Clase 7 - Cola de prioridades

martes, 11 de abril de 2023 16:26

Como una cola pero se atiende el de > prioridad y no el 1ro q' entró

(Rincón de arte)

C. de prioridad → estructura que permite como mínimo insertar y borrar mínimo (prioridad 0 > arriba)

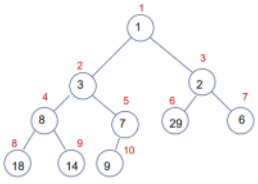
- Lista ordenada
  - Insert tiene O(N) operaciones
  - DeleteMin tiene O(1) operaciones
- Lista no ordenada
  - Insert tiene O(1) operaciones
  - DeleteMin tiene O(N) operaciones
- Árbol Binario de Búsqueda
  - Insert y DeleteMin tienen en promedio O(log N) operaciones

Heap binario → implementación sin punteros, implementa elim. y creación con $O(\log N)$

- Propiedad estructural
- Propiedad teorden

Heap es un binario completo

Ejemplo:



✓ El número de nodos n de un árbol binario completo de altura h, satisface:

$$2^h \le n \le (2^{h+1}-1)$$

Demostración:

- Si el árbol es lleno, $n = 2^{h+1}-1$
- Si no, el árbol es lleno en la altura $h-1$ y tiene por lo menos un nodo en el nivel h:

$n = 2^{h+1}-1+1-2^h$

La altura h del árbol es de **O(log n)**

Árbol binario puede merterse en arreglo.

- Raíz → pos 1
- para elem en pos I
  - h. izquierdo en $2*I$
  - h. derecho en $2*I+1$
  - Padre en $i/2$ → $(7:3=2)$

Min Heap

- Mínimo está en raíz, nodo es = o < a sus hijos
- No hay relación e/ hermanos

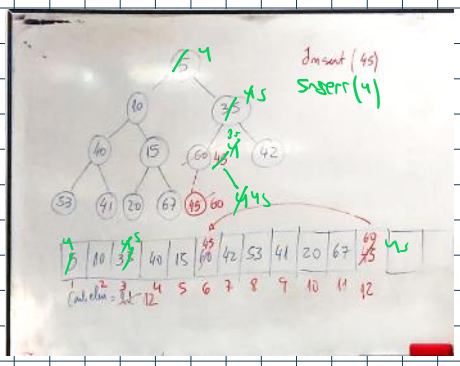
Max Heap → = pero al revés xp

Operaciones importantes

Insertar:

Si bien se inserta como último puedo hacer lío con la propiedad.

Entonces pregunto si el nro q' entró es < que su padre, si lo es, los intercambio y vuelvo a preguntar.



Filtrado hacia arriba → intercambia K hacia arriba desde lugar de inserción

→ Termina cuando K alcanza raíz o nodo cuyo padre es <

→ O(log n) intercambios

```
insert (Heap h, Comparable x) {
    h.tamaño  = h.tamaño + 1;
    n  = h.tamaño;
    while ( n / 2 > 0 & h.dato[n/2]  > x ) {
        h.dato[n]  = h.dato[n/2];
        n = n/2;
        }
    h.dato[n] = x;   // ubicación correcta de "x"
} // end del insert
```

Filtrado hacia arriba o Percolate_up

```
percolate_up (Heap h, Integer i) {
    temp = h.dato[i];
    while (i/2 > 0 & h.dato[i/2] > temp ) {
        h.dato[i] = h.dato[i/2];
        i = i/2;
        }
    h.dato[ i ] = temp;   // ubicación correcta del elemento a filtrar
} // end del percolate_up
```

Borrar mínimo:

Sé q' min es la raíz.

Para mantener estructura de binario completo, tiene q' desaparecer un elemento.

Pongo último elem a la raíz y filtro hacia abajo.

→ busco < de los hijos y lo comparo con el q' subí.

N/2 P. buscar padre. Si es >, subo el hijo y bajo el otro.

Repito todo.

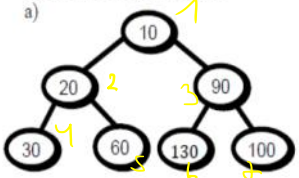
- Es similar al filtrado hacia arriba
- El filtrado hacia abajo restaura la propiedad de orden intercambiando el dato de la raiz hacia abajo a lo largo del camino que contiene los hijos mínimos
- El filtrado termina cuando se encuentra el lugar correcto dónde insertarlo
- Ya que el algoritmo recorre la altura de la heap, tiene *O(log n)* operaciones de intercambio.

```
delete_min ( Heap h, Comparable e) {
if (not esVacia(h) ) {
        e := h.dato[1];
        candidato := h.dato[ h.tamaño ];
        h.tamaño := h.tamaño - 1;
        p := 1;
        stop_perc := false;
        while ( 2* p <= h.tamaño )  and ( not stop_perc) {
            h_min := 2 * p;   // buscar el hijo con clave menor
            if h_min <> h.tamaño   //como existe el hijo derecho comparo a ambos
                if ( h.dato[h_min +1] < h.dato[h_min] )
                                        h_min := h_min + 1
            if  candidato  > h.dato [h_min] { //  percolate_down
                                        h.dato [p] := h.dato[ h_min ];
                                        p := h_min;
                                    }
            else    stop_perc := true;
        }
    h.dato[p] := candidato;
}
} // end del delete_min
```

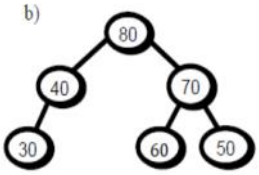
Filtrado hacia abajo o Percolate_down

1.- Indique para cada uno de los siguientes árboles binarios si son un árbol parcialmente ordenado. En caso negativo, explique por qué.

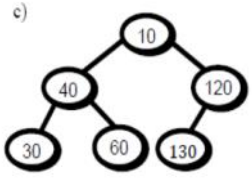
a)



b)

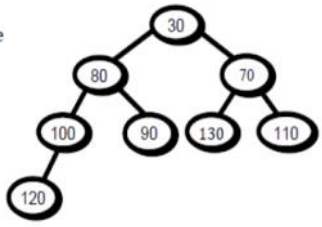


c)



2.- ¿Cuál de las siguientes opciones corresponde al almacenamiento lineal del siguiente árbol parcialmente ordenado o Heap binaria?

a) 30,70, 80 90,100, 110,120, 130
b) 30,80, 70,100, 90,130, 110, 120
c) 30,80, 100,120, 90,70, 130, 110
d) 120, 100, 90, 80, 130, 110, 70, 30



3.- Inserte los valores 60,75 y 10 a la heap anterior. Dibuje la heap resultante después de cada operación.