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

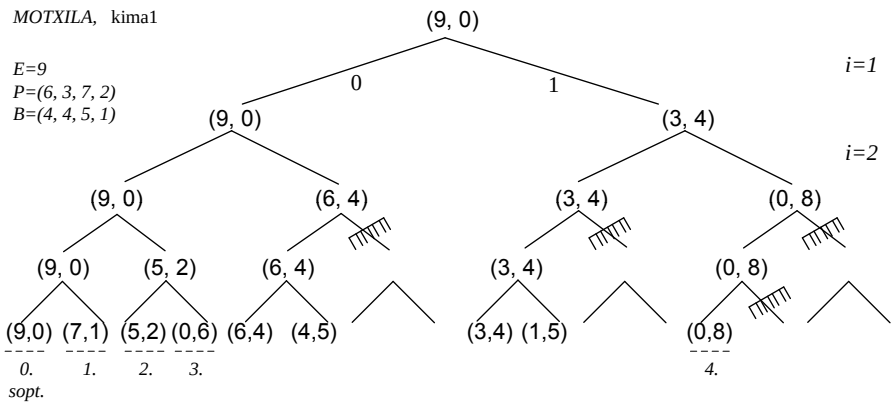
Txantiloi orokorra (1 soluzioa lortzeko)

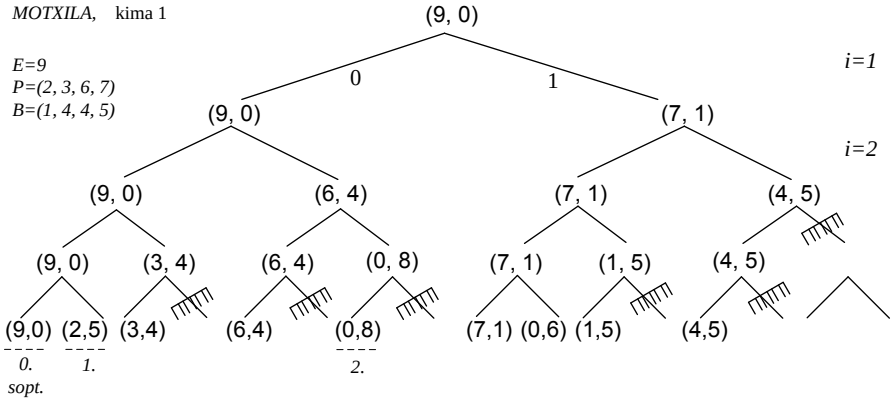
```
prozedura   saiatu   (SAIAKUNTZA, ARRAKASTA)
baldin SoluzioaDa(SAIAKUNTZA)
orduan   "Kudeatu SAIAKUNTZA saiakera"
         ARRAKASTA:=True
bestela
  errepikatu  SAIAKUNTZA hedatzeko A aukera guztiekin  egin
      baldin Onargarria (SAIAKUNTZA+A)
      orduan  saiatu (SAIAKUNTZA+A, ARRAKASTA)
   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





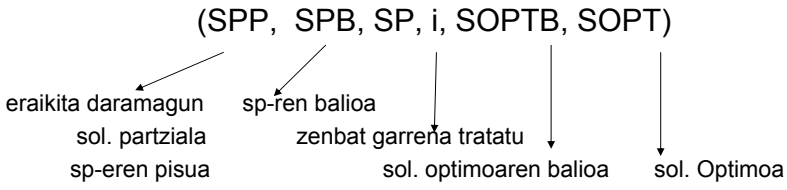
Motxila 0/1

- 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, $p(1)+p(2)+\ldots+p(n)<=E$, eta eraikiko dituen soluzio partzial kopurua (egingo dituen saiakuntza kopurua) da:

$$1+2^1+2^2+\ldots+2^n=2^{n+1}-1$$

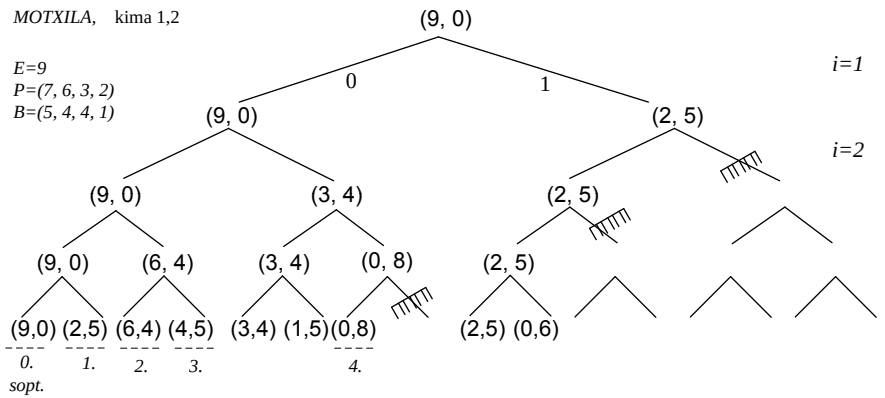
baitakigu:
  - hasierako dei bat dugula
  - ondorengoetan, beti aukera bina dugula: bi saiakuntzekin hedagarria.
  - edozein soluzio-saiakuntza luzeenaren luzera n da
- 2^n hosto (Kasu nabari) leudeke eta haien tratamendua $\Theta(n)$ da.
- Algoritmoaren denbora-ordena $O(\mathbf{n2^n})$ (zehatza, kimarik 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

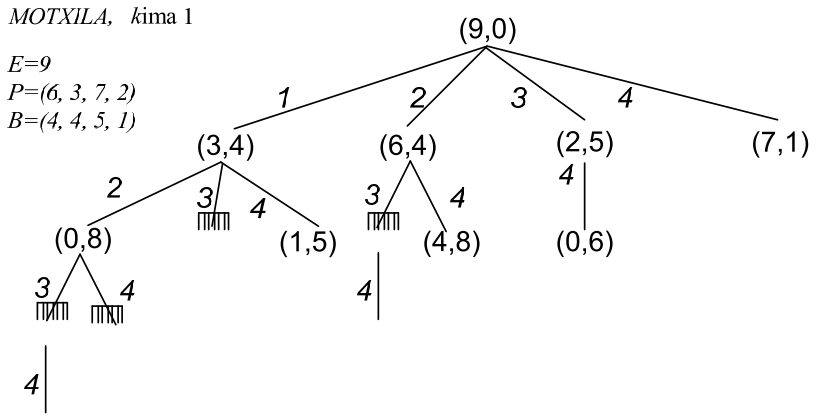
- Dagoeneko asmatu dira hainbat teknika estatistiko ESTIMATZEKO kimatzen duten algoritmo sakonak edo backtrack algoritmoek zenbat saiakuntza egingo lituzketen:
  - Monte Carlo
  - Las Vegas
- Aurten ez ditugu ikusiko
- Ondorioz, zuhaitzak gehienez edukiko lukeen tamaina soilik mugatuko dugu.
- Analisiaren 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, 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, 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∈{1,2,3,4} ≡ {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;
```

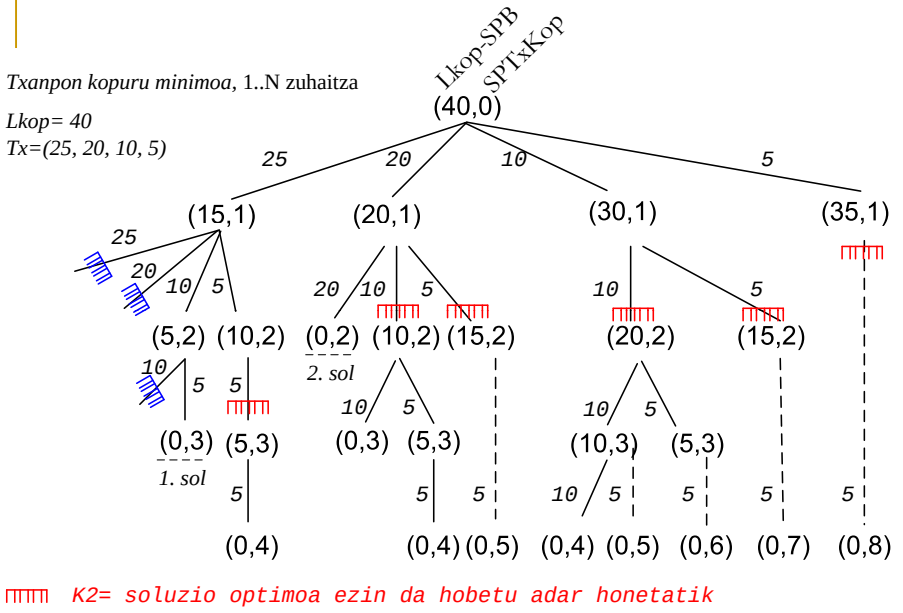

Mapak koloreztatzen. Analisia

- 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**(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



- 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)= $\min_{\mathbf{k} \le j \le n}\{B(j)\}$
      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

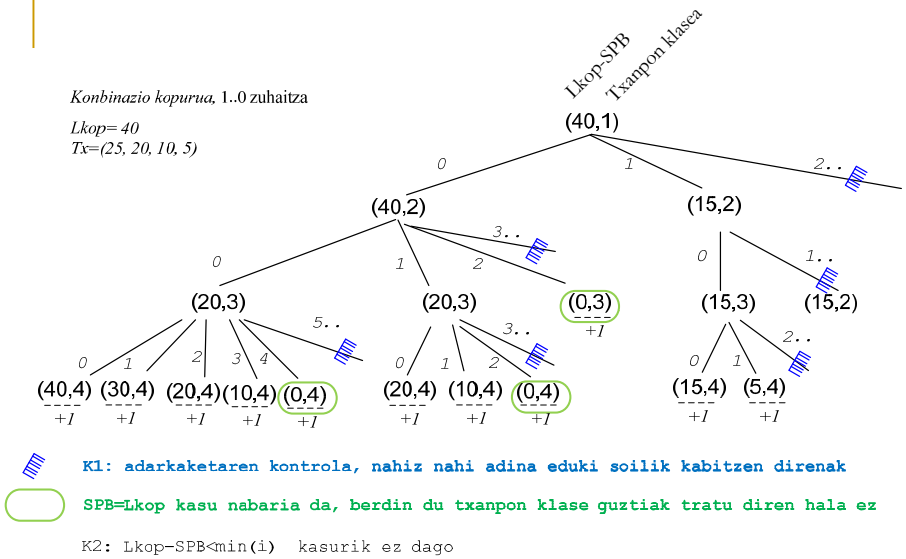
- Beste aukera bat: txanpon balioak ordenaturik aztertu: b(1)>...>b(n)

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



- 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,...} ≡ 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)= $\min_{k \le j \le n}\{B(j)\}$
      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)`

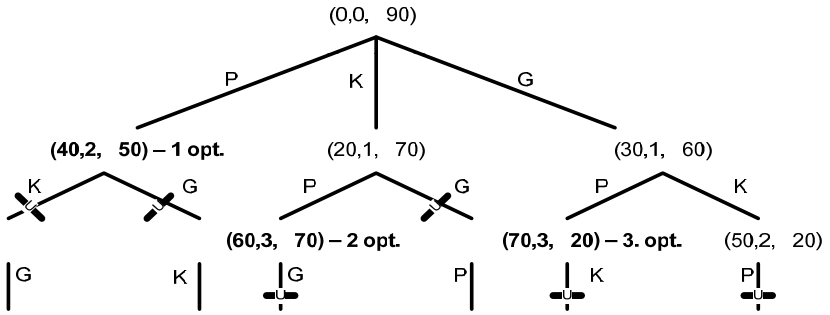
```
prozedura   TKonbinazio_Bt_1N  (i,SPB)
if SPB=L then Konbi:= Konbi+1;
else if i=n
     then if ((L-SPB) mod b(n))=0
          then Konbi:= Konbi+1 else Konbi:= Konbi+0;
          end if;

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 usteldutа baleude edo usteldуko 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.
```