt{n}) \subset O(n) \subset O(n \lg n) \subset O(n\sqrt{n}) \subset O(n^2) \subset O(n^3)$$

$$O(n^k) \subset O(a^n) \subset O(n!) \subset O(n^n)$$

$$O(kf(n)) = O(f(n))$$

a) -- Lista 0-z hasieratu

```
for IND in LISTEN_FREKUENTZIA'RANGE
loop

    LISTEN_FREKUENTZIA(IND) := 0;
end loop;
```

b) -- Handienaren posizioa aurkitu

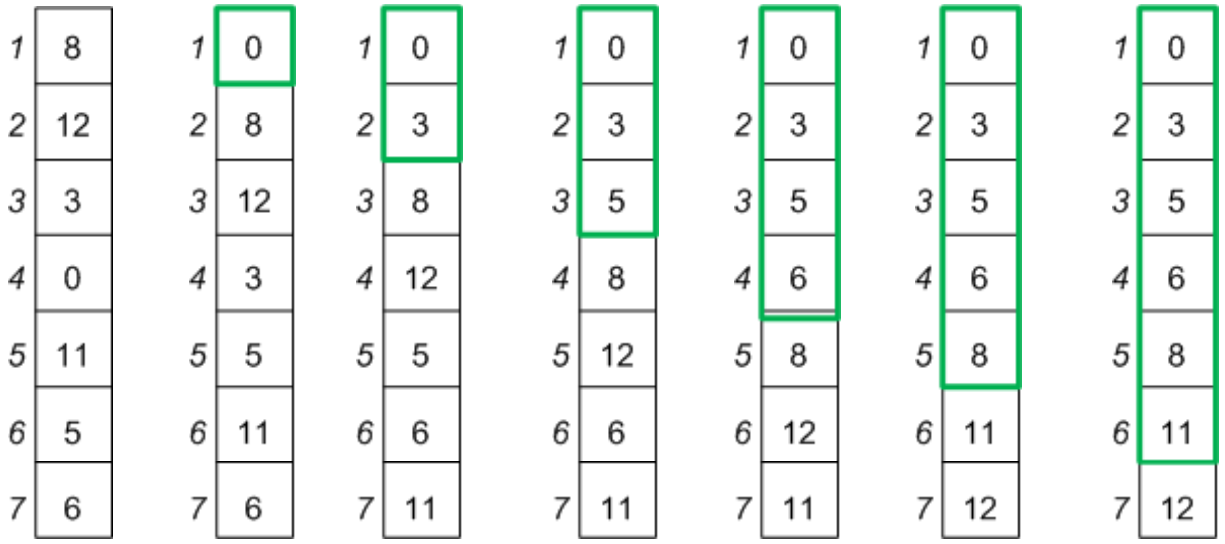
```
IND := 1;

for ZENB  in 2..LISTA'LAST loop

    if  LISTA(ZENB) > LISTA(IND)
    then  IND := ZENB;
    end if;

end loop;
```

c) -- Taula batean bilaketa lineala

```
AURKITUA := false;
IND := 0;
while IND < LISTA'LENGTH and  not
   AURKITUA loop
      IND := IND +1;
      if  ELEMENTUA = LISTA(IND)
      then      AURKITUA := true;
      end if;
end loop;
```

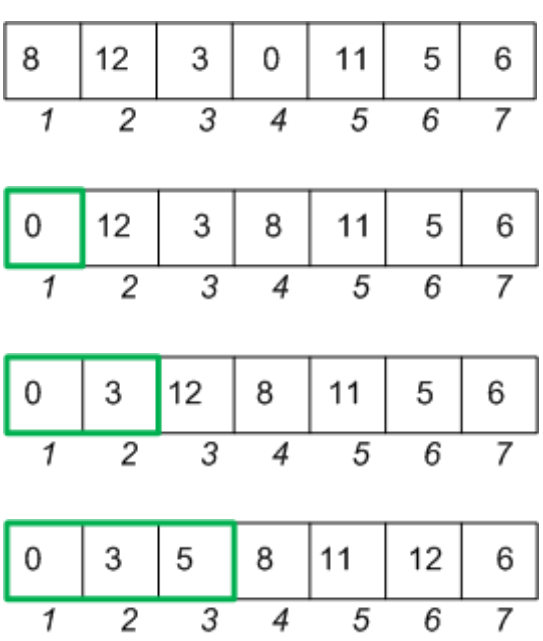
```
begin          --  ∀i (1≤i≤ n  → 1 ≤ T(i) ≤ N)
(1)for  I  in T'RANGE  loop
(2)   if T(I) > 0
(3)   then B := T(I); K := 0;
(4)          while B> 0 loop
(5)             if (B rem 10 = X)  then K := K + 1;  end if;
(6)             B := B/10;
              end loop;
(7)           TEXT_IO.PUT(INTEGER'IMAGE'T(I) &"-an: "&
                          INTEGER'IMAGE(K)&" aldiz ");
      end if;
      end loop;
end;
```

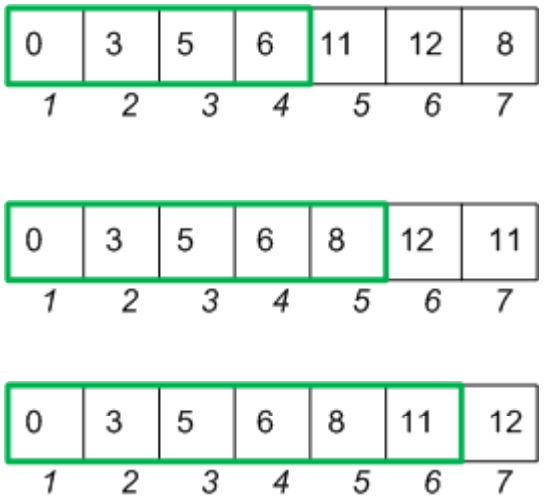
```
procedure BURBUILA (S: in out OS_SEK) is
-- BubbleSort
begin
   for  I  in S'RANGE  loop
        for J  in reverse I+1..S'LAST  loop
                if  S(J-1) >S(J) then
                        TRUKEA(S(J-1),S(J));
                end if;
        end loop;
   end loop;
end BURBUILA;
```



$$T(n)=\sum_{i=1}^{n}(n-i)=\frac{0+(n-1)}{2}\cdot n=\frac{n^2-n}{2}\in\Theta\left(\frac{1}{2}n^2\right)$$

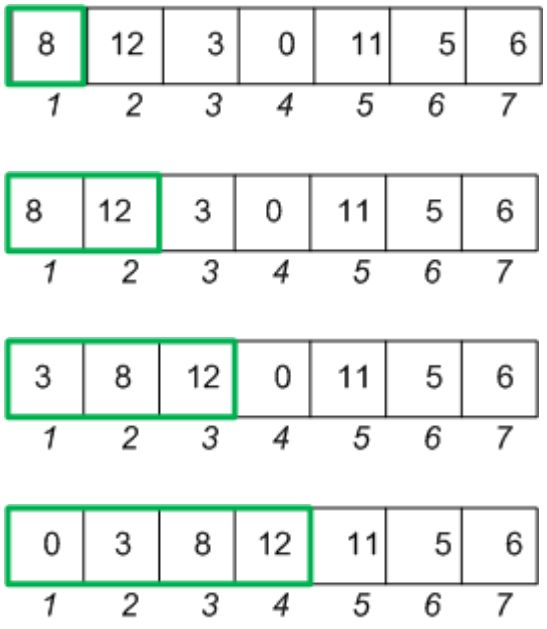
```
procedure  AUKERAKETA  ( S: in out OS_SEK) is
-- SelectionSort
 begin
   for  I  in  S'FIRST.. S´LAST-1  loop
        Min_J := I; Min_B := S(I);
        for  J  in reverse  I+1..S´LAST loop
           if S(J) ≤ Min_B
           then  Min_J := J; Min_B :=S(J); end if;
        end loop;
        S(Min_J):= S(I);  S(I):= Min_B;
   end loop;
end AUKERAKETA;
```

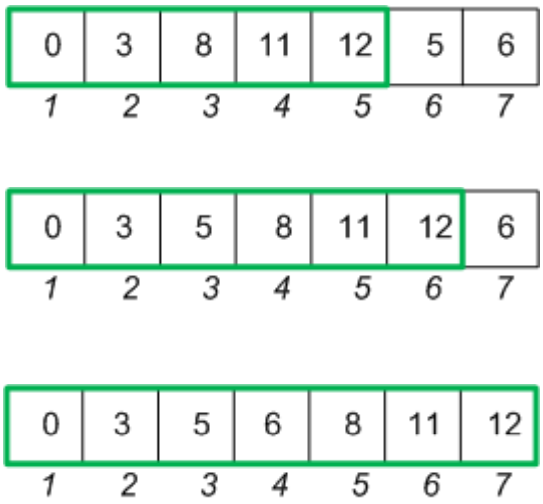




$$T(n)=\sum_{i=1}^{n-1}(n-i)=\frac{0+(n-1)}{2}\cdot n=\frac{n^2-n}{2}\in\Theta\left(\frac{1}{2}n^2\right)$$

```
procedure  TXERTAKETA  (S: in out OS_SEK) is
-- InsertionSort
begin
   for  I  in  S'FIRST+1..S'LAST  loop
        X:= S(I);  J:= I-1; JARRAITU := true;
        while  J>S'FIRST-1  and  JARRAITU  loop
          if S(J)>X  then S(J+1) := S(J); J:= J-1;
          else  JARRAITU := false; end if;
        end loop;
        S(J+1) := X;
   end loop;
end TXERTAKETA;
```





i. osagaia txertatzeraren kostua $\left(\sum_{j=1}^{i-1}\frac{1}{i}j\right)+\frac{1}{i}(i-1)=\frac{i+1}{2}-\frac{1}{i}$

T_b Kostua orora : $T_b(n)=\sum_{i=2}^{n}\left(\frac{i+1}{2}-\frac{1}{i}\right)=\frac{n2+3n-4}{2}-\sum_{i=2}^{n}\frac{1}{i}\in\Theta\left(\frac{1}{4}n^2\right)$

11 Alg. errekurtsiboen T(n)a eta ordena

- Iteratiboetako erregela mnemoteknikoak+ egiten den dei errekurtsibo kopurua eta bakoitzaren “tamaina” jaso (Ikus adibideak)

- Exekuzio-denboraren ekuazioa sinplifikatu eta ez-errekurtsiboa bilakatu. Metodoak: Hedapen metodoa, aldagai aldaketa (eta ekuazio karakteristikoaren metodoa)

- Ekuazioa sinplifikatu (baturaren erregela, limiteak,...)

- Ordena mugatu

FaktorialaIt (N,K)

K:=1;

begizta I:= 1..N errepika K:=K*I;

$T_i(n)=a+b\ n \in \Theta(n)$

Faktoriala (N,K)

baldin N<=1 orduan K:=1;

bestela Faktoriala(N-1,B); K:=N*B;

$T_e(n) = a+1\ T_e(n-1) \approx 1+ T_e(n-1)$

$= j+T_e(n-j)$

$= (n-1)+T_e(1) \in \Theta(n)$

12 Errekurtsio ekuazioen ebazpen metodoak

Hedapena

Hanoi

$T(0) = b$

$T(n) = 2\,T(n-1)+ a$

$\approx 2\,T(n-1)+ 1$

$= 2\,(2T(n-2)+ 1)+1=2^2\,T(n-2) + 2 +1$

$=2^2\,(2T(n-3)+1) + 2^1 +2^0 =2^3T(n-3)+ 2^2 + 2^1 +2^0$

$=2^iT(n-i)+ 2^{i-1}+\ldots+ 2^1 +2^0$ i pausoan

$=2^nT(0)+ 2^{n-1}+\ldots+ 2^1 +2^0$ n-i=0

$$= 2^n b + \sum_{j=0}^{i-1} 2^j = b2^n + 2^i - 1 = (b-1)2^n - 1 \in \Theta\left(2^n\right)$$

Faktoriala

$T(1) = a \approx 1$

$T(n) = T(n-1)+ 3n+7$

$\approx T(n-1)+ n$

$= T(n-2)+ (n-1)+n$

$= T(n-i) + (i-1)+\ldots+ (n-1)+n$ i pausoan

$= T(1)+ 2+\ldots+ (n-1)+n$ n-i=1

$= 1+2+\ldots+ (n-1)+n \in \Theta(n^2)$

Ek. Karakteristikoa

Prozesua:

- T(f(n)) gaiak ezkerraldean ordenean

$$a_0\ t_n + a_1\ t_{n-1} + ...+ a_k\ t_{n-k}$$

- Eskuin aldean identifikatu

$(b_1)^n\ p_1\ (n) + (b_2)^n\ p_2(n) + ...$ non $b_2 \neq b_2 \neq \ldots$

eta d_i $p_i(n)$-ren gradua den

- Jaso orain arteko kalkuluak eredua jarraituz:

$(a_0\ x^k + a_1\ x^{k-1} + ...+ a_k\ )\ (x - b_1)^{d1+1}\ (x - b_2)^{d2+1} .... = 0$

- Erroak bilatu (r_j), haien anizkoiztasunak identifikatuz

  - Erro guztiak desberdinak: $t_n = \sum_{i=1}^{n} k_i\ r_i^n$

  - Zenbait erro berdin: $t_n = \sum_{i=1}^{d} \sum_{j=0}^{m_i-1} \left(k_{ij}\ n^j \right) r_i^n$

    - r1,r2,...,rd erro desberdinak diren,
    - kij konstanteak,
    - mi: ri-erroaren anizkoitasuna
    . (errepikatzen den aldi kopurua)

Fibonacci

$T(0) = b$

$T(n) = T(n-1)+ T(n-2)+ a$

$\approx 2\,T(n-1)+ 1$

$= 2\,(2T(n-2)+ 1)+1=2^2\,T(n-2) + 2 +1$

$=2^2\,(2T(n-3)+1) + 2^1 +2^0 =2^3T(n-3)+ 2^2 + 2^1 +2^0$

$=2^iT(n-i)+ 2^{i-1}+\ldots+ 2^1 +2^0$ i pausoan

$=2^nT(0)+ 2^{n-1}+\ldots+ 2^1 +2^0$ n-i=0

$T(0) = b$

$T(n) = T(n-1)+ T(n-2)+ a$

$t_n - t_{n-1} - t_{n-2} = a$

$t_n - t_{n-1} - t_{n-2} = 1^n \times an^0$

$(x^2-x^1-x^0)\quad (x-1)^{0+1} = 0$

$$t_n = k_1\left(\frac{1+\sqrt{5}}{2}\right)^n + k_2\left(\frac{1-\sqrt{5}}{2}\right)^n + k_3 1^n \in \Theta\left(\left(\frac{1+\sqrt{5}}{2}\right)^n\right)$$

Aldagai aldaketa

Bilaketa dikotomikoa

T(n) = T(n/2) + a

$T(2^k) = T(2^k/2) + a$

$T(2^k) - T(2^{k-1}) = 1^k \times a\, k^0$

$t_k - t_{k-1} = 1^k \times a\, k^0$

$(x-1)(x-1)^{0+1} = (x-1)^2 = 0$

$n = 2^k$

$b^k \times p(k)$

$t_k = c_1\, 1^k + c_2\, 1^k\, k$

$t_k = c_1 + c_2\, k$

$t_k = T(2^k) = T(n)$

$k = \lg n$

$T(n) = c_1 + c_2 \lg n \in \Theta(\lg n)$

$$\mathbf{T(n) = 3\,(n/2)^{1/2} + 4\,T(n/4)}$$

$n=4^k$

$$\approx 4\,T(n/4) + n^{1/2}$$

$n^{1/2}=(4^k)^{1/2}=2^k$

$$T(4^k) = 4\,T(4^{k-1}) + (4^k)^{1/2}$$

$$t_k - 4\,t_{k-1} = 2^k$$

$$t_k - 4\,t_{k-1} = 2^k \times 1k^0$$

$$(x-4)\qquad (x-2)^{0+1} = 0$$

$$t_k = c_1\,4^k + c_2\,2^k =$$

$$T(n) = c_1\,n + c_2\,n^{1/2} \in \Theta(n)$$

$$\mathbf{T(n) = 2\,T(n/2) + n\,\lg n}$$

$n=2^k$

$\lg n=k$

$$T(2^k) = 2\,T(2^{k-1}) + 2^k \lg 2^k$$

$$t_k - 2t_{k-1} = 2^k\,k = 2^k \times 1k^1$$

$$(x^1-2)\qquad (x-2)^{1+1} = (x-2)^3 = 0$$

$$t_k = c_1\,2^k + c_2\,2^k\,k + c_3\,2^k\,k^2$$

$$T(n) = c_1\,n + c_2\,n \lg n + c_3\,n\,(\lg n)^2 \in \Theta(n\,(\lg n)^2)$$

Datu-egitura aurreratuak:
meta, partiketa eta grafoa

R. Arruabarrena
LSI - UPV/EHU

Sarrera

- Datu-egitura konposatu ezagunak eta erabiliak dira
  - Zerrendak, ilarak, zuhaitzak, fitxategiak, …
- Badaude beste egitura batzuk oso aproposak direnak problema jakin batzuk eraginkorki ebazteko
- Metak, partiketak eta grafoak soilik azalduko ditugu hemen
  - Egituraren ezaugarriak, ohiko eragiketak, aplikazioak, …
  - Zenbateko zehaztasunez?
    - Problemek eskatzen dituzten baldintzak bete arteraino

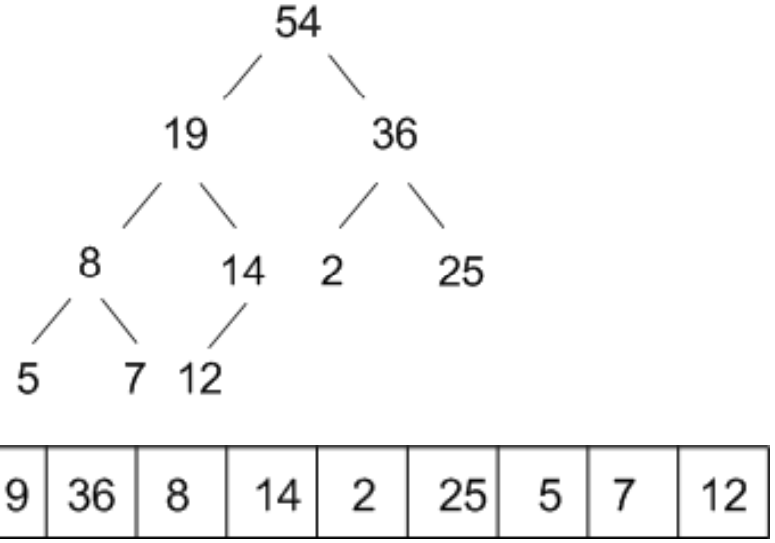
Metak

- Meta (montículo, heap):

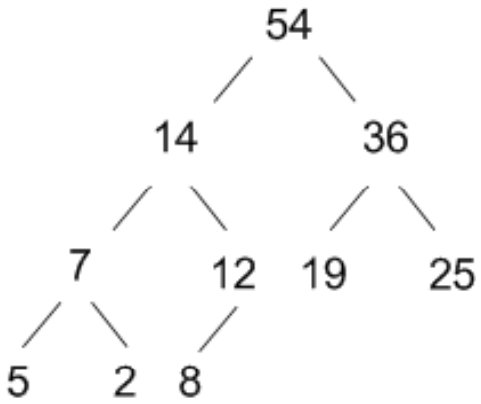
  taula baten gainean eraginkorki inplementaturik dagoen zuhaitz bitar **ia-betea**, bertako azpizuhaitz orok *maximoaren (minimoaren*) **propietatea** betetzen duelarik

- Aplikazio ugari du. Ezagunenak
  - Willias-en HeapSort ordenazio algoritmoa
  - Lehentasun dinamikoa duten ilarak
  - *n* osagaietatik bakarrik *k* txikienak/handienak behar direnean
    - aplikazio bakoitzean *k* desberdina

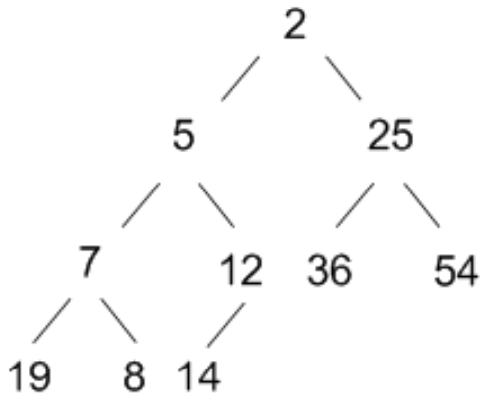
- **ia betea**: barne adabegi bakoitzak zehazki bi ume baditu, ezkerra eta eskuina, salbuespen bakarra gerta daitekeelarik: azkenaurreko mailan kokaturik dagoen eskuin alderago dagoen barne-adabegiak ume bakarra eduki dezake, ezkerra hain zuzen. Gainera, hosto guztiak azken/sakonera mailan edo azken-aurreko mailan kokaturik daude. Azken-aurreko mailan kokaturik dauden hosto oro maila bereko barne adabegi edozein baino eskuin alderago kokaturik dago.



- **maximoaren propietatea**: (Minimoaren propietatea) Erroan dagoen balioa ezker eta eskuin azpizuhaitzetan dauden balioak baino handiagoa da eta azpizuhaitz orok propietate bera betetzen du. Maximoaren propietatea betetzen duen metari maximoen meta deritzo.

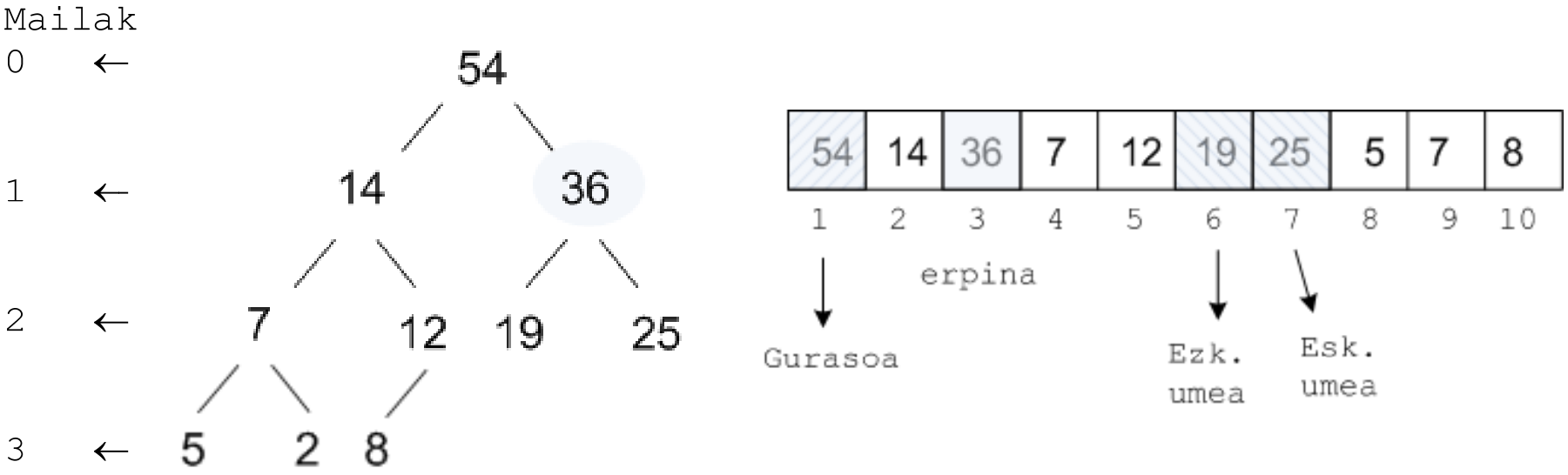


Maximoen meta



Minimoen meta

- **Metaren adierazpena**:
  - M(i)-ren gurasoa: M(i div 2); eta umeak M(2 i) eta M(2 i +1)
  - zuhaitzeko *k* mailako erpinak: taulan 2^k, 2^k+1, ..., 2^{k+1}-1 posizioetan
  - *q* osagaiko metak $\lfloor lg\ q \rfloor$ sakonera du

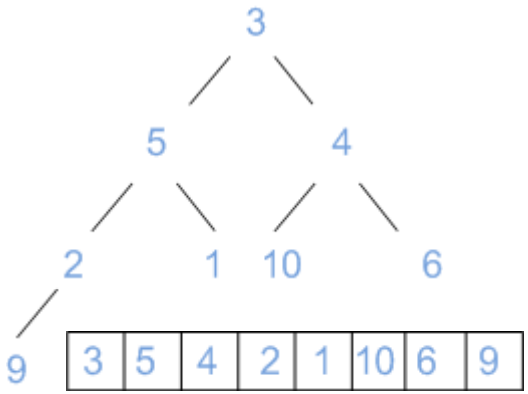


Metak

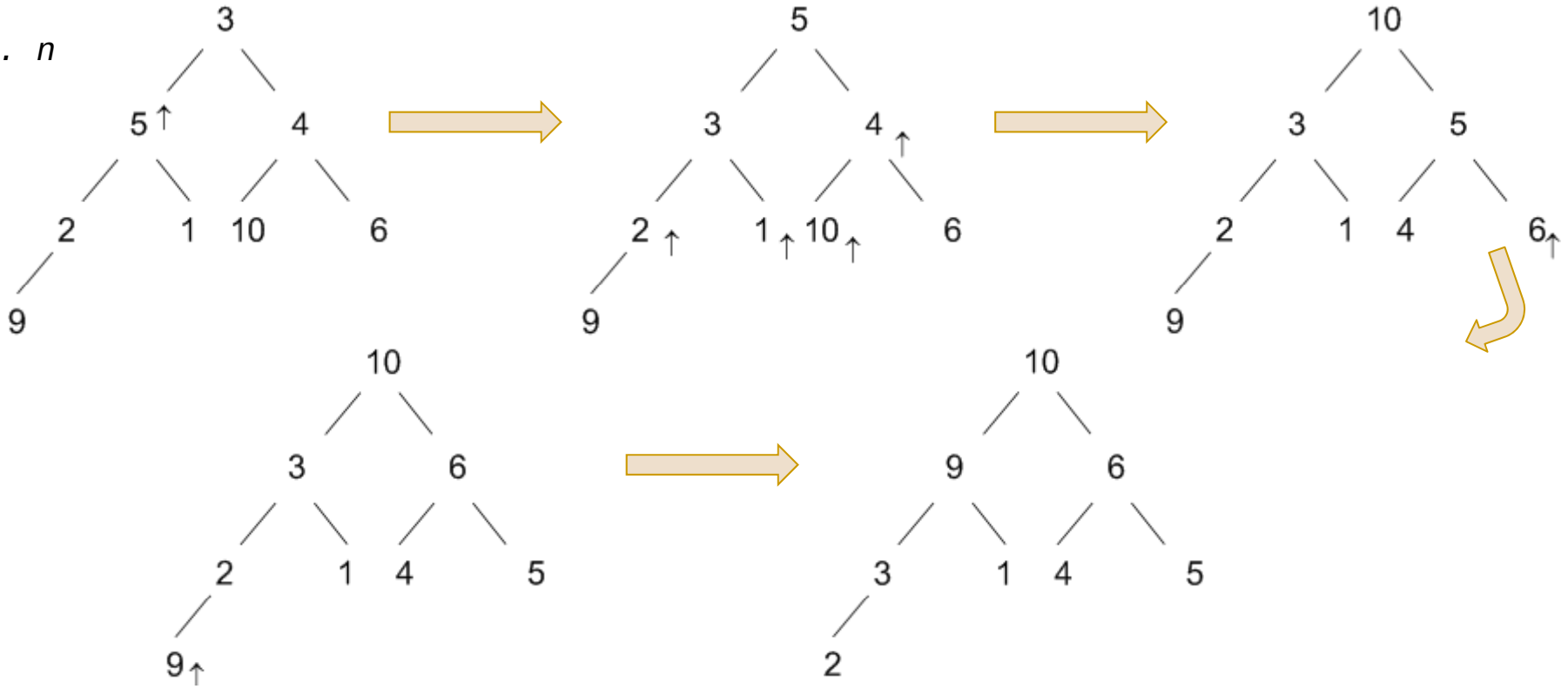
- Eragiketa erabilienak
  - `Hondoratu(Meta, Posizioa)`
    - `Hondoratu(Meta)` : erroa hondoratu
  - `Azaleratu(Meta, Posizioa)`
    - `Azaleratu(Meta)` : metako azken posizioa azaleratu
  - `MetaEraiki`
    - Azaleratuz 2..n posizioak
    - Hondoratuz (n div 2) .. 1
  - `Eguneratu(Meta, Posizioa, balioa)`
  - `Erroa(Meta,Balioa)`

```
procedure  Azaleratu  (M: in out  OsSek; P: in Indizea;) is
begin
   J:= P;
   while  M'Frist < J  and then  S(J div 2) < S(J) loop
        Trukea(M(J), M(J div 2));   J:= J div 2;
   end loop;
end;


procedure Hondoratu (M: in out OsSek; P: in Indizea; ) is
begin
   J:= P; SGiltza:=M(P);
   while  2•J ≤ M'Last  loop
        UmeMax:= 2•J ;
        if  2•J < M'Last  and then  M(2•J+1) > M(2•J)
        then  UmeMax:= 2•J+1;  end if;
        if  SGiltza < M(UmeMax)
        then  M(J):= M(UmeMax);  J:= UmeMax;
        else  exit  when  true; end if;
   end loop;
   M(J):=SGiltza;
end;
```

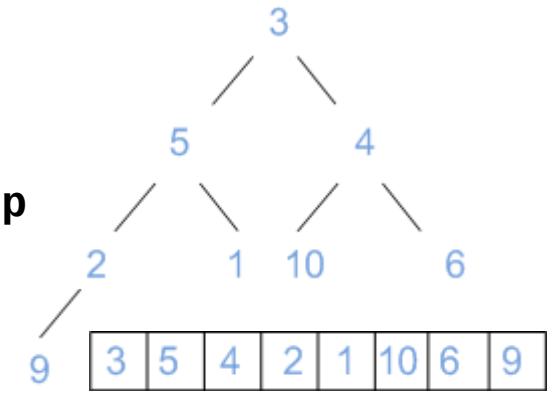
```
procedure  MetaEraiki1 (T:in out OsSek;) is
begin
   for  K in T'First+1..T'Last  loop
        Azaleratu(T(T'First,..,K),K);
         --   Azaleratu(T,K);
   end loop;
end MetaEraiki1;
```



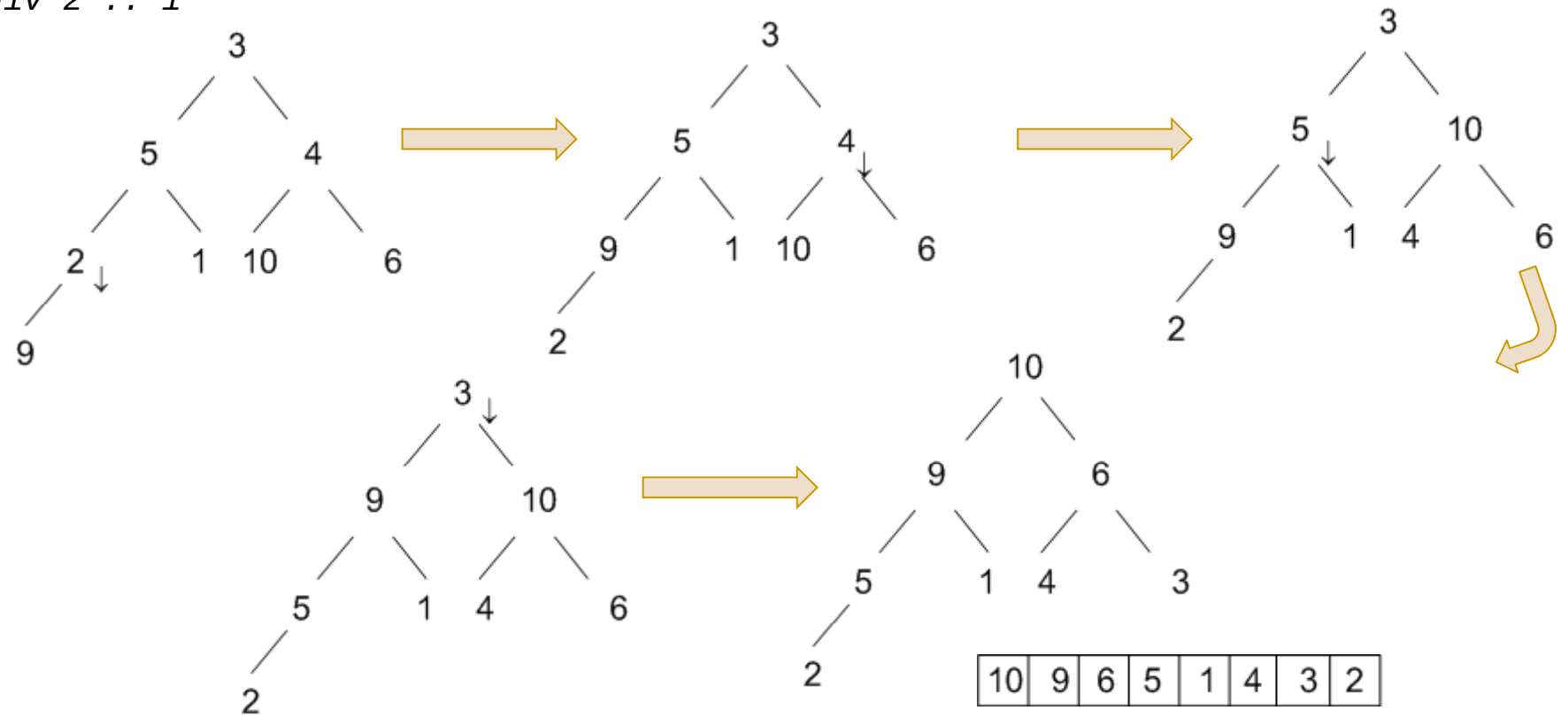
2 .. n



```
procedure  MetaEraiki2 (T:in out OsSek;) is
begin
   for I  in reverse  T'Frist.. T'Last div 2  loop
        Hondoratu(T, I);
   end loop;
end MetaEraiki2;
```



N div 2 .. 1



Meta eraiki. Analisia

- Azaleratuz

$$T(n)=\lfloor \lg 2\rfloor+\lfloor \lg 3\rfloor+\ldots+\lfloor \lg n\rfloor=\sum_{i=2}^{n}\lfloor \lg i\rfloor<\sum_{i=2}^{n}\lg i\le\frac{1}{\ln 2}\int_{2}^{n+1}\ln x dx$$

$$T(n)\le\frac{1}{\ln 2}\int_{2}^{n+1}\ln x dx=\frac{1}{\ln 2}\left[x\ln x-x\right]_{2}^{n+1}\in \mathrm{O}(n\ln n)$$

- Hondoratuz

$$T(n)=\sum_{m=0}^{s-1}2(s-m)(m.\text{ mailan dauden adabegien kopurua})$$

$$T(n)=\sum_{m=0}^{s-1}2(s-m)2^{m}=2s\sum_{m=0}^{s-1}2^{m}-2\sum_{m=0}^{s-1}m2^{m}$$

$$T(n)=2s\left(2^{s}-2^{0}\right)+2\left(2^{s}-2^{1}-(s-1)2^{s}\right)=2s2^{s}-2s+22^{s}-4-2s2^{s}+22^{s}$$

$$T(n)=42^{s}-2s-4=4\cdot 2^{\lfloor \lg n\rfloor}-2\lfloor \lg n\rfloor-4\in\Theta(n)$$

Partiketa

- Egitura:
  - Zenbakiturik dauden P osagai ditugu.
  - Osagai bakoitzak multzo, klase, baliokidetasun klase edo osagaik konexu bat adierazten du.
  - Osagai bakoitza multzo bakar batean dago.
  - Multzo guztien bilkurak P osagaiak ditu.
  - Multzo bakoitza identifikatzeko etiketa bat dugu.
  - Etiketa denak desberdinak dira.
  - Aldi berean zenbait multzo edo klase maneiatu behar dugu.
  - P osagai dituen taula batez inplementa litezke multzo edo klase guztiak.

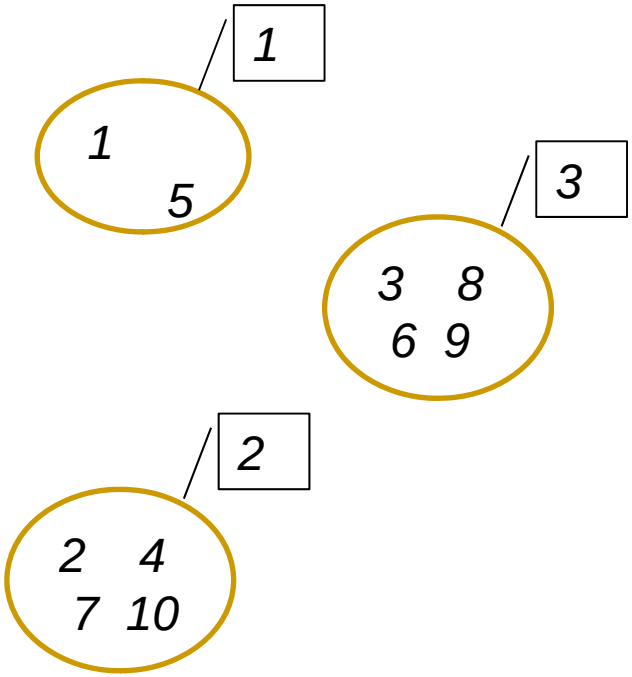
- Eragiketak egitura gain
  - Hasieratu: egitura balioz hornitu beharko da hasieran.
    - Normalki, osagai bakarreko multzo disjuntuetan
  - `Bilatu(I)`: I objektua partaidea den multzoko etiketa itzuliko du
  - `Bateratu(E1,E2)`: E1 eta E2 etiketa desberdinak izanik, bakoitzak identifikatzen duten klaseen bategitea; hots, bi klase/multzo edukitzetik, bakarra edukitzera pasa

1. hurbilketa

- Etiketa multzoko osagai txikiena da. Multzo guztiak taula gainean

```
function  BILATU1 (PARTIKETA: in P_PARTIKETA_MOTA;
                   X: in OSAG_ETIK) return OSAG_ETIK is
begin
   return  PARTIKETA(X);
end BILATU1;
```

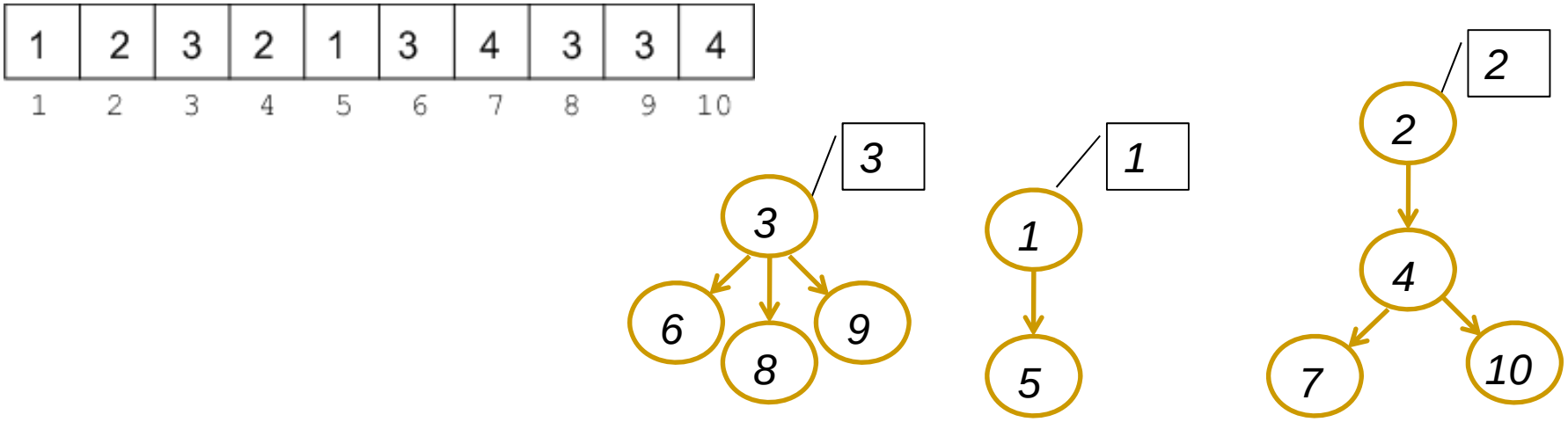
```
procedure BATERATU1 (PARTIKETA: in out P_PARTIKETA_MOTA;
                     E1,E2: in OSAG_ETIK) is
   begin
   -- E1 eta E2 etiketak dituzten multzoak bat egiten ditu.
   I:= E1; J:= E2;
   if  I>J  then  TRUKEA(I,J); end if; -- bi multzoen etiketa berria I
   for  K  in  1..P  do
        if PARTIKETA(K)=J then PARTIKETA(K):= I; end if;
   end loop;
end BATERATU1 ;
```



1	2	3	2	1	3	2	3	3	2
1	2	3	4	5	6	7	8	9	10

- Analisia: Hasierako egoeratik hasi, eta [`Bateratu1, Bilatu1`]^{m} sekuentzia posible guztien artetik denbora gehien behar duen sekuentziaren denbora-ordena **Θ(mP)** da.
  - `Bilatu1`-k Θ(1) behar du eta `Bateratu1`-k Θ(P)
  - Kasu txarrenean jarraian eginiko m `Bateratu1` eragiketek Θ(P) denbora-ordena lukete

- 2. hurbilketa
  - Etiketa multzoko objektu txikiena da. Multzo guztiak taula gainean
  - Multzo bakoitza zuhaitz baten bidez adierazten da
    - PARTIKETA(i) = i $\Rightarrow$ objektua bere multzoko etiketa da eta dagokion zuhaitzaren erroa
    - PARTIKETA(i) = j, eta i $\neq$j $\Rightarrow$ i eta j objektuak multzo berean daude eta multzo horri dagokion zuhaitzean, *j* adabegia *i*-ren gurasoa da.



```
function BILATU2 (PARTIKETA: in P_PARTIKETA_MOTA;
                  X: in OSAG_ETIK)  return OSAG_ETIK is
 I:OSAG_ETIK:= X;
begin
   while  PARTIKETA(I) /=I   loop
            I:= PARTIKETA(I);
   end loop;
   return (I)
end BILATU2;


procedure BATERATU2 (PARTIKETA: in out P_PARTIKETA_MOTA;
                     E1,E2: in OSAG_ETIK) is
begin
   if  E1<E2  then PARTIKETA(E2):= E1;
   else PARTIKETA(E1):= E2;
   end if;
end BATERATU2
```

- Analisia (2. hurbilketa)
  - Hasierako egoeratik hasi, eta [`Bateratu2`, `Bilatu2`]^m sekuentzia posible guztien artetik denbora gehien behar duen sekuentziaren denbora-ordena kalkulatu behar da.
  - `Bateratu2()` $\in \Theta(1)$
  - *m* maila dituen zuhaitz bateko sakonera mailan dagoen *x* osagaiaren etiketa bilatzeak denbora (zuhaitzaren sakoneran) lineala behar du: `Bilatu2(Partiketa,x)` $\in \Theta(m)$

  - Adib:

    1. `Bateratu2(n/2+1, n/2)`
    2. `Bateratu2(n/2, n/2-1)`

    …

    (n/2-1). `Bateratu2(3, 2)`

    (n/2). `Bateratu2(2, 1)`

    (n/2+1). `Bilatu2(n/2+1)`

    …

    n. `Bilatu2(n/2+1)`

Analisia (2. hurbilketa)

1. `Bateratu2(n/2+1, n/2)`
2. `Bateratu2(n/2, n/2-1)`

…

(n/2-1). `Bateratu2(3, 2)`

(n/2). `Bateratu2(2, 1)`

Sakonera mailan dago, sakonera n/2 da

(n/2+1). `Bilatu2(n/2+1)`

…

n. `Bilatu2(n/2+1)`

Aginduen denbora-ordena:

$\Theta((n/2) *$ `Bateratu2`-ren ordena) + $\Theta((n/2) *$ `Bilatu2(n/2+1)`-ren ordena)=

$\Theta((n/2)* \Theta(1)) + \Theta((n/2) * \Theta(n/2)) =$

$\Theta(n) + \Theta(n^2) = \mathbf{\Theta(n^2)}$

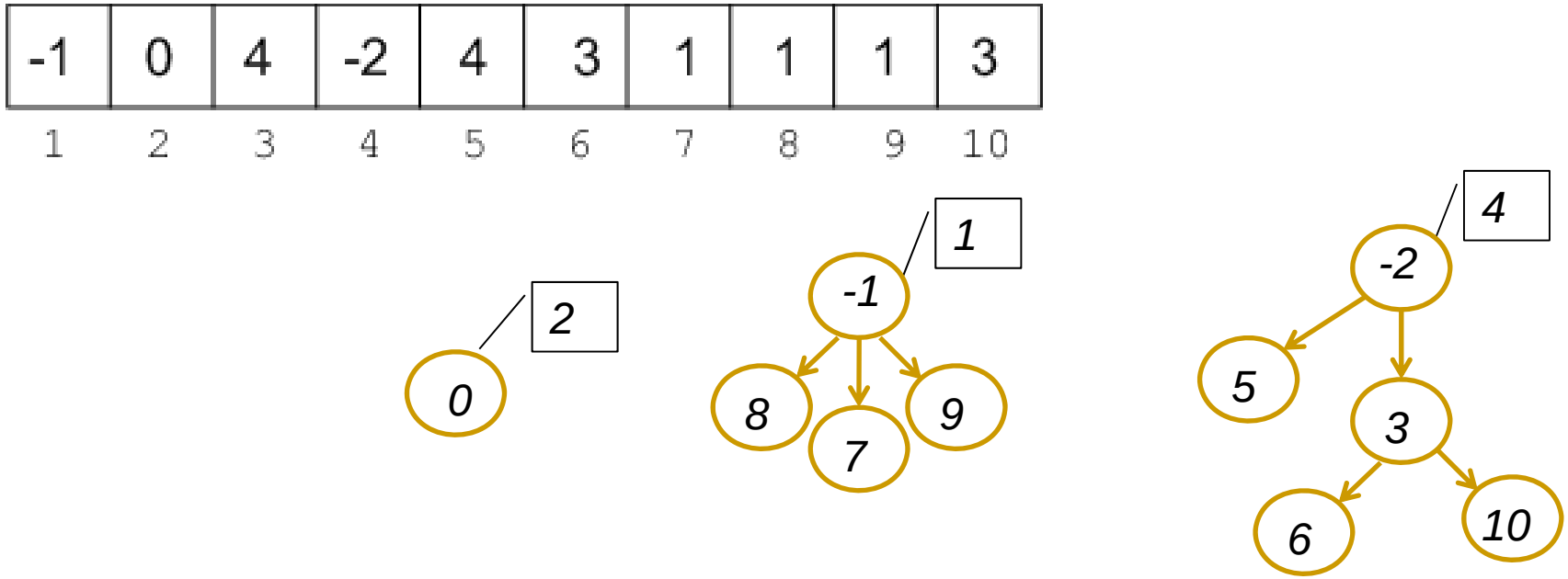
- 1 eta 2 hurbilketak
  - P≈n bi soluzioak antzekoak dira.
  - Baina, `Bateratu2()`-ren *k* dei ostean *k* sakonerako zuhaitza sor dezake. Sakonera mailako osagaien kostua handia da.
  - Sakoneraren handitzea kontrolatuko balitz, kostu aurrezpena izango genuke
  - 3. hurbilketak zuhaitz “*degeneratu*”-en sorkuntza ekiditen du

3. hurbilketa

- Multzoak zuhaitzen bidez inplementatuko dira
- Multzo guztiak taula bakar batean
- Multzoko etiketa multzoko objektu bat izanik, ez du zertan txikiena izan behar

- Estrategia: s1 eta s2 sakonera duten bi zuhaitz bateratzeko, sakonera txikiena duen zuhaitza sakonera handiagoa duen zuhaitzaren erroaren ume bihurtu.
  - lortzen den zuhaitz berriaren sakonera:
    - max(s1,s2) baldin eta s1 ≠s 2
    - 1+max(s1,s2) baldin s1=s2

- Estrategia honekin, hasierako egoeratik hasi, eta edozein `Bateratu3` sekuentzia egikaritu ondoren zuhaitzak **k osagai** baditu, zuhaitzen **sakonera** gehienez $\lfloor \mathbf{lg\ k} \rfloor$ da ⟸ **LEMA**



- `Bateratu2`$\in\Theta(1)$ eta `Bilatu2`$\in\Theta(1)$
- Hasierako egoeratik hasi, eta [Bateratu1, Bilatu1]^m sekuentzia posible guztien artetik denbora gehien behar duen sekuentziaren denbora-ordena da

Lemaren froga:

k: adabegi kopurua, s: sakonera, z: zuhaitza(multzoa)

- k=1:
  - $s_1=0$ izango da. Beraz, betetzen da $0 \leq \lfloor \lg 1 \rfloor$

- Indukzio hipotesi hedatua: $\forall i\ (1 \leq i < n \rightarrow s_i \leq \lfloor \lg i \rfloor)$
  - i adabegi dituzten zuhaitzentzat propietatea betetzen da.

- Suposa dezagun orain:
  1. z1 eta z2 zuhaitzek k1 eta k2 adabegi dituztela hurrenez hurren,
  2. k1< n eta k2< n eta k1+k2 ≥ n betetzen dela,
  3. s_{k1} eta s_{k2} z1 eta z2 zuhaitzen sakonerak direla, eta beraz
  4. i.h. aplikatuz $s_{k1} \leq \lfloor \lg k1 \rfloor$ eta $s_{k2} \leq \lfloor \lg k2 \rfloor$ beteko da

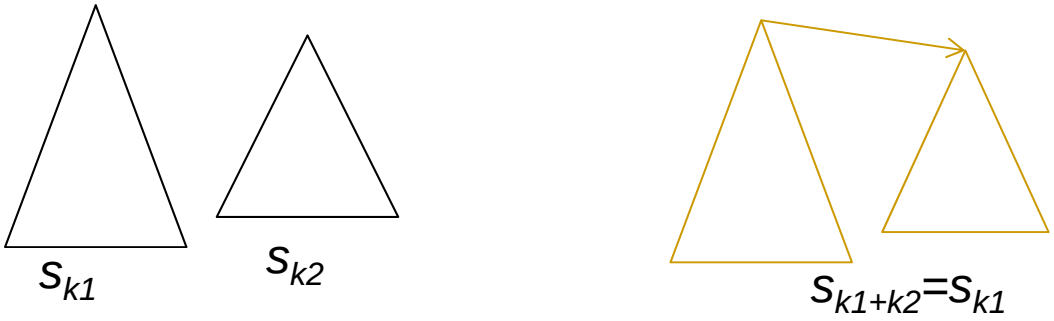
z1 eta z2 elkartzerakoan zuhaitz berriak k1+k2 adabegi izango ditu.

$$s_{k1+k2} \leq \lfloor \lg (k1+k2) \rfloor \text{ ???}$$

a) $s_{k2} \neq s_{k2}$. Suposatu $s_{k2}<s_{k1}$.

$s_{k1+k2} = s_{k1} \leq \lfloor \lg k1 \rfloor \leq \lfloor \lg (k1+k2) \rfloor$ beraz $s_{k1+k2} \leq \lfloor \lg (k1+k2) \rfloor$

- $s_{k2}>s_{k1}$ kasuaren frogaketa berdin

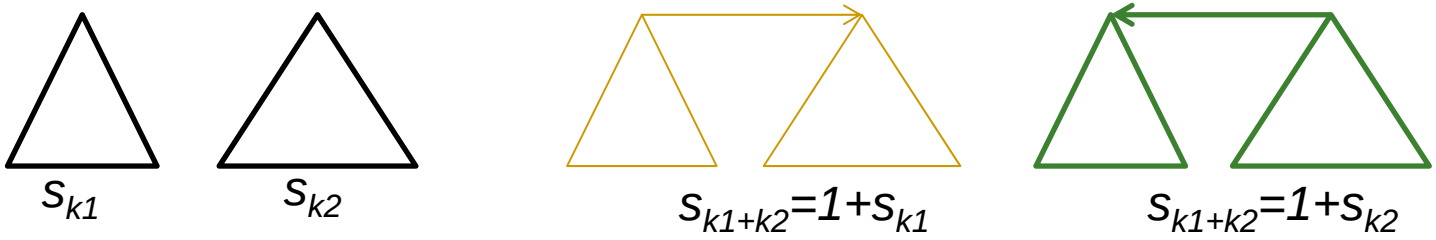


b) $s_{k2} = s_{k2}$.

$s_{k1+k2} = 1+s_{k1} \leq \lfloor \lg 2 \rfloor + \lfloor \lg k1 \rfloor \leq \lfloor \lg (2*k1) \rfloor = \lfloor \lg (k1+k1) \rfloor$, $k1 \leq k2$ suposatuz,

$= \lfloor \lg (k1+k1) \rfloor \leq \lfloor \lg (k1+k2) \rfloor$ beraz $s_{k1+k2} \leq \lfloor \lg (k1+k2) \rfloor$

- $k1 > k2$ balitz, orduan $s_{k1+k2} = 1+s_{k2}$ eginez froga



```
function BILATU3 (PARTIKETA: in P_PARTIKETA_MOTA;
                  X: in OSAG_ETIK) return OSAG_ETIK is
   I: OSAG_ETIK:= X;        -- x partaidea den multzoko etiketa itzultzen du
begin
   while  PARTIKETA(I) >0  loop
          I:= PARTIKETA(I)
   end loop;
   return (I)
end BILATU3;


procedure BATERATU3 (PARTIKETA: in out P_PARTIKETA_MOTA;
                     E1,E2: in OSAG_ETIK) is
begin
   if PARTIKETA(E1)=PARTIKETA(E2)
   then  PARTIKETA(E1):= PARTIKETA(E1)-1;  PARTIKETA(E2):= E1;
         -- lehenengo azpizuhaitzaren sakonera handitzen du
   elsif PARTIKETA(E1)<PARTIKETA(E2)  then   PARTIKETA(E2):= E1;
   else   PARTIKETA(E1):= E2;
   end if;
end BATERATU3;
```

- Analisia (3. hurbilketa)
  - Hasierako egoeratik hasi, eta [`Bateratu3`, `Bilatu3`]^m sekuentzia posible guztien artetik denbora gehien behar duen sekuentziaren denbora-ordena kalkulatu behar da.
  - `Bateratu3()`$\in\Theta(1)$ eta `Bilatu3(Partiketa,x)`$\in\Theta$(sakonera)
  - Adib:

    1. `Bateratu3(n/2+1, n/2)`
    …
    (n/2). `Bateratu3(2,1)`

    } 1+n/2 elementuko zuhaitza

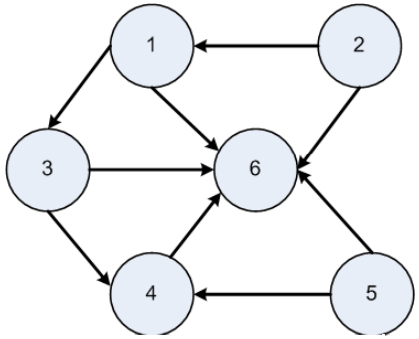
    (n/2+1). `Bilatu3(sakonerako hosto bat)`
    …
    n. `Bilatu3(sakonerako hosto bat)`

  - Lemak: K=1+n/2 => $s_k \leq \lfloor \lg (1+n/2) \rfloor$

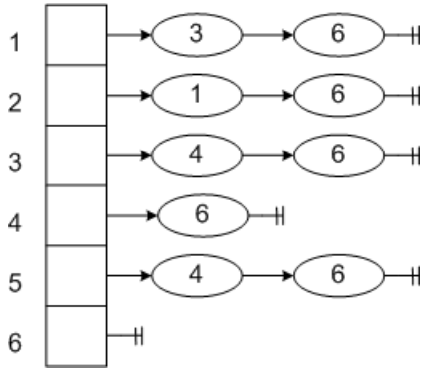
$$\Theta(n/2 * \Theta(1)) + \Theta((n/2) * \Theta(s_k) \leq \Theta(n) + \Theta(n) * \Theta(\lfloor \lg (1+n/2) \rfloor) = \mathbf{\Theta(n \lg n)}$$

Grafoa

- G=<E,A>, grafoa Erpinez eta Arkuez/ertzez osaturik dago, eta pisuduna hala ez izan liteke
  - N erpinak: hiriak, uharteak, etxebizitzak, konputagailuak, geltokiak, …
  - A arkuak/ertzark: errepideak, itsas-loturak, kaleak, erlazioak,…

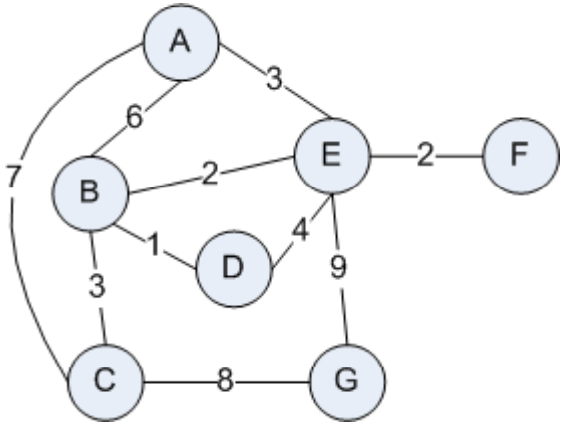


- Adierazpide erabilienak
  - Auzokideen zerrendak, auzokidetza matrizeak



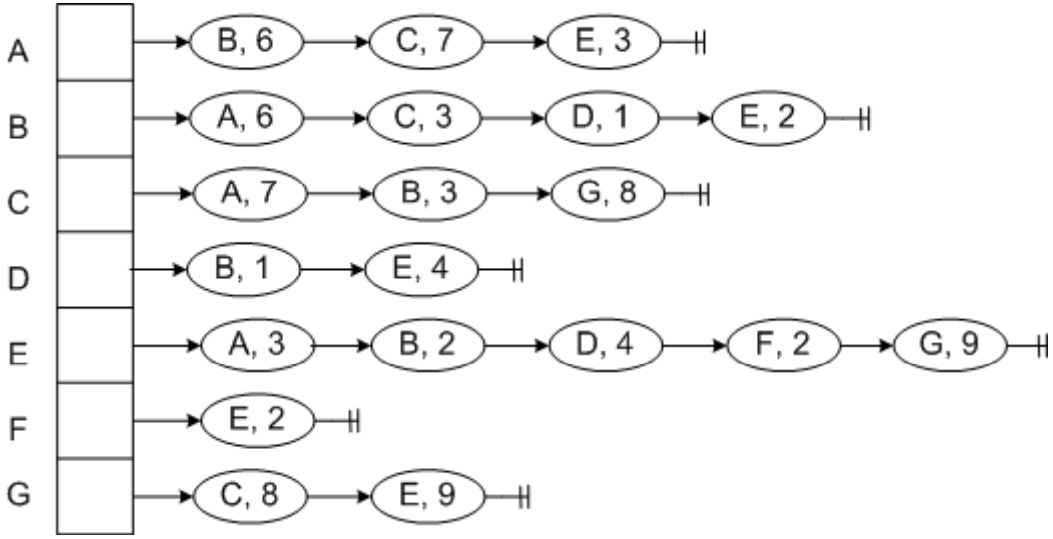
	1	2	3	4	5	6
1	0	0	1	0	0	1
2	1	0	0	0	0	1
3	0	0	0	1	0	1
4	0	0	0	0	0	1
5	0	0	0	1	0	1
6	0	0	0	0	0	0

Grafo ez-zuzendu eta pisuduna



	A	B	C	D	E	F	G
A	0	6	7	0	3	0	0
B	6	0	3	1	2	0	0
C	7	3	0	0	0	0	8
D	0	1	0	0	4	0	0
E	3	2	0	4	0	2	9
F	0	0	0	0	2	0	0
G	0	0	8	0	9	0	0

Auzokidetza matrizea



Auzokideen zerrenda

Grafo adierazpideen arteko bihurketa (1)

- Auzokideen zerrenda auzokidetza matrize bihurtzeko
  - Auzokidetza matrizea 0-ra hasieratzen da: $\Theta(N^2)$
  - *Irten* erpin bakoitzeko $\Theta(N+A)$
    - Bere auzkokide zerrenda lortu eta bertako *Iritsi* erpin bakoitzeko matrizean (*Irten*,*Iritsi*) erlazioaren existentzia erregistratu

$\Theta(N^2+(N+A)) = \Theta(N^2)$

```
procedure  BihurtuZerMatG (GZ: in GrafoaZ; GM: out GrafoaM) is
       -- Erpin kopuruaren esparrua ≡ 1..N

begin
   for  Irten in  1..N  loop
      for  Iritsi in  1..N  loop GM(Irten,Iritsi):=0; end loop;
    end loop;

   for  Irten in  1..N  loop
       Irten_Auzokideak:= Auzokideak(GZ, Irten);
       while  not( HutsaDaL? (Irten_Auzokideak))  loop
         (Iritsi, pisua):= LehenengoaL(Irten_Auzokideak);
         Hondarra(Irten_Auzokideak);
         GM(Irten, Iritsi):= pisua;        -- piduna ez balitz, 1
      end loop;
   end loop;
end BihurtuZerMatG ;
```

Adierazpideen arteko bihurketa (2)

- Auzokidetza matrize auzokideen zerrenda bihurtzeko
  - Irten erpin bakoitzeko — n aldiz
    - Bere auzokideen zerrenda hutsa egin — $\Theta(1)$
    - Iritsi erpin bakoitzeko — $\Theta(N)$
      - (*Irten,Iritsi*) erlazioak existitzen duenean, Irten erpinaren auzokideen zerrendara gehituz erregistratu

$\Theta(N^2)$

```
procedure  BihurtuMatZerG (GM: in GrafoaM; GZ: out GrafoaZ) is
      -- Erpin kopuruaren esparrua ≡ 1..N

begin
   for  Irten in  1..N  loop
        ZerrendaHutsa(GZ(Irten));

       for  Iritsi in  1..N  loop
          if GM(Irten, Iritsi)>0
          then GehituZ(GZ(Irtenn),
                       ErpinaEgin(Iritsi, GM(Irten, Iritsi));
          end if;
       end loop;
   end loop;
end BihurtuMatZerG;
```

Pisua

Grafo gaineko algoritmoak: sakonerako eta zabalerako korritzeak

R. Arruabarrena
LSI - UPV/EHU

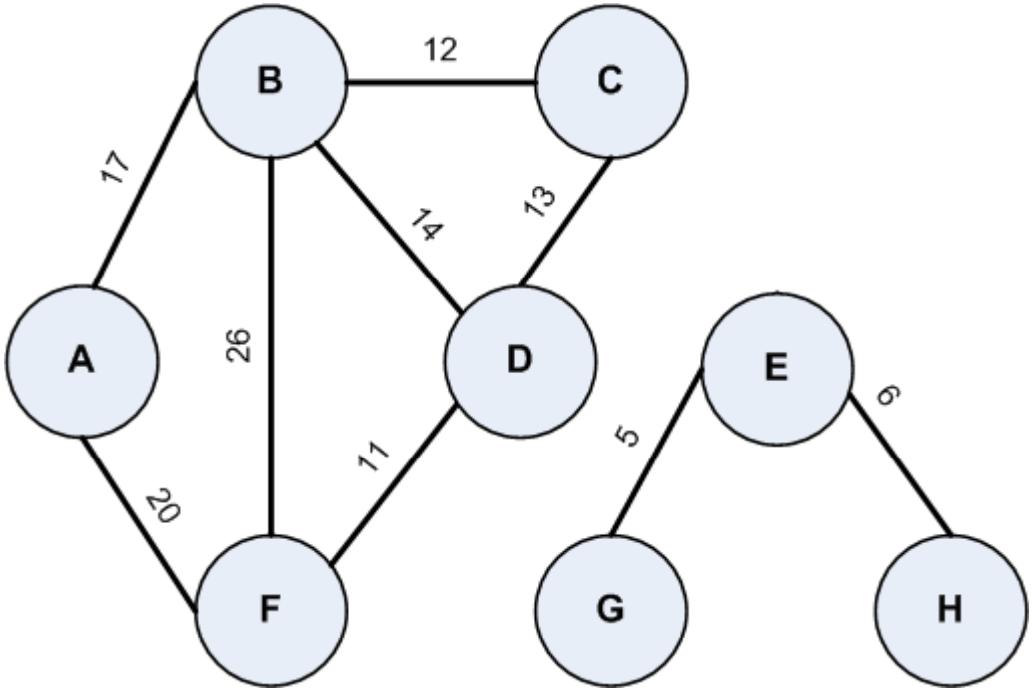
Sarrera

- Problema algoritmiko asko grafoen bidez modelatzen dira
  - Bide motzenak, hedapen zuhaitz minimoak, osagai konexuak,...
- Grafo datu-egiturak eta hauen gaineko korritzeak ematen dute soluzioa maiz
- Korritze esplizituak
  - Sakonerako korritzea
  - Zabalerako korritzea
- Korritze inplizituak
  - backtrack algoritmoak

```
procedure  GrafoaKorritu (G: in Grafoa) is
   Aztertua: constant Boolean := True;
   Bisitak: Taula(1..N):=(Others=> not (Aztertua));
    -- Erpin kopuruaren esparrua ≡ 1..N
begin

   for  Erpin_Bat  in  1..N loop
       if  not MarkatuaDago(Bisitak, ErpinBat)  then
              MarkaIpini(Bisitak, ErpinBat, Aztertua);
              SK(G, ErpinBat);            -- edo  ZK(G,
   ErpinBat);
       end if;
   end loop;

end GrafoaKorritu;
```

```
procedure  SK  (G: in Grafoa; A: in Erpina) is
   AAuzokideenKopia: ErpinenL;
   Lehena: Erpina;

Begin
   AAuzokideen_Kopia:= Auzokideak(G, A);

   while  not( HutsaDaL? (AAuzokideenKopia))  loop
        Lehena:= LehenengoaL(AAuzokideenKopia);
        Hondarra(A_AuzokideenKopia);

        if not Markatua_Dago(Bisitak, Lehena)  then
               Marka_Ipini(Bisitak, Lehena, Aztertua);
               S_K(G, Lehena);
        end if;

   end loop;
end S_K;
```

```
procedure  ZK  (G: in Grafoa; X: in Erpina) is
 Ilara: IlaraDMA:=HutsaI; YrenAuzokideak: ErpinenL;
 Y,Z: Erpina;
begin
   IlarariGehitu(Ilara, X);

   while not( HutsikDagoI?(Ilara))  loop
       Y:= Burua(Ilara);   Ilara:= IlaratikKendu(Ilara);
       YrenAuzokideak:= Auzokideak(Y).;
       while  not ( HutsikDagoL?(YRenAuzokideak))  loop
         Z:=LehenaL(YrenAuzokideak); Hondarra(YrenAuzokideak);

         if  not MarkatuaDago?(Bisitak, Z)  then
              MarkaIpini(Bisitak, Z, Aztertua);
              IlarariGehitu(Ilara, Z);
         end if;

       end loop;
   end loop;
end ZK;
```



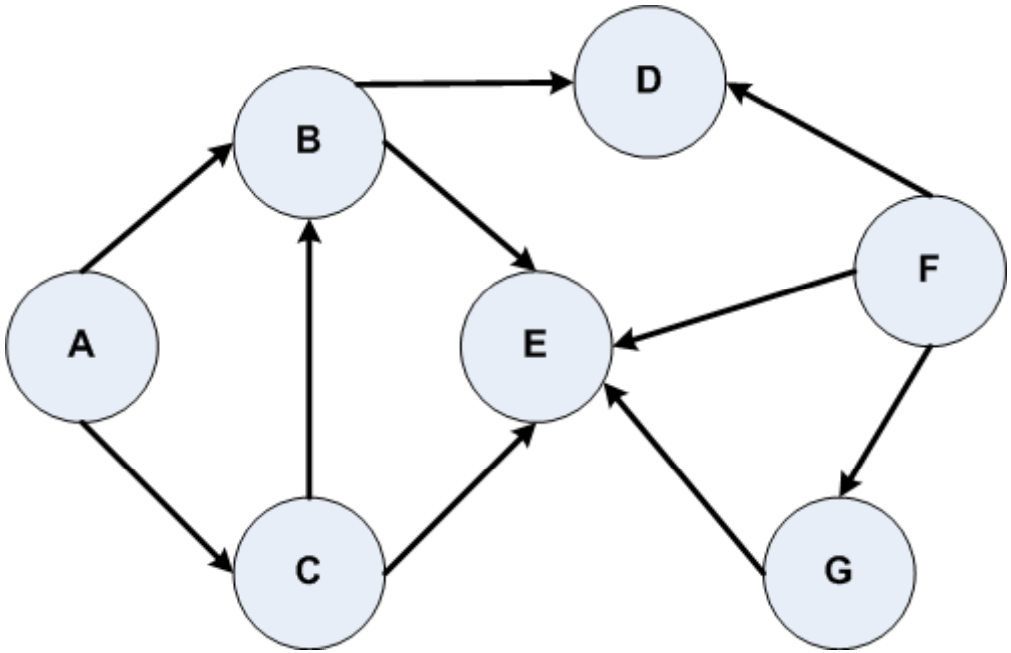
Grafo ez-zuzendu pisudunaren erpinen eta ertzen lehengo bisita ordenaren zerrendak:

- Sakonerako korritzean :

```
a (a,b) b (b,c) c (c,d) d (d,b) (d,f) f (f,a) (f,b) e (e,g) g (e,h) h
```

- Zabalerako korritzean:

```
a (a,b) (a,f) b (b,c) (b,d) (b,f) f (f,d) (c,d) e (e,g) (e,h) g h
```



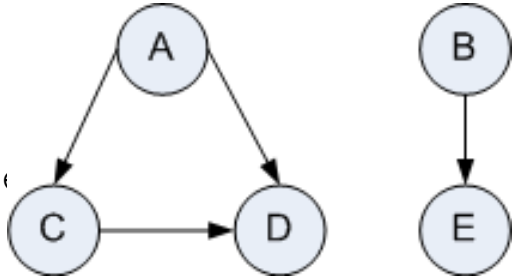
Grafo zuzenduaren erpinen eta arkuen lehengo bisita ordenaren zerrendak:

- Sakonerako korritzean : `a (a,b) b (b,d) d (b,e) e (a,c) c (c,b) (c,e) f (f,d) (f,e) (f,g) g (g,e)`
- Zabalerako korritzean: `a (a,b) (a,c) b (b,d) (b,e) c (c,b) (c,e) d e f (f,d) (f,e) (f,g) g (g,e)`

Korritzeak,

- Sakonerakoa

```
a (a,c) c (c,d) d (a,d)  b (b,e)
```



- Zabalerakoa
  - Ilaratzerakoan (hasieraketa, markatzen denean)

```
a (a,c) c (a,d) d (c,d) b (b,e) e
```

  - Ilaratik ateratzerakoan (benetako tratamendua)

```
a (a,c) (a,d) c (c,d) d b (b,e) e
```

Zuhaitz baten maila populatuena

Z zuhaitza emanik bertako zein K mailak adabegi gehien dituen kalkulatuko duen algoritmoa idaztea eta bere ordena kalkulatzea eskatzen da. Horietako maila bat baino gehiago existituko balira, maila handiena itzuli beharko luke algoritmoak

- Soluzioa uler dadin hainbat argipen
  - Zuhaitzak gehienez erpin adina maila izango ditu
  - MAILAK izeneko bektore global bat izango dugu maila bakoitzeko erpin kopurua metatzeko.
    - Mailetako kontagailuak hasieran zerora hasieratuko ditugu
  - Zabalerako korritzea erabiliz errotik urrunduz erpinak banaka aztertuko dira.
    - Erpinaren azterketak hau den mailako kontagailua inkrementatuko du
  - Korritzea bukatu ondoren, bektoreko balio maximoa maila populutsuenaren erpin kopurua da

```
procedure  GrafoaKorritu (G: in Grafoa, Erroa: in Erpina) is
   Aztertua: constant Boolean := True;
   Bisitak: Taula(ErpinKopEsparrua):=(Others=> not (Aztertua));
   Mailak: Taula(ErpinKopEsparrua):=(Others=> 0);

begin
   MarkaIpini(Bisitak, Erroa, Aztertua);
   ZK(G, Erroa);

   maxP:=maximoarenPosizioa(Mailak);
   writeln (Mailak[maxP], " erpin daude maila populatuenean,
            hau ", maxP,". Izanik.");

end GrafoaKorritu;
```

```
procedure  ZK  (G: in Grafoa; X: in Erpina) is
   Ilara: IlaraDMA:=HutsaI;  YrenAuzokideak: ErpinenL; Y,Z: Erpina;

begin
   IlarariGehitu(Ilara, (X, 0));                    -- lehenengo maila 0-a
   while not( HutsikDagoI?(Ilara))  loop
         (Y, my):= Burua(Ilara);        Ilara:= IlaratikKendu(Ilara);
         Mailak[my]= Mailak[my]+1;

         YrenAuzokideak:= Auzokideak(Y).;
         while  not  ( HutsikDagoL?(YRenAuzokideak))  loop
            Z:=LehenaL(YrenAuzokideak);Hondarra(YrenAuzokideak);
            if  not MarkatuaDago?(Bisitak, Z)  then
                 MarkaIpini(Bisitak, Z, Aztertua);
                 IlarariGehitu(Ilara, (Z, my+1));
            end if;
         end loop;
   end loop;
end ZK;
```

- Soluzioaren denbora analisia
  - `Mailak` bektorearen hasieratzea eta maximoaren kalkulua eragiketak linealak dira erpin kopuruan, $\Theta(n)$
  - Zabalerako korritzean erpin eta arku bakoitza behin tratatzen da
    - Erpin bakoitzeko kontagailu bat inkrementatuko da
    - Arkuak jomuga erpina harrapatzeko soilik aztertzen da
    - Eragiketa guztiak denboran konstantean burutzen direnez, korritzeak $\Theta(n+a)$ denbora du
- Baturaren erregela aplikatuz, $\Theta(n+(n+a)+n)=\Theta(n+a)$

Arbasoen zuhaitza

Demagun abizen jakin bateko zuhaitza genealogikoa dugula, erroa ezaguna izanik; hots, abizen hori eraman zuen lehenengo arbasoa.

Helburua sendiko edozein bi ahaideren arteko hurbilen duten lehengo arbaso komuna finkatuko duen prozedura egitea da.

Proposatutako soluzioaren denbora-analisia eta -ordena kalkula itzazu

- Soluzioa uler dadin hainbat argipen
  - Grafoa zuhaitza da, baina ez du zertan bitarra izan behar
    - odoleko zuzenen kopurua biren desberdina izan bailiteke
  - Bi ahaideak A1 eta A2 izango dira, zuhaitzean daude eta honen edozein mailatan.
  - Ahaide errepikaturik ez dago

- Sakonerako korritzea behar dugu problema ebazteko

- Erpin bakoitzeko aldagai pare bat egongo da; hots, topatu behar den ahaide bakoitzeko aldagai bat.
  - Hasieran pare guztiak False-ra hasieratuko dira

- Ahaide komun lehena topatzeko bi kasu daude
  - Ahaideak adar berean egotea
  - Ahaideak adar desberdinetan egotea

- Ahaideak adar berean
  - gurasoak pareko ahaide bat True izango du eta umeren batetik zintzilikatzen du beste ahaidea. Bigarrena, atzerako prozesaketaren ondorioz jasoko du, baina gurasoan detektatuko da adar batean daudela biak.

- Ahaideak adar desberdinetan
  - umeren bateko korritzeak ahaide bat topatzen duenean, post-prozesaketa bidez jasoko du gurasoak (adar horretatik ahaidea dagoela); beste umearekin berdin. Gurasoaren ume guztietatik abiatutako adarkatzeetatik biak topatu direnean, guraso hori da ahaide komun lehena.

```
procedure  AhaideKomunLehena (G: in Grafoa;
          Erroa, A1,A2: in Erpina;     ArbasoLehena: out Erpina) is

   Amaitua: constant Boolean := False;
   Ahaideak: PareenTaula (1..N) :=(Others=> (False,False));
   Lehena: Erpina;

begin
   if Erroa = A1 then Ahaideak(Erroa).A1:= True;
   else if Erroa = A2 then Ahaideak(Erroa).A2:= True; end if;
   end if;

   SK(G, Erroa);
   ArbasoLehena:=Lehena;

end AhaideKomunLehena;
```

```
procedure  SK  (G: in Grafoa; UnekoErp: in Erpina) is
   AAuzokideenKopia: ErpinenL;
   Erp: Erpina;
begin
   AAuzokideen_Kopia:= Auzokideak(G, UnekoErp);
   while  not( HutsaDaL? (AAuzokideenKopia))
            and not Amaitua                          loop
      Erp:= LehenengoaL(AAuzokideenKopia);
      Hondarra(A_AuzokideenKopia);
      if Erp = A1 then  Ahaideak(Erp).A1:= True;
      else if Erroa = A2 then  Ahaideak(Erp).A2:= True; end if;
      end if;

      S_K(G, Erp);
      OrBaturaGurasoEtaUmearenArtean(UnekoErp,Erp);
      If (Ahaideak(E).A1) and (Ahaideak(E).A2))
      then Lehena:= UnekoErp; Amaitua:=True;
      end if;
   end loop;

end S_K;
```

```
OrBaturaGurasoEtaUmearenArtean(E1,E2):
Ahaideak(E1).A1 := (Ahaideak(E1).A1) or (Ahaideak(E2).A1);
Ahaideak(E1).A2 := (Ahaideak(E1).A2) or (Ahaideak(E2).A2);
```

Algoritmoaren denbora analisia

- Funtsean algoritmoak zuhaitza bat sakoneran korritzen du. Erpin eta arku bakoitzeko egiten den tratamendua denbora konstantean egingarria da honakoa izanik.
  - Erpin bakoitzeko:
    - Erpinaren prozesaketa hasi aurretik: bi aldagaien hasieratzea
    - Erpinaren umeen azterketa ondoren: BiakTrue?
  - Arku bakoitzeko:
    - Arkuko jomuga erpina ahaidetako bat bada, erregistratu jomuga erpinari
    - Arku horretatik sortutako adartzetik ahaiderik zintzilik badago, gurasoak ere badu (OrBatura)
- Hortaz, soluzioren denbora-ordena O(n+a) =O(n) da, n erpin kopurua izanik eta a arku kopurua (a=n-1 baita).

Topaketako afaria. Grafo bipartigarria?

Topaketa bateko partaideen arteko *elkar ezinikusi erlazioa* ezaguna denez, topaketako antolatzaileek azken eguneko gala afarian ahalik eta girorik egon dadin, Timanfaya eta Garajonay izeneko bi jangela dituen jatetxe bat topatu dute eta haietan banatu nahi dituzte partaideak.

Idatz eta aztertu ezazu algoritmo bat, elkar ezinikusi erlazioa errespetatuz, erabakiko duena posiblea den ala ez partaideak bi jangeletan banatzea, eta hala balitz, banaketa itzuliko lukeena. Beraz, edozein partaide jangela batean esleiturik dagoenean, honek ezinikusi dituenak beste jangelan egon beharko lukete

- Problema partiketa problema da:
  - bi multzo disjuntutan banagarriak dira jankideak?
  - Problema ebazteko zabalerako (nahiz sakonerako) korritzea erabil dezakegu

- Prozesua (zabalerako korritzea
  - Jankide bat abiapuntutzat aukeratzen da, eta adibidez A jangela esleitzen zaio
  - Abiapuntua (0 mailan dagoena) eta maila bikoitietan (ertz-distantzia bikoitietara) dauden guztiak ‘A’(edo 0) jantokira esleitugarriak izan beharko lukete.
  - Aldiz, maila bakoitietan eroriko liratekeena ‘B’ (edo 1) jangelara esleitugarriak izan beharko lukete.
  - Honela, jangela esleitua duen jankide bat berriz aztertu beharko bagenu (ziklorik dagoelako edo gurasoa delako edo jatorritik bide batek baino gehiago existitzen duelako)
    - problemarik ez legoke jangela berdina esleitu beharko bagenio.
    - aldiz, jangela desberdina tokatuko balitzaio, partiketarik ez luke existituko.

- Suposaketak eta erabakiak:
  - etsai erlazioa erreziprokoa (grafo ez-zuzendua) dela
  - gerta liteke grafoa ez-konexua izatea
  - abiapuntuak beti 0 jangelara esleituko ditugu

```
procedure  GrafoaKorritu (G: in Grafoa) is
   Aztertua: constant Boolean := True;
   Bisitak: Taula(1..N):= (Others=>  not (Aztertua));
   Jangela: array (1..N) of 0..1;
   EzDagoPartiketarik: Boolean:= False;
begin
   for  ErpinBat  in  1..N loop
        -- osagai konexu edo azpigrafo bakoitzak elkarren
        -- arteko ezinikusiak jasotzen ditu.
        -- Beharrezkoa da begizta
        if  not MarkatuaDago(Bisitak, ErpinBat)  then
              MarkaIpini(Bisitak, ErpinBat, Aztertua);
              Jangela(ErpinBat):= 0;
              ZK(G, ErpinBat,0);
        end if;
   end loop;
   if EzDagoPartiketarik then write ("ez dago partiketarik");
   else IdatziTaula(Jangela);
   end if;
end GrafoaKorritu;
```

```
procedure  ZK  (G: in Grafoa; X: in Erpina) is
Ilara: IlaraDatuMota:=HutsaI; YRenAuzokideak: ErpinenL; Y,Z:Erpina;
begin
   IlarariGehitu(Ilara, X);
   while not( HutsikDagoI?(Ilara)) and (not EzDagoPartiketarik) loop
      Y:= Burua(Ilara); Ilara:= IlaratikKendu(Ilara);
      YEnAuzokideak:= Auzokideak(Y);
      while  not ( HutsikDagoL?(YRenAuzokideak))  loop
        Z:= LehenaL(YRenAuzokideak);
        Hondarra(YRenAuzokideak);
        if  not MarkatuaDago?(Bisitak, Z)
        then    MarkaIpini(Bisitak, Z, Aztertua);
                Jangela(Z):= ( (1+Jangela(Y)) mod 2);
                IlarariGehitu(Ilara, Z);
        else    if Jangela(Z)=Jangela(Y) then
                 EzDagoPartiketarik:=True; exit;
         -- else markatuta dago eta esleitutako jangela egokia da!
         end if;
      end loop;
   end loop;
end ZK;
```

Algoritmoaren denbora analisia

- Zabalerako korritzeari denbora konstantea behar duten hainbat eragiketa esleitu zaio. Zehazki,
  - erpin edo jankide bakoitza behin markatu eta jangela batera esleituko da.
  - Ertz bakoitzeko bertako erpin bat dagoeneko marka duen ala ez begiratuko da.
  - Eragiketa hauek denak dira denbora konstantean egikarigarri eta gehitu diren eragiketa berriak.

  Ondorioz,
  - $\Theta(n+a)$ ordena du soluzioak baldin eta partiketak existitzen badu (ertz eta erpin guztiak behin tratatuko direlako);
  - aldiz, partiketarik ez balego, orduan $O(n+a)$ litzateke.

Bide kopuruak Erromaraino

Jakina da bide oro Erromara doala. Gida on bat erabiliz eta errepide mapa bat erabiliz, grafo zuzendu azikliko bat (DAG, ingeleratik, direted acyclic graph) marraztu dugu hamaika ibilbide on jasotzen dituena gure hiritik Erromara iristeko.

Emandako DAGa erabiliz, gure hiritik Erromara doazen ibilbide kopurua kalkulatuko duen algoritmoa diseinatzea eskatzen zaizu bai eta haren denbora-analisia kalkulatzea ere.

- Soluziorako argipenak
  - `BideKop(1..N)` bektorea erabilko dugu
    - `BideKop(H)=z` `H` grafoko erpinetik Erromaraino `z` bide dago
    - Bektorearen betetzeak atze prozesaketa behar du
      - **Erpin baten ume guztietatik dauden bide kopuruak ezagutzen direnean, gurasoaren kopurua finka liteke**

- Abiapuntu eta jomuga erpinak (`J` erpina):
  - Mapan abiapuntu bat baino gehiago egon liteke; hots, `J` abiabuntu erpina da baldin `InDegree(J)=0`.
    - Gure hiria mapako erpin izanik, ez du zertan abiapuntu erpina izan behar: gerta liteke `InDegree(GureHiria)≠0)`. Halere ,gure abiapuntua izango da.
  - Jomuga bat baino gehiago egon litezke; hots, `Outdegree(J)=0`. Erroma bat izan liteke edo ez, Erroma “barruko” erpin bat izan bailiteke.
  - Beraz, (GureHiria,Erroma) erpin bikote arteko bideen kopurua kalkulatu behar dugu baldintza horietan:
    - `BideKop(GureHiria)`
- Oharra: gure soluzioan `BideKop(K)` ez da kalkulatuko, gure hiria eta Erroma arteko bideen artean ez badago `K` erpina.

- Grafoa sakoneran korrituko da,
  - Markatzea BideKop taula bidez egingo da.
    - -1: oraindik azter gabea adieraziko du
    - ≥0: aztertua edo aztertzen hasia
  - gure hiritik egingo lehengo deia eta Erromatik aurrera ez da sakoneran korritzen jarraitu behar. Grafoa Korritu prozeduran honako hasieratzeak beharko dira:
    - `BideKop(HasiHiria)=0;` bide kontagailua hasieratzea
    - `BideKop(Erroma)=1;` Erromatik Erromara bide 1, k.n.
- `DaudenKop`, bi hiri horien arteko bideen kopurua jasoko du.
- Egin beharko den deia

```
GrafoaKorritu(Donostia, Erroma, DaudenKop).
```

```
procedure  GrafoaKorritu (G: in Grafoa;
                          HasiHiria: in Erpina    -- Gure hiria
                          Erroma: in Erpina       -- Erroma, jo muga
                          DaudenKop: out Integer) is

   BideKop: Taula(ErpinKopEsparrua):= (Others=>  -1);
                          -- = Bisitatugabea

begin

   BideKop(Erroma):=1;     -- Erromatik Erromara bide bat.
                           -- Kasu nabari bat oraingoan hemen
                           -- Erromatik irteten direnak ez kalkulatzeko
   BideKop(HasiHiria):=0;
   SK(G, HasiHiria);
   DaudenKop:= BideKop(HasiHiria);

end GrafoaKorritu
```

```
procedure  SK  (G: in Grafoa; A: in Erpina) is
   AAuzokideenKopia: ErpinenL;        Lehena: Erpina;
Begin

   AAuzokideen_Kopia:= Auzokideak(G, A);
   while  not( HutsaDaL? (AAuzokideenKopia))  loop
        Lehena:= LehenengoaL(AAuzokideenKopia);
        HondarraL (A_AuzokideenKopia);
        if BideKop(Lehena)=-1              -- Ez da aurretik kalkulatu
        then BideKop(Lehena):=0;           -- Markatu,hasieratu+balioa lortu
             S_K(G, Lehena);
        end if;
        BideKop(A):= BideKop(A)+BideKop(Lehena);
        -- BideKop(Lehena)=0 denean Lehena hosto da edo Erromara ez
        -- doazen bideetan dago.
        -- BideKop(Lehena)=1 denean Lehena=Erroma
   end loop;
end S_K;
```

Zatitu eta irabazi teknika

(Divide y vencerás - divide & conquer)

R. Arruabarrena
LSI - UPV/EHU

Sarrera

- Goitik beherako diseinua
  - Problema azpiproblematan banatu
  - Azpiproblema bakoitza errekurtsiboki ebatzi
  - Azpiproblemen emaitzak konbinatuz jatorrizko soluzioa eraiki
  - Kostua: 3 atalen kostuen batuketa
  - Adibideak:
    - Bilaketa dikotomikoa, MergeSort, QuickSort, k.hautaketa,…
  - Abantailak:
    - Diseinu teknika naturala
  - Desabantailak
    - Azpiproblemak independienteki ebazten dira, eta azpiproblema bera behin eta berriro ebaztea gerta liteke, alferrikako T(n) gastatuz

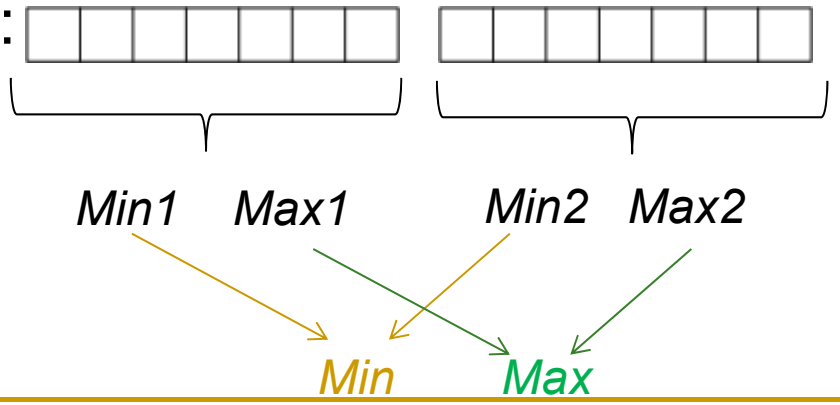
Sekuentziako balio Max eta Min

Bektore bateko balio handiena eta balio txikiena mugatzen dituen algoritmoa egizu. Soluzioak (3/2) n – 2 alderaketa gehienez eta zatitu eta irabazi teknika erabili behar du.

- Soluzio nabariak:
  - maximoa aurkitzeko: (n-1) alderaketa
  - Minimoa aurkitzeko: (n-1) alderaketa

  } *2n-2*

- Zatitu eta irabazi teknika bidez:



```
procedure MIN_MAX  ( B: in BEKTOREA; MIN, MAX: out INTEGER) is
   MIN1,MIN2,MAX1,MAX2: INTEGER;
begin
    if B'LENGTH=2 then
        if B(B'FIRST)> B(B'LAST)
         then MAX:= B(B'FIRST); MIN:= B(B'LAST);
        else MAX:= B(B'LAST); MIN:= B(B'FIRST);
        end if;

   esle MIN_MAX(B(B'FIRST..(B'FIRST+B'LAST)/2),  MIN1, MAX1);
        MIN_MAX(B((B'FIRST+B'LAST)/2)+1)..B'LAST),  MIN2, MAX2);

        if MAX1 > MAX2  then  MAX:= MAX1;
        else MAX:= MAX2;
        end if;

        if MIN1 < MIN2  then  MIN:= MIN1;
        else MIN:= MIN2;
        end if;
   end if;
```

MergeSort ordenazio algoritmoa

```
procedure  MERGESORT (S: in out OS_SEK)  is
   S_LAG: OS_SEK(S'RANGE);
begin
   if  S'LENGTH>1  then
       MERGESORT (S(S'FIRST..((S'FIRST+S'LAST)/2)-1));
                          1                (n/2)-1
       MERGESORT (S(S'FIRST+S'LAST)/2).. S'LAST));
                        (n/2)              n
       MERGE (S(S'FIRST..((S'FIRST+S'LAST)/2)-1),
                S(S'FIRST+S'LAST)/2).. S'LAST)),
                S_LAG);
       S:=S_LAG;
   end if;
end MERGESORT;
```

- Analisia:

  T(n)= #alderaketak

  - T(2)=1
  - T(n)= 2 T(n/2) + 2

  Hedapen metodoa erabiliz:

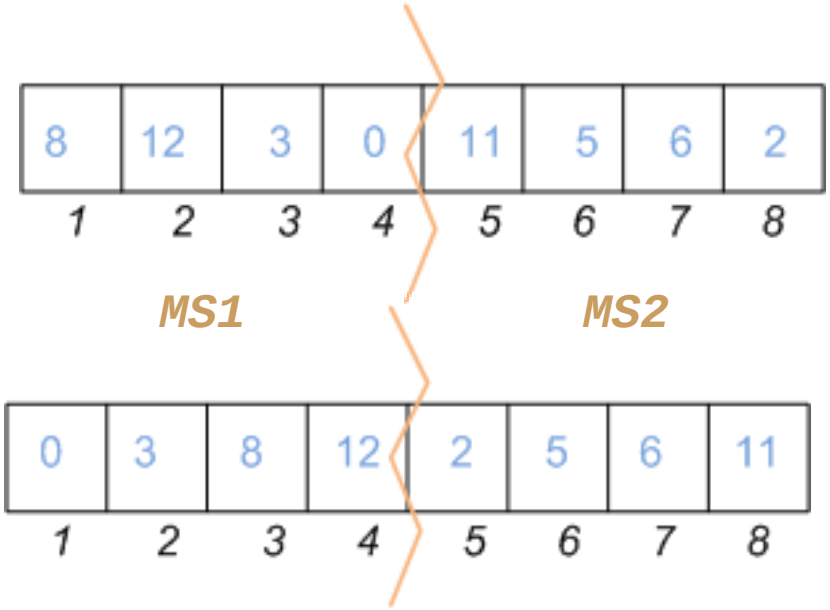
  $T(n) = 2\,T(n/2) + 2$

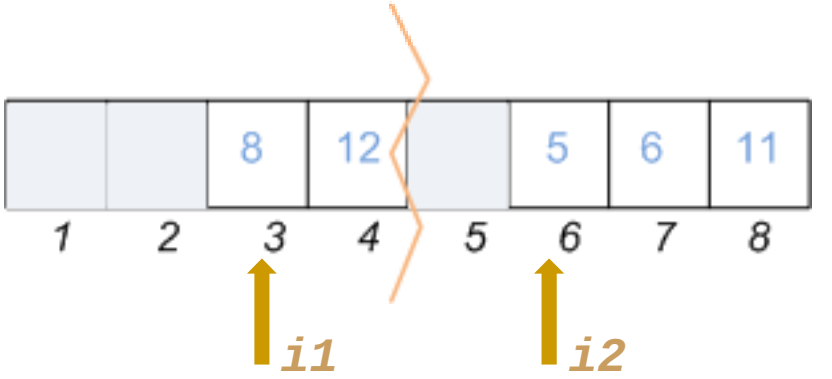
  $= 2\,(2T(n/2^2) + 2) + 2 = 2^2T(n/2^2) + 2^2 + 2$

  $= 2^iT(n/2^i) + 2^i + \ldots + 2$ i pausoan

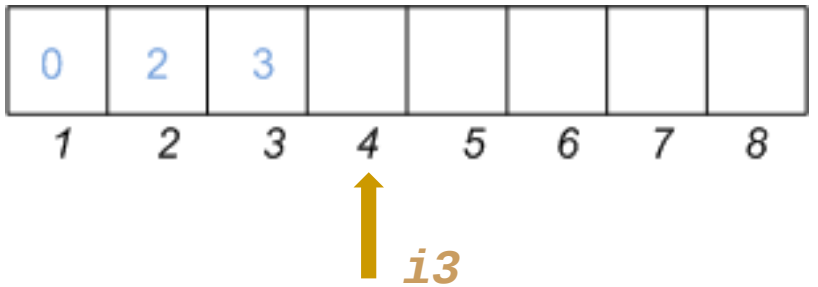
  $= 2^iT(2) + 2^{i+1} - 2$ $2 = n/2^i$; $n/2 = 2^i$

  $= 2^i\ 1 + 2\ 2^i - 2$ = **(3/2) n - 2**


8 12 3 0 11 5 6 2
1 2 3 4 5 6 7 8
MS1
MS2
0 3 8 12 2 5 6 11
1 2 3 4 5 6 7 8


8 12 5 6 11
1 2 3 4 5 6 7 8
i1
i2

MERGE


0 2 3
1 2 3 4 5 6 7 8
i3

```
procedure  MERGE  (L1,L2: in OS_SEK; L3: out OS_SEK) is

   Ind_L1: INDIZEA:= L1'FIRST; Ind_L2: INDIZEA:= L2'FIRST;
   Ind_L3: INDIZEA;

begin
   for  Ind_L3  in 1..L1'LENGTH+L2'LENGTH  loop
        if      (Ind_L1<=L1'LAST) and (Ind_L2<=L2'LAST)
        then    if      L1(Ind_L1)<L2(Ind_L2)
                then    L3(Ind_L3):=L1(Ind_L1); Ind_L1:= Ind_L1+1;
                else    L3(Ind_L3):=L2(Ind_L2); Ind_L2:= Ind_L2+1;
                end if;
        elsif (Ind_L1>L1'LAST)
        then    L3(Ind_L3):=L2(Ind_L2); Ind_L2:= Ind_L2+1;
        else    L3(Ind_L3):=L1(Ind_L1); Ind_L1:= Ind_L1+1;
        end if;
   end loop;
end MERGE;
```

- Analisia:

$T(1) = \Theta(1)$

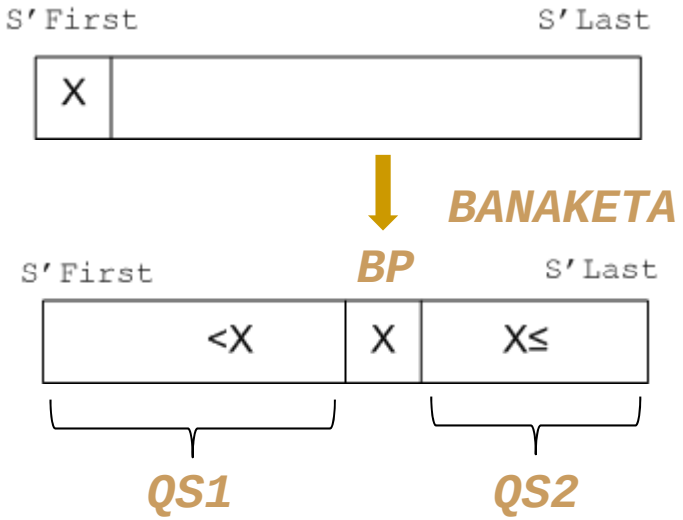
$T(n) = 2\,T(n/2) + T_{merge}(n) = 2T(n/2) + \Theta(n)$

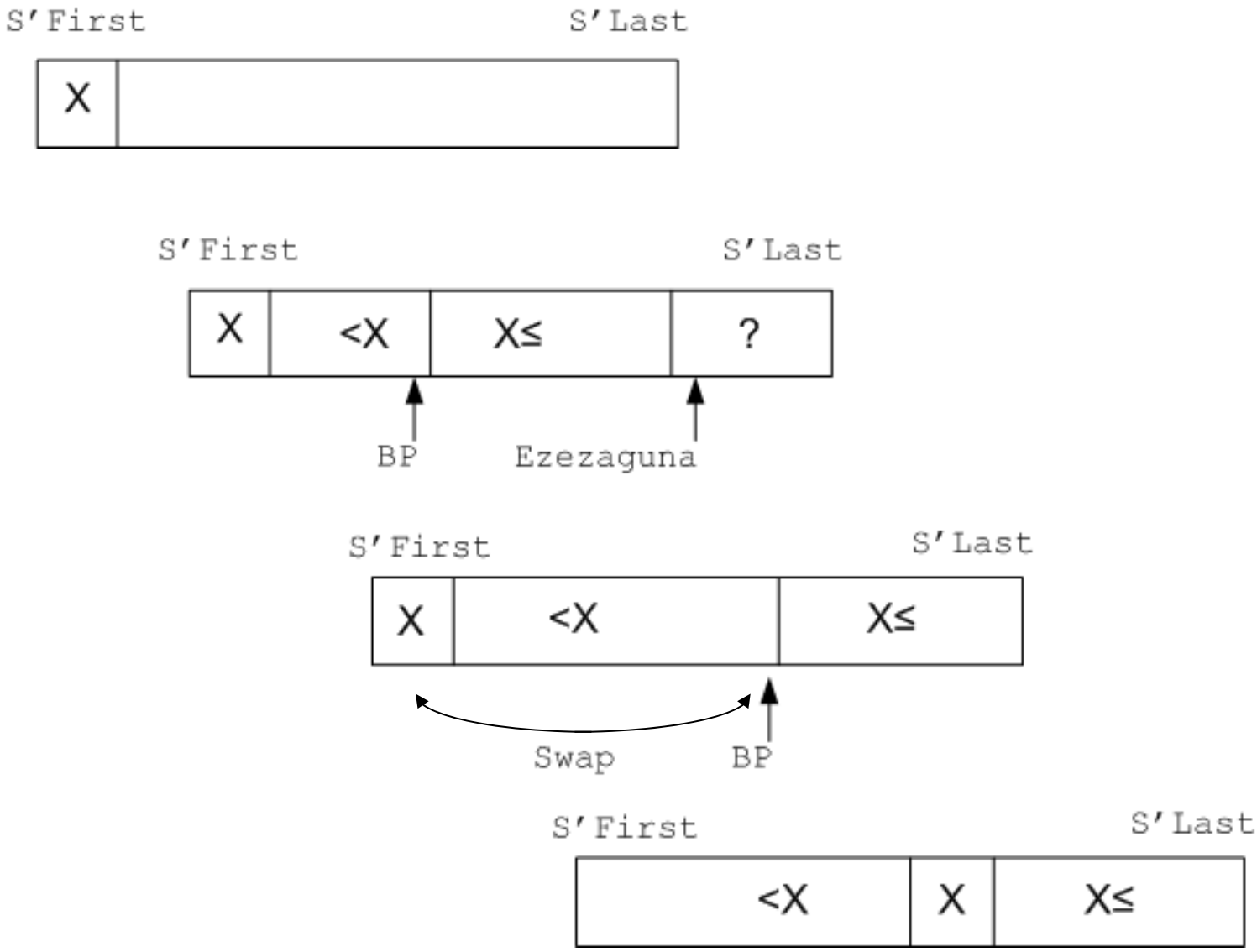
$$
\begin{aligned}
T(n) &= 2T(n/2)+n \\
&= 2(2T((n/2)/2)+n/2)+n = 2^2T(n/2^2)+2n \\
&= 2^2\,(2T((n/2^2)/2)+n/2^2)+2n = 2^3T(n/2^3)+3n \\
&\\
&= 2^i\,T(n/2^i)+i\,n \qquad i \text{ pausoan; } 1=n/2^i;\ n=2^i;\ \lg n=i \\
&= n\,T(1)+n\lg n = n + n\lg n
\end{aligned}
$$

$T(n) \in \Theta(n \lg n)$

QuickSort ordenazio algoritmoa

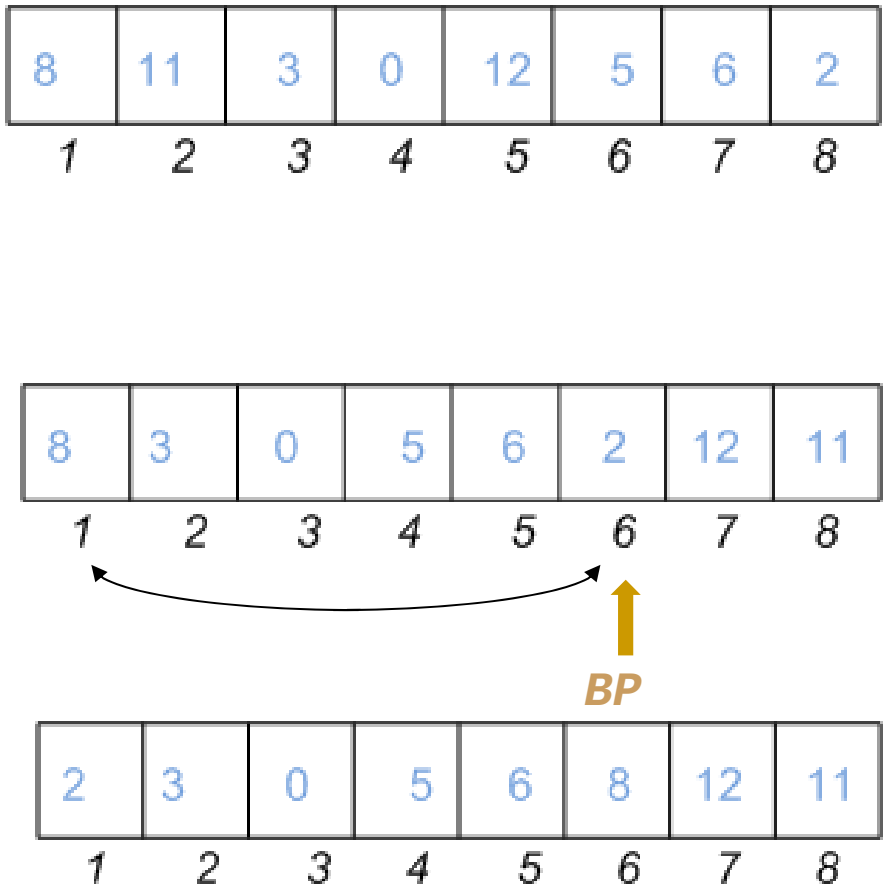
```
procedure QUICKSORT (S: in out OS_SEK) is
   BP:PUNTUA:INDIZEA;
begin
   if  S'LENGTH >1  then
       BANAKETA(S, BP);
       QUICKSORT (S(S'FIRST.. BP-1));
       QUICKSORT (S(BP+1.. S'LAST));
   end if;
end QUICKSORT;
```




S'First
S'Last
X
S'First
S'Last
X
<X
X≤
?
BP
Ezezaguna
S'First
S'Last
X
<X
X≤
Swap
BP
S'First
S'Last
<X
X
X≤

```
procedure BANAKETA (S: in out OSOEN_SEK; BP: out INDIZEA) is
   X: INTEGER; BP_LAG: INDIZEA;

begin
   X := S(S'FIRST); BP_LAG:= S'FIRST;

   for  EZEZAGUNA in  S'FIRST+1 .. S'LAST  loop
        if     S(EZEZAGUNA) < X
        then   BP_LAG:= BP_LAG+1;
               TRULEA(S(BP_LAG), S(EZEZAGUNA));
        end if;
   end loop;

   TRUKEA(S(S'FIRST), S(BP_LAG));
   BP := BP_LAG;

end BANAKETA;
```

Banaketa:


8 11 3 0 12 5 6 2
1 2 3 4 5 6 7 8
8 3 0 5 6 2 12 11
1 2 3 4 5 6 7 8
BP
2 3 0 5 6 8 12 11
1 2 3 4 5 6 7 8

- Analisia:
  - T(n)= #alderaketak
  - $T(n)= T_{banaketa}(n)+T_{qs1}(f(n))+T_{qs2}(n-f(n))$
  - Sentikorra da sarreraren antolaketarekiko
    - Kasu onena: banaketa puntua beti erdian:

      $$T(n)= 2T(n/2) + n \in \Theta(n \lg n)$$

    - Kasu txarrena: banaketa puntua beti ertz batean geratzen denean. Adibidez, ordenatua

      $$T(n)= T(n-1) + n \in \Theta(n^2)$$

    - Bataz-besteko kostua: kasu onena eta txarrenaren artekoa

- $T_b(1)=0$
- $T_b(n)= 1/n\ (\ n-1+T_{qs1}(0)+T_{qs2}(n-1)) +$ BP=1
  $1/n\ (\ n-1+T_{qs1}(1)+T_{qs2}(n-2)) +$ BP=2
  $1/n\ (\ n-1+T_{qs1}(2)+T_{qs2}(n-3)) +$ BP=3
  …
  $1/n\ (\ n-1+T_{qs1}(n-1)+T_{qs2}(0))$ BP=n

$$T_b(n) = (n-1) + \frac{1}{n} 2\sum_{i=2}^{n-1} T_b(i)$$

Batuketaren balioa indukzio bidez frogagarria: *Tb(n) ≤ k n ln n*

- n=1 denean: T(1)= 0 eta k 1 ln 1=0
- n>1 denean: indukzio hipotesi garatua $\forall i(1 \le i < n \rightarrow Tb(i) \le ki \ln i)$

$$T_b(n) \le (n-1) + \frac{2}{n} \sum_{i=2}^{n-1} ki \ln i$$

$$T_b(n) \le (n-1) + \frac{2k}{n} \int_2^n x \ln x dx = (n-1) + \frac{2k}{n} \left[\frac{x^2}{2} \ln x - \frac{x^2}{4} \right]_2^n$$

$$T_b(n) \le kn \ln n - 1 + \left(1 - k/2\right) n + 2k/n (1 - 2\ln 2)$$

$T_b(n) \in \Theta(n \lg n)$

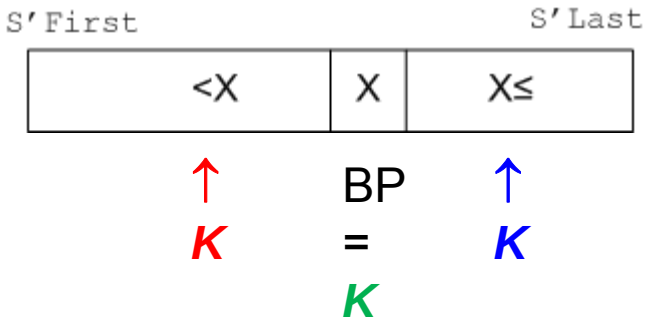
K.aren hautaketa

Osokoen taula bat emanik, bertako K osagai txikiena bilatuko duen algoritmoa idatzi eta ordena azter ezazu; hots:

m S bektoreko K. osagai txikiena da $\Leftrightarrow$

$$\#i(1 \leq i \leq n \rightarrow S(i) < m) < K \quad \text{eta} \quad \#i(1 \leq i \leq n \rightarrow S(i) \leq m) \geq K$$

- Kasu ezagunak
  - K=n/2 denean: medianaren problema; K=1: minimoa; K=n: maximoa
- Soluzio nabaria:
  - Bektorea ordenatu eta K posizioan dagoen itzuli. Kostua $\Theta(n \lg n + 1)$
- Zatitu eta irabazi teknika bidez, QuickSort-en oinarrituta
  - Bataz bestean: $T^K(n) \in O(n)$
  - Kasu txarrenean: $T^K(n) \in O(n^2)$




```
procedure KHautaketa (S:Bektorea; K: integer; BalK: Balioa)
--N=S'Last-Bek'First+1   1≤k≤N
begin
Banaketa(S,BP);
  if    BP=S'First+K-1          --BP=K
  then  BalK:= S(BP);

  elseif  BP>S'First+K-1        --BP>K
  then  KHautaketa(S(S'First..BP-1), K, BalK);

  else  -- BP<S'First+K-1       --BP<K
        KHautaketa(S(BP+1.. Bek'Last), K-(BP-S'First+1), BalK);
  end if;
end.
```

- Analisia, batazbesteko kasua:

$$T_b(n,K) = \mathbf{T^K(n)} = \text{\#alderaketak}$$

  - Oharrak:
    - BP non kokaturik geratzen den garrantzia du
      - ondorengo dei errekurtsibo mugatzen baitu
    - BP n posizio desberdinetan eror liteke.
      - Denak ekiprobableak
    - 1/n da posizio konkretu batean erortzeko probabilitatea

$$T^K(1) = 1$$

$$\begin{aligned}
T^K(n) = {} & 1/n\ (n + T^{K-1}(n-1)) + && BP=1 \wedge BP<K \\
& 1/n\ (n + T^{K-2}(n-2)) + && BP=2 \wedge BP<K \\
& \ldots \\
& 1/n\ (n + T^{K-(k-1)}(n-(K-1))) + && BP=K-1 \wedge BP<K \\
& 1/n\ n && BP=K \\
& 1/n\ (n + T^K((K+1)-1)) + && BP=K+1\ (\wedge\ BP>K) \\
& \ldots \\
& 1/n\ (n + T^K(n-1)) && BP=n\ \ (\wedge\ BP>K)
\end{aligned}$$

$$T^k(n) = n + \frac{1}{n}\left(\sum_{bp=1}^{k-1} T^{k-bp}(n-bp) + \sum_{bp=k+1}^{n} T^k(bp-1)\right)$$

- Batuketa garatzeko, goi bornatu egiten da honakoa espresioa erabiliz

$$R(n) = \max_{1 \le k \le n} \left\{ T^k(n) \right\} \qquad \forall n \;\; R(n) \le R(n+1)$$

- T(n) definizioan bornaketa espresioa ordezkatuz:

$$T^k(n) \le n + \frac{1}{n} \max_{1 \le k \le n} \left(\sum_{bp=1}^{k-1} R(n - bp) + \sum_{bp=k+1}^{n} R(bp - 1) \right)$$

$$= n + \frac{1}{n} \max_{1 \le k \le n} \left(\sum_{i=n-k+1}^{n-1} R(i) + \sum_{i=k}^{n-1} R(i) \right)$$

- R(n) goi-bornaketa lineala dela froga liteke indukzio bidez (ikus oinarrizko bibliografia):

$$R(n) = \max_{1 \le k \le n} \left\{ T^k(n) \right\} \le 4n$$

- $T^K(n)$ funtzioa R(n) bidez goi bornatu da, eta R(n) $\in$O(n); ondorioz,

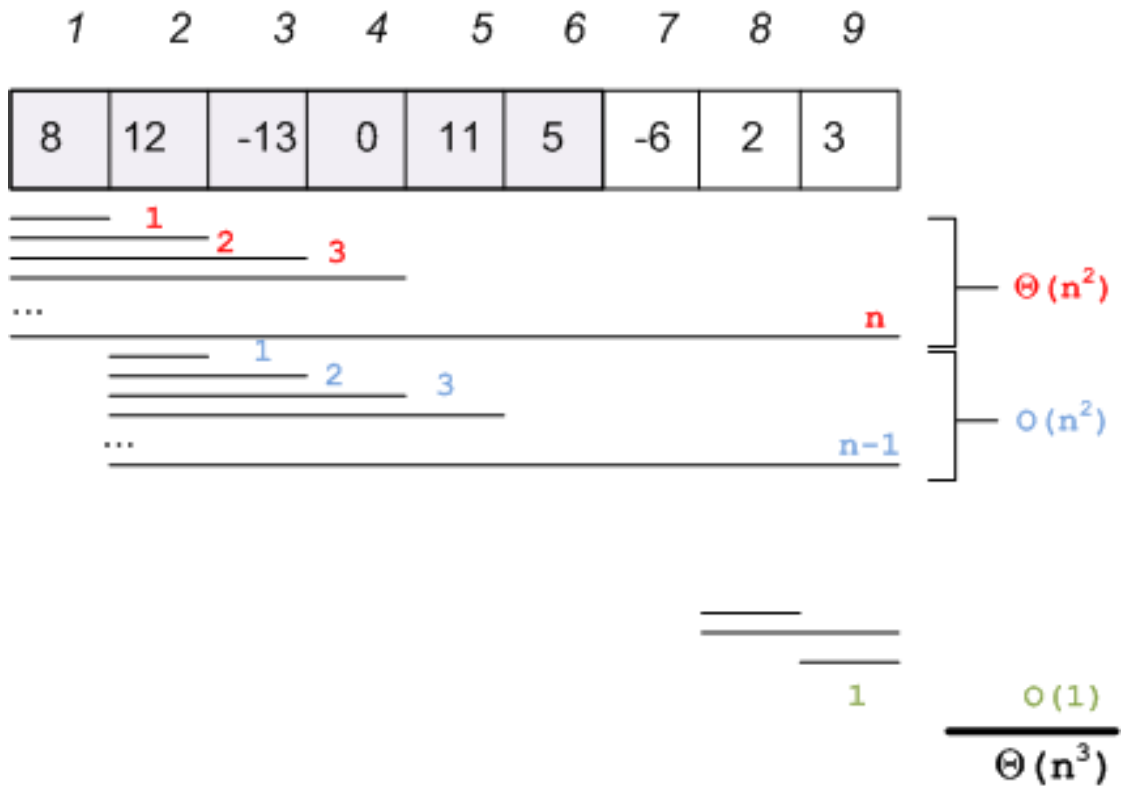
$T^K(n) \in O(n)$

Batura maximoko segmentua

Osokoen taula bat emanik, batura maximoa duen osagaien segmentua identifikatu behar da.

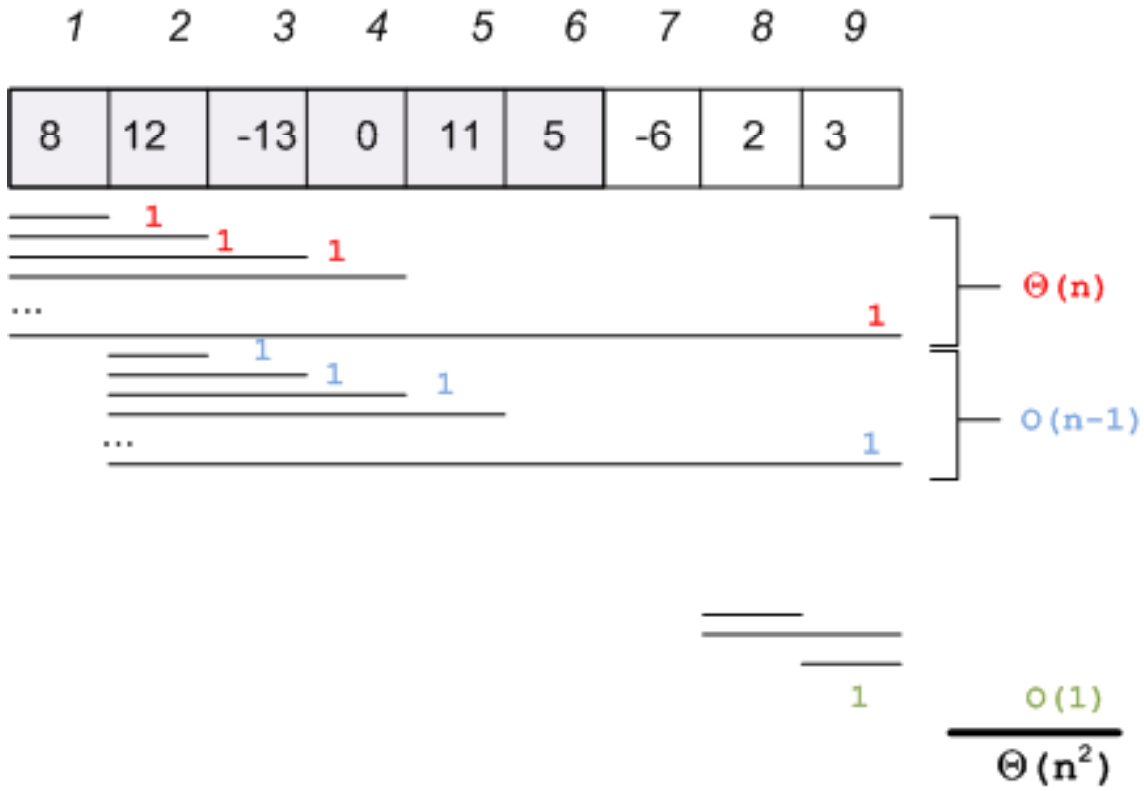
- Soluzioa nabaria:

  Kubikoa



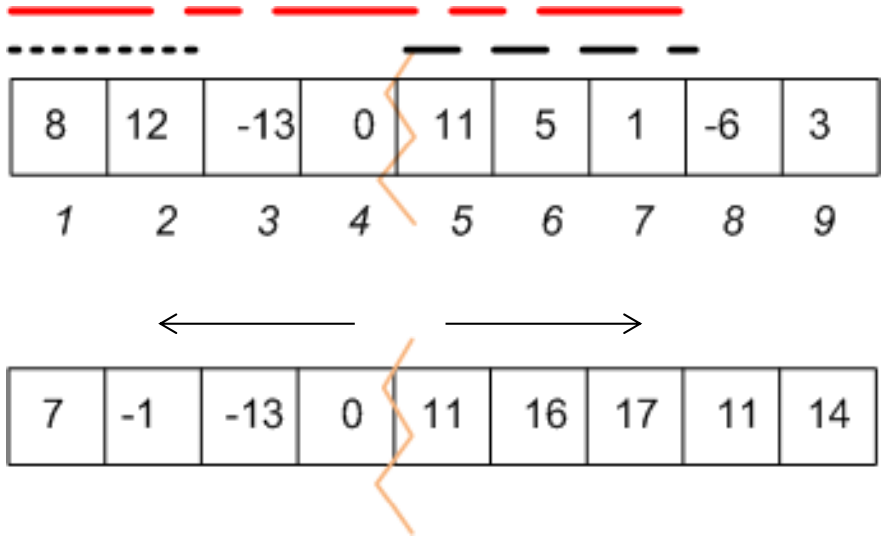
- Errazki hobe liteke:

soluzio koadratikoa



- Zatitu eta irabazi teknika bidez, magnitude hori hobetzeko, adibidez:
  - T(n)=2T(n/2)+O(n) kostuko kodea asmatu beharko genuke

- Algoritmoa:
  - 2 dei eta
  - aztertu ez diren sarrerak eskuz eraginkorki
    - Zehazki O(n)



$\in \Theta(n)$

- Eta aldagaia honela geldituko lirateke:
  - `B_Ez= 20` `Lehena_Ez=1` `Azkena_Ez=2`
  - `B_Erd_Ez+B_Erd_Es=7+17` `Posi_Erd_Ez=1` `Posi_Erd_Es=7`
  - `B_Es=17` `Lehena_Es=5` `Azkena_Es=7`

```
procedure   Batura_Max     (S: In Bektorea; B: Out Integer;
                             Lehena,Azkena: Out Indizea) Is
begin
   if  S'length=2
   then 3_artean_Max(S(S'first..S'first), S'first,S'first,
                     S(S'first..S'last), S'first,S'last,
                     S(S'last..S'last), S'last,S'last,
                     B,Lehena,Azkena);
   else Batura_Max   (S(S'first..(S'first+S'last)/2),
                     B_Ez,Lehena_Ez,Azkena_Ez);
        Batura_Max   (S(((S'first+S'last)/2)+1..S'last),
                     B_Es,Lehena_Es,Azkena_Es);
        Lag(S'first+S'last)/2):= S(S'first+S'last)/2);
        for I in reverse S'first..((S'first+S'last)/2)-1 loop
              Lag(I):= S(I)+Lag(I+1);
        end loop;
        Bektorean_Maximoa(Lag(S'first..(S'first+S'last)/2),
                          B_Erd_Ez, Posi_Erd_Ez);
```

```
        Lag((S'first+S'last)/2)+1):= S((S'first+S'last)/2)+1);
        for I in ((S'first+S'last)/2)+2..S'last  loop
                Lag(I):= Lag(I-1)+S(I);
        end loop;
        Bektorean_Maximoa(Lag((S'first+S'last)/2)+1..S'last),
                                B_Erd_Es, Posi_Erd_Es);

        3_Artean_Max(B_Ez,Lehena_Ez,Azkena_Ez,
                B_Erd_Ez+B_Erd_Es, Posi_Erd_Ez ,Posi_Erd_Es,
                B_Es,Lehena_Es,Azkena_Es,
                B,Lehena,Azkena);
end Batura_Max;
```

Programazio dinamikoa

R. Arruabarrena
LSI - UPV/EHU

Sarrera

- Goitik beherako diseinua naturala eta indartsua da
- Azpiproblema berdinen bidez soluzioa maiz lor liteke
- Ohiko inplementazioa: errekurtsioa

- errekurtsioaren murriztapenak
  - Azpiarazo bakoitza independenteki ebazten da
  - Azpiarazo bera, kalkulu bera, errepikatuz maiz
  - Alferrikako lana eta denbora

- Programazio dinamikoan
  - Azpiarazoen soluzioak metatu egiten dira
  - Aurrerago beharko balira, berrreskuratzeko

P. D.: Ezaugarriak

- Problema indukzio bidez definigarria izan behar du
- Azpiproblema desberdin bakoitza egitura batean posizio bat erlazionaturik izango du
- Azpiproblema bat ebazterakoan
  - 1.go aldian
    - azpiproblema indukzio bidezko ekuazioak dioen moduan kalkulatzen da
    - egituran hari dagokion posizioan metatzen da
  - Hurrengo alditan
    - Memoria egituran dagokion posiziotik berreskuratzen da balioa

P. D.: Ezaugarriak

- Egitura betetzeko ordenak

  Beti indukzio ekuazioak erabiliz

  - Iteratiboki gorantz: kasu nabarietatik hasiz
  - Errekurtsiboki beherantz ebatzitako kasuetaraino

- Optimizazio problemak ebazteko erabili ohi da

  - Bestelako problemak ebazteko ere balio dezake

  - **GAKOA**: **indukzio bidez definigarria izatea problema**

P. D. eta optimizazio problemen ezaugarriak

- Azpiegitura optimoa (Bellman-en optimotasun printzipioa)

> Problema bat ebazten duen
> erabakien sekuentzia optimo baten
> edozein erabakien azpisekuentzia batek,
> honek ebazten duen azpiproblemaren
> sekuentzia optimoa ere izan behar du.

Problema batek azpiegitura optimo bat duela esango dugu baldin eta problemaren soluzioak azpiproblemen soluzio optimoak barne baditu.

(≡ Indukzio bidez definigarria)

Fibonacci

1. Indukzio ekuazio ezagunak:
   - Fib(0)=1
   - Fib(1)=1
   - Fib(n)= Fib(n-1)+Fib(n-2)

Errekurtsio puruz

$$\Theta\left(\left(\frac{1+\sqrt{5}}{2}\right)^n\right)$$

P.D bidez:

2. Metatze egitura: TF(i)=x ≡ i. fibonacci zenbakia x da

TF

1	1			y	x	y+x	

Mee(n) ∈Θ (n)

3. Analisia: n. gelaxkako balio lortzeko (1..n-1) gelaxkak denbora konstantean beteko dira: **Θ(n)**

4. Iteratiboki: kasu nabarietatik interesatzen den kasu orokorra lortu arte, ekuazioek diotena inplementatuz

```
function Fib_PD (n: in Integer) is

   FT: array(1..n)of integer;

   i: integer=1;

begin
   FT(0)=1;
   FT(1)=1;

   while (i<=n) do
      FT(i):=FT(i-2)+FT(i-1);
      i=i+1;
   end loop;

   return FT(n);
end;
```

Tarteko emaitzak Metatzeko egitura

Kasu nabariak. 1 eta 2 ekuazioek markatzen duten balioak

Kasu orokor bakarra. 3 ekuazioak agintzen duen operazioak burutuz

Soluzioa

4. Memoriadun funtzioekin: soilik “Bete”-ri deia baldin kalkulatu gabe badago ekuazioak eskatzen duen azpiarazoaren balioa

```
function Fib_MemoriadunFun (n: in Integer) is

  FT:array(1..n):=(0,1=>1; others=> -1);

  proc  Bete (i: in Integer) is
  begin

    if (FT(i-2)=-1) then Bete(i-2); end if;
    if (FT(i-1)=-1) then Bete(i-1); end if;

    FT(i)= FT(i-2)+FT(i-1);
  end

begin
  if (FT(n)=-1) then Bete(n); end if;
  return FT(n);
End.
```

Azpiarazoa kalkulatu gabe dago adierazteko

Baldin azpiarazoa ez dagoen kalkulatua, kalkulatu

Soberan

Berreskuratu jada kalkulatutako (tarteko) balioak

Motxila 0/1

E edukiera duen motxilaren balioa maximizatu nahi dugu 1..n objektuak bertan osorik sartuz, jakinik p(i) eta b(i) i objektuaren pisua eta balioa direla h. h.. (non i, $1 \leq i \leq n$)

Zein da motxilaren irabazi maximoa?

Optimizazio problema: ekuazioak honekiko

Zeintzuk objektuk ematen dute irabazi hori?

Errekuperazio problema, aurrekoa osatu ostean

Garapeneko 4 pausoak:

Ekuazioak  Metatze egitura  Analisia+Kodea

Motxila 0/1: Ekuazioak.

IrabaziaM (ed, k): motxilaren edukiera “ed” izanik eta 1,2,…,k objektuak eskuragarri digula, lor litekeen irabazi maximoa objektu horiek motxilan osorik sartuz

Kasu nabariak:

(1e) IrabaziaM (ed,0)= 0

(2e) IrabaziaM (0,k)= 0

(3e) IrabaziaM (ed,k)=0 if ed<p(1) -- p(1)<....<p(N) baleude,
-- bestela arinena

Kasu orokorrak

IrabaziaM(ed,k)

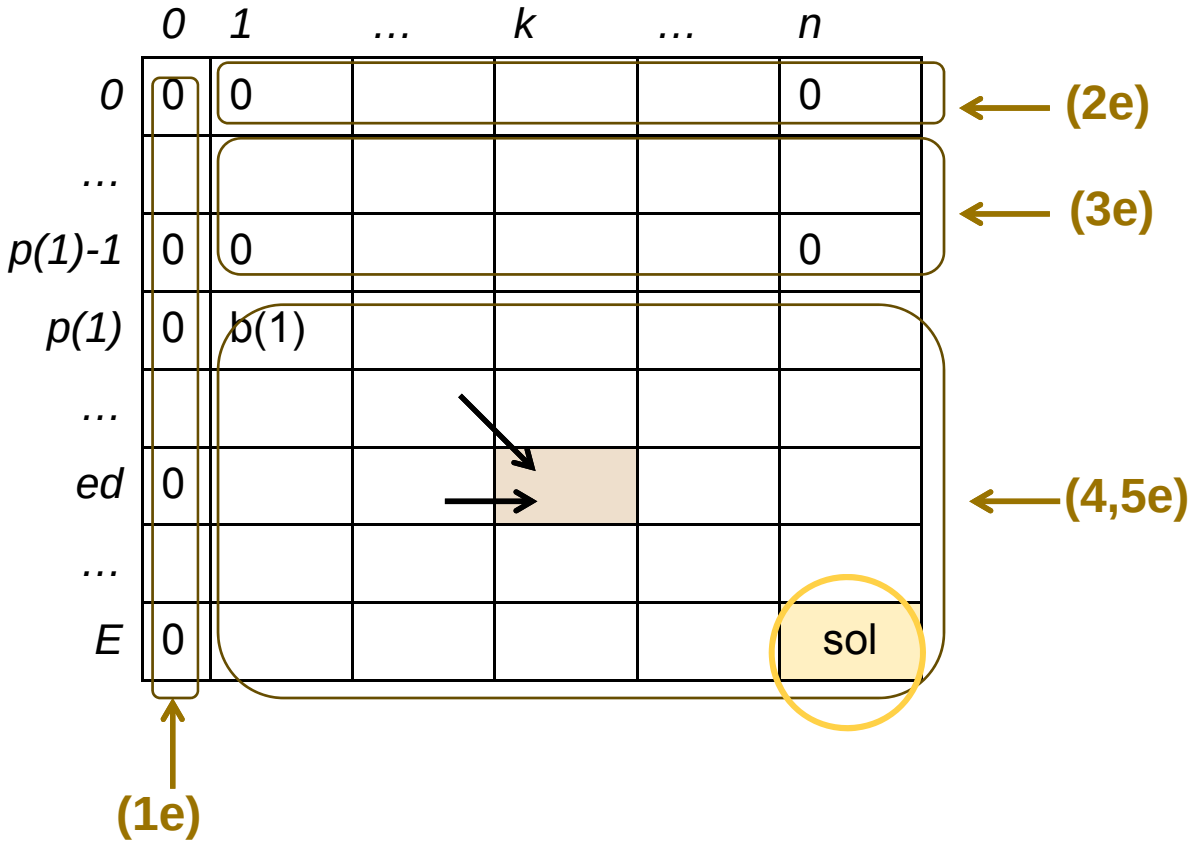
(4e) = $\underset{\text{Max}}{}$ { b(k) + IrabaziaM (ed-p(k), k-1),
IrabaziaM (ed-p(k), k-1) } if p(k)≤ed ∧ 1 ≤ K

(5e) = IrabaziaM (ed, k-1) if p(k)>ed ∧ 1 ≤ K

Motxila 0/1: Metatze egitura.

O(n x E)

Mee(n,E) ∈Θ (n x E)



- Betetze ordena erabaki:

Iteratiboki:

egitura betetze ordena (zutabeka , lerroka, diagonalka, ezker->eskuin)

Memoriadun funtzioekin:

balio lehenetsia eta **dei errekurtsiboa(k) identifikatu**

function Motxila01 (E, p(1..n),b(1..n))
TMotx(0..E,0..N)=(others= >(others => -1)) ***(kalkulatu gabea)***

```
   procedre Bete(Ed,K) is
   begin
     if TMotx(Ed,K-1)=-1 then Bete (Ed,K-1); end if;
     if p(K)<=Ed then
         if TMotx(Ed-p(K),K-1)=-1 then Bete (Ed-p(K),K-1); end if;
(4e)     TMotx(Ed,K):=MAX( b(K)+TMotx(Ed-p(K),K-1), TMotx(Ed,K-1));
(5e)  else TMotx(Ed,K):= TMotx(Ed,K-1); end if;
   end;
 begin
(1e) for Ed in 0..E loop TMotx(Ed,0):=0; end loop;
(2e) for K in 0..N loop TMotx(0,K):=0; end loop;
     for Ed in 1..p(1)-1 loop
(3e)      for K in 1..N loop TMotx(Ed,K):=0; end loop;
     end loop;
     if E>p(1) then Bete(E,N); end if;
     return TMotx(E,N);
 end;
```

```
Procedure Motxila01PD  (E, p(1..n),b(1..n))  Elem(1..N),  Irabazia)
     TMotx(0..E,0..N): float;     TObj(0..E,0..N) := (others=> (others => 0))        (Ez  sartu)
  begin
     for Ed in 0..E loop TMotx(Ed,0):=0; end loop;
(1e) for K in 0..N loop TMotx(0,K):=0; end loop;                                   Iteratiboa +
(2e) for Ed in 1..p(1)-1 loop                                                      errekuperazioa
          for K in 1..N loop TMotx(Ed,K):=0; end loop;
(3e) end loop;
   for K in 1..N loop
     for Ed in p(1)..E loop
(5e)     TMotx(Ed,K):= TMotx(Ed,K-1);
         if p(K)<=Ed and (Tmotx(Ed,K-1)< (b(K)+TMotx(Ed-p(K),K-1))
         then  TMotx(Ed,K):= b(K)+TMotx(Ed-p(K),K-1);
(4e)           Tobj(Ed,K):= 1;                                                     (Sartu)
          end if;
     Ed:=E;
     for K in 1..N loop  Elem(K):= TObj(Ed,K);
                         if (Elem(K)=1)then Ed:= Ed - p(K); end if;
     end loop;
     Irabazia:=TMotx(E,N);
  end;
```

E=10 eta 4 objektu eskuragarri izanik, zein da motxilaren irabazi maximoa? Zeintzuk objektu sartu behar ditugu? Eman zaizun ekuazio sistema erabiliz osatu ondorengo taula

	1	*2*	*3*	*4*
p(i)	4	2	6	3
b(i)	6	10	12	8

K\Ed	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
0											
1											
2											
3											
4											

E=10 eta 4 objektu eskuragarri izanik, zein da motxilaren irabazi maximoa? Zeintzuk objektu sartu behar ditugu? Eman zaizun ekuazio sistema erabiliz osatu ondorengo taula

	1	*2*	*3*	*4*
p(i)	4	2	6	3
b(i)	6	10	12	8

K\Ed	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	**0**	0	**6**	**6**	6	**6**	**6**	6	**6**
2	0	**0**	0	0	**10**	10	16	**16**	16	16	**16**
3	0	0	0	0	10	10	16	**16**	22	22	**22**
4	0	0	0	0	10	18	18	28	22	24	**24**

Txanponak. Konbinazio desberdin kopurua

$b_1,b_2,...,b_n$ N txanponen klaseak emanik eta jakinik txanpona klase bakoitzeko behar hainbat txapona eskuragarri ditugula,

L kopuru finkoa batzen duten txanpon konbinazio desberdinen kopurua kalkulatu.

- Konbi(L,i)=L kopurua itzultzeko [1..i] txanpon klaseak erabiliz dagoen txanpon konbinazio kopurua

```
Kasu nabariak:
  konbi(0,i)= ?  -- kopururik ez bada itzuli behar,

  konbi(L,1)= ?
```

Konbinazioak.

- Konbi(L,i)=L kopurua itzultzeko [1..i] txanpon klaseak erabiliz dagoen txanpon konbinazio kopurua

Kasu nabariak:

```
  konbi(0,i)= 1   -- kopururik ez bada itzuli behar, konbinazio
                  -- bakarra dago: txanponik ez erabiltzea

 konbi(L,1)= 1    if L=b1, L>0
           = 0    if L≠b1, L>0   ( Deskonposa ezina(L))
```

Kasu orokorrak:

(A aukera)

$$konbi(L,i) = konbi(L,i-1) + konbi(L-bi,i) \quad \text{if } 0<bi\leq L,\ 1<i$$

$$konbi(L,i) = konbi(L,i-1) \quad \text{if } 0<L<bi,\ 1<i$$

(B aukera)

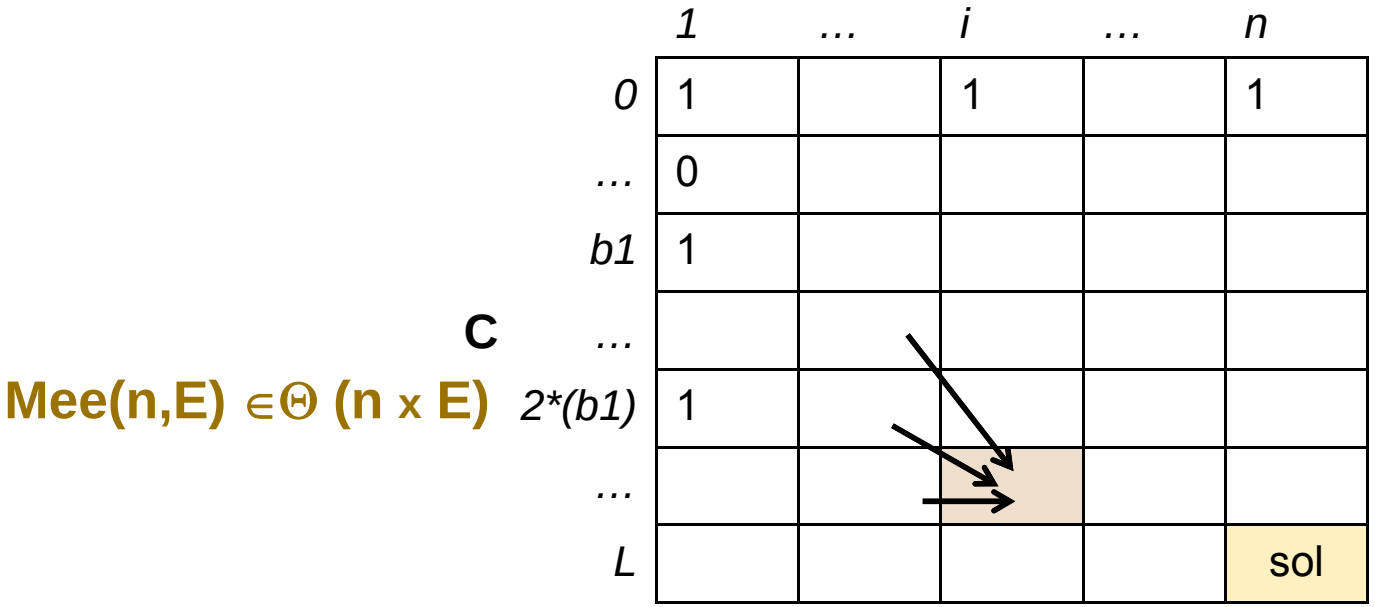
$$konbi(L,i) = konbi(L,i-1) + konbi(L-bi,i-1) + konbi(L-2bi,i-1) + \ldots + konbi(L-p*bi,i-1) \quad \text{if } 1<i,\ p = L \text{ div } bi$$

Konbinazioak.

- Konbi(L,i)=L kopurua itzultzeko [1..i] txanpon klaseak erabiliz dagoen txanpon konbinazio kopurua

```
Kasu nabariak:
  konbi(0,i)= 1   -- kopururik ez bada itzuli behar, konbinazio
                  -- bakarra dago: txanponik ez erabiltzea

 konbi(L,1)= 1    if L=b1, L>0
           = 0    if L≠b1, L>0   ( Deskonposa ezina(L))

Kasu orokorrak:
(A aukera)
   konbi(L,i)= konbi(L,i-1)+ konbi(L-bi,i)       if 0<bi≤L, 1<i
   konbi(L,i)= konbi(L,i-1)                      if 0<L<bi, 1<i

(B aukera)
   konbi(L,i)= konbi(L,i-1) + konbi(L-bi,i-1) + konbi(L-2bi,i-1)+…
               + konbi(L-p*bi,i-1)              if 1<i, p = L div bi
```

Konbinazioak. Metatze egitura eta kostua



Analisia

- i garren zutabea kalkulatzeko (i-1)garrena bakarrik erabiltzen denez, memoria espazio estra bi zutabeetara mugagarria da iteratiboki inplementatzen denean.
- Denbora ordena: Gelaxka garestiena x Gelaxka kopurua

(a aukera) **Θ (n x E)**

(b aukera) **O(n x E x m)** m= L div b1 b1<…<bn

```
procedure  KonbDestPD (L: in Kopurua; B: in Txanponak;  S: out
   Natural) is
   M: Matrizea (0..L,1..N);   -- Sup. B(1)<...<B(N)
begin
   for  i in  1.. N  loop  M(0,i) := 1; end loop;
   for  C in 1 .. L loop
     if (C mod B(1)) =0 then M(C, 1):= 1; else M(C, 1):= 0;  end if;
   end loop;

   for  i  in  2 .. N  loop      -- Kasu orokorra
        for  C  in  1 .. L  loop
        KonbKop:= M(C, i-1);
        ZenbatAldiz:= C div B(K);
        for  P  in 1.. ZenbatAldiz  loop
                 KonbKop := KonbKop + M(C- (P* B(K)), K-1);
        end loop;
        M(C,i) := KonbKop;
       end loop;
   end loop;
   S:= M(L,N);
end KonbDestPD;
```

Iteratiboa

(b aukera)

Diru itzulketa. Txanpon kopuru minimoa (TKM)

$b_1,b_2,...,b_n$ N txanponen klaseak emanik eta jakinik txanpona klase bakoitzeko behar hainbat txapona eskuragarri ditugula,

L kopuru finkoa emango duten txanpon kopuru minimoa kalkulatuko duen algoritmo bat idaztea eskatzen da.

- L deskonposa ezina izan liteke: ***kasu horretan 0 txanpona itzultzea erabakitzen bada, ez*** da desberdinduko deskonposa ezina eta 0 kopurua itzultzeko 0 txanpon behar direla

- Suposatuz: b1<b2<...<bn y $10^d <= L < 10^{d+1}$

TKM.

- (A ekuazio sistema)

  Konbi(L)=txanpon klase guztiak erabilgarri izanik, L kopurua itzultzeko behar diren txanpona kopuru minimoa

```
Kasu Nabariak:
TKM(0) = 0
TKM(bi) = 1          ∃i, 1≤i<n, L= bi
TKM(L) = ∞           1≤L<b1

Bestelako baldintzetan, kasu orokorra,:
TKM(L)= 1+ min {TKM(L-bi) |bi ≤ L, 1≤i≤n}
```

- Metatze egitura

  Egitura: T(1..L); T(j) = j kopurua itzultzeko txanpon kopuru txikiena.

- Denbora-ordena: **O(nL) = O(n 10^d);**

  gelaxka bakoitza betetzeko gehienez ***n*** txanpona klase erabiliko dira

TKM. (B ekuazio sistema)

Konbi(L,i)= [1..i] txanpon klaseak erabilgarri izanik, L kopurua itzultzeko behar den txanpona kopuru minimoa

```
TKM(L,i) = ∞          not(Deskonposagarri (L)) - barne 0≤L<b1
TKM(0,i) = 0
TKM(bj,i) = 1           if L=bj, 0≤j≤i
TKM(L,1) = L div b1     if (L mod b1)=0, L>0

TKM(L,i)=min{TKM(L,i-1), 1+TKM(L-bi,i)} bi≤L,Deskonposagarri(L-bi)
        = konbi(L,i-1)                  bi>L,Deskonposagarri(L-bi)
```

- Metatze egitura: M(1..n, 0..L)

  M(i,j)= j kopurua itzultzeko txanpon kopuru txikiena jakinik soilik 1..i klaseko txanponak erabil litezkeela.

- Denbora-ordena: **O(2nL) = O(n 10^d)** M(i,j) betetzeko $\in$ O(1)

Ariketa:

L=21 kopurua itzultzeko b={2, 5, 10} txanpona klaseak edukiz, erabaki taulak osatuz, TKM eta existitzen duten Konbinazio kopurua.

- Konbi (c aukera)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1																						
2																						
3																						

- TKM (b aukera)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1																						
2																						
3																						

Ariketa:

L=21 kopurua itzultzeko b={2, 5, 10} txanpona klaseak edukiz, erabaki taulak osatuz, TKM eta existitzen duten Konbinazio kopurua.

- Konbi (c aukera)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
2	1	0										1										2
3	1																					3

- TKM (b aukera)

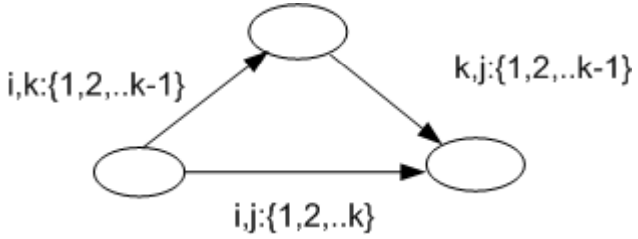
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	0	∞	1	∞	2	∞	3	∞	4	∞	5	∞	6	∞	7	∞	8	∞	9	∞	10	∞
2	0	∞					2					4					5					6
3	0	∞										4										5

Bide motzenak. Floyd.

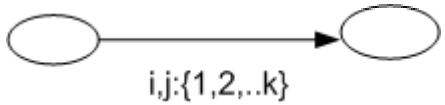
Izan bedi G grafo zuzendua eta pisuduna, non grafoko arku bakoitzak negatiboa ez den pisu bat erlazionaturik duen.

Grafoko erpin bikote bakoitzaren arteko distantzia minimoak mugatu.

- b=<e1,e2,...,ek> bideko tarteko erpinak:e2,...,ek-1
  i-tik, j-ra iristen eta {1,...,k} tarteko erpinak soilik erabiliz osa daitezkeen bide guztien artetik **b** distantzia motzena duena izan bedi.
  Grafoen teoriaren arabera, **b** bidean bi gauza gerta daitezke:



$k \in b$



$k \notin b$

Floyd

- Indukzio ekuazioak

$d_{i,j}^{k}$ i-tik hasi, j-ra iristen eta {1,...,k} tarteko erpinak soilik erabiliz osa daitezkeen bide guztien artetik distantzia motzena duena jasotzeko ekuazio sistema honakoa da :

$$d_{i,j}^{k} = \begin{cases} pisua(i,j) & if\ k = 0 \\ \min\left(d_{i,j}^{k-1}, d_{i,k}^{k-1} + d_{k,j}^{k-1}\right) & if\ k \geq 1 \end{cases}$$

Adibidez: $d_{i,j}^{0}$ i-tik j-ra zuzenean, $d_{i,j}^{1}$ i-tik j-ra zuzenean ala 1 erpinak zehakatuz,…

- Metatze egitura

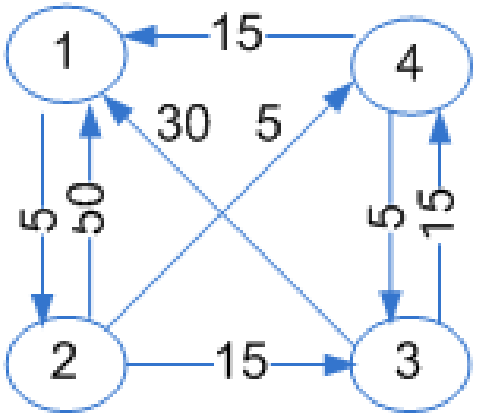
  Ez da metatze egitura estrarik behar

  k-ra hedatzerakoan matrizeko k lerroa eta k zutbeak ez dira eguneratzen.

Floyd.

```
procedure   FLOYD  (LUZERA: in MAT_ERREAL;
                          D: in out MAT_ERREAL) is
begin
   D:= LUZERA;
   for K  in  D'RANGE(1)  loop
        for I  in  D'RANGE(1)  loop
               for  J  in  D'RANGE(1)  loop
                      D(i,j) = min (D(i,j), D(i,k)+D(k,j));
               end loop;
        end loop;
   end loop;
end;
```

$\Theta(P^3)$



D^0=

	1	2	3	4
1	0	5	∞	∞
2	50	0	15	5
3	30	∞	0	15
4	15	∞	5	0

D^1=

	1	2	3	4
1				
2				
3				
4				

D^2=

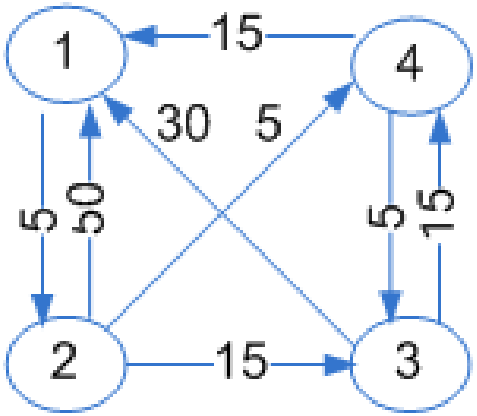
	1	2	3	4
1				
2				
3				
4				

D^3=

	1	2	3	4
1				
2				
3				
4				

D^4=

	1	2	3	4
1				
2				
3				
4				



$D^0=$

	1	2	3	4
1	0	5	∞	∞
2	50	0	15	5
3	30	∞	0	15
4	15	∞	5	0

$D^1=$

	1	2	3	4
1	0	5	∞	∞
2	50	0	15	5
3	30	**35**	0	15
4	15	**20**	5	0

$D^2=$

	1	2	3	4
1	0	5	**20**	**10**
2	50	0	15	5
3	30	35	0	15
4	15	20	5	0

$D^3=$

	1	2	3	4
1	0	5	20	10
2	**45**	0	15	5
3	30	35	0	15
4	15	20	5	0

$D^4=$

	1	2	3	4
1	0	5	20	10
2	20	0	**10**	5
3	30	35	0	15
4	15	20	5	0

Nilo.

Nilo ibaian N kai daude, bakoitzean Y ontzi aloka daitezkeelarik ibaian behera dauden beste kaietara joateko (ibaian gora ezin da joan, korronte handia du eta ibaiak). Edozein A abiapuntu-kaitik ibaian behera dagoen edozein H kaira iristeko ontzi bakar bat alokatzea garestiagoa gerta liteke Atik A+1ra, A+1etik A+6ra eta A+ 6tik H-ra joatea baino, adibidez.

A abiapuntu-kai guztietatik H helmuga-kai guztietara joateko bidai kostu posible guztien artetik bidai kostu minimoak kalkulatzea eskatzen da, bai funtzio memoriadunak erabiliz, bai hauek erabili gabe, baina bi kasuetan programazio dinamikoa teknika erabiliz. Ondoren, exekuzio-denboraren eta memoria-espazio estraren ordenak kalkula bitez.

- Datuak

Zuzenean (i,j)= i kaitik ibaian beheran dagoen j kaira zuzenean iristeko ontzi baten alogera kostua.

```
Zuzenean (i,j)=?        j<i (j ibaian goran dagoenean)
Zuzenean (i,i)= 0       kaitik bertara joateko alogera kostua 0
```

Nilo. Ekuazio sistema

Alogera(i,j)= i kaitik j ibaian beheran dagoen kaira iristeko ontzien alogeraren kostu minimoa (zuzenean ala ez)

```
Alogera(i,j)
   = 0                                  i=j
   = Zuzenean(i,j)                      i+1=j
   =min i<k<j {Zuzenean (i,j), Alogera(i,k)+Alogera(k,j)}      i+1<j   (A)

   =min i<k<j {Zuzenean (i,j), Zuzenean (i,k)+Alogera(k,j)}    i+1<j   (B)
```

- Metatze egitura: Alogera(1..n, 1..n) eta emaitza da
- Analisia
  - *MEE*: Ez du espazio estrarik erabiltzen, “Alogerak” matrizea itzuli behar baita osorik.
  - *Denbora ordena*: ($n^2/2$) gelaxka bete behar dira. Denbora gehien kalkulatzea kostatzen den gelaxka Alogerak(1,N) da, Q(n) izanik bere denbora ordena. Ondorioz, **$O(n^* (n^2/2))=O(n^3)$.**

Iteratiboki: (B) beheko lerroetatik goiko lerroetara; lerro barruan, ezkerretik eskuinera

```
procedure Nilo (Zuzen: in Matrizea; Alogerak: in out Matrizea) is
begin
for Ontzi in 1..n-1 loop                    -- kasu nabariak
   Alogerak(Ontzi,Ontzi):=0;
   Alogerak(Ontzi,Ontzi+1):= Zuzen(Ontzi,Ontzi+1);
end loop;
Alogerak(n,n):=0;

for Irten in reverse 1..n-2 loop            -- kasu orokorrak
   for Iritsi in Irten+2..n loop
      Merkeena:= Zuzen(Irten,Iritsi);
      for K in Irten+1..Iritsi-1 loop
          if Zuzena(Irten,K)+Alogerak(K,Iritsi)<Merkeena
          then Merkeena:= Zuzena(Irten,K)+Alogerak(K,Iritsi);
          end if;
          Alogerak(Irten,Iritsi):=Merkeena;
      end loop;
   end loop;
end.
```

(18,20)
(17,19),(17,20)
(16,18)(16,19)(16,20)
…
(1,3),(1,4),(1,5),…,(1,20)

Errekurtsiboki: (B ekuazio sistema)

```
procedure Nilo (Zuzen: in Matrizea; Alogerak: in out Matrizea) is
...
   prodedure  Bete (I,J: in Indizea) is
   begin
      Merkeena:= Zuzen(I,J);
      for K in I+1..J-1 loop
        if  Alogerak(K,J)=-1 then Bete(K,J); end if;
        if Zuzen(I,K)+Alogerak(K,J)<Merkeena
        then  Merkeena:= Zuzen(I,K)+Alogerak(K,J);
        end if;
      Alogerak(I,J):=Merkeena;
   end;

begin
```

```
begin
   -- balio lehenetsia, matrize osoa, sinpleagoa baita
   for Irten in 1..n loop
        for Iritsi in 1..n loop
               Alogerak(Irten,Iritsi):=-1;
        end loop;
   end loop;

   -- kasu nabariak
   for Ontzi in 1..n-1 loop
        Alogerak(Ontzi,Ontzi):=0;
        Alogerak(Ontzi,Ontzi+1):= Zuzen(Ontzi,Ontzi+1);
   end loop;
   Alogerak(n,n):=0;

   -- kasu orokorrak
   for Iritsi in 3..n loop              ←KONTUZ: Denak behar dira
        Bete(1,Iritsi);                 -- in edo in reverse
   end loop;
end.
```

Unibertsitateko irakasleen kontratazioa.

Unibertsitate bateko sail batek IK ikastordu eman behar ditu.

Horretarako, N klaseko desberdinetako irakasleak kontrata ditzake. I klaseko irakasle bakoitzak Hi ordu irakatsi ditzake gehienez lan horregatik beti Pi euro eskuratuz. IK klaseak emateko eta ordaindu beharreko euro kopurua minimoa izan dadin, zenbat irakasle kontratatu beharko liratekeen klase bakoitzeko kalkulatuko duen algoritmo bat idatzi eta soluzioaren denbora ordena eta memoria espazio estra kalkulatu.

Ekuazioak:

Ordaindu(k,j)= K ordu irakasteko eta [1..j] klaseetako irakasleak kontratagarriak izanik, unibertsitateak gutxienez ordaindu beharko lukeena.

1 aukera

Ordaindu(k,j)= K ordu irakasteko eta [1..j] klaseetako irakasleak kontratagarriak izanik, unibertsitateak gutxienez ordaindu beharko lukeena.

Kasu nabariak:

Ordaindu(0,j)=0 *-- Ez da klaserik eman behar*

K>0 kasua:

Ordaindu(K,1) = P(1) * (K div H(1)) if K mod H(1)=0

-- 1 klaseko irakasle guztiek H1 ordu irakatsiko dute

= P(1) * ((K div H(1))+1) if K mod H(1) ≠0

-- 1 klaseko azkeneko irakasle kontratatuak ez

-- ditu Ord(1) ordu irakatsiko, gutxiago baizik

Kasu orokorra: K>0, J>1

Ordaindu(K,j)

$= \min$ {Ordaindu (K,j-1),

P(j)+Ordaindu (K-H(j),j-1),

2*P(j) + Ordaindu (K-2*H(j),J-1),...,

d*P(j) +Ordaindu (K-d*H(j),J-1),

(d+1)* P(j)} d=K div H(j) eta (d+1) kasua soilik 0≠K mod H(j) bada

2 aukera

Ordaindu(k)= K ordu irakasteko eta n klaseetako irakasleak kontratagarriak izanik, unibertsitateak gutxienez ordain dezakeena.

$$\text{Ordaindu}(K) = 0 \qquad \text{if } K \leq 0$$

$$= \min_{1 \leq j \leq n} \{P(j) + \text{Ordaindu}(K - H(j))\} \qquad \text{if } 0 < K$$

Azoka.

K euro sakelan izanik azokara goaz. Honekin batera eros ditzakegun **m** produktuen zerrenda daramagu. *i* produktu bakoitzeko ($1 \leq i \leq n$) p_i prezioa eta e_i erabilgarritasun balioa (biak osoko positiboak) ezagunak ditugu. Produktu bakoitzeko gehienez 3 produktu eros ditzakegu. Gainera, eguneko eskaintzari esker, produktu beraren bigarren unitatearen salneurria 1 euro gutxiago kostatuko zaigu eta hirugarren unitatea, aldiz, 2 euro gutxiago. Erosketaren erabilgarritasuna erositako produktu bakoitzaren erabilgarritasunaren batura dela jakinik, *Programazio Dinamikoaren* teknika aplikatuz algoritmo bat idatz ezazu gehienez K euro azokako produktuetan gastatuz lor dezakegun erabilgarritasun maximoa mugatzeko bai eta erosi behar den **elementuen zerrenda** ere. Proposatutako soluzioaren denbora-ordena eta erabilitako memoria espazio estra kalkula ezazu.

Ekuazioak:

- MerkaAzoka (E,i) = lor dezakegun erabilgarritasun maximoa E euro gastatuz soilik [1...i] produktuetan (eta gehienez produktu bakoitzeko 3 unitate erosiz).

MerkaAzoka (E,i) = lor dezakegun erabilgarritasun maximoa E euro gastatuz soilik [1...i] produktuetan (eta gehienez produktu bakoitzeko 3 unitate erosiz).

$$\text{MerkaAzoka}(0,i) = 0$$

$$\text{MerkaAzoka}(d,1) = \begin{cases} 0 & \text{if } d < p_1 \\ e_1 & \text{if } p_1 \le d < 2p_1 \\ 2e_1 & \text{if } 2p_1 \le d < 3p_1 \\ 3e_1 & \text{if } 3p_1 \le d \end{cases}$$

$$\text{MerkaAzoka}(d,i) = \begin{cases} \text{MerkaAzoka}(d,i-1) & \text{if } d < p_1 \\ \max\{\text{MerkaAzoka}(d,i-1),\ e_1 + \text{MerkaAzoka}(d-p_i, i-1)\} & \text{if } p_i \le d < 2p_i \\ \max\{\text{MerkaAzoka}(d,i-1),\ e_1 + \text{MerkaAzoka}(d-p_i, i-1),\ 2e_i + \text{MerkaAzoka}(d-2p_i-1, i-1)\} & \text{if } 2p_i \le d < 3p_i \\ \max\{\text{MerkaAzoka}(d,i-1),\ e_1 + \text{MerkaAzoka}(d-p_i, i-1),\ 2e_i + \text{MerkaAzoka}(d-2p_i-1, i-1),\ 3e_i + \text{MerkaAzoka}(d-3p_i-2, i-1)\} & \text{if } 3p_i < d \end{cases}$$

Algoritmo jaleak
(Alg. Voraces - greedy algorithms)

R. Arruabarrena
LSI - UPV/EHU

Sarrera

- eguneroko bizitzan optimizazio problemak nonahi
- maiz enuntziatu sinpleak konputagailua bidezko ebazpenean denbora asko:
  - Backtracking
- praktikan aplikagarriagoak diren bi teknika:
  - hurbilketa bidezko soluzioak — Alg. Jaleak
  - soluzio optimoak — P. Dinamikoa

PD vs. Jalea

- P.D.:
  - soluzio optimoak kalkulatzen ditu
  - BAINA memoria memoria espazio estra + denbora ordena da

- JALEA:
  - sinplea, azkarra, espazio gutxi behar
  - BAINA ez du beti balio

Soluzio Jalea = Jale ZUZENA

- Soluzio jale zuzena: problemaren sarrera guztientzat soluziorik onena mugatzen duenean
  - Froga behar du izan
  - Ekintza hautaketa, Kruskal, Prim eta Dijsktra algoritmo jale zuzenak dira, euren frogak irakasgaiaren oinarrizko liburuan
- Sol. optimoaren hurbilketa jalea: Froga ez dugunean baina sarrera ia guztientzat soluzio onena kalkulatzen duenean.
  - kontrako adibideak topatzea zaila izan behar du
  - Honekin aski bada, konformatu behar. Bestela, P.D. Backtrack edo bibliografira jo

Alg. Jaleen prozesua

Soluzioa lortu arte errepika (edo hautagaiak agortu artean):

- une bakoitzean **aukera onena** egin
  - gelditzen diren hautagaietatik irabazi-etekin gehien ematen duena
- Aukeratutakoa hautagaia BEHIN tratatu:
  - onargarria bada soluzio partzialari gehitu
  - bestela baztertu

Soluzio jalearen helburua

- Optimizazio lokalak eginaz optimizazio globala lortzea
  - Lokala: gelditzen direnen artean aukera onena une bakoitzean. **Mokadu onena**
  - Globala: soluzioa zuzena izatea – optimoa
- Gure helburua:
  - Adibide zuzen batzuk ikasi eta euren adaptazioa beste problema batzuei
    - Bibliografian zuzen gehiago
  - Interesgarria denean soilik hurbilpen jaleak direnak egiten ikasi

Eskema jalea

```
function JALEA (H:Hautagai-multzoa)
                  return Hautagai-multzoa is
   SP: Hautagaien-multzoa;
begin

   MultzoHutsa(SP);
   while  not(SoluzioaDa?(SP)) and
             not(MultzoHutsaDa?(H))   do
      HautesleProzedura(H,x);          -- x hautatu jalea
      if  Osogarria?(Erantsi(SP,x))
      then  SP:= Erantsi (SP,x); end if;
   end loop;

   if  SoluzioaDa?(SP)   then return(SP)
   else MultzoHutsa(SP); return (SP); end if;
end JALEA;
```

Alg. Jaleen osagai komunak

1. hautagaien multzoa
   - Adib.: prozesa ditzakegun atazak, grafoko erpinak, arkuak edo ertzak, ...

2. SP: Soluzio Partziala
   - soluzio partziala jadanik aukeratu eta onartu diren hautagaien multzoa
   - Aurrekoaren azpimultzoa da

3. SoluzioaDa? hautagaien azpimultzo bat gure arazoaren soluzioa den ala ez erabakitzen duen funtzioa
   - nahiz eta soluzio hoberena ez izan

Alg. Jaleen osagai komunak

4. HautesleProzedura*:* oraindik aukeratu ez diren hautagaietatik soluzio onenaren bideratzailea den hautagaia aukeratuko duen prozedura

5. Helburu-funtzioa: soluzio bati dagokion balioa edo kostua itzultzen duena
   - Maximizatze / minimizatze funtzioa

6. Osogarria f.: aukeratutako azken hautagaia tarteko emaitzari erantsi garria den ala ez erabakitzen duena

Ekintza hautaketa ≡ baliabide konpartitua

Baliabide bera eskuratzearren lehiatzen duten eskaeren arteko azpimultzo bat identifikatu nahi da, non multzoaren kardinalitatea maximoa izanik, bertako eskaerak baliabidea esklusioan erabiltzen duten

- **Datuak**: baliabide bera eskuratzearren lehiatzen duten ekintza multzoa
  - N_EK = {1,...,N}.
  - *i* ekintza: *hi* hasiera-denbora; *bi* bukaera-denbora, *hi*≤*bi* betez
- **Problema**: baliabide baten erabilpen esklusiboa ekintzen artean banatu
- **Helburua**: zerbitzua lortzen duten eskaera kopurua maximatzea
- **Soluzioa**: denboran teilakatzen ez diren sarrerako datuen kardinalitate handieneko azpimultzo bat

Ekintza hautaketa. Prozedura hautesleak

a) Gutxien irauten duen ekintza aukeratu

b) Lehenen hasten dena

c) Denbora ardatzean gutxien teilakatzen dena

d) Lehenen amaitzen dena

f) Beranduen amaitzen dena

g) Beranduen hasten dena

…

Ekintza hautaketa. Prozedura hautesleak

a) Gutxien irauten duen ekintza aukeratu

Esperantza: Laburrena bada, gehiago sartuko dira eta horrela kardinalitate handiagoa.

Gezurra: B AbCa c

Jakinik :B ekintzaren hasiera denbora; b:ekintzaren amaiera denbora

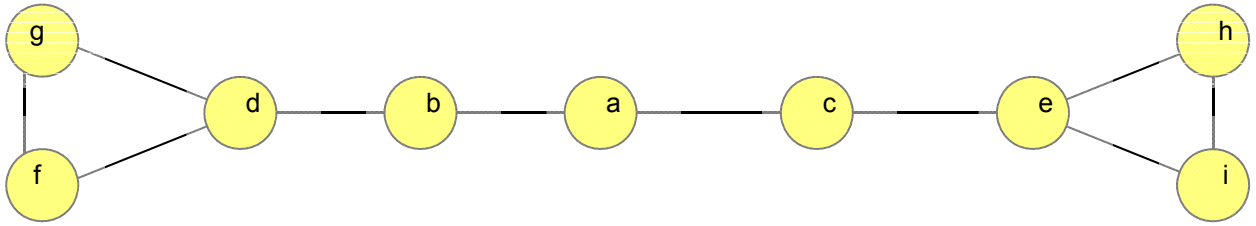
b) Lehenen hasten dena

Esperantza: Lehenago hasten bada, beste bat etorri aurretik dagoeneko martxan egongo da eta dagoeneko egiten/aprobetxatzen.

Gezurra: : A BbCc Dda

c) Denbora ardatzean gutxien teilakatzen dena

Esperantza: gutxirekin teilakatzen bada, hura soluzioan egoteak eragin txikia izango du (soilik teilakatzen diren haiengan), honela besteak (gehienak) sartzeko aukera gehiago dute



f) Beranduen amaitzen dena

Esperantza: Berandu bukatzen bada, seguruenik berandu hasiko dela, ez?

Gezurra: : A BbCc Dda

- P. hautesle onak soilik dira:

  d) Lehenen amaitzen dena

  g) Beranduen hasten dena

  Oharra: zuzentasun froga liburuan

- **Osogarria?** Baldin eta aukera berria orain arte aukeratutakoekin teilakatzen ez bada
  - J azkenekoz onartutako eskaera denean, eta I tratatzen ari garena:

$$\texttt{BukaeraT(J)} \leq \texttt{HasieraT(I)}$$

- **SoluzioaDa?** Ekoiztutako azpimultzoa beti izango da soluzioa:
  - teilakapenik ez du, eta
  - kardinalitate maximoa

Ekintza hautaketa. Algoritmoa (d prozedura hauteslea)

```
function EkintzaHauteslea (NEk : in EkintzenL)
                          return EkintzenL is
   NEkL: EkintzenL; I,J: Ekintza;
   SP: EkintzenL:= Hasieratu;
begin
   BukaeraTordenaGorakorraJarraituz(NEk, NEkL);
   LehenengoaL (NEkL, J); HondarraL(NEkL);
   ErantsiL (SP, J);
   while  not HutsaDaL?(NEkL)  loop
        LehenengoaL (NEkL, I); HondarraL (NEkL);
        if BukaeraT(J) ≤ HasieraT(I)
        then  J:= I;  ErantsiL (SP, J); end if;
   end loop;
   return SP;
end EkintzaHauteslea;
```

Analisia: $\Theta(n \lg n + n) = \Theta(n \lg n)$

Adibidea

1. “Lehenen amaitzen den eskaera” prozedura hauteslea integratua duen algoritmo jaleak zer itzuliko du ondorengo eskaerekin?

   {[8:30, 9:30), [12:00, 12:45), [11:30, 12:15), [11:00, 12:30), [10:00, 10:50) }

2. Eta “Beranduen hasten den eskara” prozedurakin?

Adibidea

1. “Lehenen amaitzen den eskaera” prozedura hauteslea integratua duen algoritmo jaleak zer itzuliko du ondorengo eskaerekin?

   { [8:30, 9:30), [12:00, 12:45), [11:30, 12:15), [11:00, 12:30), [10:00, 10:50) }

   { [8:30, 9:30), [10:00, 10:50), [11:30, 12:15), ~~[11:00, 12:30)~~, ~~[12:00, 12:45)~~**}**

2. Eta “Beranduen hasten den eskaera” prozedurakin?

   {[12:00, 12:45), ~~[11:30, 12:15), [11:00, 12:30),~~ **[10:00, 10:50), [8:30, 9:30) }**

Motxila 0/1 vs. zatiekin

- Datuak:
  - 1..n objektuak: p(i)= i objektuaren pisua eta b(i) i-ren balioa
  - Edu: Motxilaren edukiera

- Helburua 0/1: motxilaren edukiera gainditu gabe lor litekeen irabazi maximoa objektuak osorik sartuz
  - Beste bertsioa: **Objektu zatiak sar garri**

- Pareko beste enuntziatu batzuk:
  - 0/1: Tren bagoiak+ granito blokeak; furgonetak + paketeak
  - zatiekin: garraioak + kafe zakuak

Motxila. Prozedura hautesleak

a) Gehien balio duena:

Esperantza: Irabazi lokal maximoa ematen digu, daudenen artetik etekin handien ematen duena oraingoz, eta gero besteak gehituko dizkiogu.

Gezurra: E=10

p1=8 p2=4 p3=5

b1=15 b2=10 b3=10

b) Arinena:

Esperantza: Oso gutxi pisatzen duenez, gehiago sartuko zaizkigu eta ondorioz bakoitzaren etekina batuz irabazi gehiago lortuko dugu, ez?.

Gezurra: E=10

p1=8 p2=4

b1=15 b2=10

Motxila. Prozedura hautesleak

c) Max{Balioa(I)/Pisua(I)}: edo parekoa Min{Pisua(I)/Balioa(I)}

Esperantza: Bien arteko erlazioa proportzioan irabazi onena ematen diguna aukeratzea dirudi.

Gezurra: E=10

p1=7 p2=5 p3=5

b1=7 b2=4 b3=4

ONDORIOA:

Motxila 0/1 problemak EZ du ORAINGOZ soluzio jale zuzena. Frogarik ez da lortu

Aldiz, (c) p. hauteslea zuzena da **motxila zatiekin** kasurako.

Frogak existitzen du

Motxila **zatiekin.**

Osogarria: Sartzen den bitartean, osorik edo proportzio bat, baina edukiera gainditu gabe

SoluzioaDa?: Kasu honetan beti izango da soluzioa eraikitzen goazen azpimultzoa

Algoritmoa: aldagaiak

Pisua(I): I objektuaren pisua

Balioa(I): I objektuaren irabazia/prezioa/etekina,...

Propor(I):=Balioa(I)/Pisua(I);

Atzipena(I)=J; I.proportzio handiena J objektuak ematen du.

```
procedure MotxilaZatikin (E: in Integer; Pisua,Balioa: in
                          TaulaN; Irabazia: out Integer;
                                  Obj: out TaulaN)
   Proportzioa,AtzipenOrd: TaulaN;
   Irabazia,EspazioLibrea, In,Garrena:Integer;
begin
   HasieratuZeroraObj(Obj); ProporLortu(Pisua,Balioa, Propor);
   ProportzioarekikoBeheruntzOrdenatu (Propor,AtzipenOrd);
   Irabazia:= 0; EspazioLibrea:=E; Ind:=1;
   while  0<EspazioLibrea and  Ind<N+1  loop
     Garrena:=AtzipenOrd(Ind);
     if Pisua(Garrena) ≤ EspazioLibrea
     then  Obj(Garrena):=1;
           EspazioLibrea:= EspazioLibrea-Pisua(Garrena);
           Irabazia:=Irabazia+Balioa(Garrena);
     else  SartuBeharDa:= EspazioLibrea/Pisua(Garrena);
           Obj(Garrena):= SartuBeharDa; EspazioLibrea:= 0;
           Irabazia:=Irabazia+SartuBeharDa* Balioa(Garrena);
     end if;
     Ind:=Ind+1;
   end loop;
end MotxilaZatikin;
```

Adibidea

Motxila zatiekin algoritmo jalearen sarrerako datuak:

p=[7, 5, 8, 4, 8] Edu= 15

b=[7, 5, 9, 6, 5]

ProportzioarenAraberaBeheruntzOrdenatu azpiprozesua: efektua adibide bidez:

Azpiprozesuaren sarrera datuak

Proportzioa:

AtzipenOrd:

EMAITZA Obj =

Irabazia =

EspazioLibrea =

Adibidea

Motxila zatiekin algoritmo jalearen sarrerako datuak:

p=[7, 5, 8, 4, 8] Edu= 15

b=[7, 5, 9, 6, 5]

ProportzioarenAraberaBeheruntzOrdenatu azpiprozesua: efektua adibide bidez:

Azpiprozesuaren sarrera datuak

Proportzioa: 1 1 1.13 1.5 0.63

AtzipenOrd: 4 3 2 1 5

EMAITZA Obj = [0, 3/5, 1, 1, 0]

Irabazia = 6+ 9+ (3/5)

EspazioLibrea = 0

H.Z.M. - Kruskal

Grafoko erpin guztiak konektatzen dituen eta haien pisuen batura minimizatzen duen den ertzen multzoa mugatzea duen algoritmoa idatzi nahi da; hots, grafo (konexu, ez-zuzendua eta pisudunaren) **H**edapen **Z**uhaitz **M**inimoa kalkulatzea

Datuak:

- Pisuak positiboak izan behar dute
- **Kruskal**en soluzioak: grafoa ertz pisudunen **zerrenda** bidez adierazia behar du

- **Hautesle prozedura:** pisu txikieneko ertza aukeratu.
- **Osogarria**: Ziklorik ez du gehitzen *ERT Soluzio Partzialean*
- **SoluzioaDa?**: Zuhaitza bada SP

Froga liburuan kontsultagarri

Kruskalen algoritmoaren hurbilketa

```
algoritmoa KRUSKAL (G=<Erpinak, Ertzak,Pisuak) ERT: Ertz-
   multzoa
   ...
hasi
   L:= PisuenGoranzkoOrdenaJarraituzSailkatu (Ertzak);
   P := ErpinKopurua (Erpinak);
   MultzoHutsaErt (ERT);
   errepika  (Ertz_kopurua(ERT) ≠ P-1) hasi
        Pisu_txikieneko_ertza_aukeratu(L,(x,y));
        Aukeratutako_ertza_kendu(L,(x,y));
        baldin not Ziklorik_eransten_du?(ERT U {(x,y)})
        orduan ERT:= ERT U {(x,y)};
        bukatu baldin;
   bukatu errepika;
bukatu KRUSKAL
```

H.Z.M. – Kruskalen algoritmoa

```
procedure KRUSKAL(G: in GRAFOA; SErt: out Ertz_multzoa)  is
   OsKonexuak: PPartiketaMota;
   P,SErtzKop: Integer;
begin
   L:= PisuenGoranzkoOrdenaJarraituzOrdenatu(ERTZAK(G));
   P := ErpinKop (G); SErtzKop:=0; MultzoHutsaErt (SErt);
   pMultzoHaseratuBakoitzaErpinEzberdinBatekin(OsKonexuak);

   while  (SErtzKop ≠ p-1)  loop
        KontsideratuEzDenPisuTxikienekoErtza (L, (x,y));

        XBarne:=BILATU3(OsKonexuak, X);
        YBarne:=BILATU3(OsKonexuak, Y);

        if Xbarne≠Ybarne then
           BATERATU3(OsKonexuak, XBarne, YBarne);
           ErantsiErt(SErt, (x,y));
           SErtzKop:= SErtzKop+1;
        end if;
   end loop;
end KRUSKAL;
```

Kruskal. Analisia

- Hasieraketak: $\Theta(a \lg a + p + p) = \uparrow = \Theta(a \lg p)$

  $p-1 \leq a \leq (p\,(p-1))/2 \rightarrow \Theta(\lg a) = \Theta(\lg p)$

- Begizta: kasu txarrenean ordenazio zerrendako azkeneko ertza gehitzen da Sert multzora

  - `[baterat3,bilatu3]`$^n \in \Theta$`(n lgn)`

  - `2a` aldiz `Bilatu3` + `(p-1)` aldiz `Bateratu3`

    $\in \Theta((2a+p) \lg (2a+p))$

    $= \Theta(a \lg a) = \Theta(a \lg p)$

- Beraz, baturaren erregela bi agindu blokeei aplikatuz: **$\Theta(a \lg p)$**

H.Z.M. – Prim

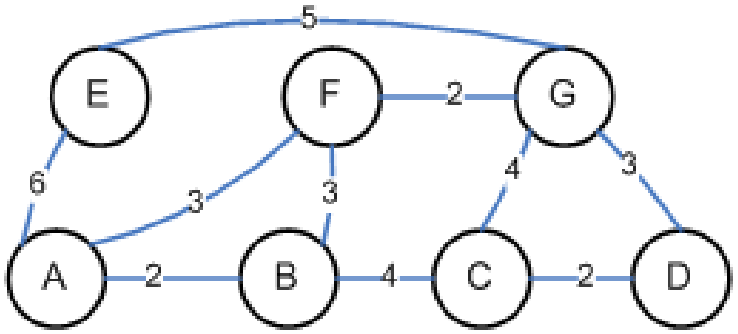
Grafoaren **H**edapen **Z**uhaitz **M**inimoaren kalkulua

Datuak + helburua:

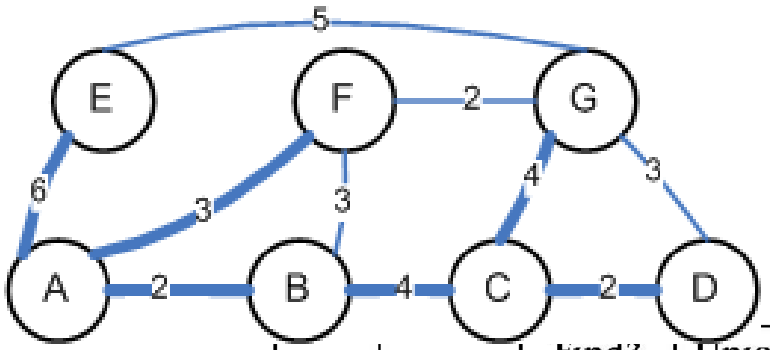
- Kruskalen problema bera prozedura hautesle desberdinez ebatzia
- Grafoa **auzokidetza matrize** bidez adierazia

Algoritmoa: aldagaiak

- ERP: erpinen multzo fiktizioa. SErt multzoko ertzetan dauden grafoko erpinez osatua
- i $\in$ Erpinak-ERP
- Auzokide(i): *i* erpinetik ERP-ekoa den eta hurbilen duen erpina
  - Hau da, (i,AUZOKIDE(i)) ertzak momentu horretako SErt multzoa eta *i* erpina modu merkeenean konektatzen dituen ertza



Bira	ertza	Find3 etiketak	Union3 Bai-Ez	H.Z.M.ri gehitu	Partiketa
0					a b c d e f g
1	(__ , __)	__ , __			a b c d e f g
2	(__ , __)	__ , __			a b c d e f g
3	(__ , __)	__ , __			a b c d e f g
4	(__ , __)	__ , __			a b c d e f g
5	(__ , __)	__ , __			a b c d e f g
6	(__ , __)	__ , __			a b c d e f g
7	(__ , __)	__ , __			a b c d e f g
8	(__ , __)	__ , __			a b c d e f g
9	(__ , __)	__ , __			a b c d e f g

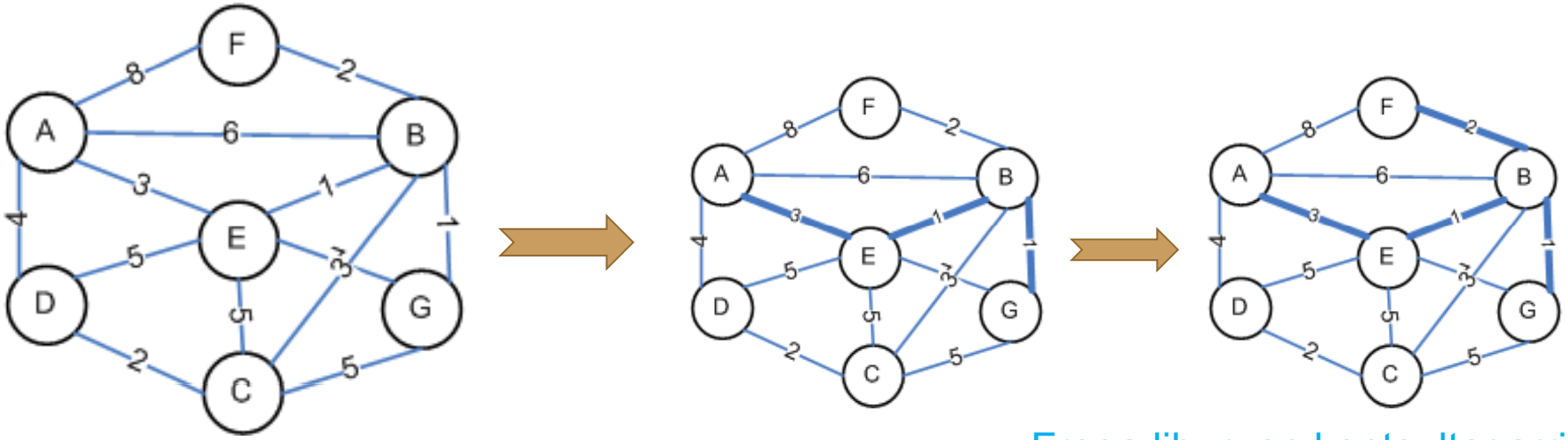


Bira	ertza	Find3 etiketak	Union3 Bai-Ez	H.Z.M.ri gehitu	Partiketa
0					0 0 0 0 0 0 0 a b c d e f g
1	(C, G)	C, G	BAI	(C, G)	0 0 -1 0 0 0 C a b c d e f g
2	(A, B)	A, B	BAI	(A, B)	-1 A -1 0 0 0 C a b c d e f g
3	(C, D)	C , D	BAI	(C, D)	-1 A -1 C 0 0 C a b c d e f g
4	(A, F)	A, F	BAI	(A, F)	-1 A -1 C 0 A C a b c d e f g
5	(D, G)	C, C	EZ		-1 A -1 C 0 A C a b c d e f g
6	(B, C)	A, C	BAI	(B, C)	-2 A A C 0 A C a b c d e f g
7	(B, G)	A, A	EZ		-2 A A C 0 A C a b c d e f g
8	(B, F)	A, A	EZ		-2 A A C 0 A C a b c d e f g
9	(A, E)	A, E	BAI	(A, E)	-2 A A C A A C a b c d e f g

H.Z.M. – Primen algoritmoa

h.z.m.-ren eraikuntza hedatuz egiten da, bertako erpinak ERP multzo fiktizioa osatuz.

Hautesle prozedura: (Erpinak-ERP) eta ERP multzoko erpinak konektatzen dituzten ertzen artetik pisu txikieneko ertza aukeratu



Froga liburuan kontsultagarri

H.Z.M. – Primen algoritmoa

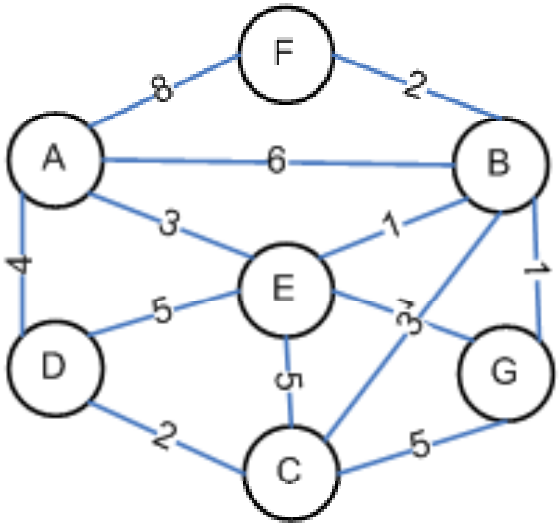
Algoritmoa: aldagaiak

- Auzokide(i):
  - i $\in$ Erpinak-ERP; hots h.z.-ean oraindik ez dagoen erpina
  - espresioak *i* erpinetik ERP-ekoa den eta hurbilen duen erpina metatzen du; hots, (i, Auzokide(i(i)) ertzak momentu horretako SErt multzoa eta *i* erpina modu merkeenean konektatzen dituena da.

- PisuMin(i):
  - (i, Auzokide(i)) ertzaren pisua
  - *i* $\in$ERP $\rightarrow$ PisuMin(i) = -1

```
procedure  PRIM (G: in Matrizea; SErt: out ErtzMultzoa) is
begin
  MultzoHutsaErt (SErt);
  for  K   in  G´First(1) +1..G'Last(1)  loop
       Auzokide(K) := 1; PisuMin := G(K,1);
  end loop;

  for Ind in  G´First(1)..G'Last(1)-1  loop
    Min := System.MAX_INT;
    for J in G´First(1)+1..G'Last(1) loop
      if 0≤PisuMin(J)<Min then Min:= PisuMin(J); K:= J;end if;
    end loop;

    ErantsiErt(SErt,(K,Auzokide(K)); PisuMin(K):= -1;
    for J in G´First(1)+1..G'Last(1) loop
      if G(K,J)<PISU_MIN(J)
      then PisuMin(J):= G(K,J); Auzokide(J):= K; end if;
    end loop;
   end loop;
end PRIM;
```

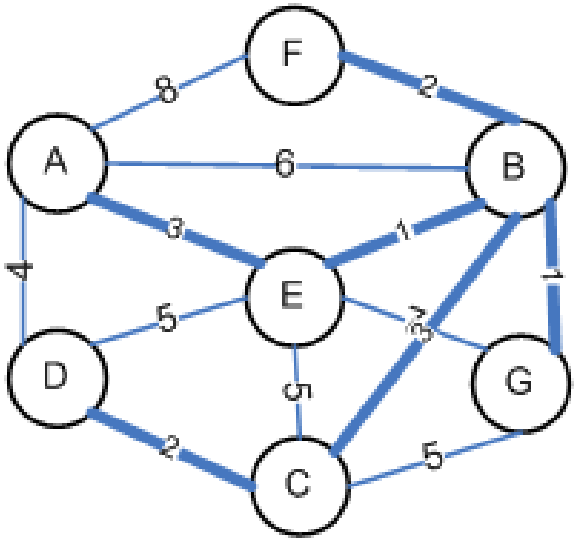
$\Theta(p^2)$



Iter.	Erpina	Ertz	Pisua
0	A		
1		(,)	
2		(,)	
3		(,)	
4		(,)	
5		(,)	
6		(,)	

Kostua Orora

	A	B	C	D	E	F	G
PisuMin							
Auzokide							



Iter.	Erpina	Ertz	Pisua
0	A		
1	E	(A,E)	3
2	B	(E,B)	1
3	G	(B,G)	1
4	F	(B,F)	2
5	C	(B,C)	3
6	D	(C,D)	2
	Kostua Orora		12

	A	B	C	D	E	F	G
PisuMin		6	∞	4	3	8	∞
Auzokide		A	A	A	A	A	A
PisuMin		1	5	4	-1	8	7
Auzokide		E	E	A	A	A	E
PisuMin		-1	3	4	-1	2	1
Auzokide		E	B	A	A	B	B
PisuMin		-1	3	4	-1	2	-1
Auzokide		E	B	A	A	B	B
PisuMin		-1	3	4	-1	-1	-1
Auzokide		E	B	A	A	B	B
PisuMin		-1	-1	2	-1	-1	-1
Auzokide		E	B	C	A	B	B
PisuMin							
Auzokide							

Kruskal vs Prim

- Kruskal $\in\Theta(a \lg a)$ eta Prim $\in\Theta(p^2)$

- Kasuak:
  - Grafoak ertz asko $\rightarrow$ a $<\approx$ (p (p-1)/2)

    Kruskal $\in\Theta(p^2 \lg p)$ izango da ($\Theta(\lg p)=Q(\lg a)$)

    Prim-en hobea

  - Grafoak ertz gutxi $\rightarrow$ a $\approx$ n

    Kruskal $\in\Theta(p \lg p)$, hortaz Prim baino eraginkorra

Distantzia minimoak - Dijkstra

G grafo zuzendua eta pisuduna: G = <Erpinak, Arkuak, Pisuak>.

Erpin bat abiapuntutzat hartuko dugu.

Helburua: abiapuntu erpinetik grafoko beste erpinetara doazen bide motzenak mugatzea

- H hautagai erpinen multzoa.
- E erpin hautatu eta onartuen multzo fiktizioa.
- **Bide berezia:** abiapuntu-erpinetik beste erpin batera doan bidea eta E-ko erpinez osatuta dagoena soilik.
- Bide berezi motzenak bakarrik kontuan hartzeak bide motzenak ekoizten dituztela frogagarria da (ikus liburua)

- Kalkuluak erraztearren :
  - grafoko erpinak: {1,2, .. , P}
  - abiapuntu-erpina: 1
  - Auzokidetza matrizea

    $G(i,j) \geq 0$ -> $\exists(i,j) \in G$ eta bere pisua da

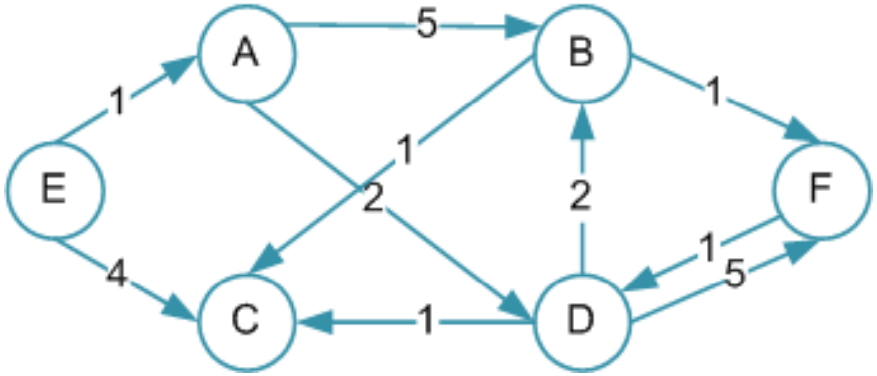
    $G(i,j) = \infty$ -> $\neg\exists(i,j) \in G$

Prozesua:

Froga liburuan kontsultagarri

- Begizta jalearen bira bakoitzean:
  - bide berezi motzena aukeratzen da, bide berezi motzena bide motzena bihurtuz.
  - E-koa ez den erpin berria E-ri gehitzen zaio (H-tik kenduz),
  - erpin berria gehitzeagatik sortu diren bide berezi berriak aurrekoak baino motzagoak badira erregistratzen dira.

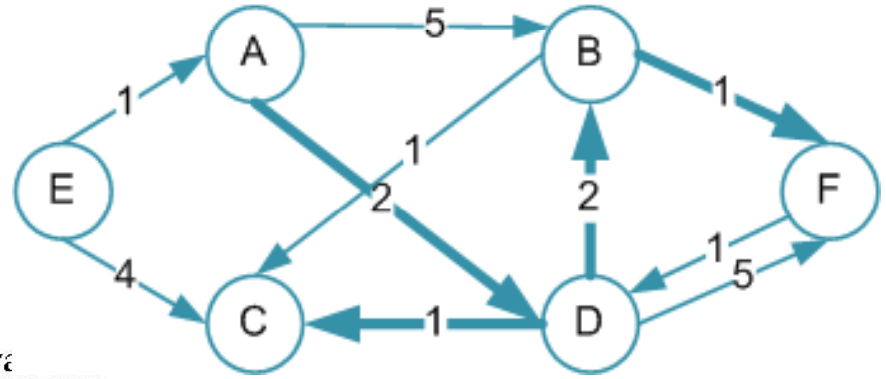
```
procedure  DIJKSTRA ( G: in GRAFOA;  D: out TAULA) is
   P: Integer:=G'LENGTH(1);
begin
   HASIERATU2_P(G,H);
   for K in  G'FIRST(1)+1..G.'LAST(1) loop D(K):= G(1,K);
   end loop;

   for  I in  P-2  loop
      GertuenDagoenErpina(H,D,X);          ∈Θ(p²)
      HautagaietatikKendu(H,X);            ∈Θ(p²)
      HK:= H;
      for J in 1..ErpinKopurua(H) loop
       Y:= LehenengoaL(HK);
       AurreraEgin(HK);
       if D(Y)>D(X)+G(X,Y) then D(Y):= D(X)+G(X,Y); end if;
      end loop;                            (p-2) + (p-3)+ ... + 1 ∈Θ(p²)
   end loop;
end DIJKSTRA ;                                        Θ(p²)
```



Distantzia taularen hasieratzea:

A	B	C	D	E	F

Harrapatutako erpina	Distantzia taularen eguneraketa:
1 bira	A B C D E F
2 bira	A B C D E F
3 bira	A B C D E F
4 bira	A B C D E F



Distantzia taularen hasiera

X	5	∞	2	∞	∞
A	*B*	*C*	*D*	*E*	*F*

	Harrapatutako erpina	Distantzia taularen eguneraketa:					
1 bira	D	X	4	3	2	∞	7
		A	*B*	*C*	*D*	*E*	*F*
2 bira	C	X	4	3	2	∞	7
		A	*B*	*C*	*D*	*E*	*F*
3 bira	B	X	4	3	2	∞	5
		A	*B*	*C*	*D*	*E*	*F*
4 bira	F	X	4	3	2	∞	5
		A	*B*	*C*	*D*	*E*	*F*

Backtrack:
Atzera jotze “inteligentea”

R. Arruabarrena
LSI - UPV/EHU

Sarrera

- Bilaketa sakona/exhaustiboa erabiltzea da aukera bakarra soluzioa bilatzeko hainbat problementzat
- Ebazten dituen problema motak dira:
  - Erabakitze problemak: soluziorik badu?
  - Konbinazio problemak:
    - zenbat konbinazio desberdin daude soluzio direnak?
    - konbinazio onena zein da? (=optimizazio problemak)
    - Soluzio den lehenengo konbinazioa zein da?
- eskema generikoa egileetan zehar desberdina da
  - [liburuan beste bat]
- Oraingoan, soluzioa eraikitzeko modua:
  - Saiakuntzak eginez soluzio partzialak hedatzen saiatuko gara

Sarrera

- Saiakeren bidezko prozesua
  - Saiakuntza neutrotik hasi: []
  - tarteko pausoan [x1,x2,...,xi] soluzio partziala eraiki dugu
  - hurrengo pausoan xi+1 **aukera guztiekin** hedagarria ote den aurreko saiakuntza soluzio partziala aztertzen dugu:
    - [x1,x2,...,xi,xi+1] saiakuntza onargarria bada, [x1,x2,...,xi] soluzio partziala [x1,x2,...,xi,xi+1]ra hedatzen da.
    - xi+1 –en aukerak [x1,x2,...,xi,xi+1] onartezinak egiten badituzte, honek adieraziko du [x1,x2,...,xi] soluzio partziala ez dela hedagarria (hots, ez du soluziorik emango); ondorioz soluzio partzialari gehitutako azkeneko aukera kentzen diogu, xi-a, eta jarraitzen dugu [x1,x2,...,xi-1] saioan beste aukera bat aztertzen.
  - **saio bat onargarria** den ala ez erabakitzeko irizpidea
    - propioa da problema bakoitzeko
    - ezinbesteko garrantzia du eraginkortasunean

Sarrera

- Estrategia hau zuhaitz egitura bat sakoneran korritzearen baliokide da
  - korritzea inplizitua da, zuhaitza inplizitua baita (dma-k ez du existitzen)
  - zuhaitzaren errotik hasten da
  - adabegia = saio partziala
  - arkua = aztertzen ari garen aukera bat
  - saiakuntza bat onartezina denean, zuhaitzeko **adarra kimatzen** da
    - bertatik aterako litzatekeen azpizuhaitza ez da eraikiko, ez da denbora hartan inbertituko
  - zenbat eta algoritmoa “argiago”/”azkarragoa” izan, orduan eta kimatze gehiago egongo da eta eraginkorragoa izango da
  - hostoak:
    - hedagarriak ez diren saioak, ala
    - soluzioak

Sarrera

- Backtrack esplorazio-zuhaitza irudikatu
  - 01, 1N, iN
  - Kasu nabariak
  - Kasu orokorretako adarketak
  - Onargarria noiz
  - Kimak
- Aldagaiak, lokalak, globalak,
- Algoritmoa

Txantiloi orokorra

```
prozedura   saiatu   (SAIAKUNTZA)
baldin SoluzioaDa(SAIAKUNTZA)
orduan   "Kudeatu SAIAKUNTZA saiakera"
{bestela}
   errepikatu  SAIAKUNTZA hedatzeko A aukera guztiekin  egin
       baldin   OnargarriaDa (SAIAKUNTZA+A)
       orduan   saiatu (SAIAKUNTZA+A)
```

-- {bestela} klausula kendu behar da soluzio batzuk beste batzuen aurrizkiak direnean

Txantiloi orokorra (1 soluzioa lortzeko)

```
prozedura   saiatu   (SAIAKUNTZA, ARRAKASTA)
baldin SoluzioaDa(SAIAKUNTZA)
orduan   "Kudeatu SAIAKUNTZA saiakera"
          ARRAKASTA:=True
bestela
  errepikatu  SAIAKUNTZA hedatzeko A aukera guztiekin  egin
       baldin Onargarria (SAIAKUNTZA+A)
        orduan  saiatu (SAIAKUNTZA+A, ARRAKASTA)
       baldin  ARRAKASTA orduan "amaitu errepikatu"
```

Motxila 0/1

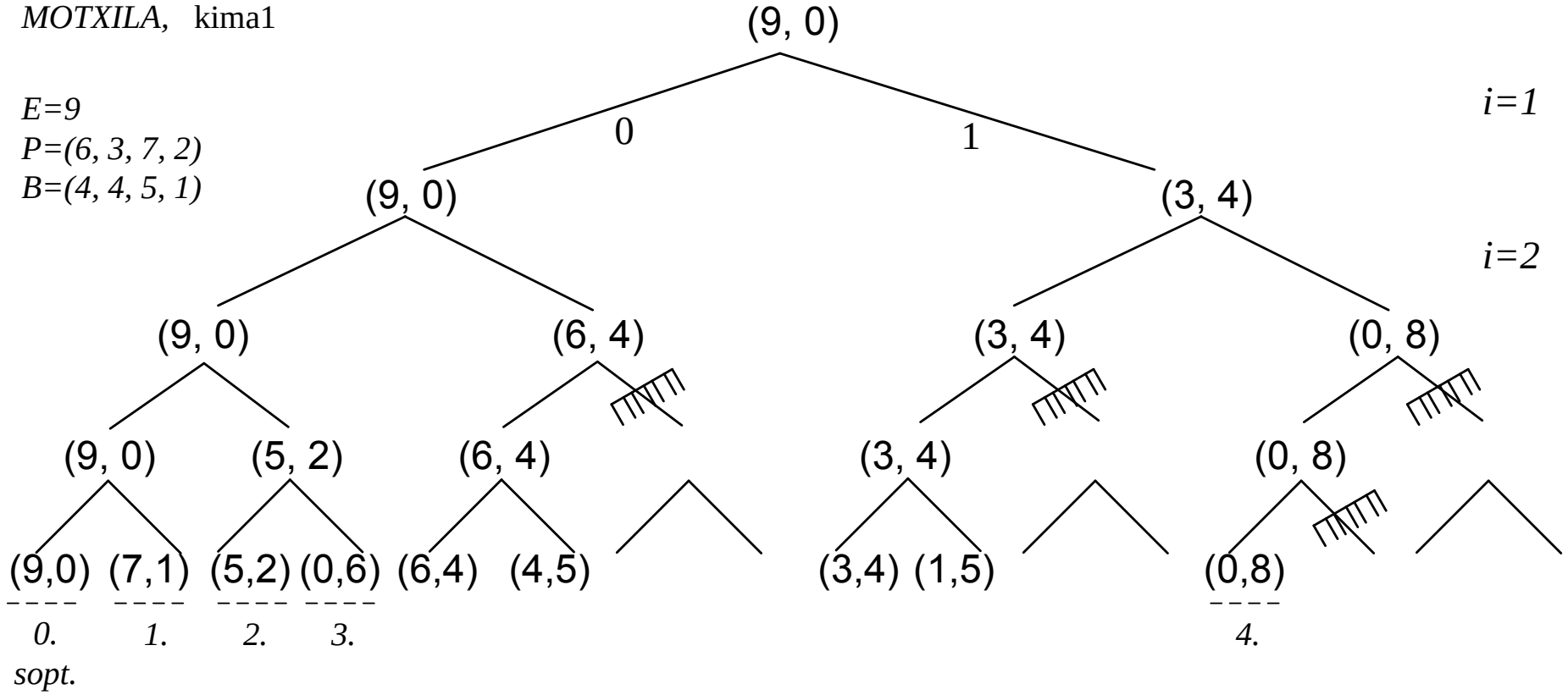
- Datuak:
  - E: motxilaren edukiera
  - N objektu, i objektu bakoitzeko: P(i) pisua eta B(i) balioa
- Helburua:
  - Objektuak osorik sartuz hauek motxilari gehitzen dioten irabazia maximizatu motxilaren edukiera gainditu gabe.

- Optimizazio problema dela eta:
  - orain arteko soluzio onena zein den eraman behar da: SOPT
  - komeni da haren balio optimoa ere eramatea: SOPTB

- Soluzioaren eta soluzio partzialen formatua:
  - n osagai dituen sekuentzia: [x1,...,xi,...,xn], non
  - osagai bakoitza xi $\in$ {0,1} $\equiv$ {EzDaSartzen,Sartzen da}

MOTXILA, kima1

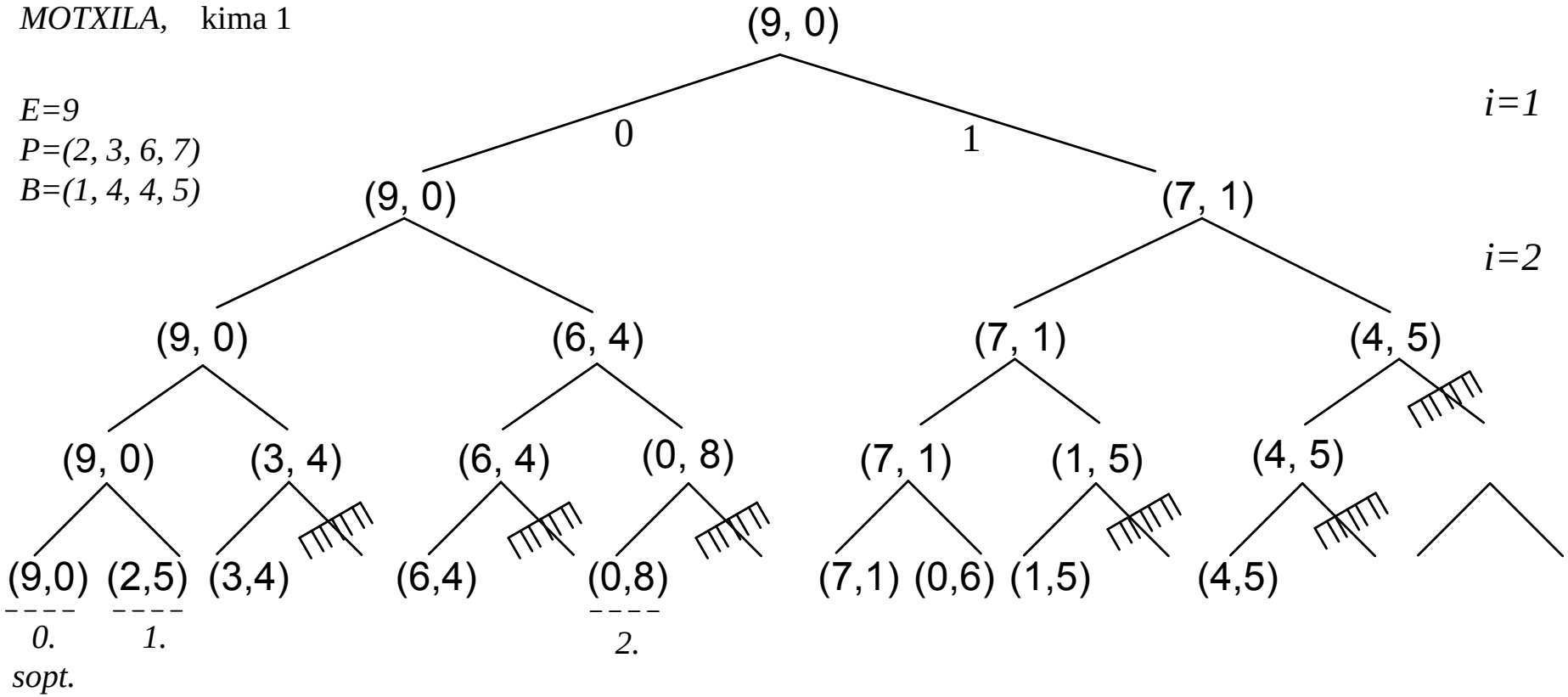
E=9
P=(6, 3, 7, 2)
B=(4, 4, 5, 1)



sopt.

MOTXILA, kima 1

E=9
P=(2, 3, 6, 7)
B=(1, 4, 4, 5)



Motxila 0/1

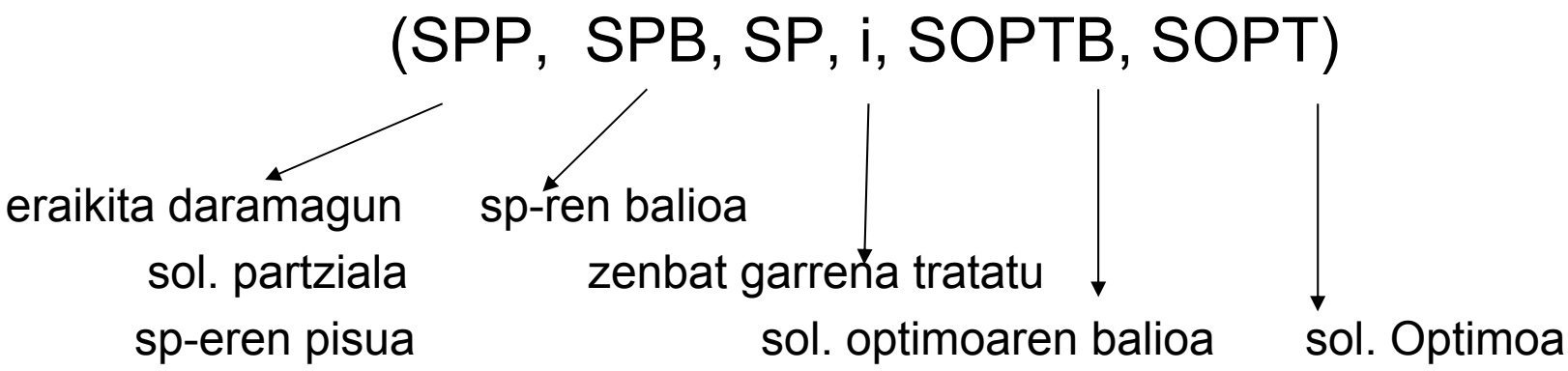
- Komeni da orain arte eraikita daramagun SP=[x1,...,xi] soluzio partzialeko hainbat datu ezagutzea:
  - dituen osagai kopurua: i
  - $SPP = \sum_{k=1}^{i} pisua(k) \times X_k$
  - $SPB = \sum_{k=1}^{i} balioa(k) \times X_k$

- Soluzio partziala noiz soluzioa da?
  - n osagai dituenean
  - [Espazio librerik ez dagoenean]
  - [Gelditzen den espazio librea objektu arinena baino txikiago denean]

  eta, SP berria orain arteko optimoa baino hobea bada: erregistratu

Motxila 0/1

- Parametrizazioa, adibidez:



- Lehenengo deia: `MotxilaBt01K1(0,0,[],1,0,[])`

- MotxilaBt01K1**(0,0,[],1,0,[])**

```
prozedura     MotxilaBt01K1 (SPP,SPB,SP,i,SOPTB,SOPT)

if i=n+1  then if SPB>SOPTB then  SOPT(1..n):=SP(1..n);
                                  SOPTB:= SPB;
               end if;

else
   for  A  in  1..0   loop
       if  SPP+A × Pisua(i) <= E
       then  SP(i):=A;
             MotxilaBt01K1 (SPP+ A × P(i),
                            SPB+A × B(i),
                            SP, i+1, SOPTB, SOPT)
       end if;
   end loop;
end if;
```

Motxila. Analisia

- Algoritmo hauen eraginkortasuna kalkulatzea ez da erreza
- Zehazki zenbat lan egiten duten kalkulatzea zaila da
  - “kima” funtzioaren (“inteligentzia”) kostuaren kalkuluak zailtzen du
- Modurik egokiena eraginkortasun kostuaren hurbilpen bat lortzeko inplizituki aztertzen duten saiakuntzen zuhaitza aztertuaz egiten da
  1. Aztertzen diren saiakuntzak zenbatu
  2. Saiakuntza bakoitza hedatzeko aukera guztien eraikuntzaren kostua aztertu
  3. Saiakuntza berria egitearen kostua aztertu datuen/parametroen prestaketa
     - dei errekurtsiboaren aurretik eta
     - itzuleran aurreko saiakuntzako datuen berreskurapena
     - saiakuntzak kasu nabarian eta bestetan duten kostua aztertu
  4. Aukera berria daukagun saiakuntzarekin onargarria den erabakitzearen kostua
     - hedagarria da SP+A? Honen kostua

- 2,3 eta 4 ataletako eragiketen kostua denbora konstantean egiten dira, kasu nabarietako bektorearen erregistroa salbu. Hauek Q(n) behar dute
  - 2^n da hosto kopurua eta
  - gehienez hauetan guztietan egingo da balioen hobekuntza

- 1 atalak erabakiko du ondorioz ordena.

- Kasu txarrenean, ez du adarrik kimatuko, *p(1)+p(2)+…+p(n)<=E,* eta eraikiko dituen soluzio partzial kopurua (egingo dituen saiakuntza kopurua) da:

$$1+2^1+2^2+...+2^n=2^{n+1}-1$$

baitakigu:
  - hasierako dei bat dugula
  - ondorengoetan, beti aukera bina dugula: bi saiakuntzekin hedagarria.
  - edozein soluzio-saiakuntza luzeenaren luzera n da

- 2^n hosto (Kasu nabari) leudeke eta haien tratamendua $\Theta(n)$ da.

- Algoritmoaren denbora-ordena $O(\mathbf{n2^n})$ (zehatza, kimarik ez balego) ($KN \in O(n)$)

Backtrack: analisia

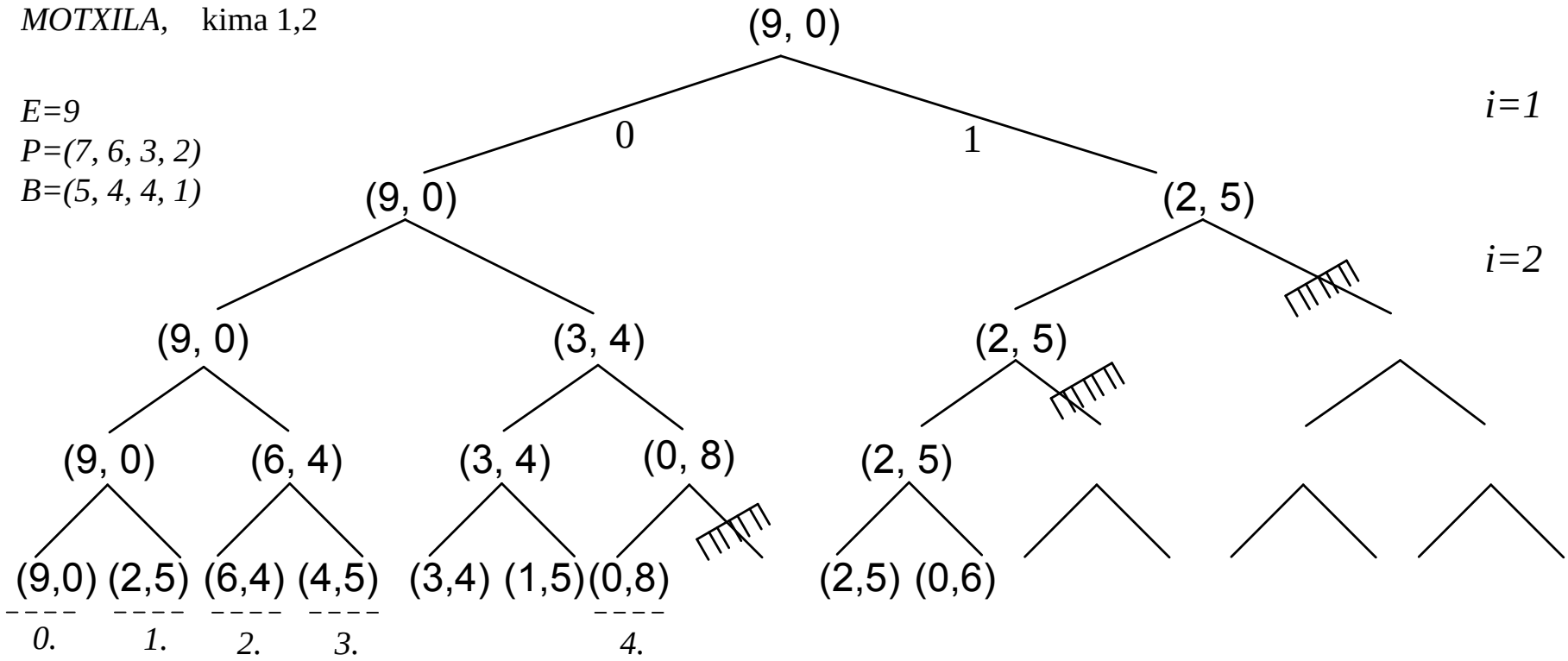
- Oro har, kalkulatzen den denbora ordena **goi-ordena** izaten da
- Benetako ordena (Θ) kalkulatzea eragiketa zaila da guztiz
  - $\equiv$ zehazki zenbat adabegi izango dituen zuhaitza inplizituak jakitea
  - Kimatzeko erabiltzen den funtzioa, zenbat eta hobea izan orduan eta eraginkorragoa izango da soluzioa eta goi ordena horretatik urrunduko da
- Ez da ahaztu behar
  - kimak ez ditu saiakuntza hedagarri onargarriak (soluzio posibleak) galdu behar
  - kimatu behar ote den kalkuluak denbora behar du
  - kiman ala zuhaitz handiagoa eraikitzen inbertitu denbora?

Oreka bilatu edo une horretan interesatzen dena

- Dagoeneko asmatu dira hainbat teknika estatistiko ESTIMATZEKO kimatzen duten algoritmo sakonak edo backtrack algoritmoek zenbat saiakuntza egingo lituzketen:
  - Monte Carlo
  - Las Vegas
- Aurten ez ditugu ikusiko
- Ondorioz, zuhaitzak gehienez edukiko lukeen tamaina soilik mugatuko dugu.
- Analisiaren goi-ordena soilik lortzeko aukera izango dugu.
- Soluzio esponentziala lortzen badugu, algoritmoa bakarrik erabilgarria izango da oso tamaina txikia duten sarrerekin.
- Kima onak lortuko bagenitu (AA heuristikoak ikusiko dituzue), adar osoak eraikiko ez direla eta, soluzio erabilgarriagoak lortuko genituzke, eta haien ordena ESTIMATZEKO goian aipatutako hurbilpen estatistikoak erabili beharko zenituzkete.

MOTXILA, kima 1,2

E=9
P=(7, 6, 3, 2)
B=(5, 4, 4, 1)



sopt.

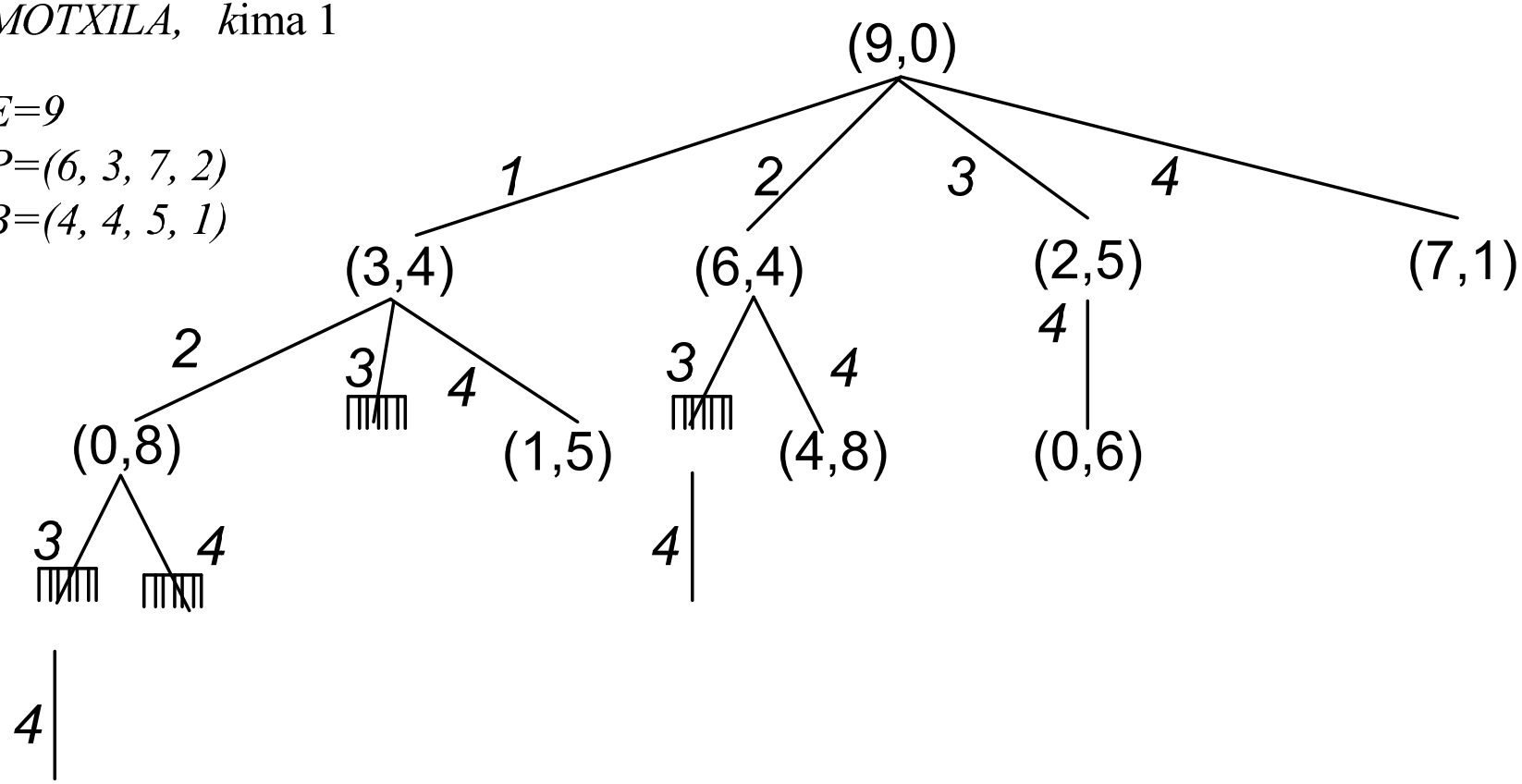
Motxila, 01 zuhaitza, 2 kimekin

- 1 deia: `MotxilaBt01K2` **(0,0,[],1,0,[],** $\sum_{j=1}^{n} Balioa(j)$ **)**

```
prozedura MotxilaBt01K2(SPP,SPB,SP,i,SOPTB,SOPT,KanpokoenBal)

if i=n+1  then if SPB>SOPTB then  SOPT(1..n):=SP(1..n);
                                  SOPTB:= SPB;
               end if;

else
   for  A  in  1..0   loop
       if  SPP+A × Pisua(i) <= E and SPB+KanpokoenBal > SOPTB
       then  SP(i):=A;
             MotxilaBt01K2 (SPP+ A × P(i),
                            SPB+ A × B(i),
                            SP, i+1, SOPTB, SOPT,
                            KanpokoenBal-B(i))
       end if;
   end loop;
end if;
```

MOTXILA, *k*ima 1

E=9
P=(6, 3, 7, 2)
B=(4, 4, 5, 1)



Motxila, 1N zuhaitza, 2 kimekin

- Kalkulatu eta global utzi: $Out(i) = \sum_{j=i}^{n} Balioa(j)$
- 1 deia: `MotxilaBt1NK2(0,0,[],1,0,[])`

```
prozedura MotxilaBt1NK2(SPP,SPB,SP,i,SOPTB,SOPT)

if i=n+1 or (E-SPP<Pisua(n))
                -- pisua(1)>...>pisua(n), bestela arinena Min{p(1),..p(n)}
then  if SPB>SOPTB then  SOPT:=SP; SOPTB:= SPB;
      end if;
else
   for  A  in  i..n  loop
       if  SPP+A × Pisua(i) <= E and SPB+Out(A)> SOPTB
        then  GehituBurutikZer(SP, A);
              MotxilaBt1NK2 (SPP+ A × P(i), SPB+ A × B(i),
                             SP, A+1, SOPTB, SOPT);
              KenduZerBurua(SP, A);
       end if;
   end loop;
end if;
```

Mapak koloreztatzen 4 kolore erabiliz

- Mapa geografiko bat emanik eta 4 kolore, mapa koloretan margotu nahi da baina elkarren ondoan dauden herrialdeak kolore desberdinetan egon behar dutela jakinik

- Zein herri zein koloretan margotu behar da?
- Problema orokorragoa: k-koloreztapena
- Soluziorik badago? k≥4, bai

- Datuak:
  - MG mugen grafo bat auzokidetza matrize bidez adierazia
  - erpinak: herrialdeak
  - ertzak: muga duten herrialde bikoteak

- Soluzio partzialak edo saiakuntzak:
  - erpin bakoitza zein koloretan margotuko den jaso beharko du
  - Saiakuntzak MARGOA bektore bakarrean jasoko ditugu

- n erpinak 1..n bidez zenbakituak ditugularik:
  - MARGOA=[x1,...,xj,...,xi], non xj$\in${1,2,3,4} $\equiv$ {Urdina, Gorria, Berdea, Horia}
  - k luzera duen saiakuntzan MARGOA(1..k) balioak soilik izango dira baliagarriak
    - lehenengo k erpinen koloreak izango dira

- Saiakuntza soluzioa da:
  - MARGOA-ko n balioak adierazkorrak direnean

- k luzerako saiakuntza hedatzea:
  - grafoko (k+1) erpinari kolore bat esleitzea izango da

- k luzerako saiakuntza hedagarria da?:
  - grafoko (k+1) erpinari kolore bat eslei badiezaiokegu
    - kontuan hartu: grafoa eta MARGOA(1..k) koloreak, 1..k erpinetako herrialde mugalarien koloreetako bat ez erabiltzeko

- Parametrizazioa:
  - aldagai globalak: MARGOA, MG grafoa, guztira erabilgarri diren kolore kopurua
  - saiakuntzaren luzera eramango dugu soilik; hots, MARGOA-ko balio adierazkorren kopurua.

- 1 deia: `LauKolore(0)`

```
prozedura LauKolore(i)
if i=n  then inprimatu(MARGOA(1..N))
else
   for  K  in  1..4   loop
      MARGOA(i+1):=K;
      if  OnargarriaDaKolorea?(MG(1..n,1..n), MARGOA, i+1)
      then    LauKolore (i+1);
      end if;
   end loop;
end if;
```

```
function OnargarriaDaKolorea? (MG, MARGOA,azkena)
begin
  onargarri:= true
  herrialde:=1
  while   herrialde<azkena   and   onargarria   loop
      if   MG (herrialde,azkena)=1
           and  MARGOA(herrialde)= MARGOA(azkena)
      then onargarri:=false;
      end if;

      herrialde:= herrialde+1;
   end loop;
   return (onargarri);
end fun;
```

Mapak koloreztatzen. Analisia

- Saiakuntza bakoitza hedatzeko 4 aukera ditugu
- Soluzio–saiakuntzen luzera n da
- Zuhaitz inplizituak izango dituen adabegi edo saiakuntzen kopuruaren goi bornea: $1+4^1+4^2+...+4^n=(4^{n+1}-1)/3$
- Bestalde, saiakuntza kolore berriarekin hedagarria den aztertzeko behar den denbora ordena ez da konstantea.
  - i+1 garren erpinaren auzokideak ote diren 1..i erpinak aztertu behar da eta auzokideak diren haien koloreak zeintzuk diren ikusi.
  - `OnargarriaDaKolorea?` funtzioaren denbora ordena O(n) da.
- Saiakuntza kasu nabaria bada, eta n osagaiak inprimatu behar dira: O(n)
- `LauKolorek` gehienez (n4n) beharko du, hots, bere denbora ordena $O(\mathbf{n4^n})$ da

Txanpon kopuru minimoa itzuli (nahi adina)

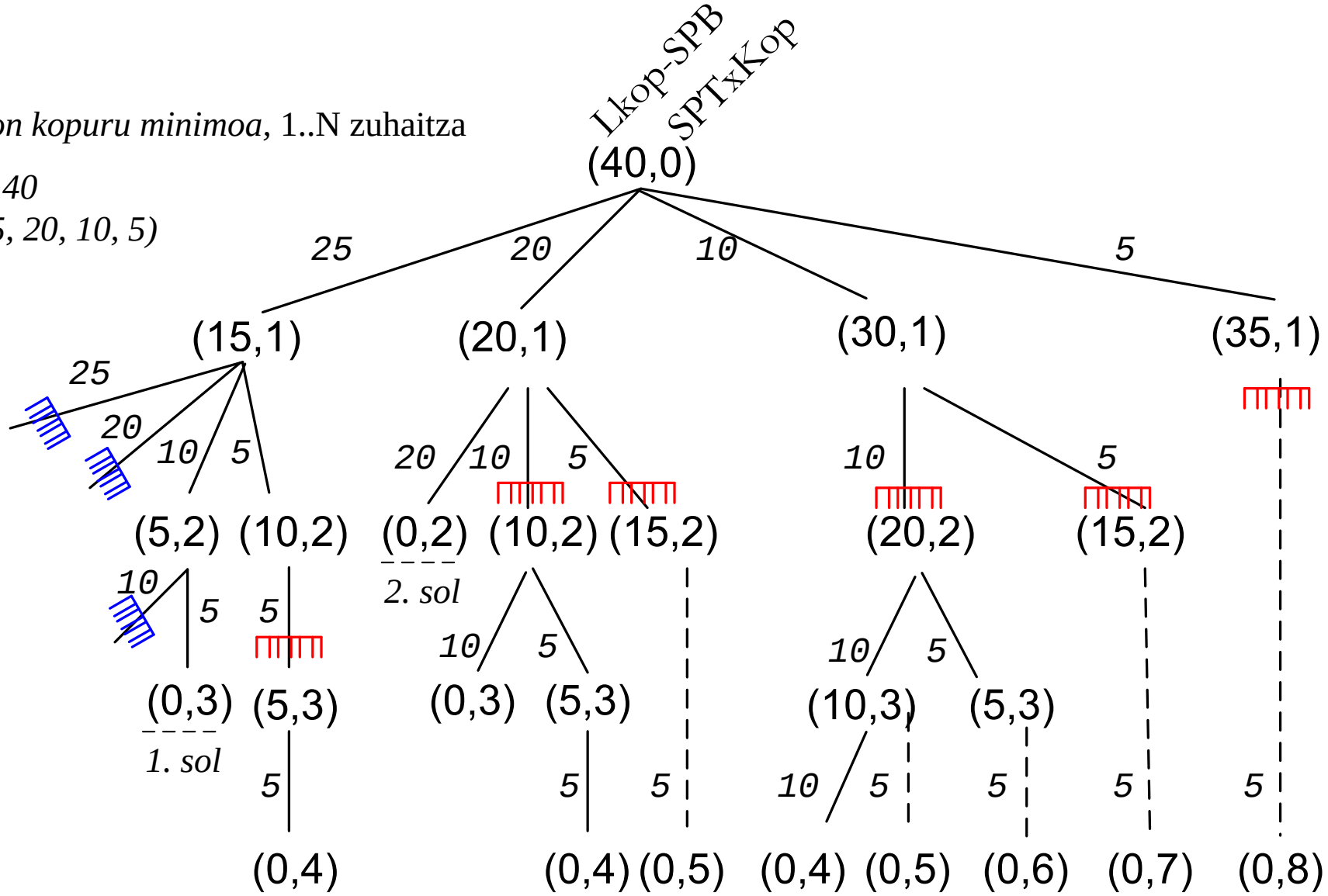
- Datuak:
  - L: Itzuli behar den kopurua
  - N txanpon klase, i txanpon klase bakoitzeko:
    - B(I) da beren balioa,
    - Nahi adina txanpon,
    - [Murriztapenekin, muga(i)=ki i klaseko eskuragarri ditugun txaponak]
- Helburua:
  - L kopurua zehazki itzultzeko behar diren txanpon kopuru minimoa
  - Nahi izanez gero, zeintzuk txanponak konkretuak ematen duten balio minimo hori ere kalkula dezakegu → noski, kostuan eragina du

---

- Optimizazio problema denez:
  - soluzio onenaren balio optimoa eramatea: SOptTxKop
  - orain arteko soluzio onena zein den eraman dezakegu: SOPT

Txanpon kopuru minimoa, 1..N zuhaitza

Lkop= 40
Tx=(25, 20, 10, 5)



K2= soluzio optimoa ezin da hobetu adar honetatik

- Soluzioaren eta soluzio partzialen formatua:
  - n osagai dituen sekuentzia: SP=[z_1,...,z_i,...,z_n], non
  - osagai bakoitza $z_i \in \{0,1,2,...\} \equiv$ i klasekorik {Ez daramagu, bat, bi,..}

- SP noiz soluzioa da?
  - n osagai dituenean **edo L=SPB**
  - eta, SP berria orain arteko optimoa baino hobea bada: erregistratu

- Komeni da orain arte eraikita daramagun SP=[z1,...,zi] soluzio partzialeko hainbat datu ezagutzea:
  - dituen osagai kopurua: i
  - $SPB = \sum_{k=1}^{i} Balioa(k) \times Z_k$ non k klaseko Z_k=SP(K) txanpon daramazkigun
  - $L >= SPB = \sum_{k=1}^{i} Balioa(k) \times Z_k$
  - $SPTxKop = \sum_{k=1}^{i} Z_k = \sum_{k=1}^{i} SP(k)$

Aldagaiak

- i: zenbat garren txanpon klasea tratatu behar den; zuhaitzaren maila da.
- SP: i. klasea aztertu ostean, soluzio partzialak [z1,z2,…,zi] itxura du, zi: i klasetik erabiliko diren txanpon kopurua da
- SPB: i. klasea aztertu ostean, SP-k daramazkien txanponen balioen batura; BATUKARIA(j=1..i, b(j)*SP(j))
- SPTxKop : Berdin, baina SP-k daramazkien txanpon kopurua; hots, BATUKARIA(j=1..i, SP(j))
- SOpt: soluzio optimoa. Beti N balio ditu eta [z1,z2,…,zn] itxura.
- SOptTxKop: SOpt-ek duen txanpon kopurua; BATUKARIA(j=1..i, SOpt(j)). Hasierako balioa ∞
- Lkop: Txanponekin osatu/itzuli behar den kopurua

- **Kimak**: 1 eta 2 ohikoak
  - Kima1: nahiz nahi adina txanpon eduki txanpona klase bakoitzeko, soilik sartzen direnak erabili behar dira zuhaitza hedatzeko.
  - Kima2: eraikitzen ari garen soluzio partzialak, soluzio optimoak une horretan duen kopurua ezin hobe badezake, ez da jarraitu behar hedatzen.
  - Kima 3: Deskonposezina (tartean):
    - T. merkea, inplementagarria,

      MIN(1..N) behin osatu ondoren, non MIN(K)= $\min_{\mathbf{k}\leq j\leq n}\{B(j)\}$

      denbora konstantean (L-SPB) <MIN(K)
    - T. garestia: adib. 116 eta (5,10,20,50,100) klaseak, deskonposa ezina detekzioa

Txanponak (TKM), 01 zuhaitza

- Lehenengo deia: `TKM_Bt_01 (1, 0, 0)`

```
Procedure TKM_Bt_01 (i,SPB,SPTxKop)
begin
if (L-SPB)=0 then if SPTxKop < SOPTTxKop
                  then SOPTTxKop:= SPTxKop;
                       SP(i..n):=ZerozOsatu; SOPT:= SP;
                  end if;

elsif i=n then if (L-SPB) mod b(n) = 0
               then  Azkenak:= (L-SPB) div b(n);
                     if SPTxKop+Azkenak < SOPTTxKop
                     then  SOPTTxKop:= SPTxKop+Azkenak;
                     SP(n):= Azkenak; SOPT:= SP;
                     end if;
               endif;   -- {b(n)-z deskonposaezina
```

```
else   -- (L-SPB) ≠0  eta i≠n
    if (L-SPB) > MIN(A)^k then
        -- deskonposagarria izan liteke oraindik
        for  A  in ((L-SPB) div b(i))^k..0  loop
           if  SPTxKop+A< SOPTTxKop^k
           then  SP(i):= A;
                  TKM_Bt_01 (i+1, SPB+A*b(i), SPTxKop+ A);
           endif;
         end loop;
          end if;
end if;
end;
```

Txanponak (TKM), iN zuhaitza,

```
procedure TKM_Bt_iN (i, SP, SPB,SPTxKop)
begin
if (L-SPB)k =0 then if SPTxKop < SOPTTxKop
                    then SOPTTxKop:= SPTxKop;
                         KopiatuM(SP,SP);
                    end if;

elsif i=n then if (L-SPB) mod b(n) = 0
               then  Azkenak:= (L-SPB) div b(n);
                     if SPTxKop+Azkenak < SOPTTxKop
                     then  SOPTTxKop:= SPTxKop+Azkenak;
                           SOPT:= SP;
                           KopiatuM(SP,SP);
                           GehituAldiz(SOPT, b(n),Azkenak);
                     end if;
                -- {b(n)-z deskonposaezina
               endif;
```

```
else   -- (L-SPB) ≠0  eta i≠n
    if (L-SPB) > MIN(A)^k then
        -- deskonposagarria izan liteke oraindik
        for  A  in i..n  loop
           if  SPTxKop+1< SOPTTxKop^k and then SPB+b(A)≤L
           then TKM_Bt_iN(i+1, SP ∪{A}, SPB+b(A), SPTxKop+1);
           endif;
         end loop;
          end if;
   end if;
end;
```

- Lehenengo deia: `TKM_Bt_iN (1,`*`Hutsa`*`,0,0)eta SOPTTxKop = ∞`
- Zuhaitzak: adarketa i..N
- 20+10 eta 10+20 konbinazio bera kontsideratzen da

- Beste aukera bat: txanpon balioak ordenaturik aztertu: b(1)>...>b(n)

Konbinazioak kopuru bat itzultzeko (nahi adina)

- Datuak:
  - Lkop: Itzuli behar den kopurua
  - N txanpon klase, i txanpon klase bakoitzeko:
    - B(I) da beren balioa,
    - Nahi adina txanpon,
    - [Murriztapenekin, muga(i)=ki i klaseko eskuragarri ditugun txaponak]
- Helburua:
  - Lkop kopurua zehazki itzultzeko dauden konbinazio desberdinen kopurua

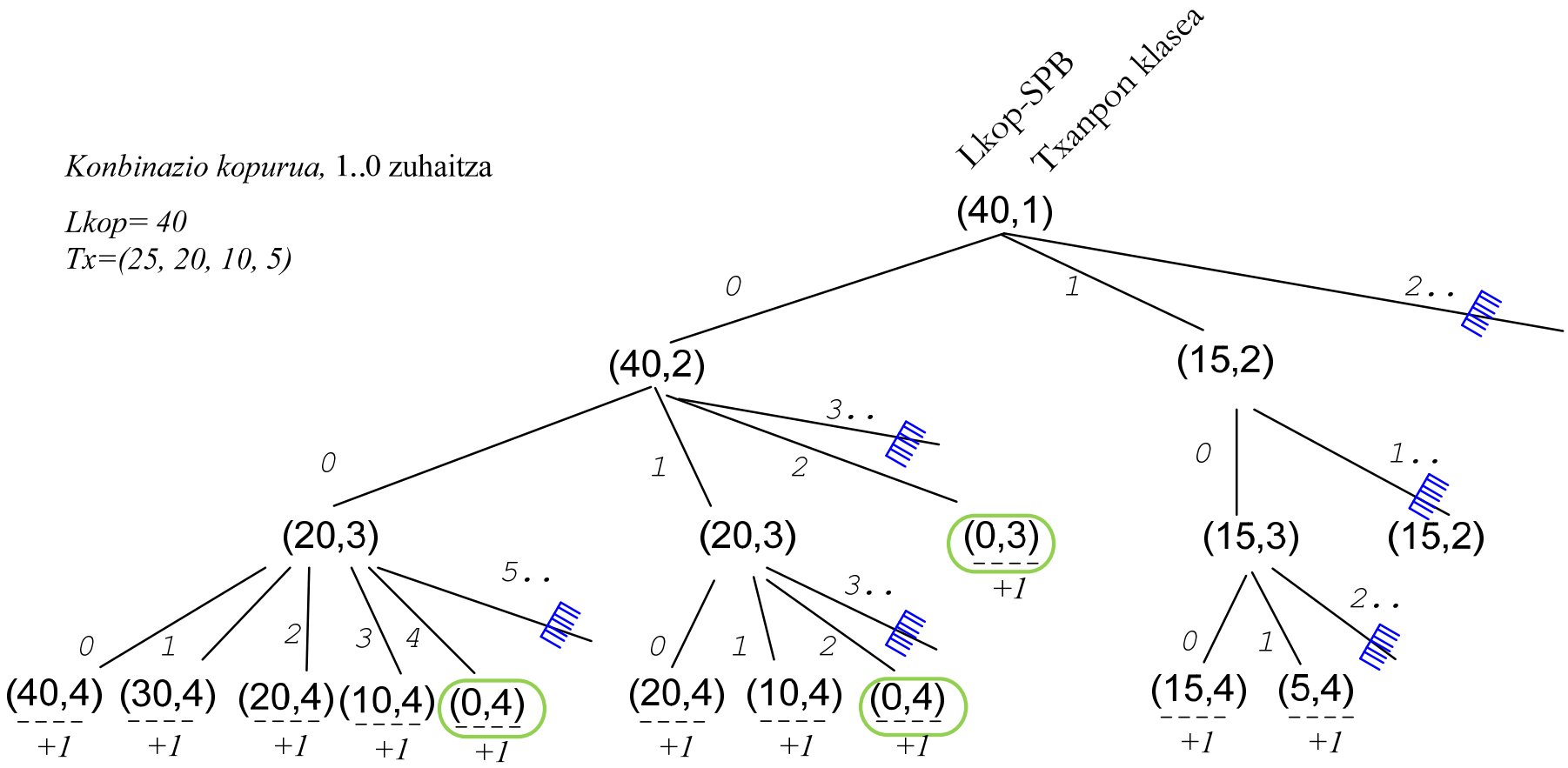
---

- Ez da optimizazio problema

- Ez dugu kalkulatuko zeintzuk diren konbinazioak, soilik zenbat dauden
- SP fiktizioa izango da
- 20+50 eta 50+20 konbinazio bakarra da

Konbinazio kopurua, 1..0 zuhaitza

Lkop= 40
Tx=(25, 20, 10, 5)



K1: adarkaketaren kontrola, nahiz nahi adina eduki soilik kabitzen direnak

SPB=Lkop kasu nabaria da, berdin du txanpon klase guztiak tratu diren hala ez

K2: Lkop-SPB<min(i) kasurik ez dago

- Zuhaitzaren maila bakoitzak txanpon klase bat tratatzen du: 0 aldiz, 1, 2,… klase bera

- Aldagaiak
  - Aldagai globalaK
    - Konbi: Dauden konbinazio kopurua jasotzeko.
    - Lkop: Itzuli behar den kopurua

  - Parametroak:
    - i: zenbat garren txanpon klasea tratatu behar den
    - SPB: orain arteko 1..(i-1) txanpon klaseek batu duten kopurua
    - (SP fiktizioa, SP=[z1,...,zi,...,zn], non osagai bakoitza zi $\in${0,1,2,...} $\equiv$ i klasekorik {Ez daramagu, bat, bi,..})

- Kasu nabariak
  - **LKop=SPB**
  - n txanpon klasea tratatzerakoan aztertu kasua: **(L-SPB) mod b(n)= ?**
    - Eta anizkoitza bada, konbinazio bat gehiago; bestela 0

- Komeni da orain arte eraikita daramagun SP=[z1,...,zi] soluzio partzialeko hainbat datu ezagutzea:
  - SP-k duen osagai kopurua: i (hots, tratatutako txanpon klaseak)
  - $SPB = \sum_{k=1}^{i} Balioa(k) \times Z_k$ non k klaseko Zk=SP(K) txanpon daramazkigun

- **Kimak**: 1 eta 2 ohikoak

  - Kima1: i txanpon klase bakoitzeko nahiz nahi adina eduki, soilik erabilgarriak direnak erabili, besteak ez: hau da [((L-SPB) div balioa(i))+1 ..) aukerak ez

  - Kima2: Deskonposezina. Ez genuke hedatzen jarraitu behar, eraikitzen ari garen konbinazioak ezin duenean L jatorrizko kopurua eman (hots, goazen moduan deskonposa ezina denean)

    - T. merkea, inplementagarria,

      MIN(1..N) behin osatu ondoren, non MIN(K)= $\min_{\mathbf{k}\leq j\leq n}\{B(j)\}$

      denbora konstantean (L-SPB) <MIN(K)

    - T. garestia: adib. 116 eta (5,10,20,50,100) klaseak, deskonposa ezina detekzioa

- 1 deia: `TKonbinazio_Bt_01 (1,0)`

```
prozedura   TKonbinazio_Bt_01 (i,SPB)
if SPB=L then Konbi:= Konbi+1;

else if i=n  then if ((L-SPB) mod b(n))=0
                  then Konbi:= Konbi+1 else Konbi:= Konbi+0;
                  end if;

else if (L-SPB) >= MIN(i)k2                    -- Kima2 merkea
     then for A in 0.. (L-SPB) div Balioa(i)) k1   loop    -- Kima1
               TKonbinazio_Bt_01 (i+1, SPB+ A * b(i))
          end loop;
     end if;
end TKonbinazio_Bt_01;
```

- 1 deia: TKonbinazio_Bt_1N (1,0)

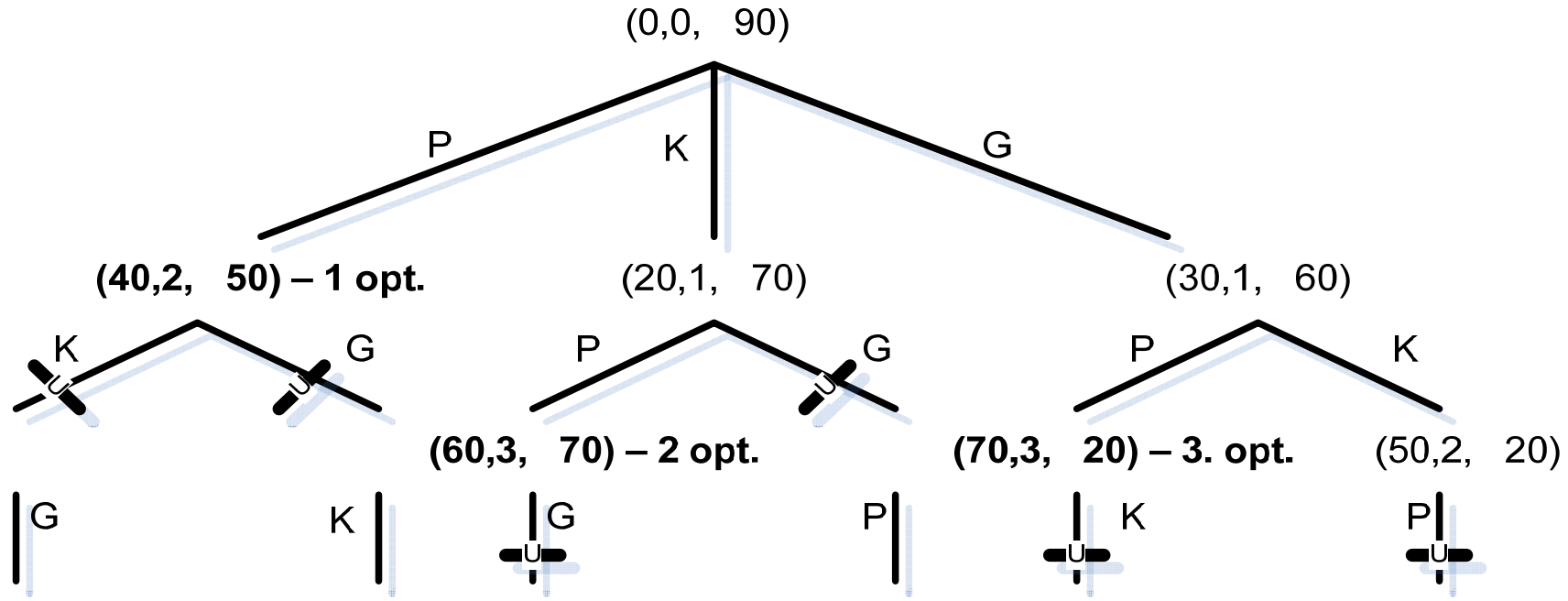
```
prozedura   TKonbinazio_Bt_1N  (i,SPB)
if SPB=L then Konbi:= Konbi+1;
else if i=n
     then if ((L-SPB) mod b(n))=0
          then Konbi:= Konbi+1 else Konbi:= Konbi+0;
           end if;

else if (L-SPB) >= MIN(i)k2                    -- Kima2 merkea
     then for A in 1.. n  loop
           if SPB+b(A) ≤ L
            then TKonbinazio_Bt_1N (A, SPB+ b(A))
            end if;
         end loop;
     end if;
end TKonbinazio_Bt_1N;
```

Pernandoren baratzak

Pernando Abarkazaharrek *n* baratz ditu, bakoitzak fruitarbola mota bat duena. Fruituak heldu dira eta uzta garaia da. Pernandok badaki zenbateko etekina ekoitziko dion baratz bakoitzeko uztaren bilketak; hots, *i* baratzerako ETEKINA(i) euro. Bestalde, baratz bakoitzeko uzta bilketak egun kopuru desberdin bat behar duen arren, kopuruak Pernandok ezagutzen ditu; honela, DENBORA(i) egun behar du *i* baratzeko uzta bilketak. Azkenik, Pernandok ere ezagutzen du baratz bakoitzeko fruituak zenbat egun tardatzen duten usteltzen; hots, *i* baratzerako USTELDU(i) egunaren ondoren ustelduko dira. Bestalde, traktorea eraman behar dela eta, baratzak osorik biltzen dira edo ez dira biltzen, eta biltzen direnean, fruitu onak soilik biltzen dira, hauek ustelduekin nahastuko balira uzta osoa galduko bailitzateke.

- Datuak:
  - N baratz
  - i baratzeko fruituak
    - ETEKINA(i) euro etekin ematen du
    - DENBORA(I) egun behar dira uzta osoa jasotzeko
    - USTELDU(I) egunaren ondoren usteltzen hasten dira

Fruitarbola	Etekina(i)	Usteldu(i)	Denbora(i)
Pikuak	40	3	2
Klaudiak	20	2	1
gereziak	30	1	1



- **Aldagai** globalak izango dira:
  - Ematen zaizkigun datuak: ETEKINA(i),USTELDU(i),DENBORA(i) (i fruitarbola,1..n)
  - SOPT: fruitarbolen jasotze antolaketa optimoa
  - SOPTET: SOPT-ek ematen duen etekina

- Dei errekurtsiboko parametro izango dira:

  BTPernando (SP,SPET,SPEG,KanpokoenET, AztertzekoZuhaitzak)
  - SP: eraikitzen dugun soluzio partziala: Fruitarbolen jasotze ordena baten proposamena: (X1,X2,...Xk), non k<=n; lehendabizi X1 fruitarbolak bildu, ondoren X2,…
  - SPET: SPko zerrendako antolamendua burutzeagatik lortuko den etekina
  - SPEG: SPko zerrendako antolamendua burutzeko behar ditugun egun kopurua (zuhaitzak ordena horretan jasotzeko). Hasieran 0 da, eta zuhaitzean zenbat garrengo egunean gauden adieraziko du.
  - KanpokenET: Oraindik aztertu ez diren zuhaitzen jasotzeak emango lukeen etekina.
  - AztertzekoZuhaitzak: Aztertzeko gelditzen diren fruitarbolen zerrenda

- **Kimak**
  - Kima 1: Zuhaitz mota bat biltzen hasten garenean, eta jakinik bilketak hainbat egun behar dituela, hura bukatu aurretik fruituak ezin zaizkigu usteldu. Hots, uztan garaian ustelduta baleude edo ustelduko balira, ez bildu:

    `SPEG+DENBORA(i) > USTELDU(i)`

  - Kima 2: SP-ko irabaziari (hots, SPETri) oraindik tratatu gabeko zuhaitzek emango luketen etekina batuz, soluzio optimoaren etekina hobetzen ez bada, ez genuke jarraitu beharko:

    `SPET+KanpokoenET< SOPTET`

- Kasu Nabaria: Zuhaitz mota gehiago aztergai ez ditugunean.

- *Lehenengo deia:*
  - BackTrackPernando ([],0,0,Batukari(i=1..n, ETEKINA(i)),[1..n]))

```
procedure BTPernando (SP, SPET, SPEG, KanpokoenET,
                      AztertzekoZuhaitzak)
begin
  if hutsaDaZ(AztertzekoZuhaitzak)
  then if SOPTET<SPET then SOPTET:= SPET; SOPT:=SP; end if;
  else
   while not(HutsaDaZ(AztertzekoZuhaitzak) loop
     Zu:= Lehenengoa (AztertzekoZuhaitzak);
     Hondarra(AztertzekoZuhaitzak);
     if SPEG+DENBORA(Zu) <= USTELDU(Zu)k1
              and  SPET+KanpokoenET> SOPTET
     then  BTPernando (GehituAtzetik(SP,(Zu,SPEG)),
                       SPET+ETEKINA(Zu), SPEG+DENBORA(Zu),
                       KanpokoenET- ETEKINA(Zu),
                       KenduAgerpena(AztertzekoZuhaitzak,Zu));
     else if SOPTET<SPET then SOPTET:= SPET; SOPT:=SP; end if;
     end if;
   end loop;
  end if;
end.
```