estudios de postgrado
en computación

Análisis y Diseño de Algoritmos (AyDA)

Isabel Besembel Carrera

MONTÍCULOS BINOMIALES Y DE FIBONACCI CONJUNTOS DISJUNTOS

Montículos binomiales

- **Implementan eficientemente la mezcla de dos colas por prioridad**
- **Se componen de un bosque de árboles binomiales**
- **Árbol binomial de altura Br es un árbol donde:**
  - **Bo consiste en un único nodo**
  - **Br se compone de dos árboles de Br-1, donde uno de los dos está adosado a la raíz del otro como su hijo extremo derecho**

Propiedades de los árboles binomiales

- Hay 2^r nodos
- Altura r
- Hay exactamente $\binom{r}{i}$ nodos de profundidad i=0, 1, 2,…, r
- La raíz tiene grado r mayor que cualquier grado de otro nodo en Br. Si los hijos de la raíz se numeran de izquierda a derecha por r-1, r-2, …, 0, el hijo i es la raíz del subárbol Bi
- El máximo grado de cualquier nodo en un árbol binomial de n nodos es lg n

Mayo. 2000

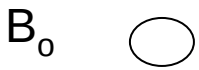
Especificación Clase MontMezcla

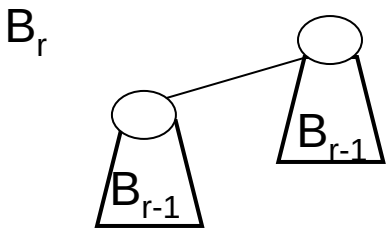
Clases: NodoMont, TipoClave, Boolean

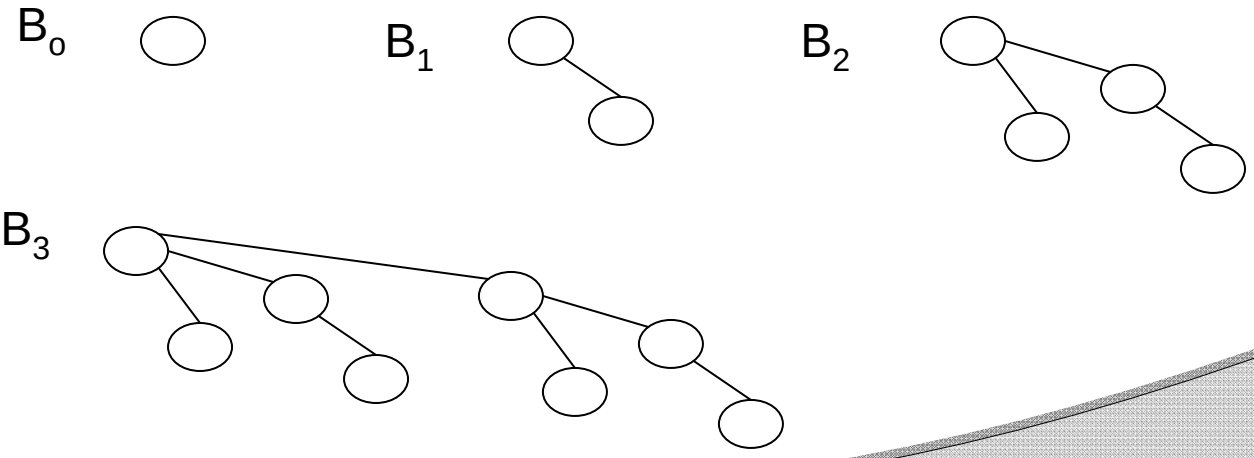
1	*Especificación sintáctica:* crea()→MontMezcla, inserta(MontMezcla,NodoMont) .. →MontMezcla, elimina(MontMezcla,NodoMont) .. →MontMezcla, min(MontMezcla)→NodoMont, extMin(MontMezcla)→NodoMont, union(MontMezcla,MontMezcla)→ .. MontMezcla, decreClave(MontMezcla,NodoMont, .. TipoClave)→Boolean, destruye(MontMezcla)→.	-*crea():* Crea un montículo vacío. -*inserta():* Ingresa un nuevo nodo con la clave nueva. -*elimina():* Elimina el nodo dado. -*min():* Regresa la referencia al nodo con clave mínima. -*extMin():* Elimina el nodo con clave mínima regresando su referencia. -*union():* Crea y regresa un nuevo montículo que contiene todos los nodos de ambos montículos y destruye los argumentos. *decreClave():* Cambia la clave del nodo por la dada, si ella es menor. *destruye():* Destruye el montículo.
2	*Variables:* MontMezcla: mb TipoClave: k NodoMont: nb	
3	*Especificación semántica:* min(crea()) = Nulo extMin(crea()) = Nulo min(inserta(crea(),nb)) = nb extMin(inserta(crea(),nb)) = nb	

Operación	Montículo binario W(n)	Montículo binomial W(n)
crear()	$\Theta(1)$	$\Theta(1)$
inserta()	$\Theta(\lg n)$	$O(\lg n)$
min()	$\Theta(1)$	$O(\lg n)$
extMin()	$\Theta(\lg n)$	$\Theta(\lg n)$
union()	$\Theta(n)$	$O(\lg n)$
decreClave()	$\Theta(\lg n)$	$\Theta(\lg n)$
elimina()	$\Theta(\lg n)$	$\Theta(\lg n)$

Árboles binomiales



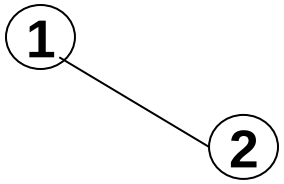




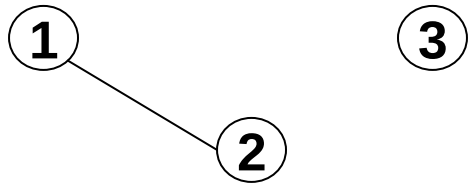
Montículos binomiales

- **Un montículo binomial (H) es un montículo mezclable conformado por un conjunto de árboles binomiales que satisfacen:**
  1. **Cada árbol binomial en H es un montículo ordenado. La clave del nodo X es mayor o igual a la clave de su padre, por lo que la clave de la raíz es la menor**
  2. **Hay a lo sumo un árbol binomial en H cuya raíz es de grado dado. H consiste de a lo más $\lfloor \lg n \rfloor + 1$ árboles binomiales**

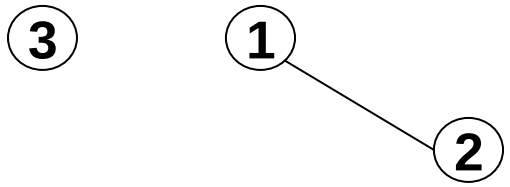
Montículos binomiales



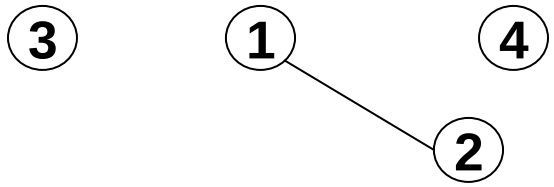
Montículos binomiales



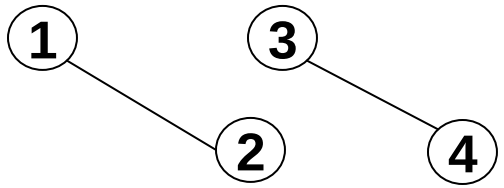
Montículos binomiales



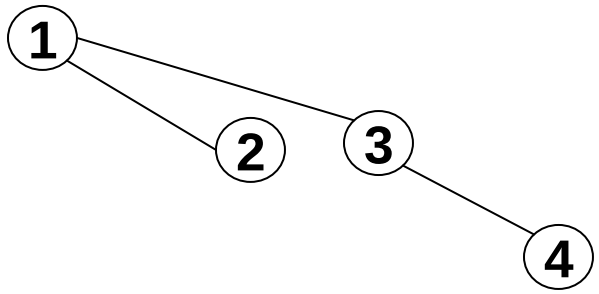
Montículos binomiales



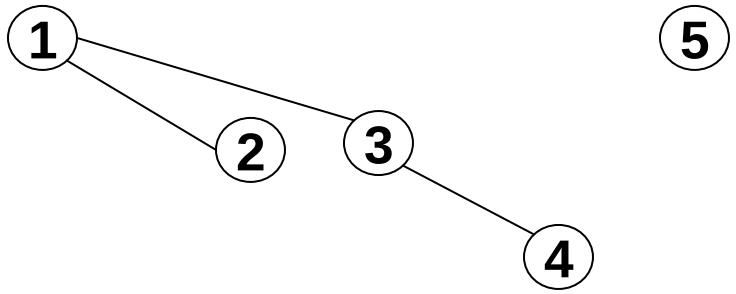
Montículos binomiales



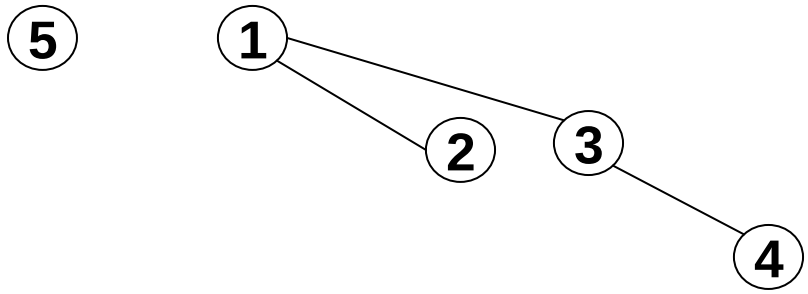
Montículos binomiales



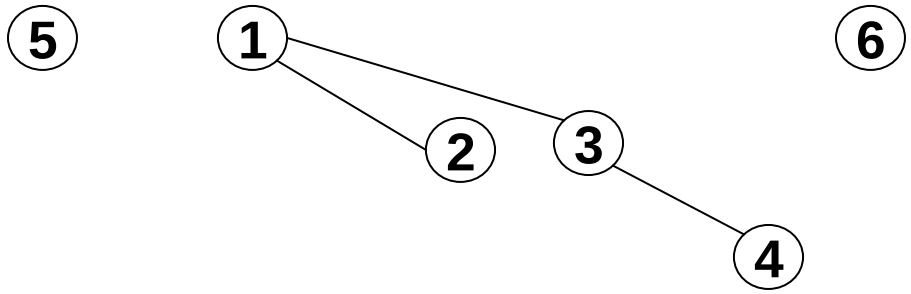
Montículos binomiales



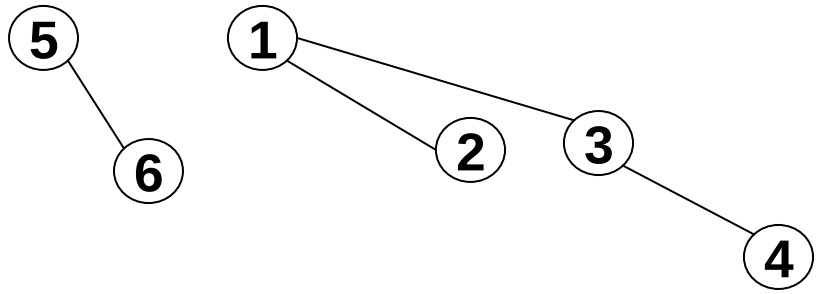
Montículos binomiales



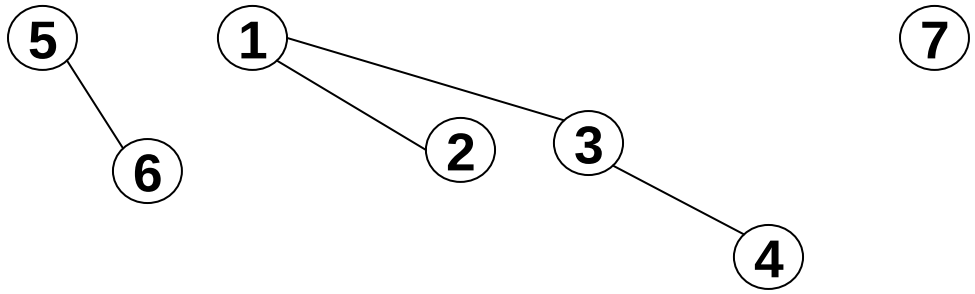
Montículos binomiales



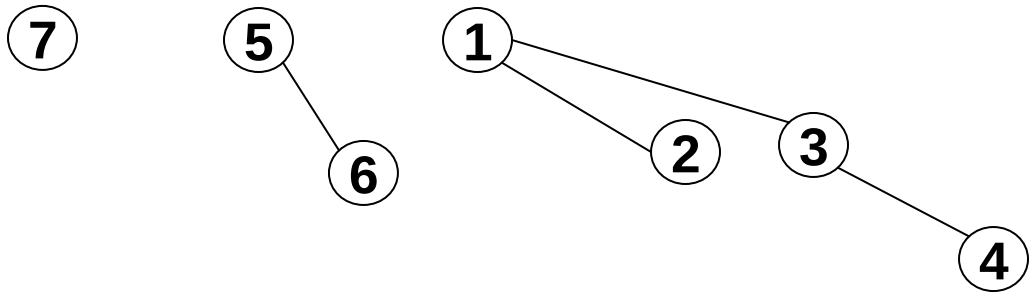
Montículos binomiales



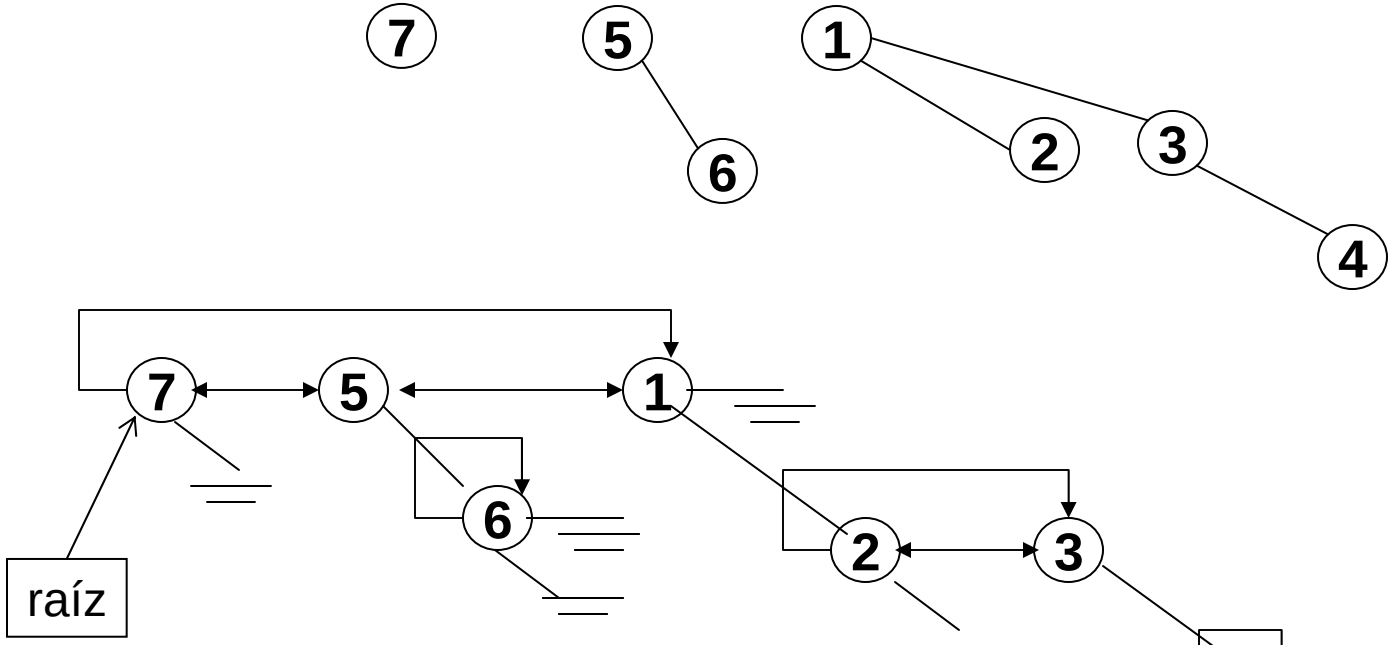
Montículos binomiales



Montículos binomiales

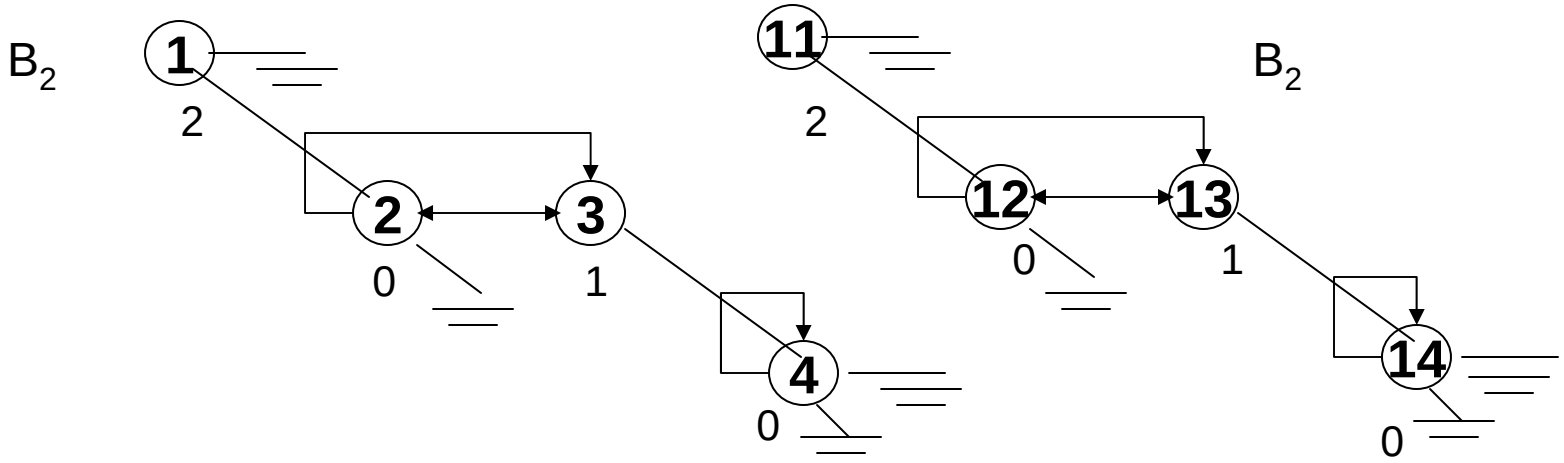


Montículos binomiales



Implementación
con lista de árboles

Unión de montículos binomiales



Implementación

NodoMont: p clave grado hijo hermano

p: Apuntador al nodo padre

clave: Clave del nodo

grado: Número de hijos del nodo

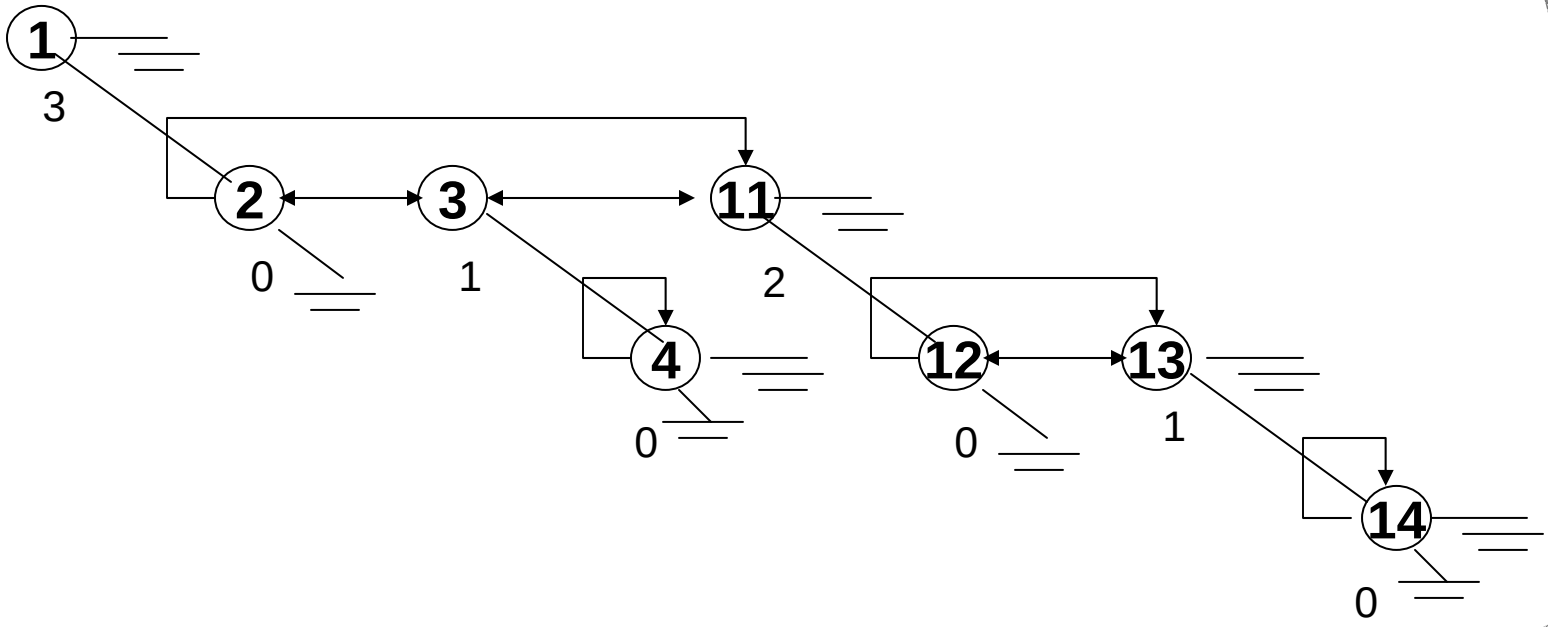
hijo: Apuntador al primer nodo hijo

hermano: Apuntador al nodo hermano derecho

Unión en 2 fases:

1. Mezcla las 2 listas de raíces siguiendo el grado de sus nodos en forma ascendente
2. Mezcla las raíces con igual grado hasta tener una raíz de cada grado

Unión de montículos binomiales



Montículos de Fibonacci

- Es un montículo mezclable conformado por una colección de árboles, pero sin relación de orden entre ellos
- Se recomiendan cuando el número de extMin() o extMax() y elimina() es menor en relación al número de ocurrencias de las otras operaciones
- Diferencia con los binomiales: los de Fibonacci tienen una estructura menos rígida, cuyo mantenimiento se retrasa hasta que sea conveniente, permitiendo una menor complejidad en tiempo

Formato del nodo

NodoMont: | p clave grado marca hijo izq der |

p: Apuntador al nodo padre

clave: Clave del nodo

grado: Número de hijos del nodo

marca: Es verdadero si el nodo ha perdido un hijo durante el periodo en que él fue hecho hijo de otro nodo. Los nodos recién creados tienen marca Falso.

hijo: Apuntador a un nodo hijo

izq: Apuntador al nodo hermano a la izquierda

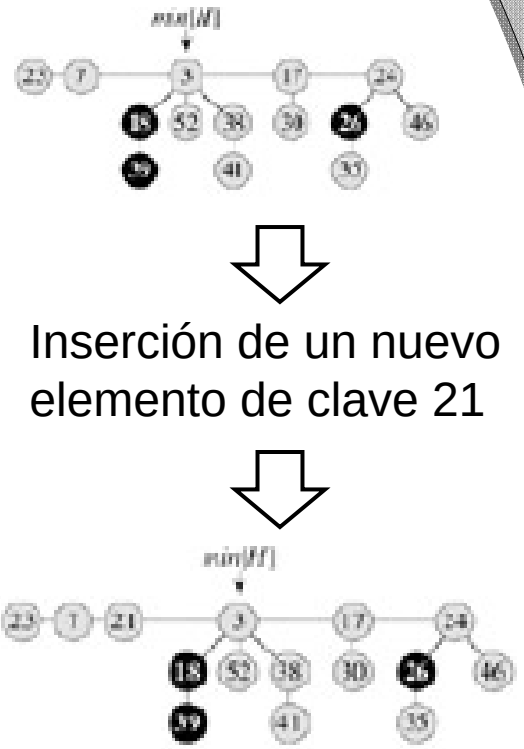
der: Apuntador al nodo hermano a la derecha

Mayo 2000

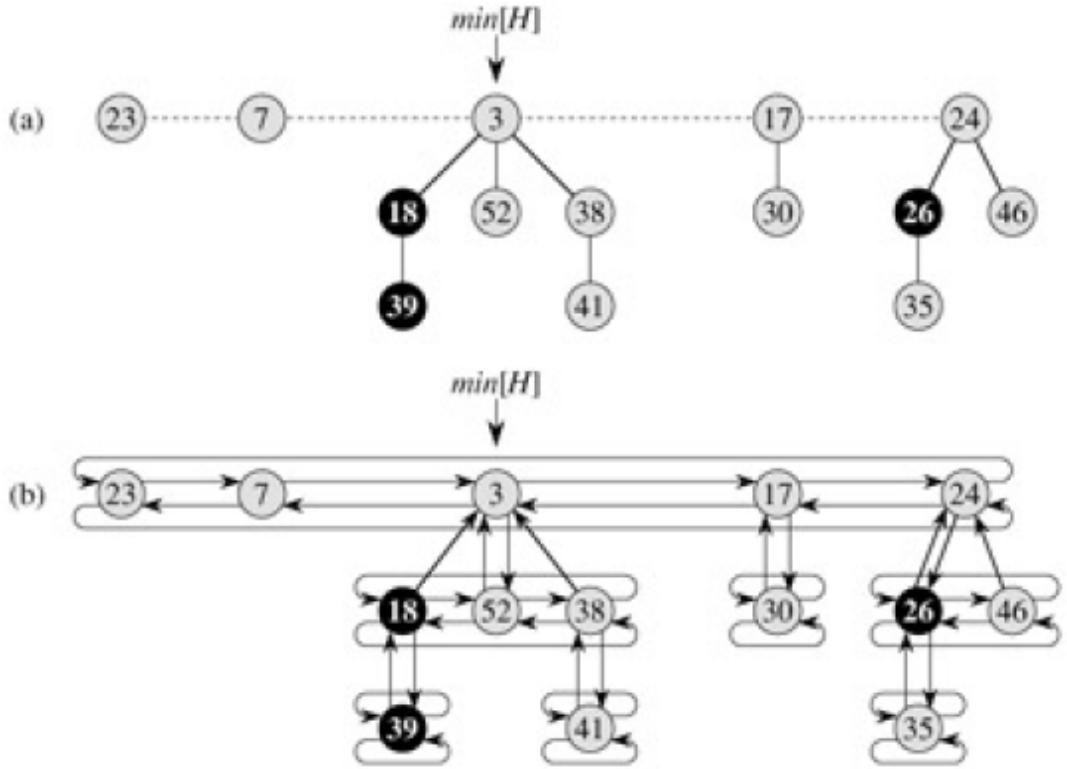
Implementación Clase MontFibo

Classes: NodoMont, ApuntadorA, TipoClave, Entero+

1	*Superclases:* MontMezcla	-**pMin**. Referencia a la lista de raíces.
2	*Estructura:* Privado: pMin: ApuntadorA NodoMont n: Entero+	-**montFibo()**: *Constructor.*
3	*Funciones:* Público montFibo() min(): ApuntadorA NodoMont union(MontFibo: h2): MontFibo inserta(ApuntadorA NodoMont: px) extMin(): ApuntadorA NodoMont decreClave(ApuntadorA NodoMont: .. pa, TipoClave: k) elimina(ApuntadorA NodoMont: pa) Privado: concatena(ApuntadorA .. NodoMont: p) consolida(ApuntadorA .. NodoMont: p) corte(ApuntadorA .. NodoMont: px, py) propCorte(ApuntadorA .. NodoMont: py)	-**min()**:*Observador.* Regresa la referencia al nodo mínimo. -**union()**: *Transformador.* Une dos montículos en uno. -**inserta()**: *Transformador.* Ingresa un nuevo nodo a la estructura. -**extMin()**: *Transformador.*Saca y regresa el nodo mínimo. -**decreClave()**: *Transformador.*Cambia la clave del nodo enviado. -**elimina()**: *Transformador.* Elimina el nodo enviado. -**concatena()**: *Transformador.* Concatena dos listas de raíces. -**n()**. Número actual de nodos. -**consolida()**: *Transformador.* Actualiza la lista de raíces. -**corte()**: *Transformador.* Coloca px como raíz. -**propCorte()**: *Transformador.* Propaga los cortes hacia arriba.



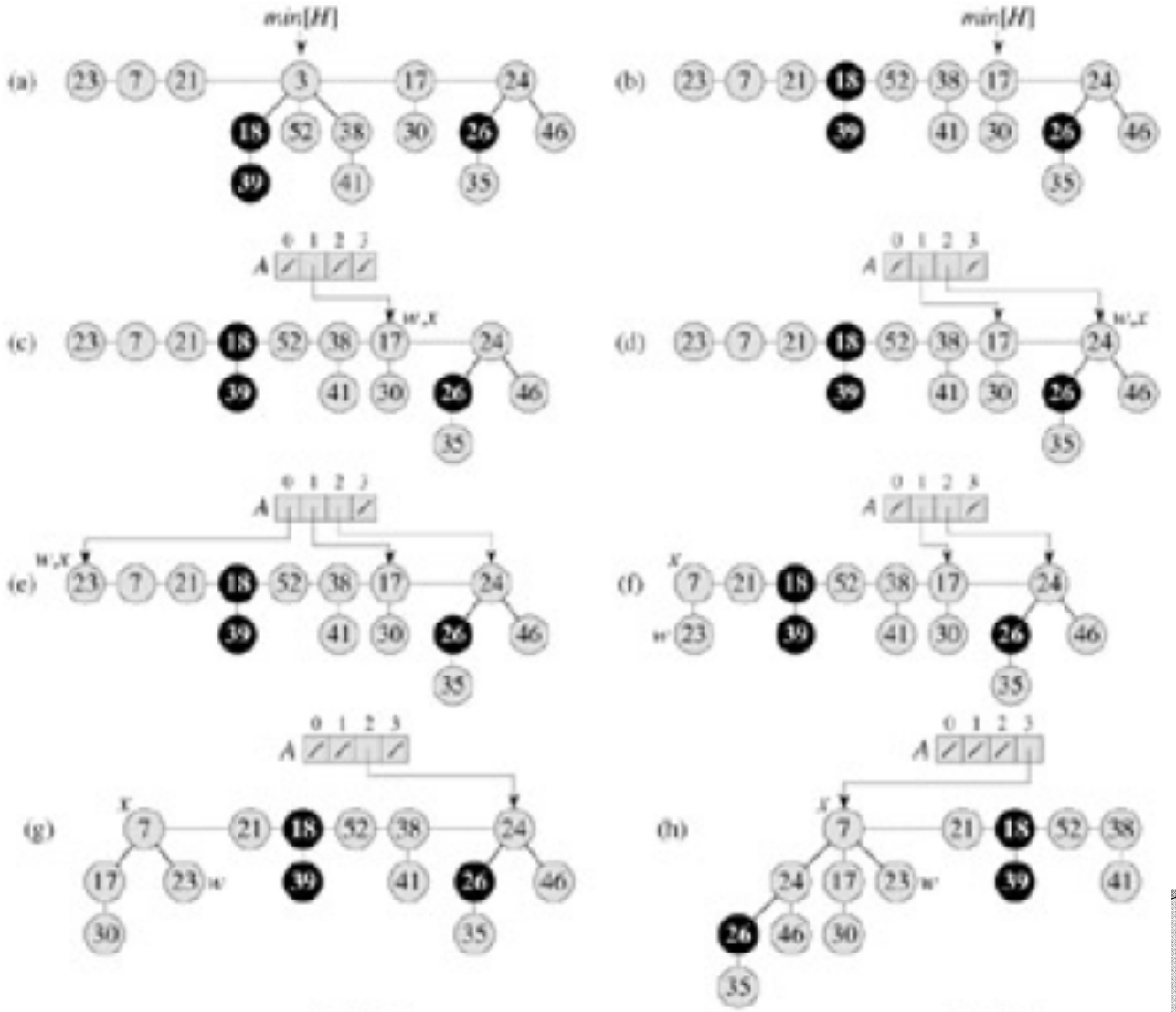
Montículos de Fibonacci



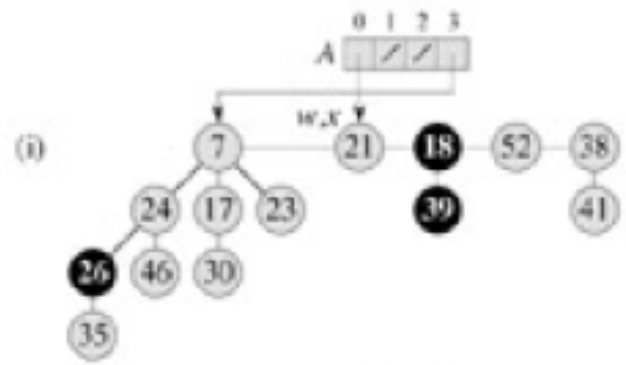
Operación	Montículo binario W(n)	Montículo binomial W(n)	Montículo Fibonacci Amort.
crear()	$\Theta(1)$	$\Theta(1)$	$\Theta(1)$
inserta()	$\Theta(\lg n)$	$O(\lg n)$	$\Theta(1)$
min()	$\Theta(1)$	$O(\lg n)$	$\Theta(1)$
extMin()	$\Theta(\lg n)$	$\Theta(\lg n)$	$O(\lg n)$
union()	$\Theta(n)$	$O(\lg n)$	$\Theta(1)$
decreClave()	$\Theta(\lg n)$	$\Theta(\lg n)$	$\Theta(1)$
elimina()	$\Theta(\lg n)$	$\Theta(\lg n)$	$O(\lg n)$

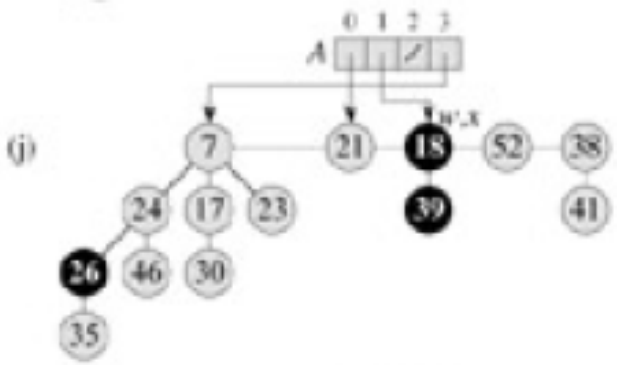
- Lista de raíces (enlazadas con - - - -)
- Bosque de árboles montículos sin alguna forma o tamaño predefinido
- Puede manejar un conjunto de colas por prioridad disjuntas

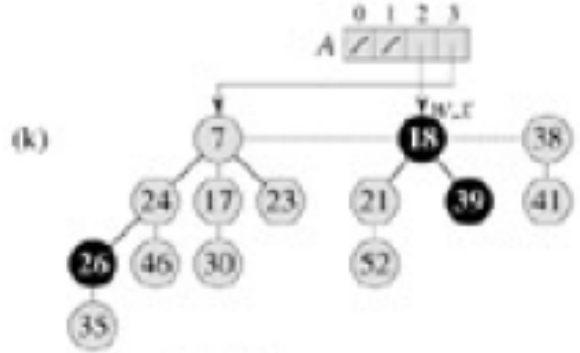
Extracción del mínimo

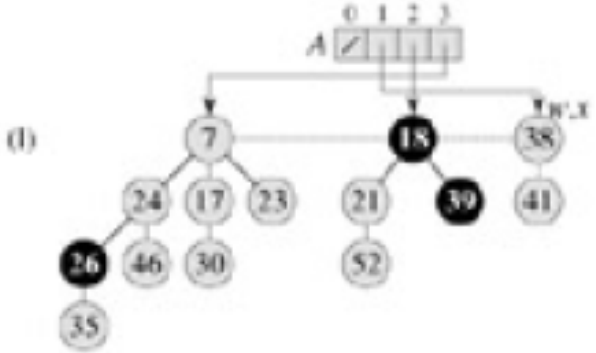


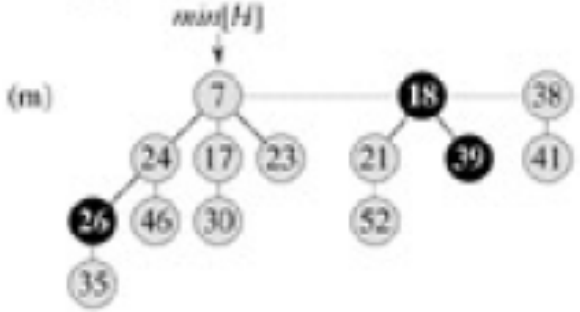
Extracción del mínimo



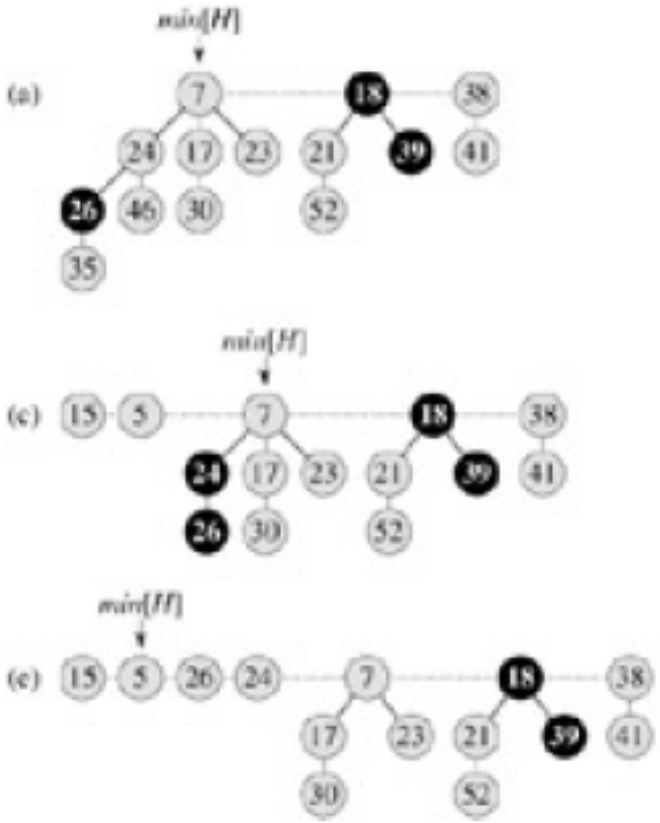








Decremento de la clave



Conjuntos disjuntos

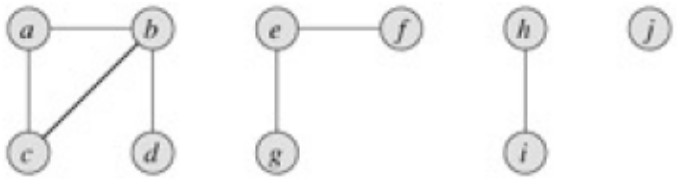
Mantiene una colección de conjuntos dinámicos, $S=\{S_1,...,S_k\}$

- Cada conjunto en S contiene un miembro que lo representa e identifica, denominado su representante X
- Operaciones:
  - Crea(X): crea un nuevo conjunto con un solo elemento que a su vez es su representante, el cual no pertenece a ningún otro conjunto de miembros
  - Union(X,Y): unifica los 2 conjuntos dinámicos que contienen a X e Y, S_X, S_Y, en un nuevo conjunto con todos los miembros de S_X y de S_Y

Operaciones del conjunto disjunto

- El representante de la unión se escoge entre sus miembros, pero normalmente se selecciona uno de los 2 representantes de los conjuntos unidos.
- Los conjuntos unidos son eliminados, pues no pueden haber repetidos
- busca(X): regresa una referencia al representante que contiene X

Ejemplo de aplicación



(a)

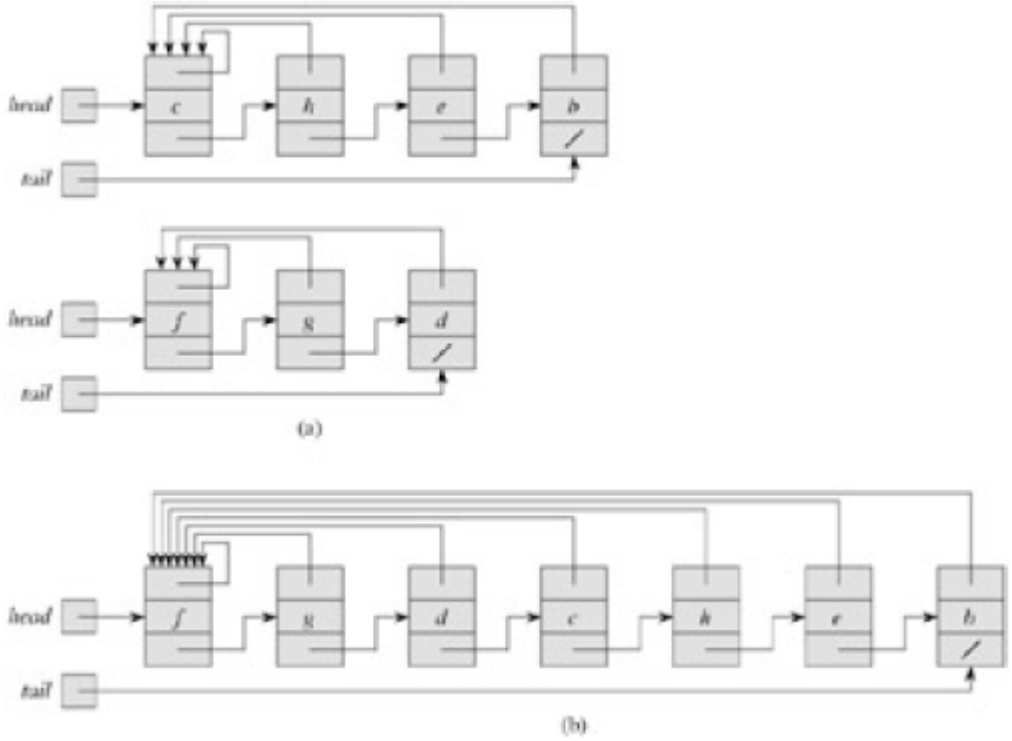
Edge processed	Collection of disjoint sets									
initial sets	{a}	{b}	{c}	{d}	{e}	{f}	{g}	{h}	{i}	{j}
(b,d)	{a}	{b,d}	{c}		{e}	{f}	{g}	{h}	{i}	{j}
(e,g)	{a}	{b,d}	{c}		{e,g}	{f}		{h}	{i}	{j}
(a,c)	{a,c}	{b,d}			{e,g}	{f}		{h}	{i}	{j}
(h,i)	{a,c}	{b,d}			{e,g}	{f}		{h,i}		{j}
(a,b)	{a,b,c,d}				{e,g}	{f}		{h,i}		{j}
(e,f)	{a,b,c,d}				{e,f,g}			{h,i}		{j}
(b,c)	{a,b,c,d}				{e,f,g}			{h,i}		{j}

- Determinación de las componentes conexas de un grafo no dirigido

```
CONNECTED-COMPONENTS(G)
1  for each vertex v ∈ V[G]
2       do MAKE-SET(v)
3  for each edge (u, v) ∈ E[G]
4       do if FIND-SET(u) ≠ FIND-SET(v)
5             then UNION(u, v)

SAME-COMPONENT(u, v)
1  if FIND-SET(u) = FIND-SET(v)
2       then return TRUE
3       else return FALSE
```

Representación con listas enlazadas



- Algoritmos son igual de rápidos en ambas representaciones, pero si se incluye la unión por rango y la compresión de caminos, es mejor el bosque de árboles

- Representación como bosque de árboles
- Cada nodo apunta al padre
- El nodo representante apunta a él mismo