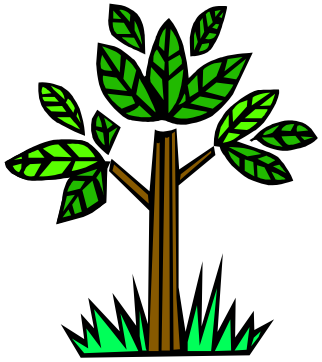
Datu egiturak

Zuhaitzak



Aurkibidea

- ***Zer dira?***
  - Zergatik dira garrantzitsuak
  - Ezaugarriak
  - Definizioak
    - Ez errekurtsiboa
    - Errekurtsiboa
  - Terminologia
- ***Inplementazio adibideak***
  - Ez errekurtsiboa
  - Errekurtsiboa
- ***Metodo nagusiak***
- ***Zuhaitz bat korritu***
  - Aurre-ordenean
  - In-ordenean
  - Post-ordenean
- ***Zuhaitz bereziak***
  - Bilaketa-zuhaitz bitarrak
  - AVL Zuhaitzak

Zuhaitzak

Zergatik dira garrantzitsuak? Helburuak

Gure helburua,

Elementu talde batean, oinarrizko eragiketak <u>denbora gutxienean</u> exekutatzea da:

- **Bilaketa**
- **Ezabaketa**
- **Txertaketa**

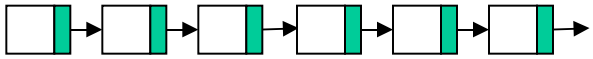
Zuhaitzak

Zergatik dira garrantzitsuak? Helburuak

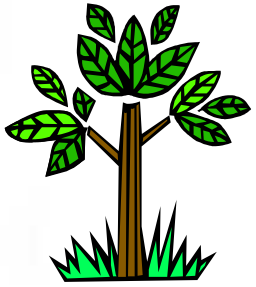
Arrayak (ordenatua)

Elementu baten atzipena
Posizio batean: O(1)
Eduki bidez : O(n)
Elementu baten txertaketa:O(n)

Zerrenda Estekatua(ordenatua)



Elementu baten atzipena :
Posizio batean : O(n)
Eduki bidez : O(n)
Elementu baten txertaketa :O(1)



ZUHAITZAK

Elementu baten bilaketa(eduki edo posizio) O(log n)

Elementu baten txertaketa O(log n)

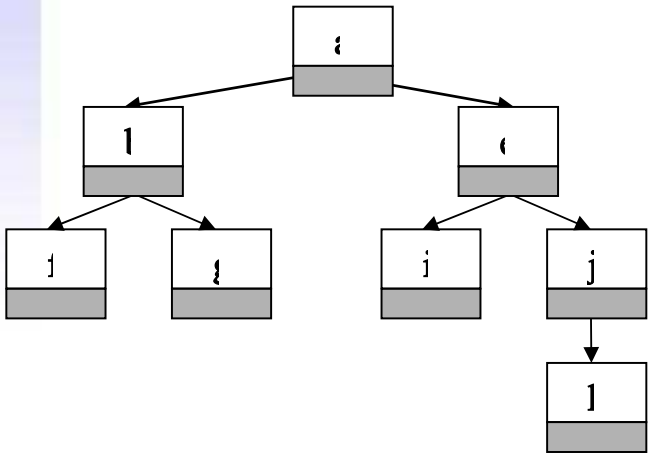
Funtzioen haziketa f(n)

log n	√n	n	n log n	n^2	n^3	2^n
1	1,4	2	2	4	8	4
2	2,0	4	8	16	64	16
3	2,8	8	24	64	512	256
4	4,0	16	64	256	4.096	65.536
5	5,7	32	160	1.024	32.768	4.294.967.296
6	8,0	64	384	4.096	262.144	$1,8 * 10^{19}$
7	11,0	128	896	16.536	2.097.152	$3,4 * 10^{38}$

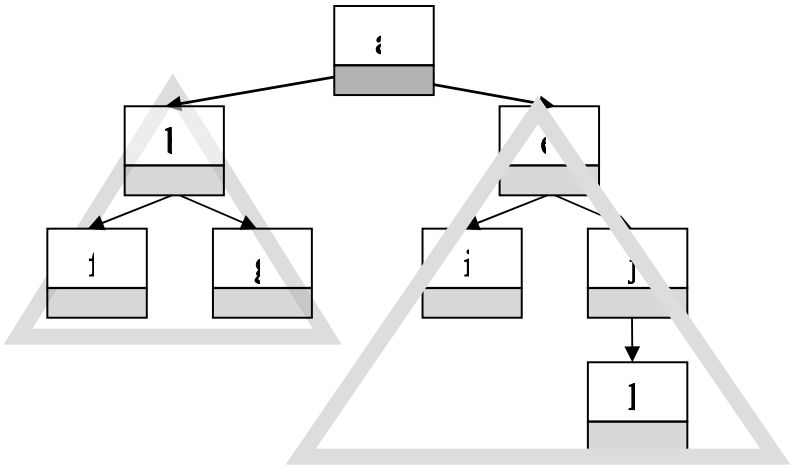
Zuhaitzak

Zer dira ? Ezaugarriak

- **Zuhaitz** bat elementuak **hierarkikoki** gordetzen dituen datu egitura **ez lineal** bat da
- Bi eratara definitu daiteke:



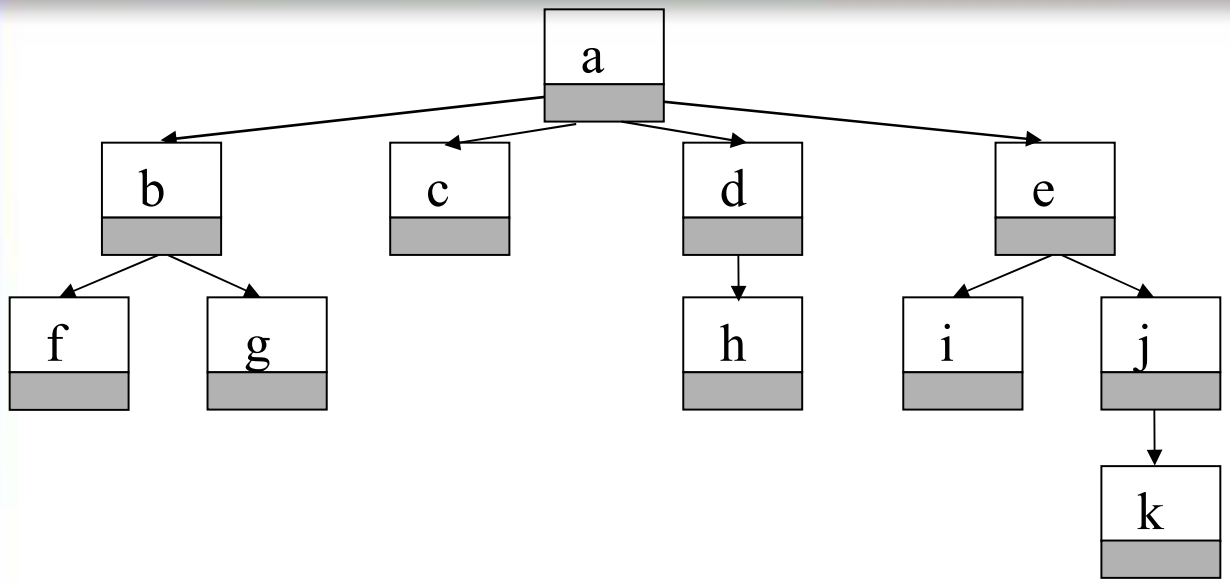
Definizio EZ Errekurtsiboa



Definizio Errekurtsiboa

Zuhaitzak

Definizio ez errekurtsiboa



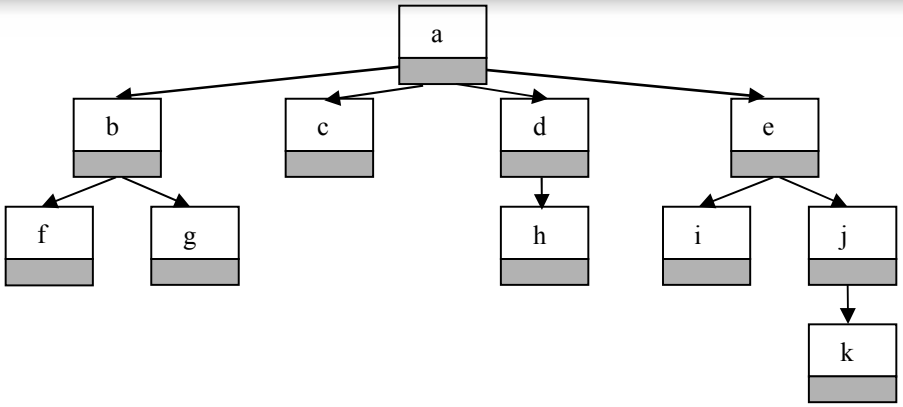
Zuhaitz bat osatzen dute:

Adabegi multzo bat

Zuzendutako arku talde bat. Arku bakoitzak bi adabegi konektatzen ditu **guraso-umea** erlazioaren bidez

Zuhaitzak

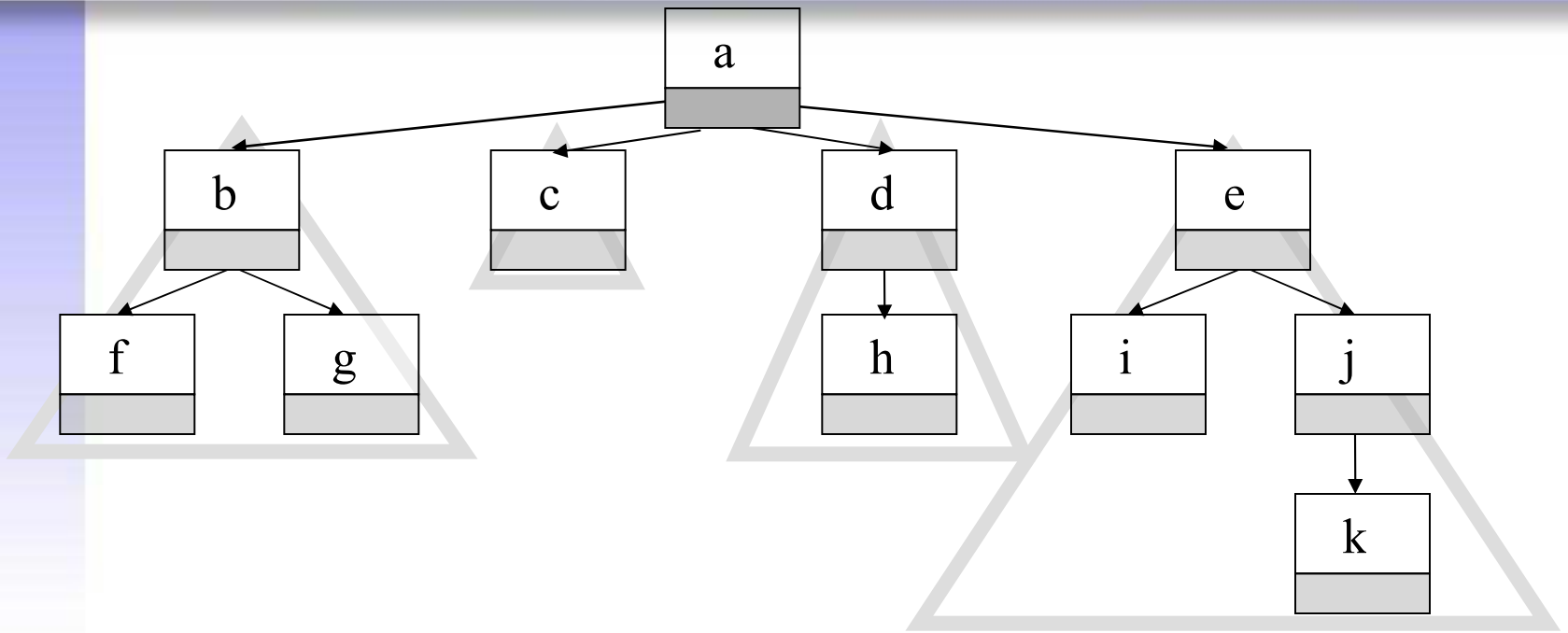
Definizio ez errekurtsiboa



- Zuhaitz batean adabegi “a” bakar bat bereizten da **erroa** bezala (ez dauka gurasorik)
- Adabegi “u” bakoitza (erroa izan ezik) **arku** baten bidez konektatua dago “g” adabegi bakar bateri. Esaten da, “g” adabegia “u” adabegiaren **gurasoa** dela eta “u” “g”-ren **umea** dela.
- Umeak ez dauzkaten adabegiei **hostoak** deritzaie

Zuhaitzak

Definizio errekurtsiboa

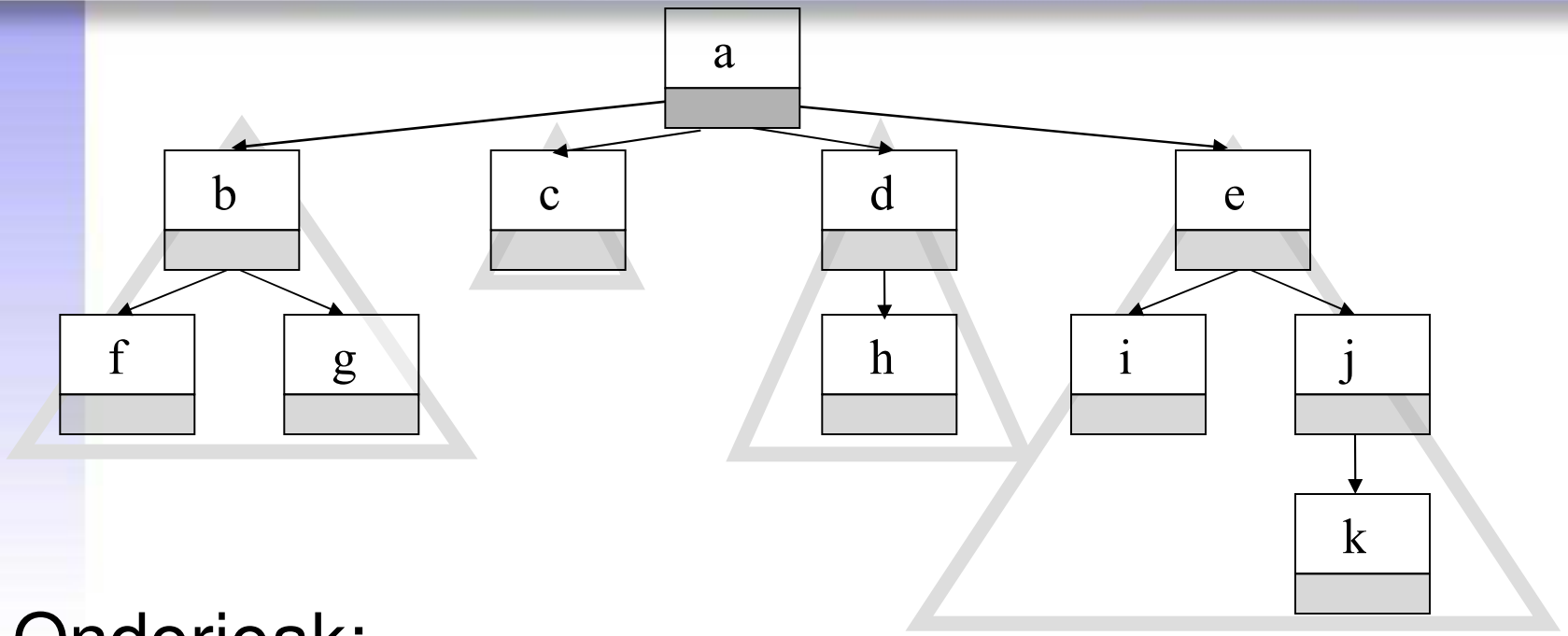


Z **zuhaitz** bat hutsa da edo osatzen dute:

- **Erro** bat
- Zero edo n **azpizuhaitz** ez hutsak Z1,Z2,Z3, etb. non beren erroak **arku** baten bidez konektatuta daude Z-ren errora.

Zuhaitzak

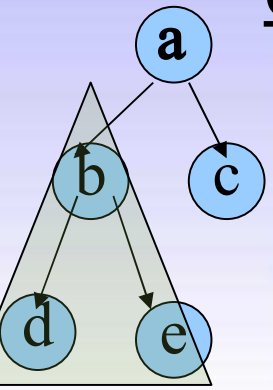
Definizio errekurtsiboa



- Ondorioak:
  - Zuhaitzaren adabegi bakoitzak azpizuhaitz bat definitzen du (erroa bezala duena)
  - Erlazio estu bat dago adabegi eta azpizuhaitz baten artean.

Zuhaitzak
Terminologia eta oinarrizko propietateak

- ***Aurrekoak eta ondorengoak*** (Def errekurtsiboa)

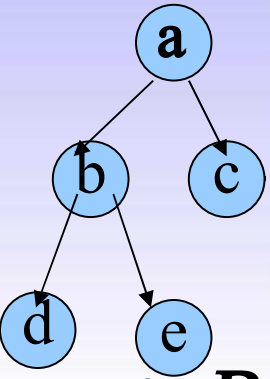


  - Adabegi baten **aurrekoak**, adabegi berbera eta bere gurasoaren aurreko guztiak dira.
  - ***Zuhaitz baten adabegi guztientzat erroa adabegia aurreko bat da.***
  - "a1" adabegia beste adabegi "a2" baten **ondorengoa** dela esaten da, "a2" adabegia "a1" adabegiaren **aurrekoa** bada.
  - Z zuhaitzaren **azpizuhaitz** bat "a" adabegian, zuhaitz bat da "a"-ren *ondorengo* adabegi guztiekin Z

Zuhaitzak

Terminologia eta oinarrizko propietateak

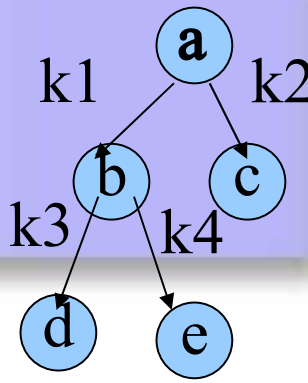
- ***Ahaidetasuna***
  - "g" adabegia "u" adabegiaren **gurasoa** bada, orduan "u" adabegia "g" adabegiaren **umea** da
  - Bi adabegi guraso berdinaren umeak badira, **anaiak** direla esaten da.



- ***Barne eta kanpo adabegiak***:
  - **Barne** adabegiak ume bat edo gehiago duten adabegiak dira
  - **Kanpo** adabegiak umeak ez dituzten adabegiak dira
  - Kanpo adabegiei **hostoak** deritzaie

Zuhaitzak

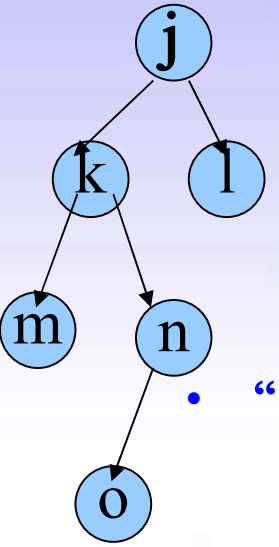
Terminologia eta oinarrizko propietateak



- **Bide bat**
  - Zuhaitzaren bi adabegi lotzen dituen arku sekuentzia da.
- **2 adabegien arteko bidearen *luzera***
  - Bi adabegien arteko arku kopurua
- **Adabegi baten tamaina**
  - Ondorengo adabegien kopurua (bera barne)
- **Zuhaitzaren tamaina**
  - Erro adabegiaren ondorengoen kopurua

Zuhaitzak

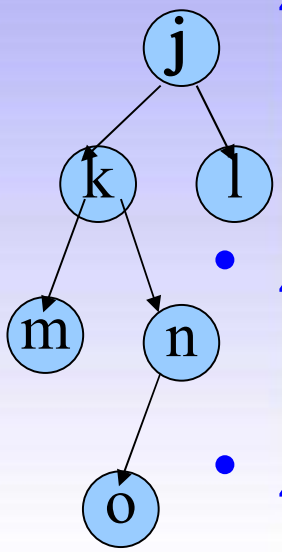
Terminologia eta oinarrizko propietateak



- **"a" adabegi baten sakonera (3 definizio)**
  - bidearen luzera errotik "a" adabegiraino
  - "a" adabegiaren ondorengoen kopurua (bera izan ezik)
  - Definizio errekurtsiboa
    - Erro adabegia bada: sakonera = 0 (kasu nabaria)
    - Beste kasuan: sakonera = 1 + gurasoa adabegiaren sakonera
- **"a" adabegi baten altuera (maila) (2 definizio)**
  - "a" adabegitik "hosto" sakoneraino doan bidearen luzera.
  - Definizio errekurtsiboa
    - Hosto bat bada (kanpo adabegia): Altuera = 0 (Kasu nabaria)
    - Beste kasuan: altuera = 1 + bere umeen altuera maximoa

Zuhaitzak

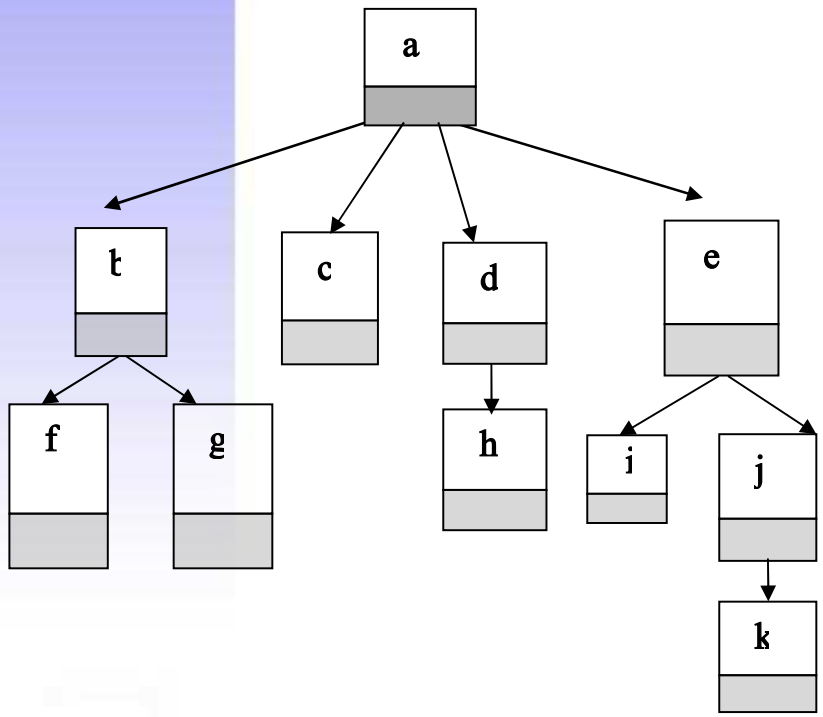
Terminologia eta oinarrizko propietateak



- **Zuhaitz baten sakonera**
  - Bere adabegi baten sakonera maximoa. Beti bere **kanpo adabegi** baten (hosto) sakonera izango da.
- **Zuhaitz baten altuera**
  - Bere erro adabegiaren **altuera** da
- **Zuhaitz ordenatuak**
  - Zuhaitz bat **ordenatua** dagoela esaten da adabegi guztien umeentzako **orden lineal** bat existitzen bada. Hau da, definituta dago, zein ume “bisitatzen” da lehendabizi, zein bigarrena, etb.

Zuhaitzak

Terminologia eta oinarrizko propietateak

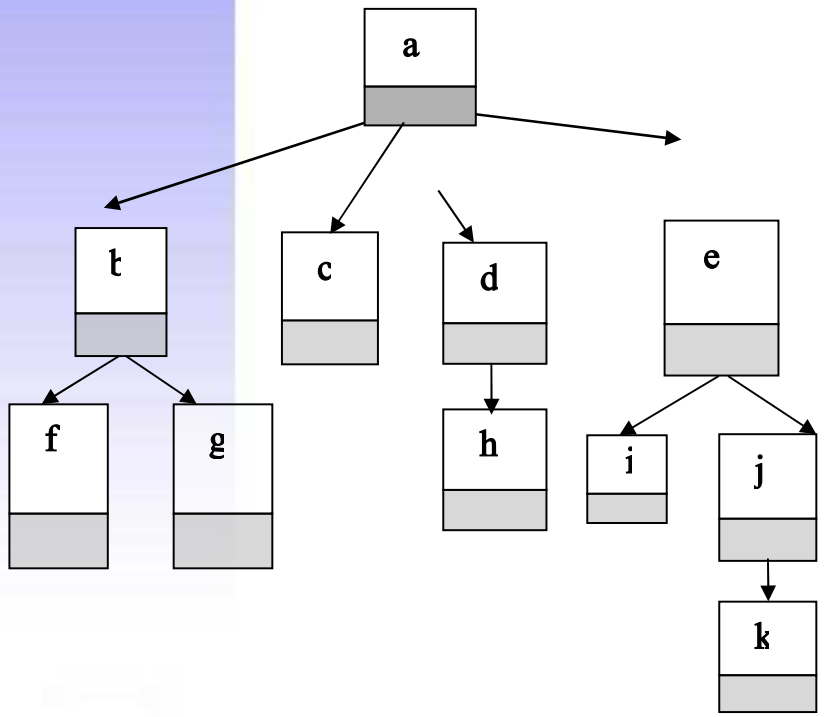


Adab	Altuera	sakonera	Tamaina	Barne Kanpo
a	3	0	11	Barne
b	1	1	3	Barne
c	0	1	1	Barne
d	1	1	2	Barne
e	1	1	4	Barne
f	0	1	1	Kanpo
g	0	2	1	Kanpo
h	0	2	1	Kanpo
i	0	2	1	Kanpo
j	1	3	2	Barne
k	0	3	1	Kanpo

Zuhaitzaten **tamaina**: 11
Zuhaitzaren **altuera**: 3
Zuhaitzaren **sakonera**: 3

Zuhaitzak

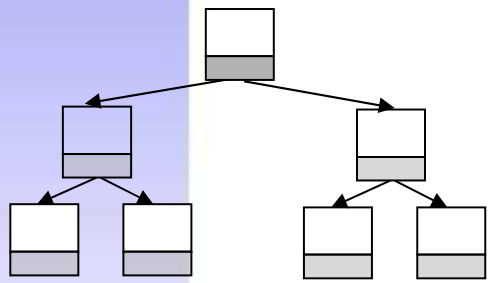
Terminologia eta oinarrizko propietateak

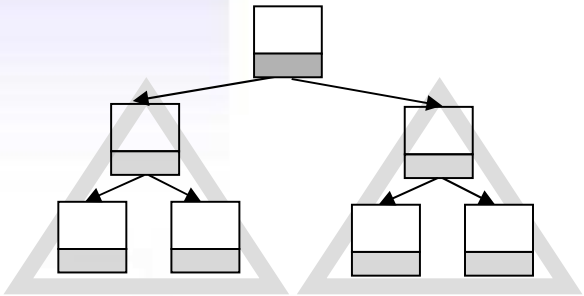


Adab	Altuera	sakonera	Tamaina	Barne Kanpo
a	3	0	11	Barne
b	1	1	3	Barne
c	0	1	1	Barne
d	1	1	2	Barne
e	1	1	4	Barne
f	0	1	1	Kanpo
g	0	2	1	Kanpo
h	0	2	1	Kanpo
i	0	2	1	Kanpo
j	1	3	2	Barne
k	0	3	1	Kanpo

Zuhaitzaten **tamaina**: 11
Zuhaitzaren **altuera**: 3
Zuhaitzaren **sakonera**: 3

Zuhaitz bitarrak





- ***Zuhaitz bitarrak:***

– Zuhaitz bitar bat adabegi bakoitzak **gehienez bi ume** dituen zuhaitz bat da

- Lehenego ume adabegiari **ezkerra** deritzaio
- Bigarren ume adabegiari **eskubia** deritzaio

– Zuhaitz bitarrak ere definitu daitezke era **errekurtsiboan.** Kasu honetan esaten da zuhaitza hutsa dela edo osatuta dagoela:

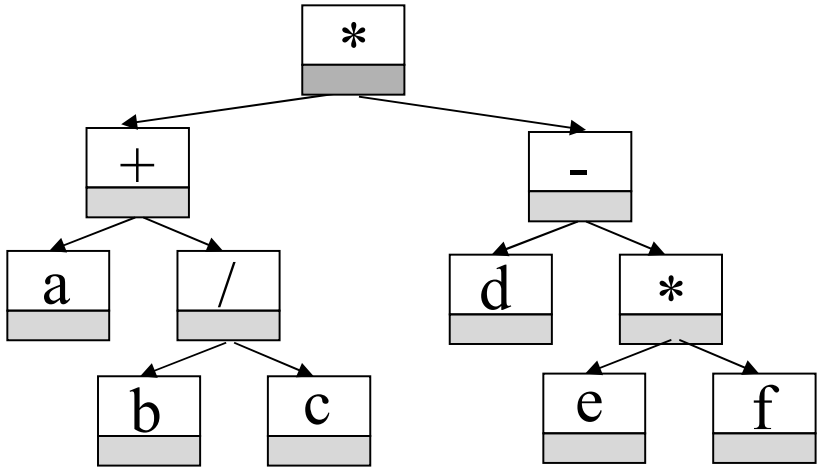
- **Erroa** adabegiaz
- **Azpizuhaitz ezkerraz** eta
- **Azpizuhaitz eskubiaz**

Zuhaitz bitarrak

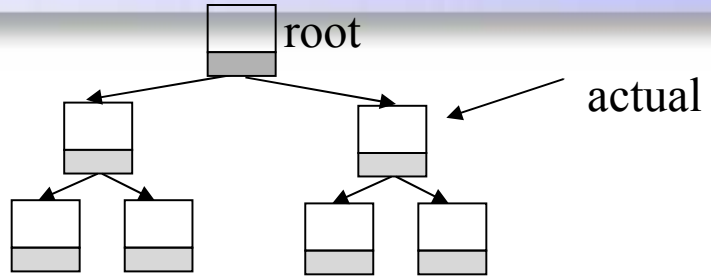
Adibideak

- Espresio aritmetikoak eragile bitarrekin.
- Txapelketa baten egitura
- Zuhaitz genealogikoak

Adibideak: (a+b/c)*(d-e*f)



Zuhaitz bitarren inplementazioa
BinTree:egitura dinamikoa

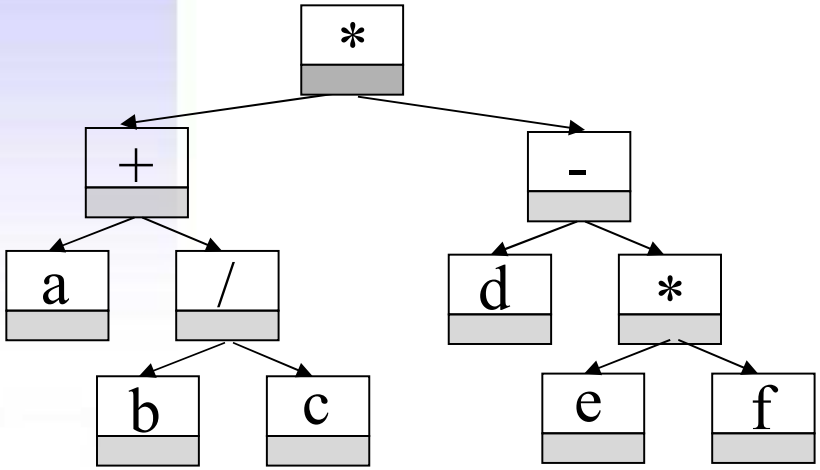



```
public class BinTree<T>{
 BTNode<T> root;
 BTNode<T> actual;
}
```

```
public class BTNode<T>{
  T content;
  BTNode<T> left;
  BTNode<T> right;
  public BTNode(T el){
    content=el;
  }
}
```

Zuhaitz bitarren inplementazioa
Egitura estatikoa Array-en bidez

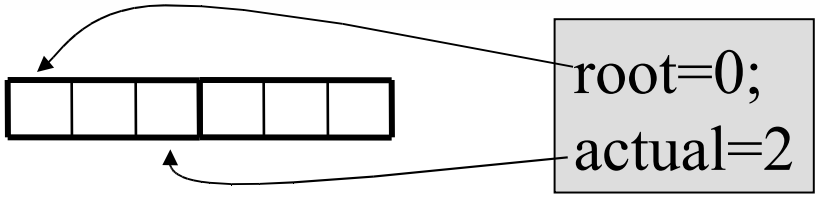
Zuhaitzen datu egitura errepresentazio estatiko baten bidez adierazi daiteke. Inplementazio hau erabiltzen da memoria esleipen dinamikoa onartzen ez duten lengoaietan.



Pos	content	left	right
1	*	2	3
2	+	4	5
3	-	6	7
4	a	0	0
5	/	8	9
6	d	0	0
7	*	10	11
8	b	0	0
9	c	0	0
10	e	0	0
11	f	0	0

Zuhaitz bitarren inplementazioa

BinTree:egitura estatikoa




```
public class BinTree<T>{
 int root; //erroaren posizioa
 BTNode<T>[] tree;
 int actual;
}
```

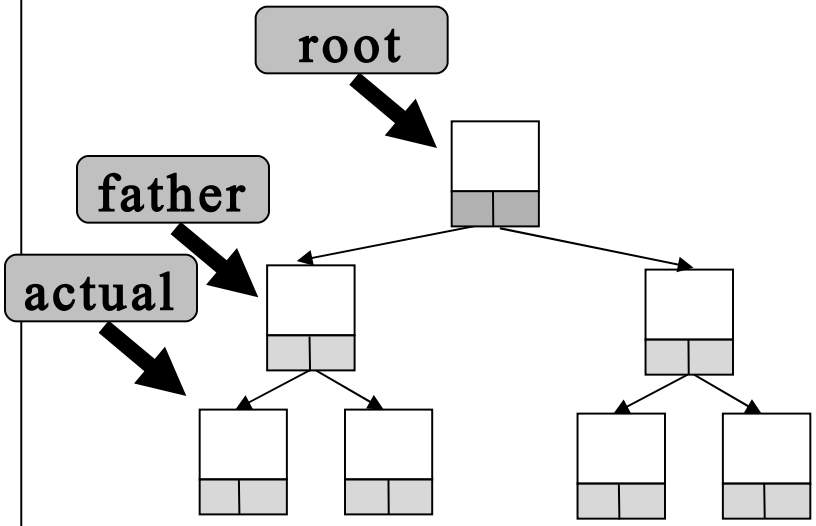
```
public class BTNode<T>{
   T content;
   int left;
   int right;
}
```

Zuhaitz bitarren inplementazioa

BinTree klasearen metodoak

class BinTree<T>

- Atributua: BTNode<T> root;
- Atributua: BTNode<T> actual
- Atributua: BTNode<T> father
- Metodoak:
  - void goLeft();
  - void goRight();
  - void goBrother();
  - void goRoot();
  - void inserRoot(T o);
  - void insertLeftChild(T o);
  - void insertRightChild(T o);
  - T getActual();



BinTree

Metodoak Java-n

Desplazamendu metodoak	***Esanahia***
void **goRoot()**	*actual* kokatzen du zuhaitzaren **erroan**
void **goLeft()**	*actual* kokatzen du *actual* elementuaren **ume ezkerrean**
void **goRight()**	*actual* kokatzen du *actual* elementuaren **ume eskubian**
void **goBrother()**	*actual* kokatzen du *actual* elementuaren **anaian**

Kontsulta metodoak	***Esanahia***
boolean **isInternal()**	*true* itzultzen du *actual* **barne** adabegia bada
boolean **isExternal()**	*true* itzultzen du *actual* **kanpo** adabegia bada (hostoa)
boolean **isRoot()**	*true* itzultzen du *actual* adabegia **erroa** bada (hostoa)
int **size()**	Zuhaitzaren **tamaina** itzultzen du
boolean **isEmpty()**	*true* itzultzen du **zuhaitza** hutsa bada
T **getActual()**	*actual* adabegiaren **edukia** itzultzen du
void **setActual**(T o)	*actual* adabegiaren edukian o objektua **ipintzen** du

BinTree

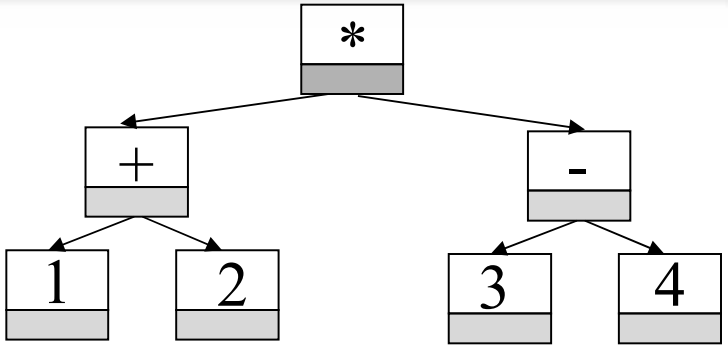
Metodoak Javan

Txertaketa metodoak	*Esanahia*
void **insertRoot** (T o)	o objektua txertatzen du zuhaitzaren erroa bezala
void insertLeftChild (T o)	o objektua txertatzen du *actual* adabegiaren ume ezkerra bezala
void insertRightChild (T o)	o objektua txertatzen du *actual* adabegiaren ume eskubia bezala

Zuhaitz bitarrak

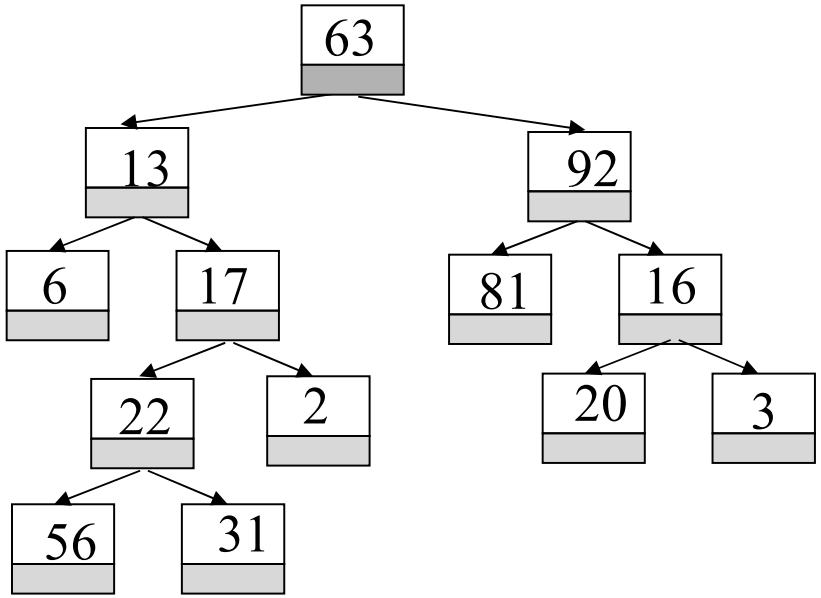
Zuhaitz baten sorketa BinTree klasearen bidez

```
BinTree<String> bTree=new BinTree<String>();
bTree.insertRoot(new String(“*”));
bTree.insertLeftChild(new String(“+”));
bTree.insertRightChild(new String(“-”));
bTree.goLeft();
bTree.insertLeftChild(new String(“1”));
bTree.insertRightChild(new String(“2”));
bTree.goRoot();
bTree.goRight();
bTree.insertLeftChild(new String(“3”));
bTree.insertRightChild(new String(“4”));
```



Zuhaitz bitarrak
Ariketa I

Osoko zuhaitz bitar bat inplementatzen duen BinTree klasea emanda eta x elementu bat, elementu hori zuhaitzan agertzen den ala ez erabakitzen duen metodo bat inplementatu BinTree klasean.



Adibidea: binTree.find(22) = true

Zuhaitzen errepresentazioa
Ariketa I

I atala : Espezifikazioa/Parametrizazioa

✔**Sarrera:**

Zuhaitz bat eta x zenbaki oso bat

✔**Irteera:**

Balio boolear bat, *true* izango da zuhaitzaren adabegiren bat x balioa badu eta *false* kontrako kasuan.

Zuhaitzen errepresentazioa
Ariketa I

II Atala: Kasu nabarien eta orokorren ebazpena

✔**Kasu nabaria(k)**

→ Zuhaitza hutsa bada orduan *false* itzuli

→ Zuhaitza hutsa ez bada eta bere erroa bilatzen ari garen elementua berdina bada, orduan *true* itzuli

✔**Kasu orokorra(k)**

→ Zuhaitza hutsa ez bada eta bere erroa bilatzen ari garen elementua desberdina bada, orduan *true* itzuli elementua ezkerreko edo eskubiko azpizuhaitzan agertzen bada eta *false* beste kasuan.

Zuhaitzen errepresentazioa
Ariketa I

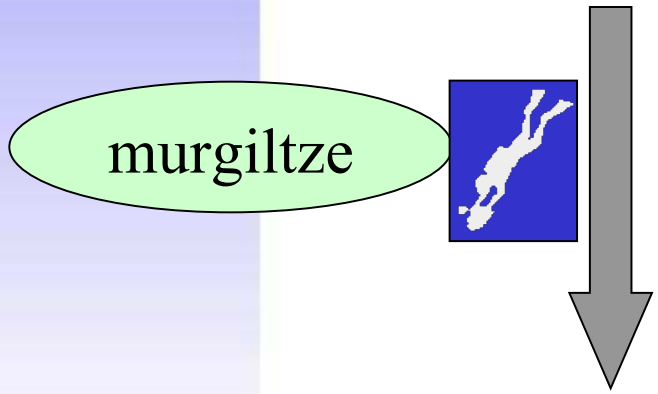
III Atala : Azpiprogramaren diseinua

find(x)

```
IF zuhaitza EZ hutsa  THEN
      IF zuhaitzaren erroa=bilatutako elementua THEN
            true tzuli
       ELSE
         itzuli (find (azpizuhaitz ezkerra, elementua) EDO
                        find (azpizuhaitz eskuina, elementua) )
 ELSE
   false itzuli
```

Zuhaitzen errepresentazioa
Ariketa I: murgiltze arazoa

boolean find(T x)



Pauso bakoitzean x elementuaz gainera, zein **adabegian** gaude jakin behar dugu

boolean find(BTNode<T> theNode, T x)

Metodo berri bat definitzen da parametro gehiagorekin

Zuhaitzen errepresentazioa
Ariketa I

IV atala: Programazioa

```
private boolean find(BTNode<T> pNode, T pValue) {
  T i;
  if (pNode!=null) {
   i=pNode.content;
   if (i.compareTo(pValue)==0 )
    return true;
    else
     return (find(pNode.left, pValue) ||
                find(pNode.right,pValue) );
   }
   return false;
}
```

```
public boolean find(T pValue) {
   return find(root,pValue);
}
```

Zuhaitzen errepresentazioa

Errekurtsioaren 3 oinarrizko erregelak

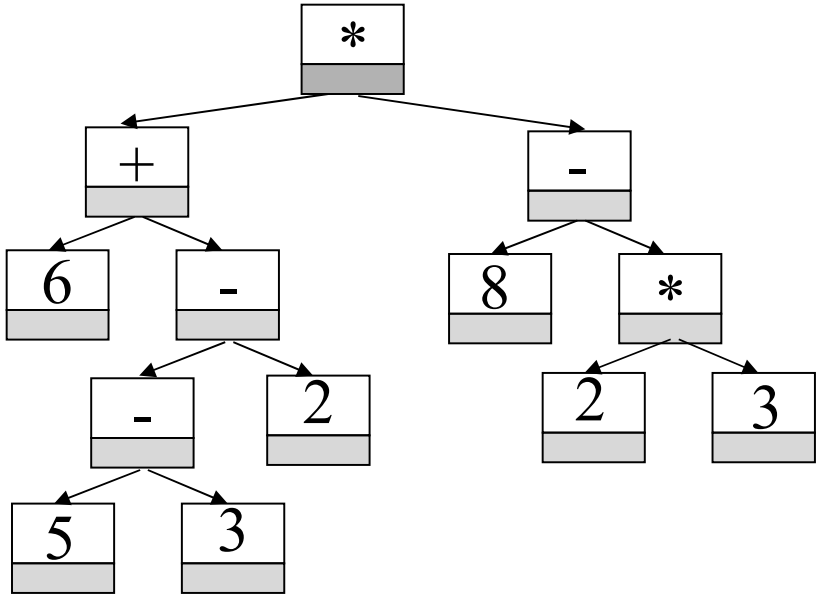
1. *Kasu nabaria(k):*Gutxienez kasu bat egon behar da errekurtsiorik gabe ebatzi daitekena

2. *Progresioa*: Deialdi errekurtsibo guztiak kasu nabarietara hurbildu behar dira

3. *"Sinestu daiteke":* egiten diren barne deialdi errekurtsibo guztiak emaitza zuzena itzultzen dutela sinestu.

Zuhaitzen errepresentazioa
Ariketa II

Espresio aritmetiko bat errepresentatzen duen Zuhaitz bitar BinTree klasea emanda, zuhaitzaren balioa lortzen duen metodo bat inplementatu



Adibidea: binTree.evaluate()=12

Zuhaitzen errepresentazioa
Ariketa II

I atala : Espezifikazioa/Parametrizazioa

✔**Sarrera:**

Zuhaitz bitar bat espresio aritmetiko baten errepresentazioarekin

✔**Irteera:**

Zuhaitza espresioaren emaitza.

Zuhaitzen errepresentazioa
Ariketa II

II Atala: Kasu nabarien eta orokorren ebazpena

✔**Kasu nabariak**

- ➔ Zuhaitza hutsa bada orduan 0 itzuli
- ➔ Zuhaitzak erroan zenbaki oso bat badauka orduan itzuli zenbakia

✔**Kasu orokorrak**

- ➔ Zuhaitzak erroan eragile bat badauka
  - ➔azpizuhaitz ezkerra ebaluatu
  - ➔azpizuhaitz eskubia ebaluatu
  - ➔erroaren eragilea aplikatu lortutako emaitzei

Zuhaitzen errepresentazioa
Ariketa II

III Atala : Azpiprogramaren diseinua

```
IF zuhaitza EZ hutsa  THEN
     IF zuhaitzak erroan elementu oso bat badu THEN
          elementuaren balioa itzuli
      ELSE //zuhaitzak erroan eragiketa bat badauka
        ebaluatu (azpizuhaitz ezkerra)
        ebaluatu (azpizuhaitz eskubia)
        erroaren eragiketa aplikatu lortutako emaitzei eta
        emaitza itzuli
 ELSE
   itzuli 0
```

Zuhaitzen errepresentazioa
Ariketa II

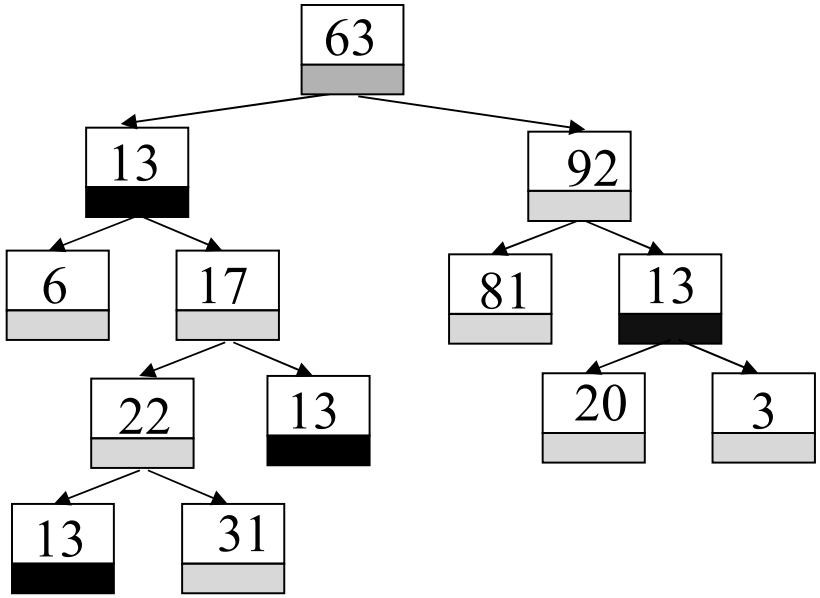
IV atala: Programazioa

```
public int evaluate(BTNode<T> pNode) {
  Character c;
  if (pNode!=null) {
   c=pNode.content;
   if (Character.isDigit(c.charValue()) )
    return Integer.parseInt(c.toString());
    else {
     int left=evaluate(pNode.left);
     int right=evaluate(pNode.right);
     return calculate(c.charValue(),left,right);
     }
   }
   return 0;
}
```

```
public int evaluate() {
   return this.evaluate(root);
}
```

Zuhaitzen errepresentazioa
Klaseko ariketa

x elementuaren agerpen kopurua itzultzen duen **`int count(T x)`** metodoa inplementatu.



binTree.count(13)=4

Zuhaitzen adabegiak “bisitatu”

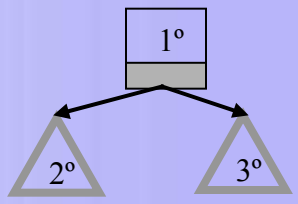
Zuhaitz batean egiten den ohizko eragiketa bat **adabegi guztietan** eragiketa bat exekutatzea izaten da.

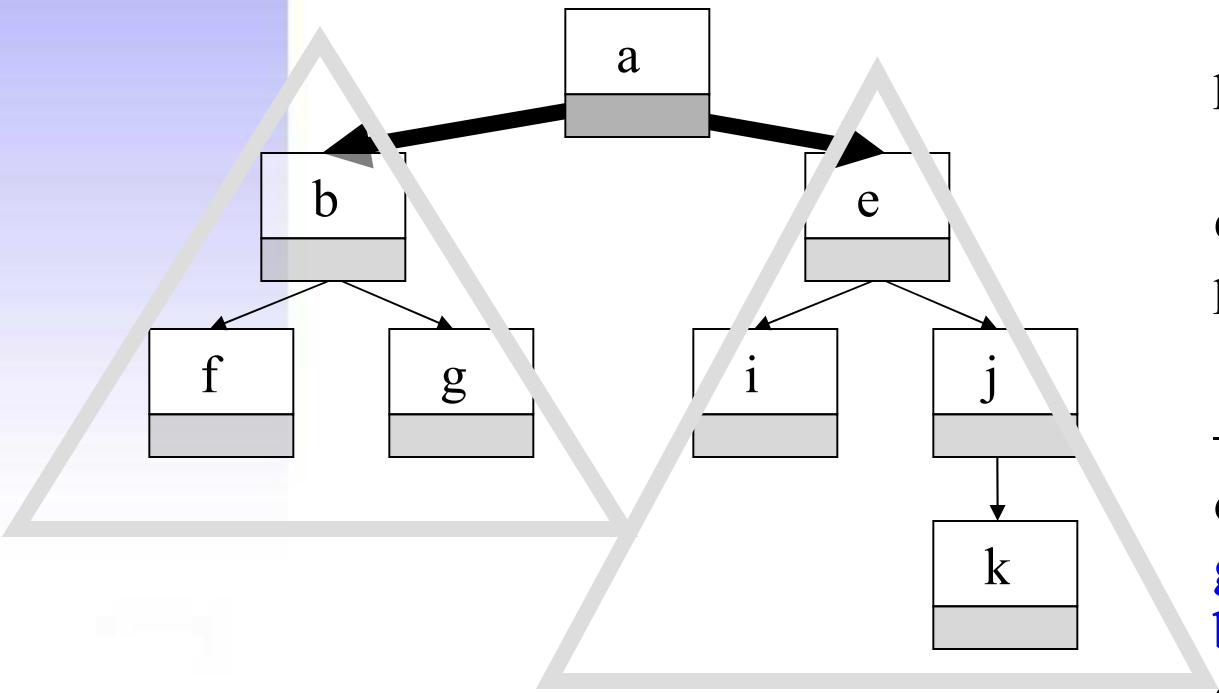
Zuhaitzaren elementu guztietan eragiketa bat exekutatzea, adabegi guztiak “**bisitatzea**” dakar. Prozesu hau 3 erataran egin daiteke:

1. **Aurre-Ordenean**
   - Erroa bisitatu
   - Ezkerreko azpizuhaitza aurre-Ordenean korritu
   - Eskubiko azpizuhaitza aurre-Ordenean korritu
2. **Post-Ordenean**
   - Ezkerreko azpizuhaitza post-Ordenean korritu
   - Eskubiko azpizuhaitza post-Ordenean korritu
   - Erroa bisitatu
3. **In-Ordenea**
   - Ezkerreko azpizuhaitza in-Ordenean korritu
   - Erroa bisitatu
   - Eskubiko azpizuhaitza in-Ordenean korritu

Zuhaitz bitarrak

Aurre-ordenean “bisitatu”





Aurre-ordenean (Def)

- Lehendabizi **adabegia** prozesatzen da
- Jarraian **bere umeak** errekurtsiboki aurre-ordenean prozesatzen dira

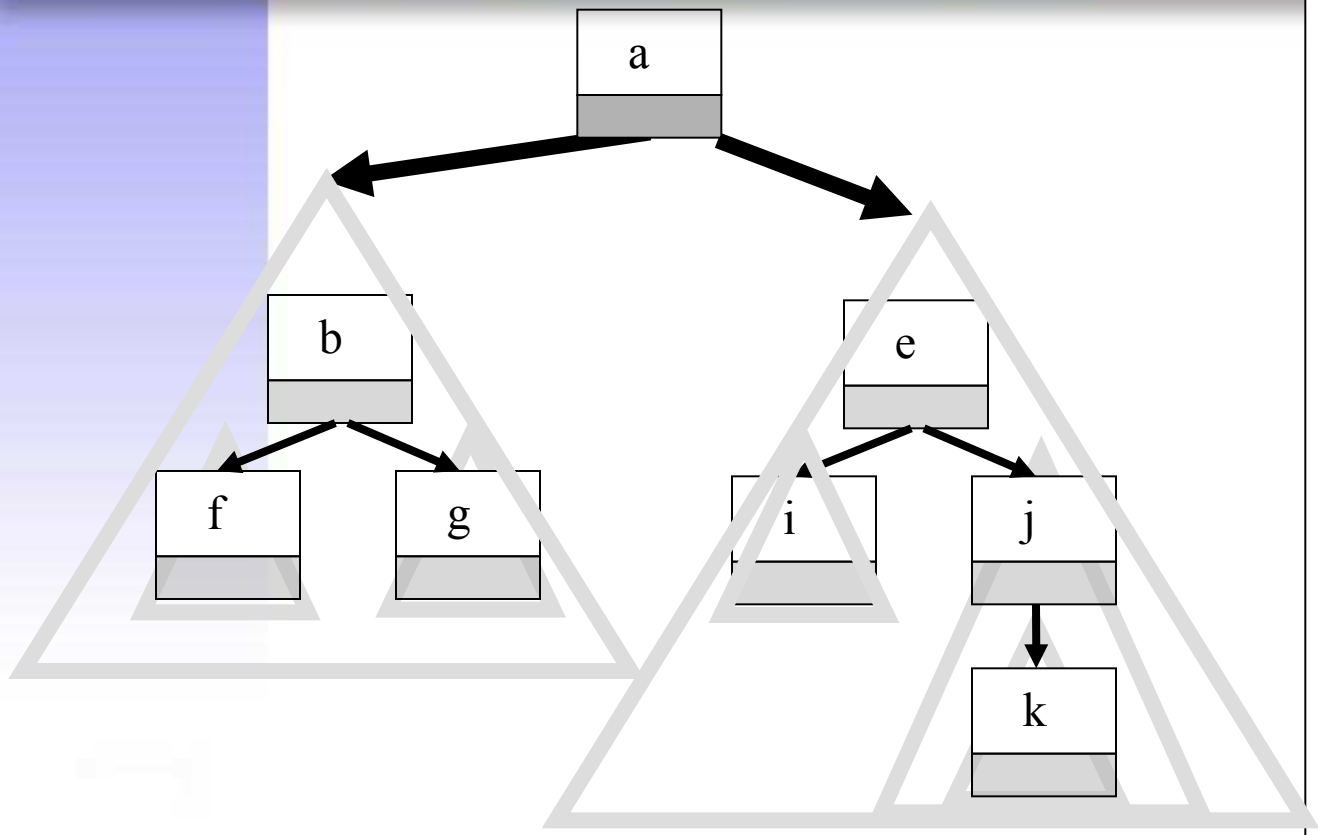
• ***Erabilpena***

– Ordenazio lineal bat lortu nahi dugunean **guraso adabegi guztiak beti bere umeak baino lehenagoko** agerpenarekin.

– Ej: liburu baten kapituloak

Zuhaitz bitarrak

Aurre-ordenean “bisitatu”



Aurre-ordenean

1. **Lehendabizi adabegia (a)**
2. **Azpizuhaitz ezkerra**
   1. Lehendabizi adabegia **(b)**
   2. Azpizuhaitz ezkerra
      1. Lehendabizi . adabegia **(f)**
      2. Azpizuhaitz. ezkerra (null)
      3. Azpizuhaitz eskubia (null)
   3. Azpizuhaitz eskubia
      1. Lehend. adabegia **(g)**
      2. Azpizuhaitz ezkerra (null)
      3. Azpizuhaitz eskubia (null)
3. **Azpizuhaitz eskubia**
   1. Lehendabizi adabegia **(e)**
   2. Azpizuhaitz ezkerra
      1. Lehendabizi adabegia **(i)**
      2. Azpizuhaitz ezkerra (null)
      3. Azpizuhaitz eskubia (null)
   3. Azpizuhaitz eskubia
      1. Lehendabizi adabegia **(j)**
      2. Azpizuhaitz ezkerra (null)
      3. Azpizuhaitz eskubia
         1. Lehend. adabegia **(k)**
         2. Azp. ezkerra (null)
         3. Azp. eskubia (null)

Adabegiak prozesatzen diren ordena:

a,b,f,g,e,i,j,k

Zuhaitz bitarrak

Ariketa III: Aurre-ordenean “bisitatu”

I atala : Espezifikazioa/Parametrizazioa

✔**Sarrera:**

Zuhaitz bitarra

✔**Irtera:**

Zuhaitzaren elementu guztien inprimaketa aurre-ordenean.

II Atala: Kasu nabarien eta orokorren ebazpena

✔**Kasu nabariak**

- ➔ Zuhaitza hutsa bada orduan ezer ez idatzi

✔**Kasu orokorrak**

- ➔ Erroaren balioa inprimatu
- ➔ Ezkerreko zuhaitza idatzi aurre-ordenean
- ➔ Eskubiko zuhaitza idatzi aurre-ordenean

Zuhaitz bitarrak

Ariketa III: Aurre-ordenean “bisitatu”

III atala : Azpiprogramaren diseinua

```
void AurreOrden(Node)

 IF Node EZ hutsa ORDUAN
    imprimatu Node-ren balioa
    Imprimatu AurreOrdenean Node-ren azpizuhaitz ezkerra
    Imprimatu AurreOrdenean Node-ren azpizuhaitz eskubia
  ELSE
   ezer ez egin
```

Zuhaitz bitarrak

Ariketa III: Aurre-ordenean “bisitatu”

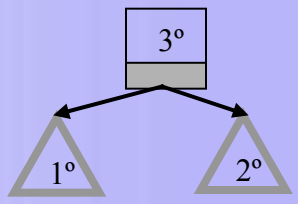
IV atala : Programazioa

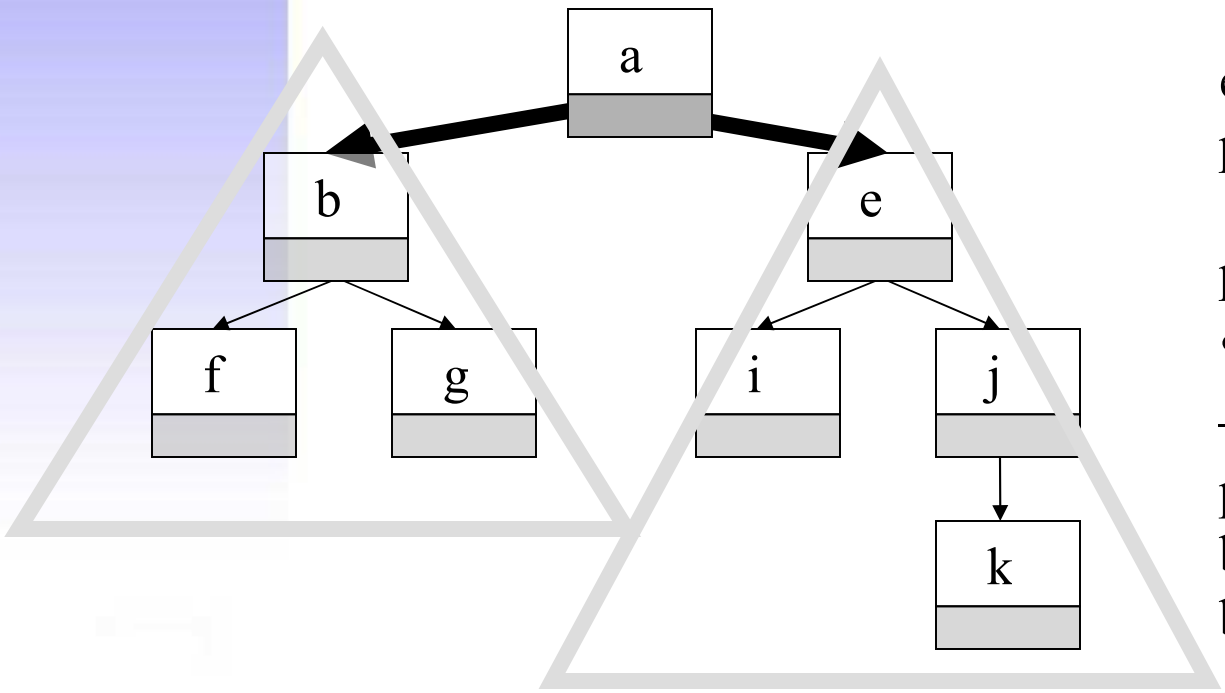
```
public void printAurreOrden() {
this.printAurreOrden(root);
}
```

```
public void printAurreOrden(BTNode pNode) {
  if (pNode!=null) {
    System.out.println(pNode.content);
    printAurreOrden(pNode.left);
    printAurreOrden(pNode.right);
  }
}
```

Zuhaitz bitarrak

Post-ordenean “bisitatu”





Post-ordenean (Def)

- Lehendabizi **bere umeak** errekurtsiboki post-ordenean prozesatzen dira
- Jarraian **adabegia** prozesatzen da

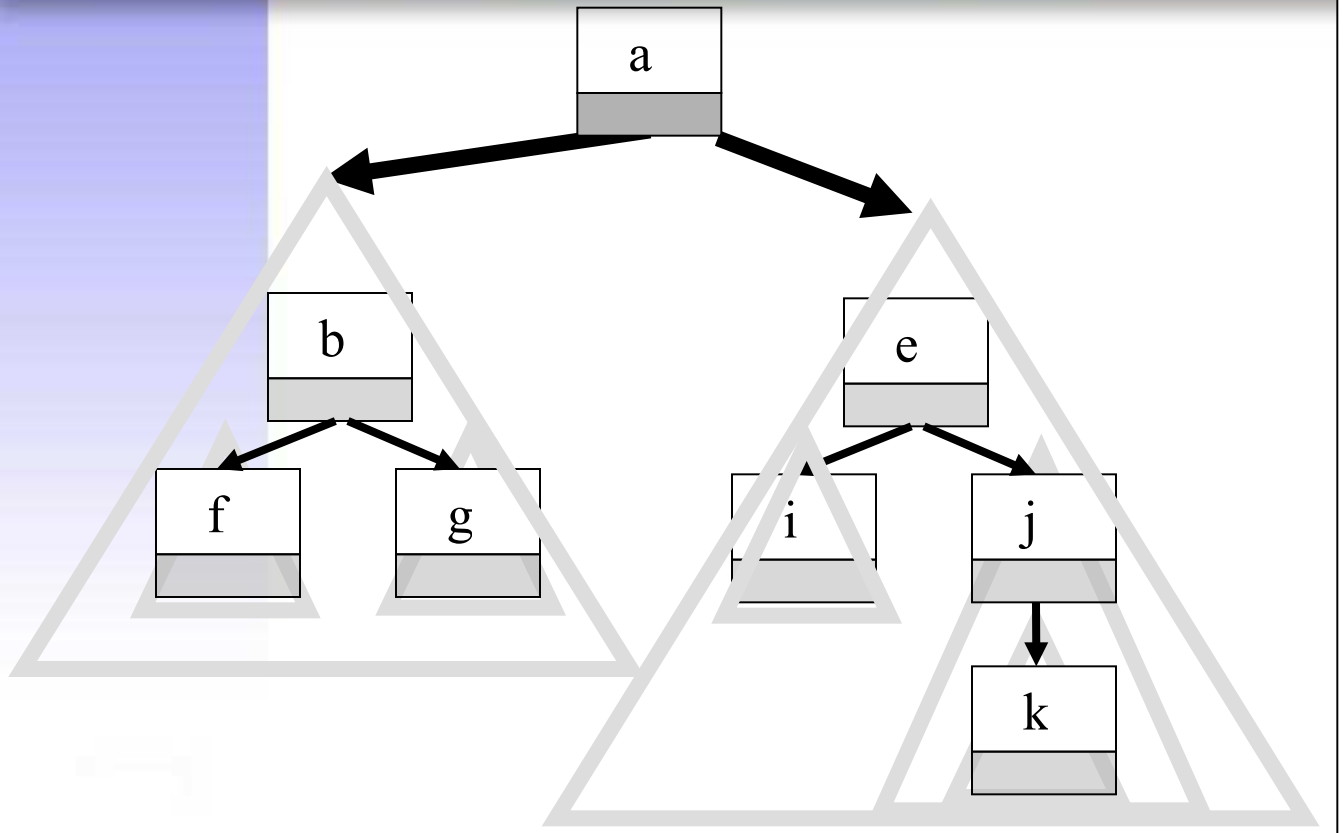
• ***Erabilpena***

– Guraso adabegiaren problema ebazteko, lehendabizi bere umeen problema ebatzi behar denean

Adibidea: **Direktorio baten tamaina** fitxategi sistema batean.

Zuhaitz bitarrak

Post-ordenean “bisitatu”



Adabegiak prozesatzen diren ordena:

f,g,b,i,k,j,e,a

Post-ordenean

1. Zuhaitz ezkerra

1. Zuhaitz ezkerra
   1.Zuhaitz ezkerra (null)
   2. Zuhaitz eskubia (null)
   3. Adabegia **(f)**
2. Zuhaitz eskubia
   1.Zuhaitz ezkerra (null)
   2. Zuhaitz eskubia (null)
   3. Adabegia **(g)**
3. Adabegia **(b)**

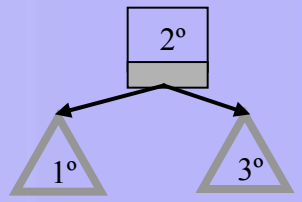
2. Zuhaitz eskubia

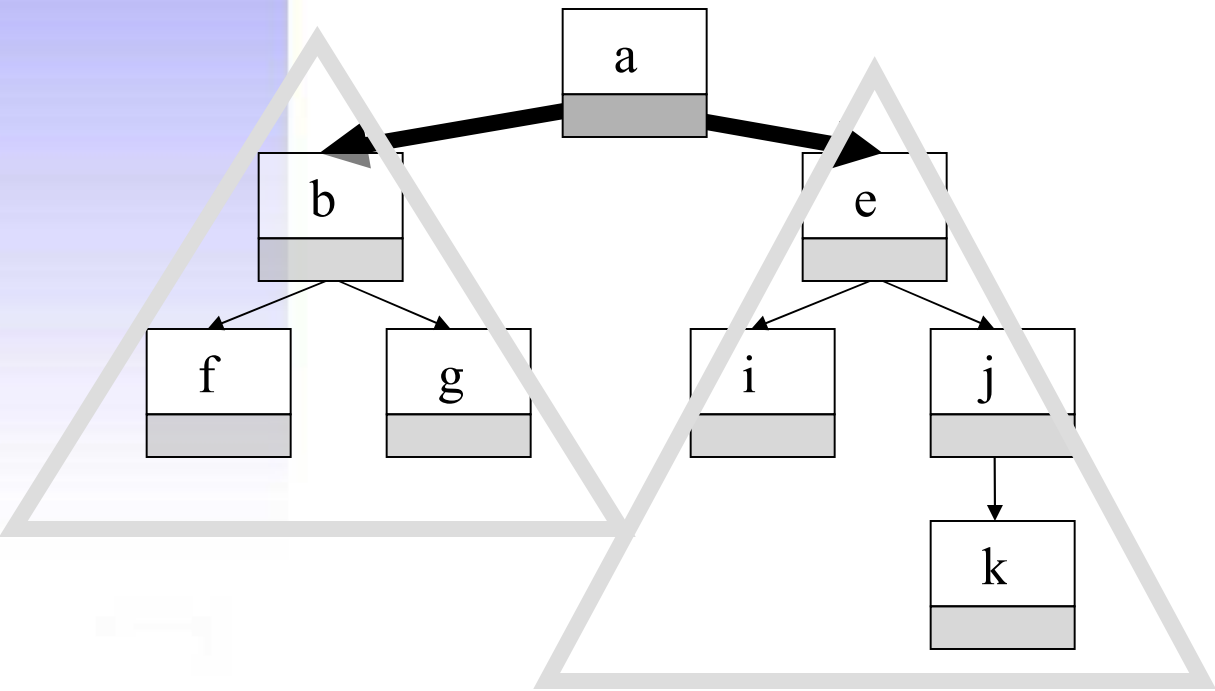
1. Zuhaitz ezkerra
   1. Zuhaitz ezkerra (null)
   2. Zuhitz eskubia (null)
   3. Adabegia **(i)**
2. Zuhaitz eskubia
   1. Zuhaitz ezkerra (null)
   2. Zuhaitz eskubia (null)
      1. Zuhaitz ezkerra. (null)
      2. Zuhaitz eskubia (null)
      3. Adabegia (null) **(k)**
   3. Adabegia **(j)**
3. Adabegia (e)

3. Adabegia (a)

Zuhaitz bitarrak

In-ordenean “bisitatu”: (orden simetrikoan)





In-ordenean (Def)

- Lehendabizi **ezkerreko** zuhaitza prozesatzen da
- Jarraian **adabegia**
- Azkenik **eskubiko** zuhaitza

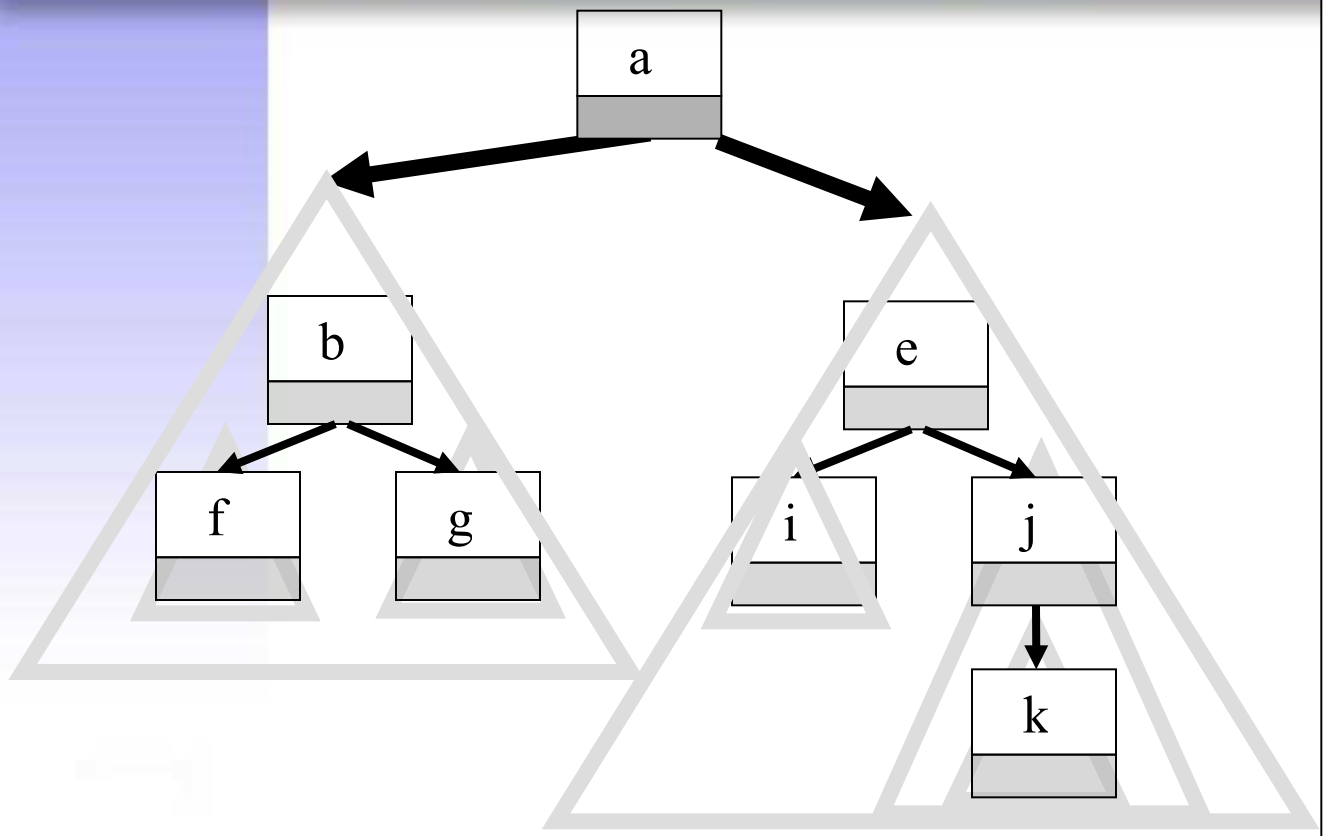
- ***Erabilpena***

– Gurasoa adabegia bere bi umeen artean prozesatu behar denean.

– Adibidea: Espresio algebraiko bat sortu zuhaitz baten errepresentazioengatik.

Zuhaitz bitarrak

In-ordenean “bisitatu”



Adabegiak prozesatzen diren ordena :

f,b,g,a,i,e,j,k

In-ordenean

1. Zuhaitz ezkerra

1. Zuhaitz ezkerra

1.Zuhaitz ezkerra (null)

2. Adabegia **(f)**

3. Zuhaitz eskubia (null)

2. Adabegia **(b)**

3. Zuhaitz eskubia

1. Zuhaitz ezkerra (null)

2. Adabegia **(g)**

3. Zuhaitz eskubia (null)

2. Adabegia (a)

3. Zuhaitz eskubia

1. Zuhaitz ezkerra

1. Zuhaitz ezkerra (null)

2. Adabegia **(i)**

3. Zuhaitz eskubia (null)

2. Adabegia **(e)**

3. Zuhaitz eskubia

1. Zuhaitz ezkerra (null)

2. Adabegia **(j)**

3. Zuhaitz eskubia (null)

1.Zuhaitz ezkerra (null)

2. Adabegia (null) **(k)**

3. Zuhaitz eskubia (null)

Zuhaitzen errepresentazioa

Datuen sailkapena

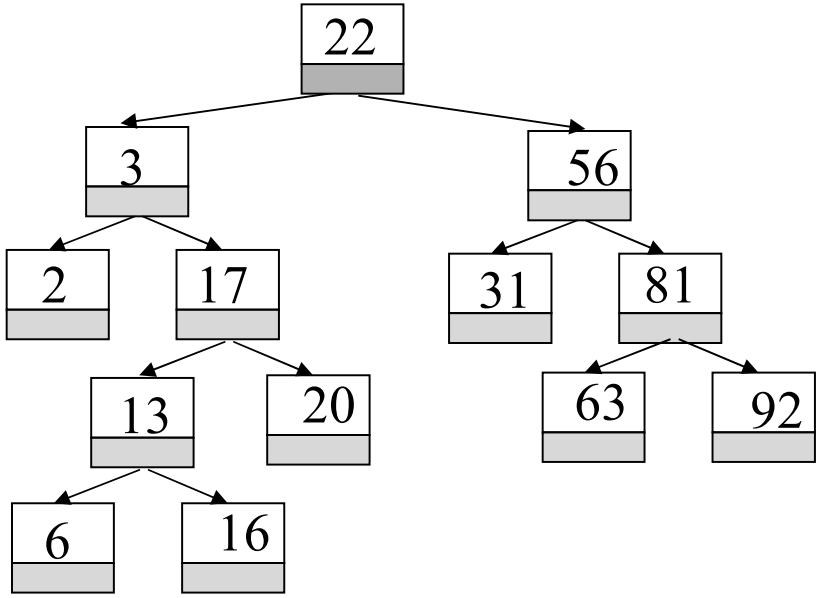
find metodoaren lehenengo inplementazioan, orokorrean zuhaitz guztia korritzen da elementuaren bila, elementua zuhaitzan dagoen ala ez kontuan hartu gabe.

PROZESU GARESTIA
(Eraginkortasun eskasa)

Nola antolatu zuhaitzaren elementuak bilaketa eraginkorragoa izan dadin?

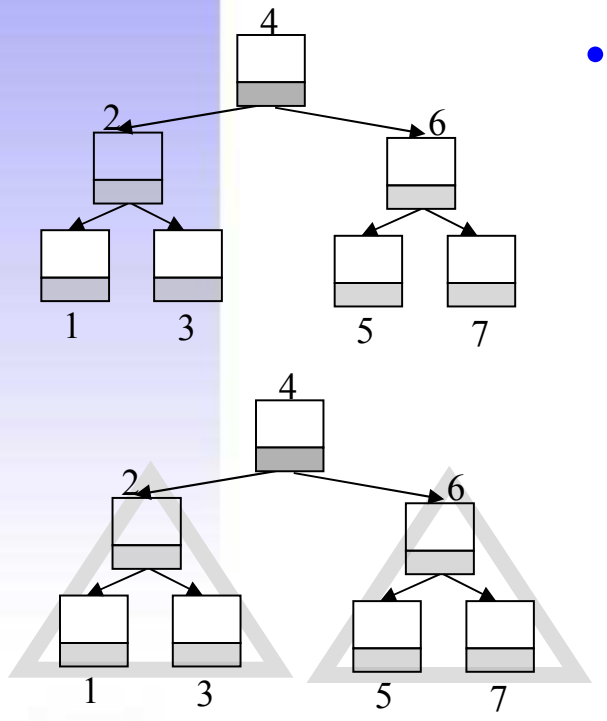
Bilaketa-zuhaitz bitarrak

Zein da zuhaitz honen ezaugarri nabarmenena ?



Zuhaitz honen adabegi guztientzat, ezkerreko azpizuhaitzen adabegi guztien balioak erroa baino txikiago dira, eta eskubiko azpizuhaitzen adabegi guztien balioak handiagoak.

Bilaketa-zuhaitz bitarrak

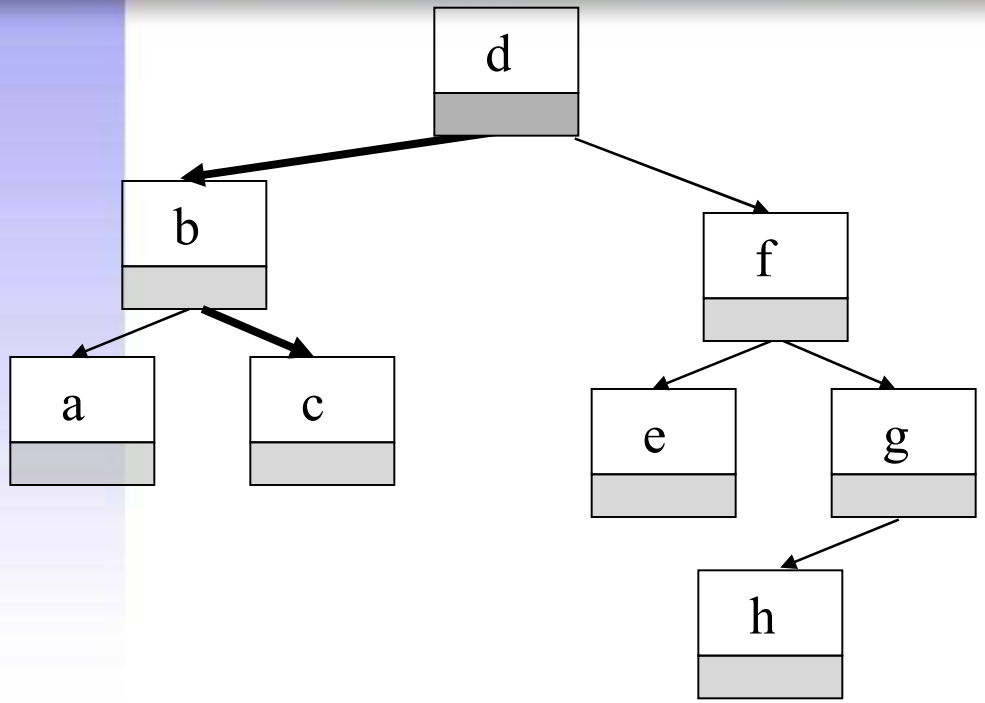


- B***ilaketa-zuhaitz bitarrak:***
  - zuhaitz bitar bat da, adabegi bakoitzean hurrengo propietatekin:
    - balio txikiagoa duten elementu guztiak, adabegi horren **azpizuhaitz ezkerrean** daude
    - balio haundiagoa duten elementu guztiak, adabegi horren **azpizuhaitz eskubian** daude
    - Ez dira onartzen elementu errepikatuak

bilaketa-zuhaitz bitarrak oinarrizko datu egitura bat da elementu talde bat errepresentatzeko, elementuak *orden lineal batengatik* sailkatuta daudenean.

Bilaketa-zuhaitz bitarrak

Oinarrizko eragiketak: Bilaketa



Adibidez “c” bilatzen dugu:

- c<d: Azpizuhaitz ezkerra
- c>b: Azpizuhaitz eskubia
- c=c: elementua aurkitua

- ***Bilaketa***: Ezkerreko edo eskubiko arkuetatik mugitzen gara konparaketaren arabera:
  - Bilatutako elementua eta adabegiaren erroa berdinak badira, orduan aurkitu dugu bilatutako elementua
  - Adabegiaren erroa baino txikiagoa bada, bilatuko dugu ezkerreko azpizuhaitzan.
  - Adabegiaren erroa baino haundiagoa bada, bilatuko dugu eskubiko azpizuhaitzan.

Bilaketa-zuhaitz bitarrak

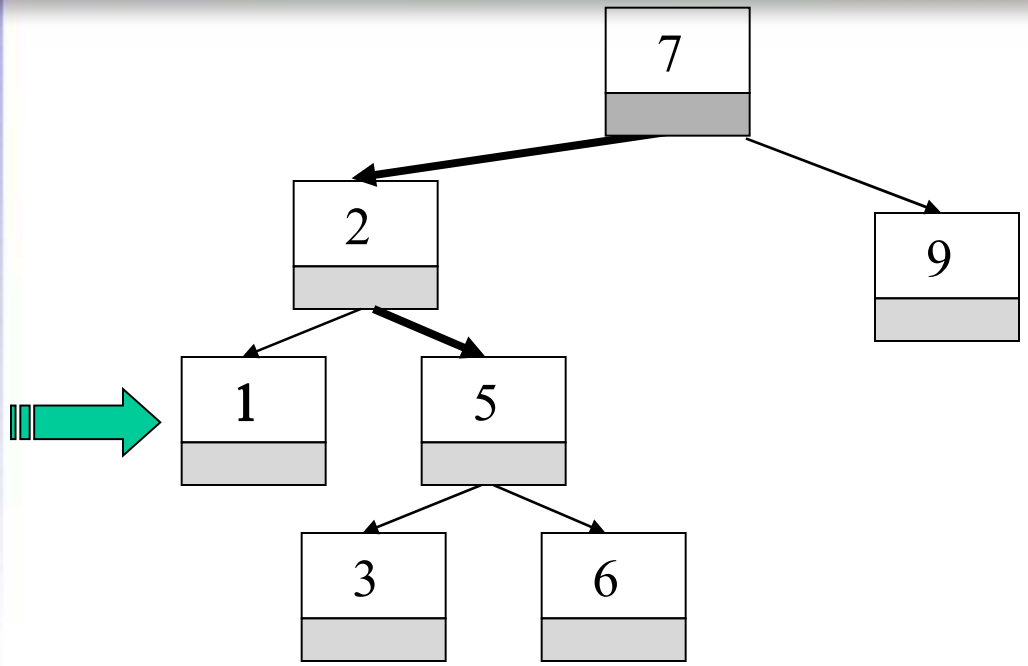
Oinarrizko eragiketak: Bilaketa. Inplementazioa

```
public boolean find(T x) {
 return find(x,root); //murgilketa}
```

```
public boolean find(T x, BTNode<T> pNode) {
 if (pNode==null)
  return false;
  else
   {
   T edukia=pNode.content;
   if (edukia.compareTo(x)==0) return true;
    else if (x.compareTo(edukia)>0) return find(x,pNode.right);
     else return find(x,pNode.left);
   }
}
```

Bilaketa-zuhaitz bitarrak

Oinarrizko eragiketak: txikiaren bilaketa.



Elementu txikiena **ezkerrago kokatuta dagoen hostoa** da (Node.left==null)

Bilaketa-zuhaitz bitarrak

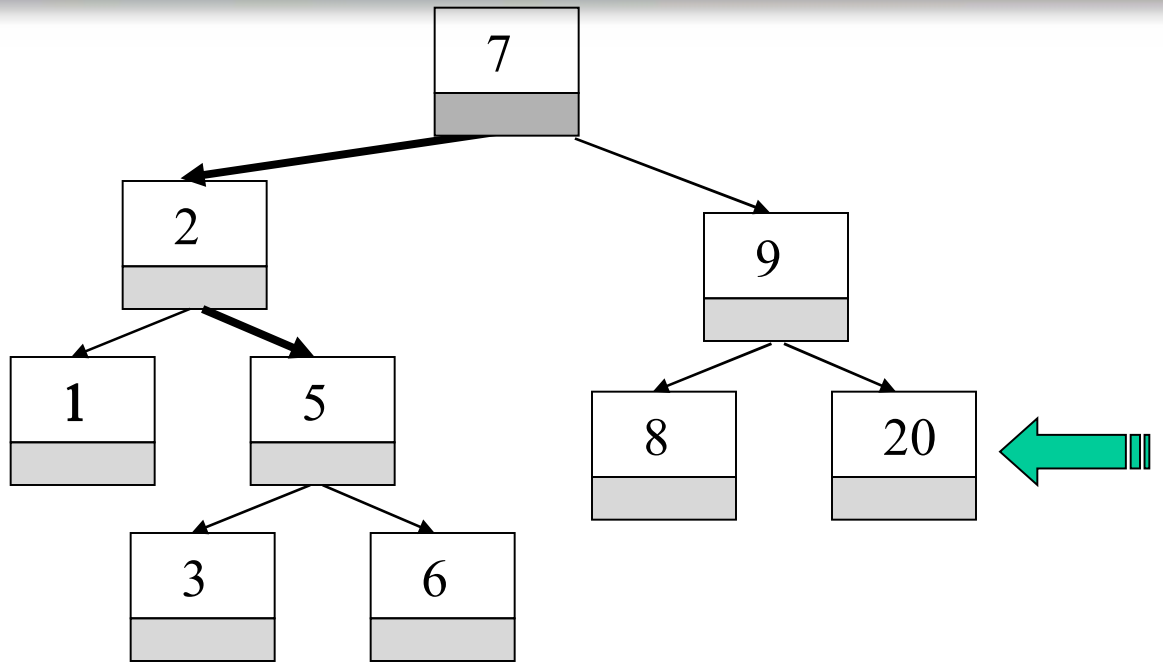
Oinarrizko eragiketak: txikiaren bilaketa. Inplementazioa

```
public int findMin() {
 return findMin(root); //murgilketa
```

```
public T findMin(BTNode<T> pNode) {
 if (pNode.left!=null)
  return findMin(pNode.left);
   else
    return pNode.content;
}
```

Bilaketa-zuhaitz bitarrak

Oinarrizko eragiketak: haundienaren bilaketa.



Elementu haundiena **eskubiago kokatuta dagoen hostoa** da (Node.right==null)

Bilaketa-zuhaitz bitarrak

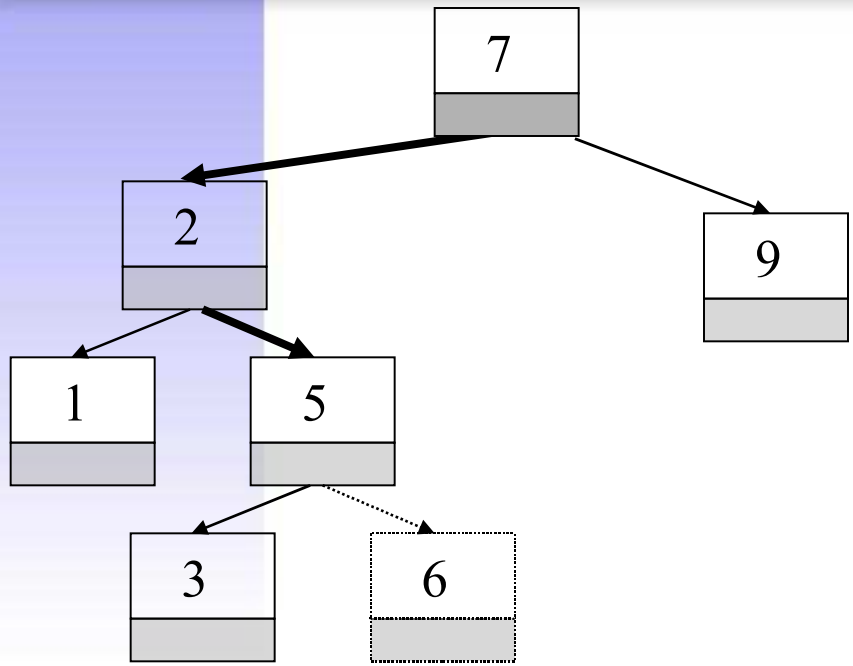
Oinarrizko eragiketak: haundienaren bilaketa. Inplementazioa

```
public int findMax() {
 return findMax(bTree.root);}
```

```
public T findMax(BTNode<T> pNode) {
 if (pNode.right!=null)
  return findMax(pNode.right);
   else
    return pNode.content;
}
```

Bilaketa-zuhaitz bitarrak

Oinarrizko eragiketak: Txertaketa



Adibidez 6 txertatu

- ***Txertaketa*** : Konparaketaren arabera mugitzen gara. Hosto batera ailegatzen garenean txertatzen dugu
  - Elemetuaren balioa eta adabegiaren erroa berdinak badira, ez dugu ezer egiten.
  - Hosto batera ailegatzen garenean
    - **Txikiagoa bada ezkerrean txertatzen dugu**
    - **Haundiagoa bada eskubian txertatzen dugu**
  - Txikiagoa bada ezkerreko azpizuhaitzan bilatzen dugu
  - Haundiagoa bada eskubiko azpizuhaitzan bilatzen dugu

Bilaketa-zuhaitz bitarrak

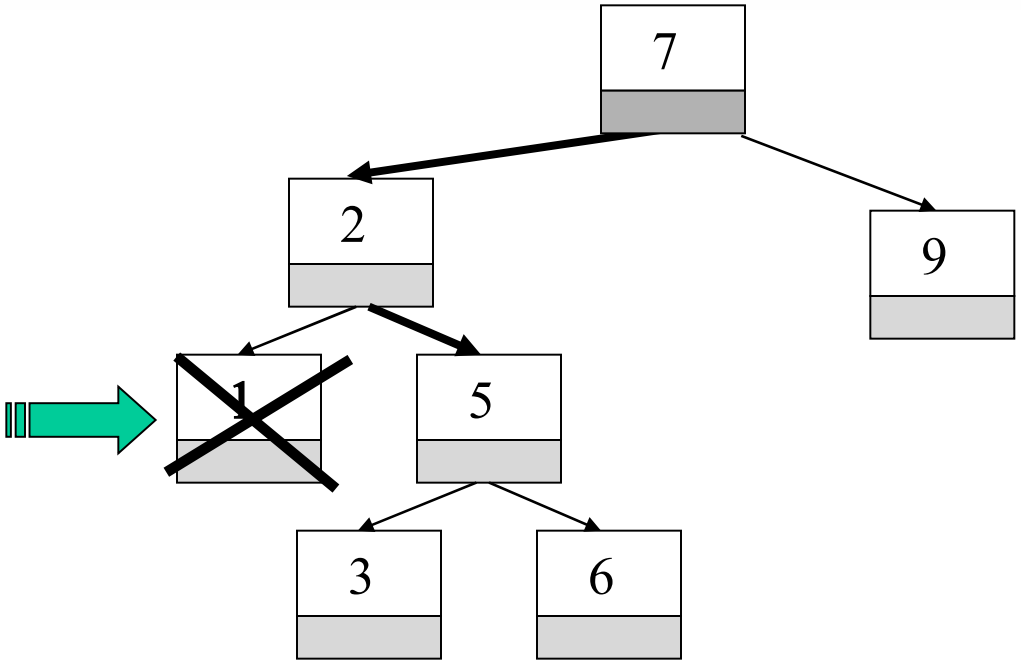
Oinarrizko eragiketak: Txertaketa. Inplementazioa

```
public void insert (int i) {
 root=insert(i,root);}
```

```
public BTNode insert (T pElem, BTNode<T> pNode) {
if (pNode==null)
 pNode=new BTNode(pElem,null,null);
 else {
  if ( pElem.compareTo(pNode.content))<0)
     pNode.left=insert(pElem,pNode.left);
  else
   pNode.right=insert(pElem,pNode.right);
 }
return pNode;
}
```

Bilaketa-zuhaitz bitarrak

Oinarrizko eragiketak: Txikienaren ezabatu



<u>Ezkerrago kokatuta dagoen</u> elementua ezabatzen da (node.left==null)

Bilaketa-zuhaitz bitarrak

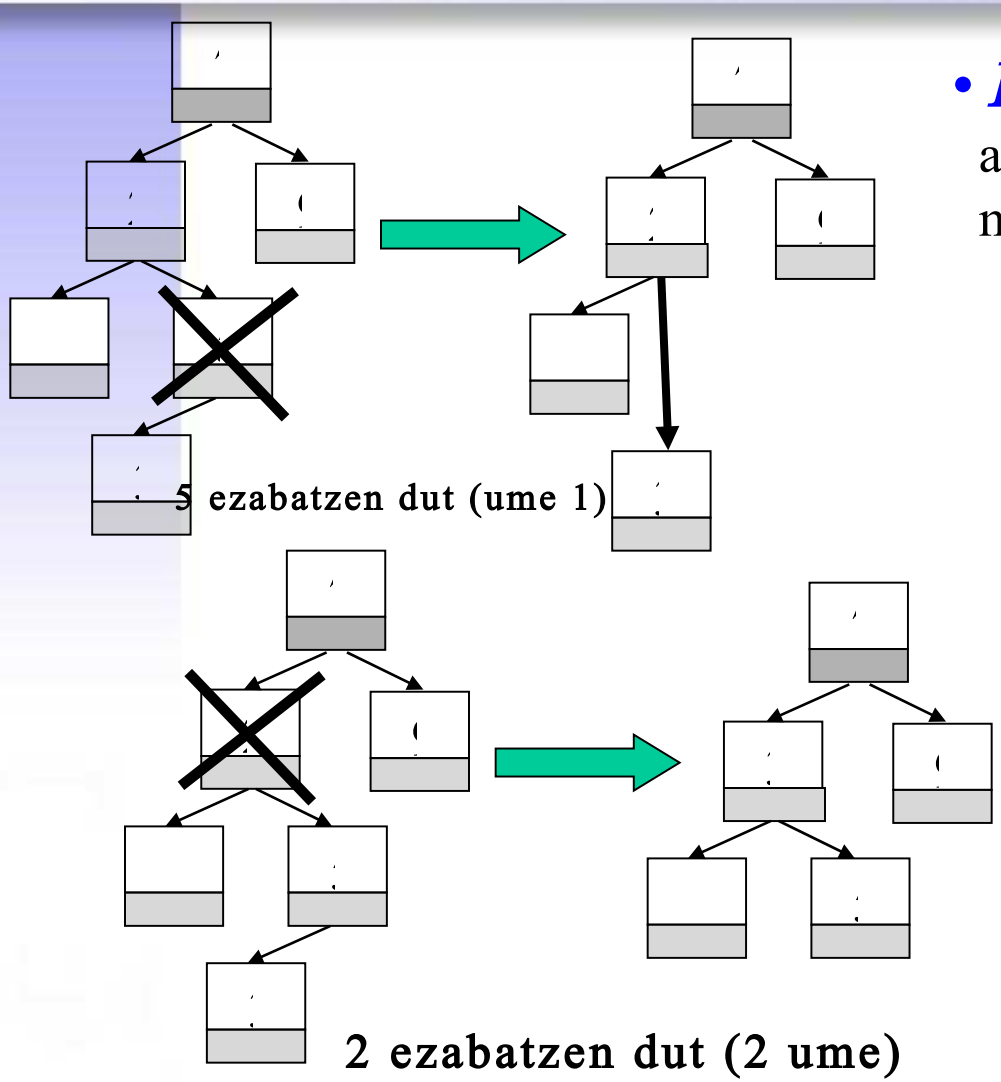
Oinarrizko eragiketak: txikienaren ezabaketa. Inplementazioa

```
public void removeMin() {
 root=removeMin(root);}
```

```
public BTNode removeMin(BTNode<T> pNode) {
 if (pNode.left!=null)
   pNode.left=removeMin(pNode.left);
  else
   pNode=pNode.right;
 return pNode;
}
```

Bilaketa-zuhaitz bitarrak

Oinarrizko eragiketak: Ezabaketa



- ***Ezabaketa*** : konplexua da barne adabegiek zuhaitza konektatuta mantentzen bait dute.Ezabatzeko:
  - Hosto bat bada zuzenean ezabatzen da
  - Ume bakar bat badauka, adabegia ezabatzen da eta bere gurasoa bere umeari seinalatzen dio
  - **Bi ume** badauka :
    - Adabegia aldatzen da eskubiko azpizuhaitzaren elementu txikienengatik
    - Ezabatzen da eskubiko azpizuhaitzaren elementu txikiena

Bilaketa-zuhaitz bitarrak

Oinarrizko eragiketak: Ezabaketa. Inplementazioa

```
public void remove(T x) {
root=remove(x,root);}
```

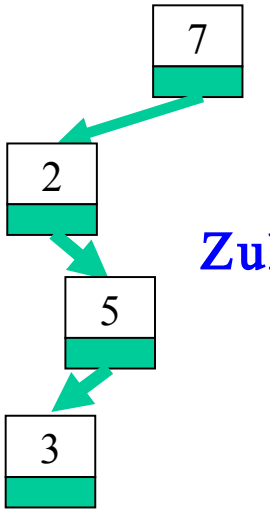
```
public BTNode<T> remove(T x, BTNode<T> pNode) {
  T balioa=pNode.content;
  if (x.compareTo(balioa)>0)
   pNode.right=remove(x,pNode.right);
   else if (x.compareTo(balioa)<0)
     pNode.left=remove(x,pNode.left);
     else
      if (pNode.left!=null && pNode.right!=null) {
       pNode.content=findMin(pNode.right);
       pNode.right=removeMin(pNode.right);
      }
      else
       if (pNode.left!=null) pNode=pNode.left;
        else pNode=pNode.right;
  return pNode;
}
```

Bilaketa-zuhaitz bitarrak

Eragozpenak

Suposatu datuen sarrera hurrengoa dela:

7 2 5 3

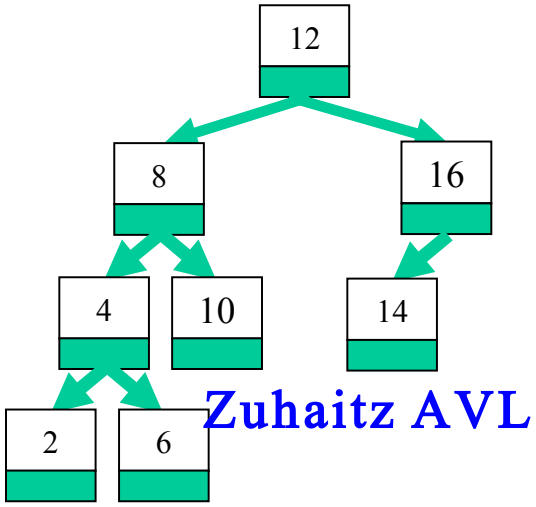


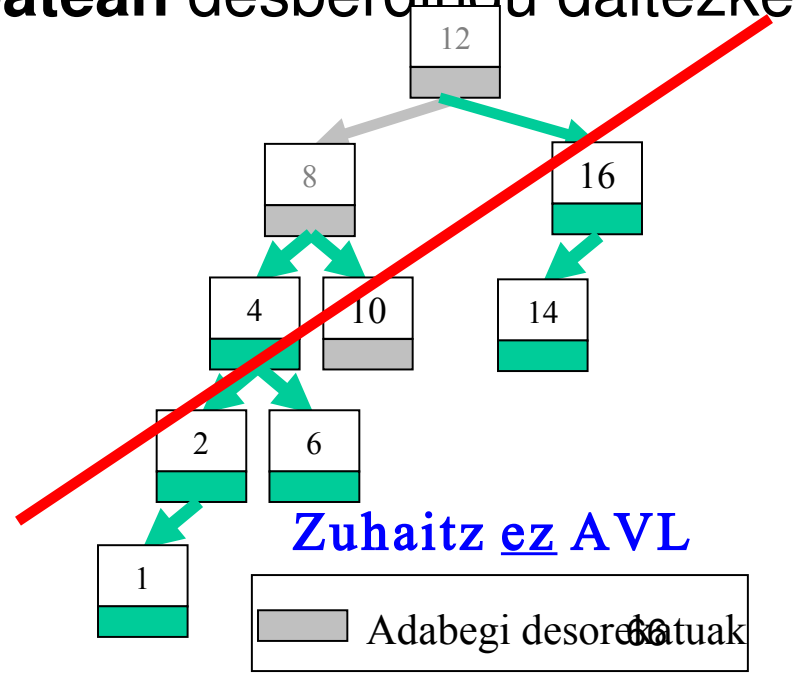
Zuhaitz degeneratua O(N)

Problema: Zuhaitzaren tamaina sarreraren ausazpenaren menpe dago

Zuhaitz AVL
(Adelson-Velskii eta Landis):

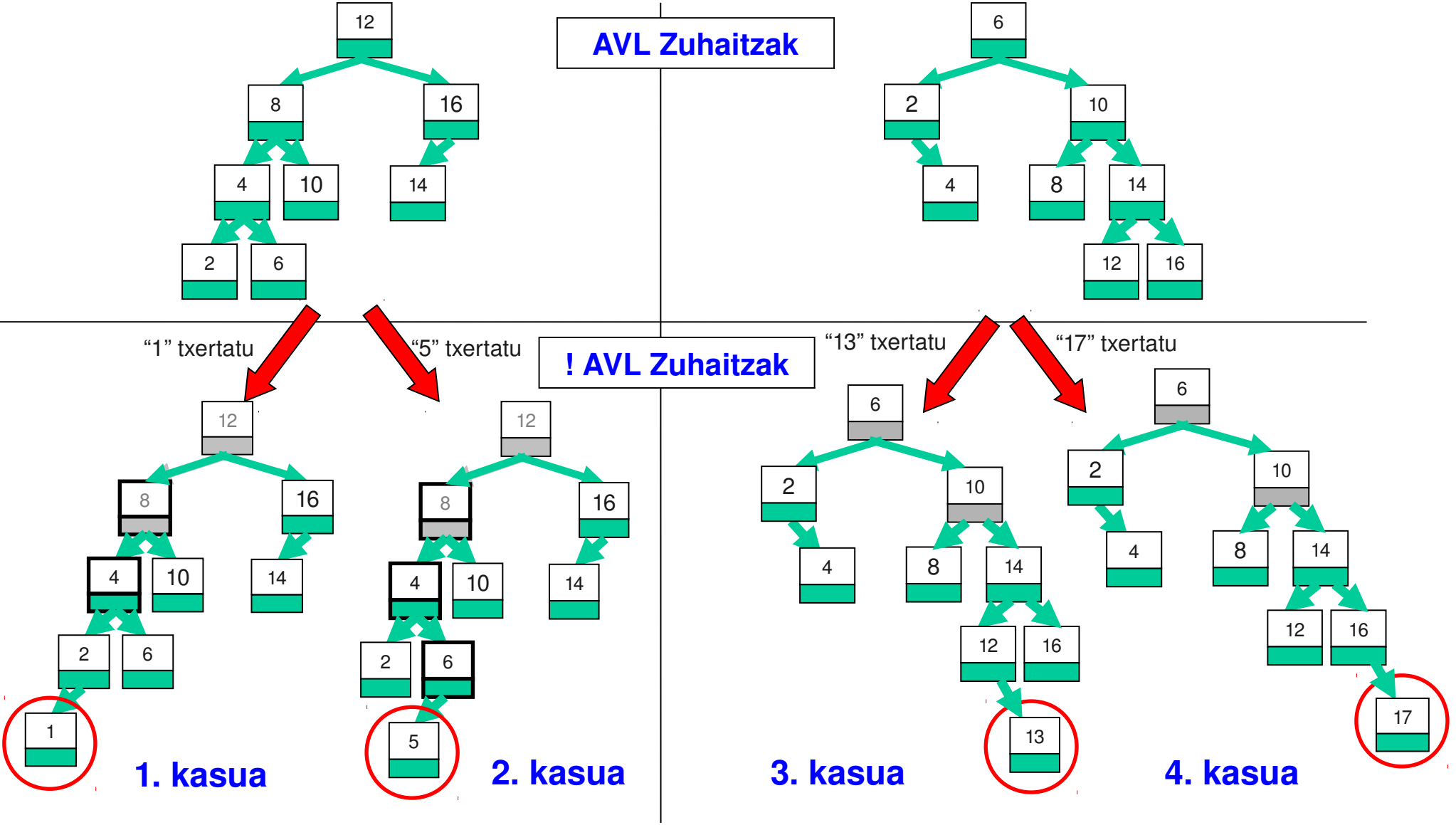
- **Definizioak**:
  - Bilaketa-zuhaitz bitarrak dira, orekazko propietate osagarri batekin:
  - Ezkerreko eta eskubiko umeen **altuerak gehienez unitate batean** desberdindu daitezke





AVL Zuhaitzak

Txertatzerakoan, oreka hausteko 4 aukera ezberdin:

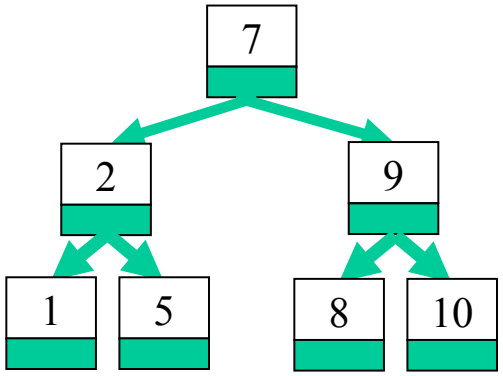


Oreka berreskuratzeko egin behar diren eragiketei buruz gehiago jakiteko:
http://www.c.conclase.net/edd/index.php?cap=008#inicio

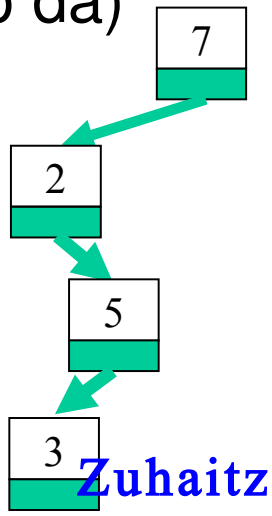
bilaketa-zuhaitz bitarrak

Oinarrizko eragiketen konplexutasuna:

- ***Eragiketa baten konplexutasuna*** proportzionala da azken bisitatutako adabegiaren sakontasunera.
  - Zuhaitza orekatuta badago (AVL) orduan konplexutasuna logaritmikoa izango da(logN) (bilaketa bitarren berdina da)
  - Zuhaitza degeneratuta badago, orduan lineala O(N) (bilaketa lineala izango da)



Zuhaitz orekatua O(log N)



Zuhaitz degeneratua O(N)

Bilaketa-zuhaitz bitarrak

Oinarrizko eragiketen konplexutasuna:

- Oinarrizko eragiketa baten konplexutasuna
  - Bilaketa
  - Txertaketa
  - Ezabaketa

  Proportzionala da kontsultatutako adabegi kopuruari
- Adabegi batera ailegatzeko balioa:

1+ (adabegiaren sakonera)

Emaitzak

Bilaketa-zuhaitz bitarrak

- Bilaketa-zuhaitz bitarrak oso garrantzitsuak dira algoritmoen diseinuetan.
- **Oinarrizko eragiketa** guztiak dauzka:
  - bilaketa
  - txertaketa
  - ezabaketa
- Eragiketa guztiak **denbora logaritmikoan** egiten dira (AVL)