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)
  - 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?

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

✍ Alg1 Konp2 makinan urtebetez utziz, n handiena:

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

$$n=44,8421;$$

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;        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) ≤ **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

⇒ gutxienez itxaron beharko dugun denbora

Kasu txarrena: $T_t(n)$

n tamainako sarrera guztien artetik denbora maximoa behar duen haren denbora

⇒ erantzun denbora kritikoa denean

Batez besteko kasua: $T_b(n)$

n tamainako sarrera guztien exekuzio-denboren batez besteko denbora

⇒ exekuzio asko sarrera oso desberdinekin

✍ Txanpon faltsua mugatzen

$T(n)$= pisatze kopurua

alg1

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

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

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

alg2

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

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

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

alg3 suposatuz: $n=2^k$

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

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

$T_b(32)=5$

✍ Bilaketa $T(n)$= konparazio kop.

ez-ord

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

$T_b(n)=p/n\ \mathbf{(}1+2+\ ...+\ n)+(1-p)\ n$

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

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

Ord+lineala

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

$T_b(n)=p/n\ \mathbf{(}1+2+\ ...+\ n)+(1-p)/(n+1)\ (1+...+n+n)$

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

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)

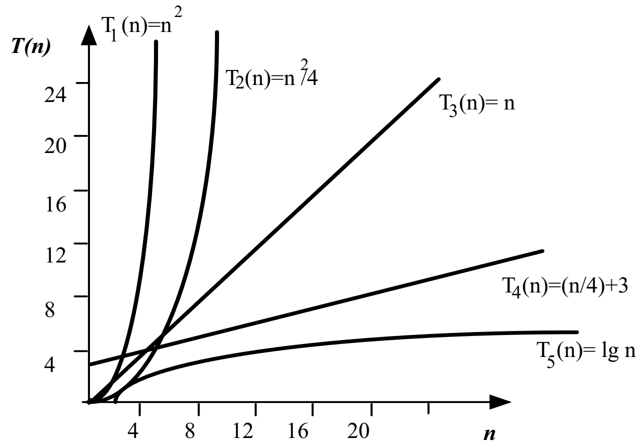
- Konstante biderkatzaileak funtzioek ordena berdina dutenean bakarrik kontuan hartuko ditugu: inplementazio txikikerietan sartzea ekiditen du

- 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 $\rightarrow$ 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)

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

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

Omega larria : Ω (gutxienez edo berdin)

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

- Ω(f(n)) irakurriko *f(n) omega ordena*

Teta larria: Θ (zehazki)

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

- Θ(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 $\Rightarrow 2^n \le d\,(3^n)$

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

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

- Edo limiteak erabiliz:

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

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

- $\frac{1}{2} n^2+4n+5 \in \Theta(n^2)$?   *Bai*

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

  $\Rightarrow c\, n^2 \le \frac{1}{2} n^2+4n+5 \le d\, n^2$

  $\boxed{\Theta(\frac{1}{2} n^2+4n+5) = \Theta(n^2)}$

- L'Hopital

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

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

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

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

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

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

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

10 Algoritmo iteratiboen T(n)a eta ordena

- Erregela mnemoteknikoak erabiliz T(n)-a idatzi
  1. parametroek $\rightarrow$ sarreraren tamaina
  2. :=, >,<,<=,...,I/O,... $\in O(1)$
     Salbuespenak: array-ekin adi!
  3. agindu-sekuentziak: $T_1+\ldots+T_k$, 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 $\rightarrow$ $T_1(n) \in \Theta(f(n))$ P2 $\rightarrow$ $T_2(n) \in \Theta(g(n))$

  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 oinarrial 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 gaineko 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: $f, g : I\!N \to I\!R$

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. $\lim_{n\to\infty} \frac{f(n)}{g(n)} = K \in I\!R^+ \Rightarrow \Theta(f(n)) = \Theta(g(n))$ 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) $f^{\star\prime}(x)$ eta $g^{\star\prime}(x)$(funtzio hedatuak deribagarriak badira) eta
   (d) $\exists \lim_{x\to\infty} \frac{f^{\star\prime}(x)}{g^{\star\prime}(x)}$

orduan

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

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

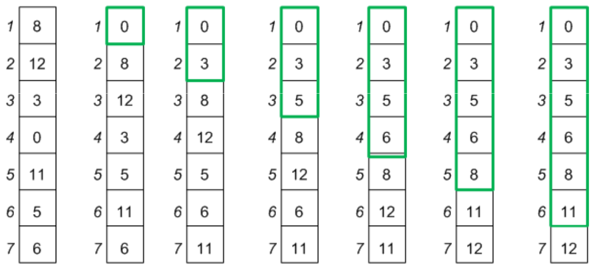
```
IND := 1;

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;
```

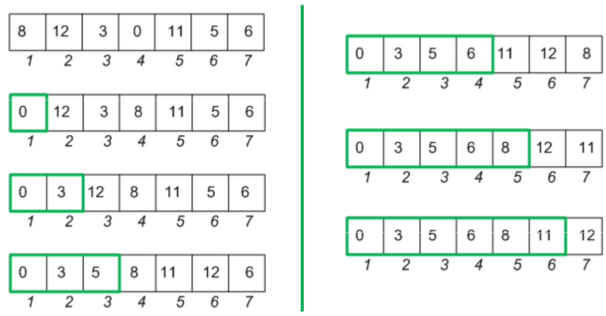
```
begin          --  ∀i (1≤i≤ n  → 1 ≤ T(i) ≤ N)
(1)for  I  in T'RANGE  loop
(2)   if T(I) > 0
(3)   then B := T(I); K := 0;
(4)         while B> 0 loop
(5)            if (B rem 10 = X)  then K := K + 1;  end if;
(6)            B := B/10;
            end loop;
(7)         TEXT_IO.PUT(INTEGER'IMAGE'T(I)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



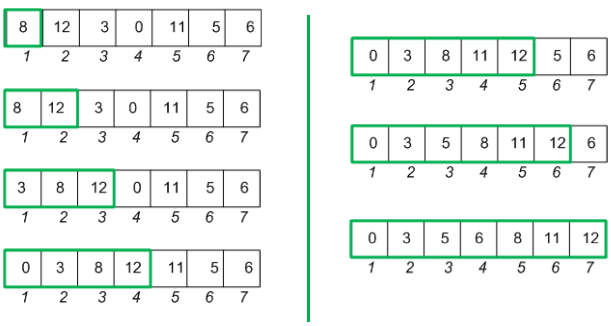
$$T(n)=\sum_{i=1}^{n}(n-i)=\frac{0+(n-1)}{2}\cdot n=\frac{n^2-n}{2}\in\Theta\left(\frac{1}{2}n^2\right)$$


```
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;

$$
\begin{aligned}
T_e(n) &= a+1\,T_e(n-1) \approx 1+T_e(n-1)\\
&= j+T_e(n-j)\\
&= (n-1)+T_e(1) \in \Theta(n)
\end{aligned}
$$

12 Errekurtsio ekuazioen ebazpen metodoak

Hedapena

Hanoi

$$
\begin{aligned}
T(0) &= b\\
T(n) &= 2\,T(n-1)+a\\
&\approx 2\,T(n-1)+1\\
&= 2\,(2T(n-2)+1)+1=2^2\,T(n-2)+2+1\\
&= 2^2\,(2T(n-3)+1)+2^1+2^0=2^3T(n-3)+2^2+2^1+2^0\\
&= 2^iT(n-i)+2^{i-1}+\ldots+2^1+2^0 && \text{i pausoan}\\
&= 2^nT(0)+2^{n-1}+\ldots+2^1+2^0 && \text{n-i=0}\\
&= 2^n b+\sum_{j=0}^{i-1}2^j = b2^n+2^i-1=(b-1)2^n-1 \in \Theta\left(2^n\right)
\end{aligned}
$$

$$
\begin{aligned}
T(1) &= a \approx 1\\
T(n) &= T(n-1)+3n+7\\
&\approx T(n-1)+n\\
&= T(n-2)+(n-1)+n\\
&= T(n-i)+(i-1)+\ldots+(n-1)+n && \text{i pausoan}\\
&= T(1)+2+\ldots+(n-1)+n && \text{n-i=1}\\
&= 1+2+\ldots+(n-1)+n \in \Theta(n^2)
\end{aligned}
$$

Ek. Karakteristikoa

Prozesua:

- T(f(n)) gaiak ezkerraldean ordenean
  $a_0\,t_n + a_1\,t_{n-1} + \ldots + a_k\,t_{n-k}$
- Eskuin aldean identifikatu
  $(b_1)^n\,p_1(n) + (b_2)^n\,p_2(n) + \ldots$ non $b_2 \neq b_2 \neq \ldots$
  eta d_i $p_i(n)$-ren gradua den
- Jaso orain arteko kalkuluak eredua jarraituz:
  $(a_0\,x^k + a_1\,x^{k-1} + \ldots + a_k)\,(x-b_1)^{d1+1}\,(x-b_2)^{d2+1}\ldots = 0$
- 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)

Hanoi

$$
\begin{aligned}
T(0) &= b\\
T(n) &= T(n-1)+T(n-2)+a\\
&\approx 2\,T(n-1)+1\\
&= 2\,(2T(n-2)+1)+1=2^2\,T(n-2)+2+1\\
&= 2^2\,(2T(n-3)+1)+2^1+2^0=2^3T(n-3)+2^2+2^1+2^0\\
&= 2^iT(n-i)+2^{i-1}+\ldots+2^1+2^0 && \text{i pausoan}\\
&= 2^nT(0)+2^{n-1}+\ldots+2^1+2^0\\
&= \ldots && \text{n-i=0}
\end{aligned}
$$

$$
\begin{aligned}
T(0) &= b\\
T(n) &= T(n-1)+T(n-2)+a
\end{aligned}
$$

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

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

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

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

Aldagai aldaketa

Bilaketa dikotomikoa

$\boxed{\mathbf{T(n) = T(n/2) + a}}$

$n=2^k$

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

$b^k \times p(k)$

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

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

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

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

$t_k = c_1 + c_2\,k$

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

$k = \lg n$

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

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

$n=4^k$

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

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

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

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

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

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

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

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

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

$n=2^k$

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

$\lg n = k$

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

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

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

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