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

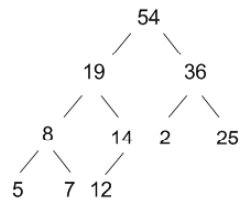
Metak

- Meta (montículo, heap):

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

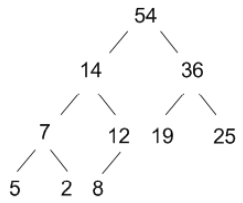
- Aplikazio ugari du. Ezagunenak
  - Willias-en HeapSort ordenazio algoritmoa
  - Lehentasun dinamikoa duten ilarak
  - *n* osagaietatik bakarrik *k* txikienak/handienak behar direnean
    - aplikazio bakoitzean *k* desberdina

- **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.

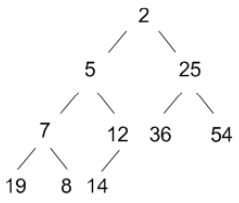


54	19	36	8	14	2	25	5	7	12

- **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.

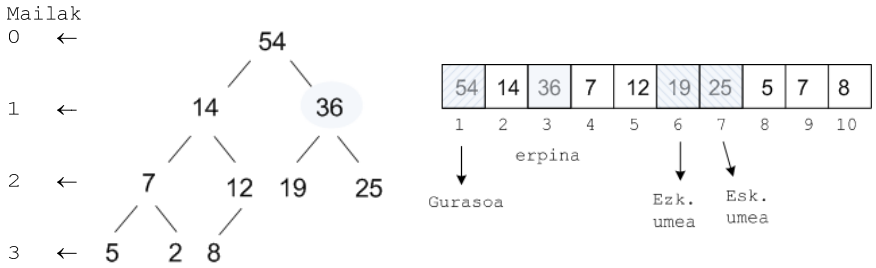


Maximoen meta



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

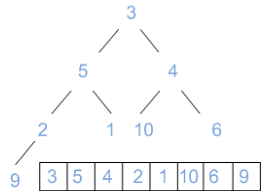


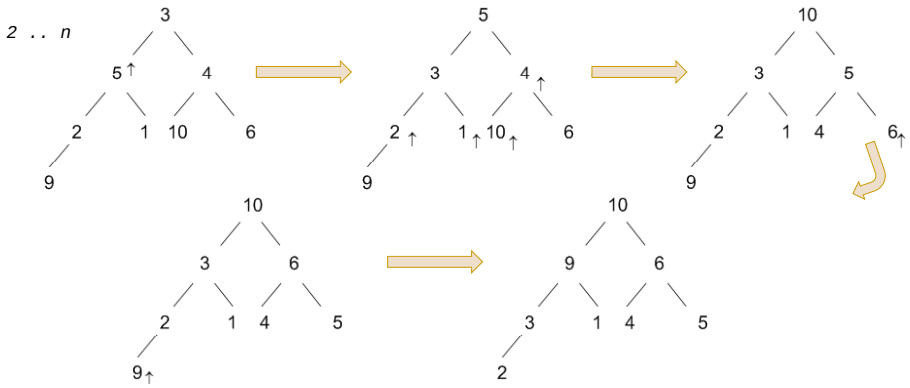
Metak

- 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)`

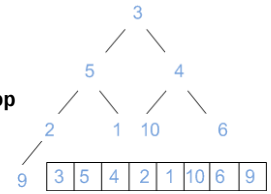

```
procedure  Azaleratu  (M: in out  OsSek; P: in Indizea;) is
begin
   J:= P;
   while  M'Frist < J  and then  S(J div 2) < S(J) loop
        Trukea(M(J), M(J div 2));   J:= J div 2;
   end loop;
end;


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;
```

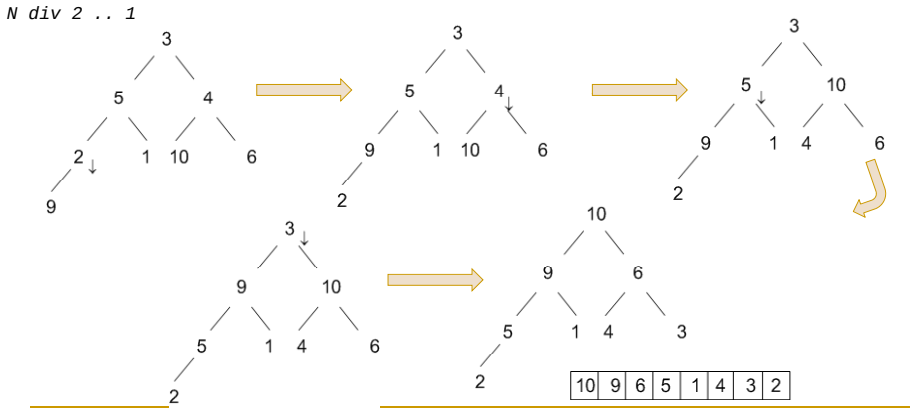
```
procedure  MetaEraiki1 (T:in out OsSek;) is
begin
   for  K in T'First+1..T'Last  loop
       Azaleratu(T(T'First,..,K),K);
        --   Azaleratu(T,K);
   end loop;
end MetaEraiki1;
```






```
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;
```





Meta eraiki. Analisia

- Azaleratuz

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

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

- 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.

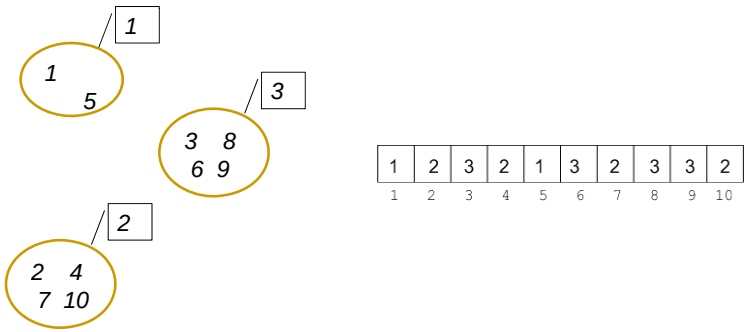
- Eragiketak egitura gain
  - Hasieratu: egitura balioz hornitu beharko da hasieran.
    - Normalki, osagai bakarreko multzo disjuntuetan
  - `Bilatu(I)`: I objektua partaidea den multzoko etiketa itzuliko du
  - `Bateratu(E1,E2)`: E1 eta E2 etiketa desberdinak izanik, bakoitzak identifikatzen duten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

1. hurbilketa

- Etiketa multzoko osagai txikiena da. Multzo guztiak taula gainean

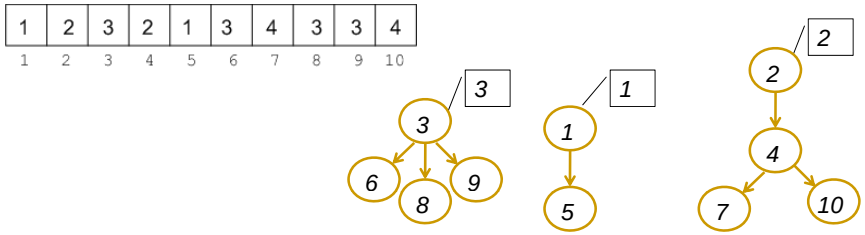

```
function  BILATU1 (PARTIKETA: in P_PARTIKETA_MOTA;
                   X: in OSAG_ETIK) return OSAG_ETIK is
begin
   return  PARTIKETA(X);
end BILATU1;
```

```
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 ;
```

- 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 ⇒ objektua bere multzoko etiketa da eta dagokion zuhaitzaren erroa
    - PARTIKETA(i) = j, eta i ≠j ⇒ 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:

Θ((n/2) * `Bateratu2`-ren ordena) + Θ((n/2) * `Bilatu2(n/2+1)`-ren ordena)=

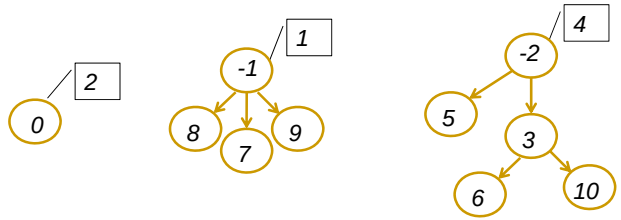
Θ((n/2)* Θ(1)) + Θ((n/2) * Θ(`n/2`)) =

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

- 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.
    - Iortzen 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 $\lfloor$**lg k**$\rfloor$ da ⟸ **LEMA**

-1	0	4	-2	4	3	1	1	1	3
1	2	3	4	5	6	7	8	9	10



- `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 $0 \le \lfloor \lg 1 \rfloor$
- Indukzio hipotesi hedatua: $\forall i\ (1 \le i < n \rightarrow s_i \le \lfloor \lg i \rfloor)$
  - 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 $s_{k1} \le \lfloor \lg k1 \rfloor$ eta $s_{k2} \le \lfloor \lg k2 \rfloor$ beteko da

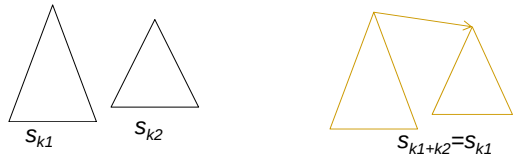
z1 eta z2 elkartzerakoan zuhaitz berriak k1+k2 adabegi izango ditu.

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

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

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

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

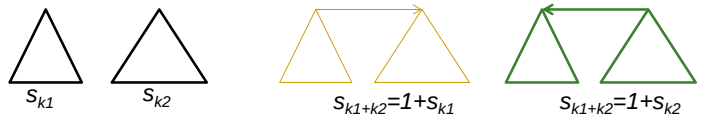


b) $s_{k2} = s_{k2}$.

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

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

- 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

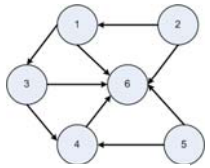
    (n/2+1). `Bilatu3(sakonerako hosto bat)`
    …
    n. `Bilatu3(sakonerako hosto bat)`

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

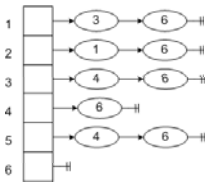
$$\Theta(n/2 * \Theta(1)) + \Theta((n/2) * \Theta(s_k)) \le \Theta(n) + \Theta(n) * \Theta(\lfloor \lg(1+n/2) \rfloor) = \mathbf{\Theta(n \lg n)}$$

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,…

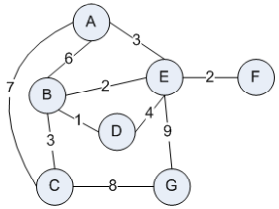


- Adierazpide erabilienak
  - Auzokideen zerrendak, auzokidetza matrizeak



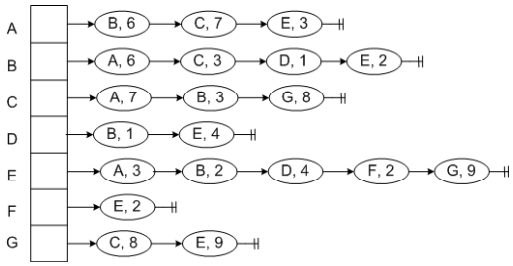
	1	2	3	4	5	6
1	0	0	1	0	0	1
2	1	0	0	0	0	1
3	0	0	0	1	0	1
4	0	0	0	0	0	1
5	0	0	0	1	0	1
6	0	0	0	0	0	0

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