TEMA 4
El tipus conjunt

PROGRAMACIÓ I ESTRUCTURES DE DADES

Tipus conjunt

- 1. Definicions generals
- 2. Diccionari
  - 2.1. Taula de dispersió
- 3. Cua de prioritat
  - 3.1. Montícul
  - 3.2. Cua de prioritat doble
    - 3.2.1. Montícul doble

1. Tipus conjunt

DEFINICIONS

- Un conjunt és una col·lecció d'elements, cada un dels quals pot ser un conjunt o un element primitiu que rep el nom d'àtom.
- Tots els membres del conjunt són distints.
- L'orde dels elements no és important (diferent de les llistes)

Notació de conjunts

- Es representa tancant el seus membres entre claus {1,2,5}
- Relació fonamental, la de pertinença: $\in$ {x / x $\in$ Naturals} , {x / x < 8}
- Hi ha un conjunt especial sense elements: $\varnothing$
- A $\subset$ B si tot element de A també ho és de B
- Conjunt Universal: format per tots els possibles elements que pot contenir

1. Tipus conjunt

SINTAXI

MÒDUL GENERIC ModuloConjunto
MÒDUL Conjunto **USA** Boolean, Natural
SINTAXI

Crear ( ) → Conjunto
Insertar(Conjunto, Ítem) → Conjunto
Eliminar(Conjunto, Ítem) → Conjunto
Pertenece(Conjunto, Ítem) → Boolean
EsVacio(Conjunto) → Boolean
Cardinalidad(Conjunto) → Natural
Unión(Conjunto, Conjunto) → Conjunto
Intersección(Conjunto, Conjunto) → Conjunto
Diferencia(Conjunto, Conjunto) → Conjunto

VAR

C, D: Conjunto; x, y: Ítem;

1. Tipus conjunt

SEMÀNTICA(I)

```
EsVacio( Crear ) = Cierto
EsVacio( Insertar( C, x ) ) = Falso
Insertar( Insertar( C, x ), y ) =
        si ( x == y ) entonces Insertar( C, x ) //no es permeten elements repetits
        sino Insertar( Insertar( C, y ), x )   //dona igual l'ordre d'inserció dels elem
Eliminar( Crear, x ) = Crear
Eliminar( Insertar( C, x ), y ) =
        si ( x == y ) entonces C       // i si tinguerem elements repetits?
        sino Insertar( Eliminar( C, y ), x )
Pertenece( Crear, x ) = Falso
Pertenece( Insertar( C, x ), y ) =
        si ( x == y ) entonces Cierto
        sino Pertenece( C, y )
Cardinalidad( Crear ) = 0
Cardinalidad(Insertar(C,x)) = 1+Cardinalidad(C)
Union( Crear, C ) = C
Union( Insertar( C,x ), D ) =
        si ( Pertenece( D, x ) ) entonces Union( C, D )
        sino Insertar( Union( C, D ), x )
```


1. Tipus conjunt

SEMÀNTICA(II)

```
Diferencia( Crear, C ) = Crear
Diferencia( Insertar( C,x ), D ) =
        si ( Pertenece( D, x ) )
        entonces Diferencia( C, D )
        sino Insertar( Diferencia( C, D ), x )
Interseccion( Crear, D ) = Crear
Interseccion( Insertar( C,x ), D ) =
        si ( Pertenece( D, x ) )
        entonces Insertar( Interseccion( C, D ), x )
        sino Interseccion( C, D )
```

1. Tipus conjunt

IMPLEMENTACIÓ

- **Per mitjà d'un vector**

  -Vector de bits/enters (cada component correspon a un element del conjunt universal)

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

  -Vector de elements

1	9	0	4	2	

- **Emmagatzemar els elements a mesura que s'inserisquen (per mitjà de llistes, arbres, ...):**

  Espai proporcional al conjunt representat

1. Tipus Conjunt

EXERCICI

Complexitat espacial:
Quantitat de recursos espacials (memòria) que un algorisme consumeix o necessita per a la seua execució

- **Complexitats espacials (pitjor cas):**

m=elem. conj. n=elem. conj. Univ.	**Vector de Bits**	**Llista Ordenada**	**Llista desordenada**
Cerca/Inserció/Unió	**O(n)**	**O(m)**	**O(m)**

1. Tipus Conjunt

EXERCICI

- **Emplenar la següent taula de complexitats temporals (pitjor cas):**

m=elem. conj. n=elem. conj. Univ.	Vector de Bits	Llista Ordenada	Llista desordenada
Cerca			
Inserció			
Unió			

2. DICCIONARI

DEFINICIÓ

- **Subtipus del CONJUNT, amb les operacions:**
  - **CREAR**
  - **INSERIR**
  - **ESBORRAR**
  - **CERCA**

2. DICCIONARI

IMPLEMENTACIÓ

- **Implementacions senzilles:**
  Per mitjà de llistes o vectors
  Complexitat temporal de cerca, inserció i esborrat:
  Llistes: O (n)
  Vector bits: O (1)
  Vector elements: O (n)

- **Per mitjà d'una taula de dispersió (HASHING)**

2.1 TAULA DE DISPERSIÓ (HASHING)

DEFINICIÓ

- **HASHING: Utilitzarem la informació de l'element que s'ha emmagatzemat per a buscar-ne la posició dins de l'estructura**

- **Operacions:**
  - **Cerca O(1)**
  - **Inserció O(1)**
  - **Esborrat O(1)**

2.1 TAULA DE DISPERSIÓ (HASHING)

MÈTODE

- Dividir el conjunt en un nombre finit "B" de classes.
- S'usa funció de dispersió H, tal que H(x) serà un valor entre 0 i B-1

Formes de dispersió:

- Oberta: No imposa limit de grandària al conjunt.
- Tancada: usa un grandària fixa d'emmagatzemament (limita la grandària).

2.1 Taula Hash. Dispersió tancada

DEFINICIÓ

- **Els elements s'emmagatzemen en taula de grandària fixa (TAULA DE DISPERSIÓ).**
- **La taula es divideix en B classes, i cada una podrà emmagatzemar S elements. En els nostres exemples, S=1**
- **La funció de dispersió s'implementa per mitjà d'una funció aritmètica. Exemple:**

$$H(x) = x \ MOD \ B$$

2.1 Taula Hash. Dispersió tancada

INSERCIÓ

Cas **col·lisió**: x1, x2 (SINÒNIMS/ H(x1) = H(x2))

ESTRATÈGIA DE REDISPERSIÓ:

Triar successió de localitats alternes dins de la taula, fins a trobar-ne una buida

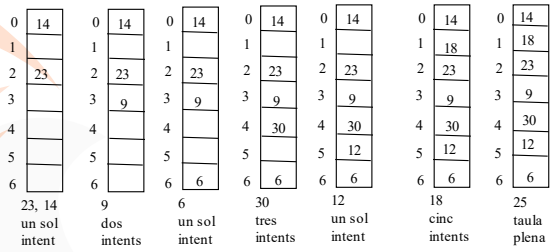
H(x), h1(x), h2(x), h3(x), … , h[B-1](x)

Si cap està buida: no és possible inserir

2.1 Taula Hash. Dispersió tancada

INSERCIÓ. EXEMPLE

Exemple. Inserir en una taula de dispersió de grandària *B=7*, amb funció de dispersió *H(x)= xMOD B*, i amb estratègia de redispersió *la següent posició de la taula*, els elements següents: *23, 14, 9, 6, 30, 12, 18, 25*



NOMBRE TOTAL D'INTENTS FINS A LA CLAU 18:

14

2.1 Taula Hash. Dispersió tancada

BUSCA. ESBORRAT

CERCA D'ELEMENTS

Cercar en successió de localitats alternes dins de la taula, fins a trobar-ne una buida:

H(x), h1(x), h2(x), h3(x), ...

ESBORRAT D'ELEMENTS

Cal distingir durant la cerca:

- Caselles buides
- Caselles suprimides

Durant la inserció les caselles suprimides es tractaran com a espai disponible.

2.1 Taula Hash. Dispersió tancada

ANÀLISI (I)

- *ESTRATÈGIA DE REDISPERSIÓ LINEAL ("següent posició")*:

  - No eficient. Llarga seqüència d'intents

  $h_i(x) = (H(x) + 1 \cdot i) \; MOD \; B$ / c=1 $\quad \boldsymbol{h_i(x) = (h_{i-1}(x) + 1) \; MOD \; B}$

- *ESTRATÈGIA DE REDISPERSIÓ ALEATÒRIA*:

  $h_i(x) = (H(x) + c \bullet i) \; MOD \; B$ / c>1 $\quad \boldsymbol{h_i(x) = (h_{i-1}(x) + c) \; MOD \; B}$

Continua produint AMUNTONAMENT (següent intent només en funció de l'anterior)

c i B no han de tindre factors prims comuns majors que 1

- *E.R. AMB 2a FUNCIÓ D'HASH*:

$k(x) = (x \; MOD \; (B-1)) + 1$

$h_i(x) = (H(x) + k(x) \bullet i) \; MOD \; B$ $\quad \boldsymbol{h_i(x) = (h_{i-1}(x) + k(x)) \; MOD \; B}$

B ha de ser prim

2.1 Taula Hash. Dispersió tancada

ANÀLISI (II)

- *LA MILLOR FUNCIÓ DE DISPERSIÓ*:
  - Que siga fàcil de calcular
  - Que minimitze el nombre de col·lisions
  - Que distribuïsca els elements de forma atzarosa
  - Ha de fer ús de tota la informació associada a les etiquetes

2.1. Taula Hash. Dispersió Tancada

ANÀLISI (III)

- Estratègia de redispersió aleatòria
  - c i B no han de tindre factors prims comuns majors que 1 per tal de buscar en totes les posicions de la taula
    - Exemple→ c=4; B=6

      $h_i(x) = (H(x) + c \bullet i) \text{ MOD } B = (h_{i-1}(x) + c) \text{ MOD } B$

      H(10)=10 MOD 6=4

      $h_1(10)=(4+4 \bullet 1) \text{ MOD } 6=(4+4) \text{ MOD } 6=2$

      $h_2(10)=(4+4 \bullet 2) \text{ MOD } 6= 12 \text{ MOD } 6=0$

      $h_3(10)=(4+4 \bullet 3) \text{ MOD } 6=4$; $h_4(10)=(4+4 \bullet 4) \text{ MOD } 6=2$

X		X		X	
0	1	2	3	4	5

    - Exemple→ ¿c=6; B=9? ¿c=2; B=9?

- **Estratègia de redispersió amb 2ª funció hash**
  - B ha de ser prim per a que es busque en totes les posicions de la taula
    - c=k(x) → 1…B-1 // k (x) = (x MOD (B-1)) + 1

      $h_i(x) = (H(x) + k(x) \bullet i) \text{ MOD } B = (h_{i-1}(x) + k(x)) \text{ MOD } B$

      B=7; k(x)=1…6 però B=6; k(x) = 1 ... 5 (**inclou 4**)

      B=11;k(x)=1…10

2.1. Taula Hash. Dispersió Tancada

EXERCICIS

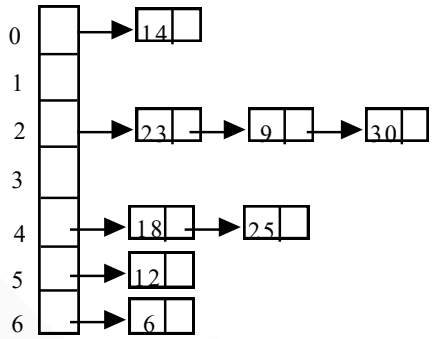
1) **Insertar en una taula de dispersió tancada de grandària *B=7*, amb funció de dispersió *H(x)= x MOD B*, i amb estratègia de redispersió *segona funció hash*, els següents elements: *23, 14, 9, 6, 30, 12, 18***

2.1 Taula Hash. Dispersió oberta

DEFINICIÓ

Elimina problema del CLUSTERING SECUNDARI (col·lisions entre claus no sinònimes)

Les col·lisions es resolen utilitzant una llista enllaçada



2.1 Taula Hash

FACTOR DE CÀRREGA (I)

$$\alpha = \frac{n}{|B|}$$

n = nº elem. de la taula. B = grandària de la taula

HASH TANCAT: $0 \leq \alpha \leq 1$

HASH OBERT: $\alpha \geq 0$ (No hi ha límit en el nombre d'elements en cada casella).

2.1 Taula Hash

FACTOR DE CÀRREGA (II)

E: Nombre esperat d'intents en cerca en el pitjor cas
c.éx: amb èxit. s.éx: sense èxit.

	H.C.L.		H.C. Aleat.		H. Obert	
α	**E c.ex**	**E s.ex.**	**E c.ex**	**E s.ex.**	**E c.ex**	**E s.ex.**
0.1	1.06					
0.25	1.17					
0.5	1.50					
0.75	2.50	8.5	1.9	4.0	1.8	2.0
0.9	5.50	50.5	2.6	10.0	1.9	2.0
0.95	10.50					

2.1 Taula Hash

COMPARACIÓ HASH OBERT I TANCAT

- H.O. és més eficient i amb menor degradació (quant més ple funciona millor que el HT)

- H.O. requereix espai per als elements de la llista, per la qual cosa H.T. és més eficient espacialment.

•Reestructuració de les taules de dispersió:

$n \geq 0{,}9\ B$ (H.T.)

$n \geq 2\ B$ (H.O.)

→ Nova taula amb el doble de posicions

Problemes:

- Les restriccions de sobre B i c de les estratègies de redispesió aleatòria i de segona funció de Hash s'han de complir
- Cal tornal a calcula H(x) i tornar a inserir tots els elements

3. CUA DE PRIORITAT

DEFINICIÓ (I)

- Conjunt d'elements ordenats amb les operacions:

  Crear ( ) → ColaPrioridad

  EsVacio () → Boolean

  Insertar (ColaPrioridad, Item) → ColaPrioridad

  BorrarMínimo (ColaPrioridad) → ColaPrioridad

  BorrarMáximo (ColaPrioridad) → ColaPrioridad

  Búsqueda (ColaPrioridad, Item) → Boolean

  Cardinalidad (ColaPrioridad) → Natural

  Copiar (ColaPrioridad) → ColaPrioridad

  Mínimo (ColaPrioridad) → Item

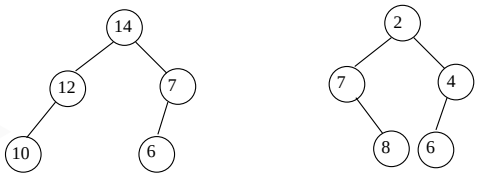
  Máximo (ColaPrioridad) → Item

3. CUA DE PRIORITAT

DEFINICIÓ (I)

- *Arbre mínim (màxim):*

Arbre en què l'etiqueta de cada node és menor (major) que la dels fills.

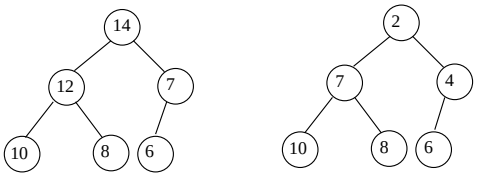


3. CUA DE PRIORITAT

DEFINICIÓ (II)

- *Heap mínim (màxim):*

Arbre binari complet que a més, és ARBRE MÍNIM o MÀXIM.



HEAP MÀXIM HEAP MÍNIM

3. CUA DE PRIORITAT

DEFINICIÓ (III)

- *Implementació cua prioritat:*
  - LLISTA DESORDENADA:
    INSERCIÓ: O(1)
    ESBORRAT: O(n)
  - LLISTA ORDENADA: ascendent o descendentement.
    INSERCIÓ: O(n)
    ESBORRAT: O(1)
  - ARBRE BINARI DE CERCA:
    INSERCIÓ: O(n)
    ESBORRAT: O(n)

3. CUA DE PRIORITAT

DEFINICIÓ (IV)

- *Implementació cua prioritat:*
  - **HEAP o MONTICLE:**
    INSERCIÓ: O(log n)
    ESBORRAT: O(log n)

3.1 HEAP MÀXIM (MÍNIM)

INSERCIÓ

- *MÈTODE:*

1.- Inserir en la posició corresponent perquè continue sent un arbre complet.

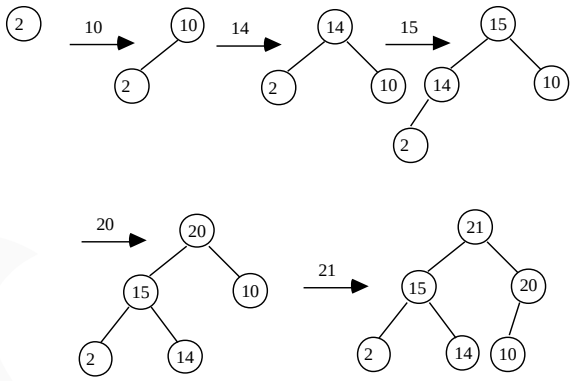
2.- Reorganitzar perquè complisca les condicions de HEAP:

Comparar amb el node pare: si no acompleix les condicions d'arbre mínim/màxim, aleshores intercanviar els dos

3.1 HEAP MÀXIM (MÍNIM)

INSERCIÓ. EXEMPLE

- *Inserir: 2, 10, 14, 15, 20 i 21 en un heap màxim inicialment buit*



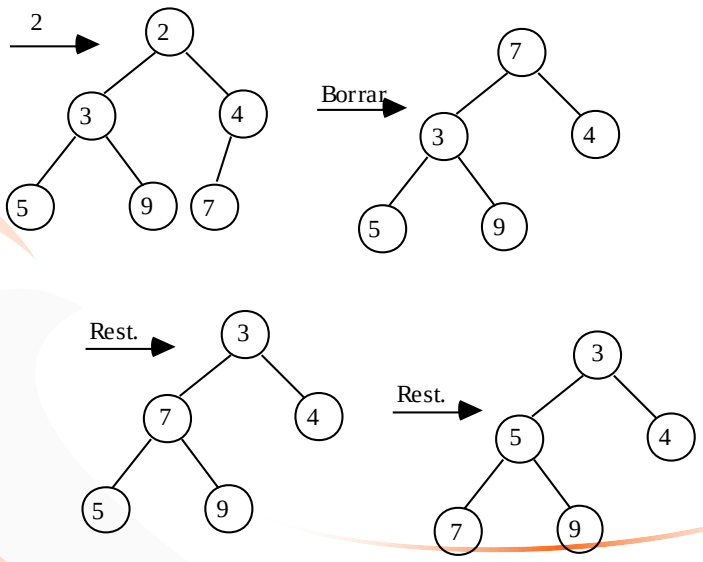
3.1 HEAP MÀXIM (MÍNIM)

ESBORRAT.

- *Se substitueix l'arrel amb l'element més a la dreta en l'últim nivell*
- *Mentre no siga un HEAP s'afona aquest element substituint-lo amb el més gran (montícul màxim) o més menut(montícul mínim) del seus fills*

3.1. HEAP MÀXIM (MÍNIM)

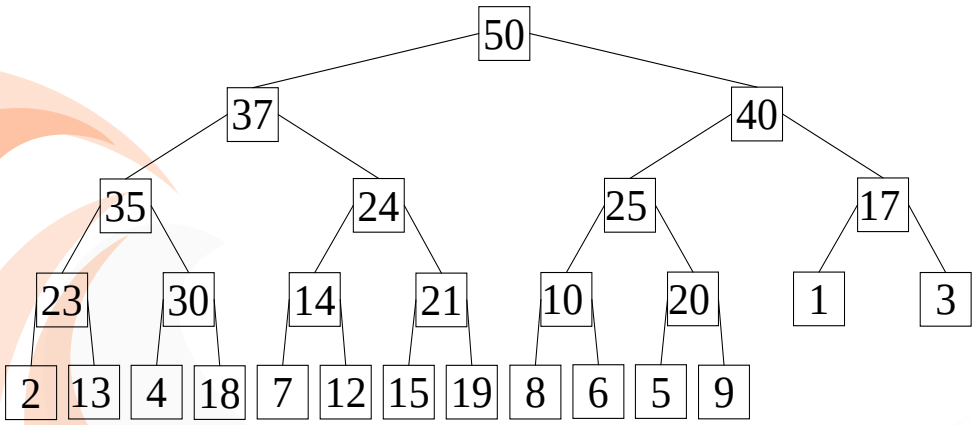
ESBORRAT. EXEMPLE



3.1. HEAP MÀXIM (MÍNIM)

ESBORRAT. EXEMPLE II

- Realitza dos esborrats sobre el següent montícul màxim.
- Sobre el resultat anterior, realitza les insercions: 60, 36



3.1. HEAP MÀXIM (MÍNIM)

INSERCIÓ. EXEMPLE II

- Sobre el resultat anterior, realitza les insercions: 60, 36

3.1 HEAP MÀXIM (MÍNIM)

REPRESENTACIÓ

- ENLLAÇADA: problema en inserció ja que necessita fer recorreguts ascendents.

- SEQÜENCIAL(en un vector):
  Fills de p[i] són p[2•i] i p[2•i+1]. Pare de p[i] és p[i DIV 2] amb DIV la divisió entera

3.1 HEAP MÀXIM (MÍNIM)

APLICACIÓ: HEAPSORT

Algorisme d'ordenació d'un vector d'elements.

MÈTODE:

1) Inserir els elements en un HEAP
2) Realitzar esborrats de l'arrel del HEAP

IMPLEMENTACIÓ (UN SOL VECTOR):

1) Deixar la part esquerra del vector per al HEAP, i la part dreta per als elements encara no inserits.
2) Esborrar l'arrel del HEAP portant-la a la part dreta del vector.

COMPLEXITAT:

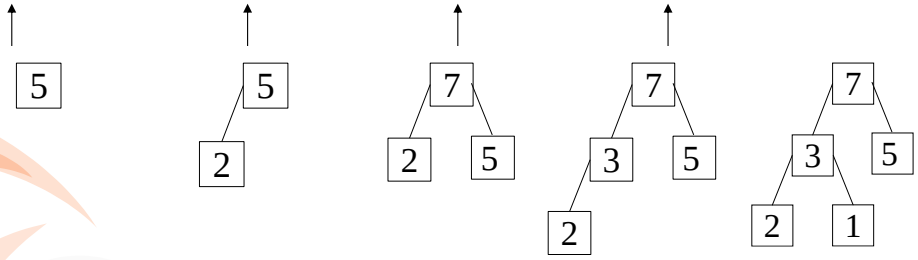
O (n log n)

3.1. HEAP MÀXIM (MÍNIM)

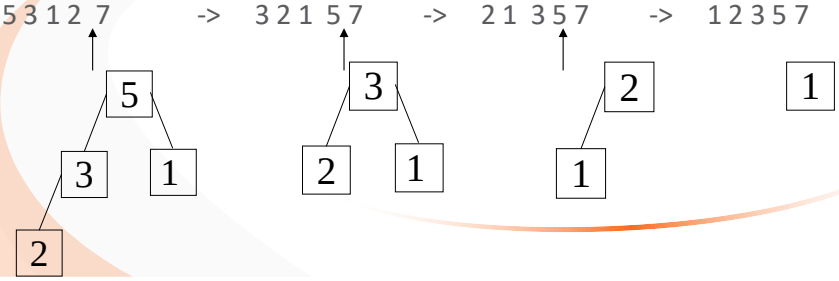
HEAPSORT. EXERCICI

- Ordenar el vector 5 2 7 3 1 utilitzant un heap màxim

1) 5 2 7 3 1 -> 5 2 7 3 1 -> 7 2 5 3 1 -> 7 3 5 2 1 -> 7 3 5 2 1



2) 5 3 1 2 7 -> 3 2 1 5 7 -> 2 1 3 5 7 -> 1 2 3 5 7



3.1. HEAP MÀXIM (MÍNIM)

HEAPSORT. EXERCICI

- Ordenar el vector 9 5 7 4 8 6 2 1 utilitzant un heap mínim

3.2 CUA DE PRIORITAT DOBLE (DEAP)

DEFINICIÓ (I)

Cua de Prioritat doble: TAD cua de prioritat en la qual es suporta l'operació d'esborrat de la clau màxima i mínima.

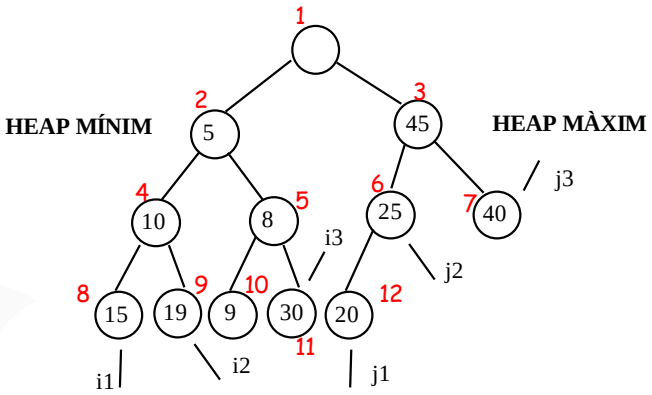
DEAP: És un Heap que suporta les operacions de cua de prioritat doble.

DEFINICIÓ: és un arbre binari complet el qual o és buit o satisfà les propietats següents:

1) L'arrel no conté elements
2) El subarbre esquerre és un HEAP mínim
3) El subarbre dret és un HEAP màxim
4) Si el subarbre dret no és buit:
   - Siga "i" qualsevol node del subarbre esquerre.
   - Siga "j" el node corresponent en el subarbre dret. Si "j" no existeix, llavors siga el node del subarbre dret que correspon al pare de "i".
   - Aleshores: clau (i) < clau (j)

3.2 CUA DE PRIORITAT DOBLE (DEAP)

DEFINICIÓ (II)



3.2 CUA DE PRIORITAT DOBLE (DEAP)

IMPLEMENTACIÓ

Igual que en un HEAP, encara que la primera posició no s'utilitzarà.

$$j = i + 2^{\lceil \log_2 i \rceil - 1}$$

Si j > n aleshores j = j **DIV** 2

Exemple:
simètric de i1 (i=8). $j = 8 + 2^{(\log_2 8) - 1} = 8 + 2^2 = \underline{\mathbf{12}}$
simètric de i2 (i=9). $j = 9 + 2^{(\log_2 9) - 1} = 9 + 2^2 = 13$. com j>n --> j = 13 DIV 2 = **6**

3.2 CUA DE PRIORITAT DOBLE (DEAP)

INSERCIÓ

1) S'insereix l'element en el següent índex de l'arbre complet
2) Es compara el node inserit amb el node simètric corresponent, fent l'intercanvi en cas que no es complisca la condició 4 de la definició del DEAP:

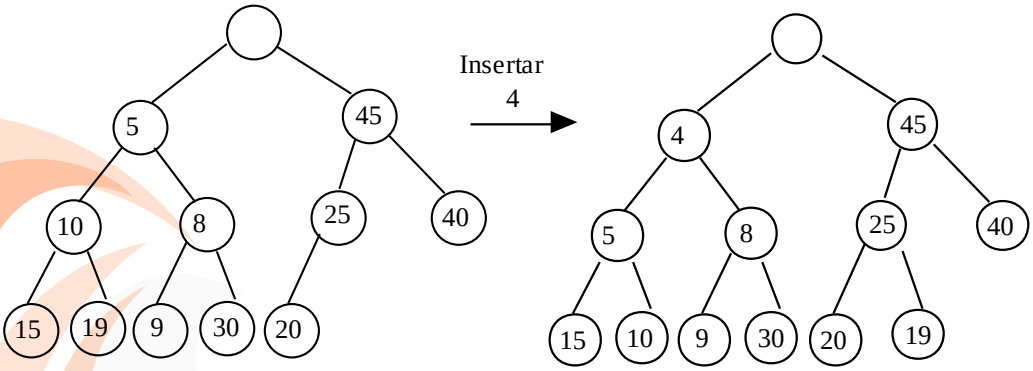
$$i = n - 2^{\lceil \log_2 n \rceil - 1}$$

3) Actualitzar el HEAP per mitjà del procés "d'ascensió" de l'element inserit. **[Aquest pas s'ha d'executar també si no s'ha fet intercanvi en el pas 2]**

Exemple:
simétric de j1 (j=12). $i = 12 - 2^{(\log_2 12) - 1} = 12 - 2^2 = \underline{\mathbf{8}}$

3.2. CUA DE PRIORITAT DOBLE (DEAP)

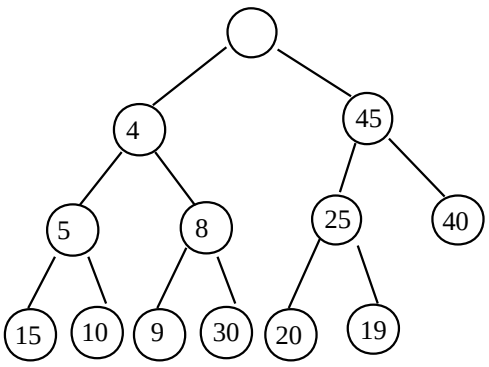
INSERCIÓ



3.2. CUA DE PRIORITAT DOBLE (DEAP)

INSERCIÓ

- Sobre el DEAP següent insertar: 35, 7, 50, 17, 12, 27, 55



3.2 CUA DE PRIORITAT DOBLE (DEAP)

ESBORRAT

1) Intercanviar l'arrel del HEAP mínim (o màxim) amb l'element més a la dreta de l'últim nivell de l'arbre, i esborrar aquest.

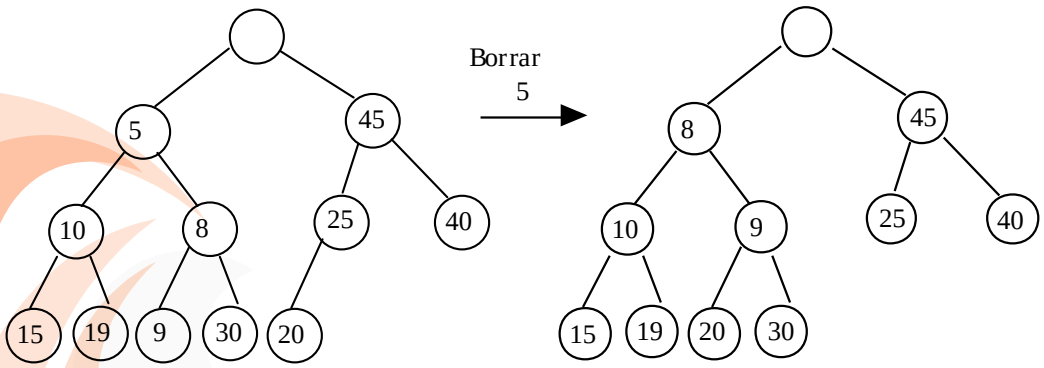
2) Actualitzar HEAP mínim (o màxim), "afonant" la clau intercanviada.

3) Comprovar que la clau intercanviada no incomplisca la condició del DEAP amb el seu corresponent node simètric, i intercanviar si cal

4) Actualitzar el monticle en què quede la clau intercanviada

3.2. CUA DE PRIORITAT DOBLE (DEAP)

ESBORRAT



3.2. CUA DE PRIORITAT DOBLE (DEAP)

ESBORRAT

MONTÍCUL DOBLE: Esborrar els elements mínim i màxim de forma succesiva.