Heaps

Algoritmos y Estructuras de Datos II

1^{er} cuatrimestre de 2018

Motivación

Implementar una cola de prioridad. Operaciones que nos interesan:

Máximo	Determinar el elemento más prioritario.
Agregar	Agregar un elemento.
Sacar máximo	Sacar el elemento más prioritario.
Conj→cola	Convertir un conjunto en una cola de prioridad.

La prioridad puede ser cualquier **relación de orden total**.
(¿Qué era una relación de orden total?).

Hablamos siempre del **máximo**.

Posibles implementaciones (sin usar heap)

	Máximo	**Agregar**	**Sacar máximo**	**Conj→cola**
Lista	?	?	?	?
Lista + máximo	?	?	?	?
Lista ordenada	?	?	?	?
AVL + máximo	?	?	?	?

Posibles implementaciones (sin usar heap)

	Máximo	**Agregar**	**Sacar máximo**	**Conj→cola**
Lista	$O(n)$	$O(1)$	$O(n)$	$O(n)$
Lista + máximo	$O(1)$	$O(1)$	$O(n)$	$O(n)$
Lista ordenada	$O(1)$	$O(n)$	$O(1)$	(sorting)
AVL + máximo	$O(1)$	$O(\log n)$	$O(\log n)$	$O(n \log n)$

Spoiler

¿Para qué queremos un heap si podemos usar un AVL?

Spoiler

¿Para qué queremos un heap si podemos usar un AVL?

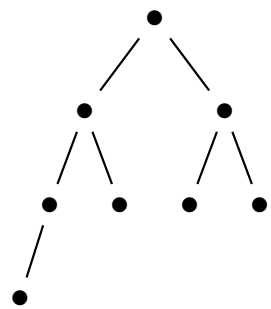
- Más sencillo de implementar.
- Mejores constantes.
- Se puede hacer sin punteros.
- La operación **Conj**$\rightarrow$**cola** es estrictamente mejor.

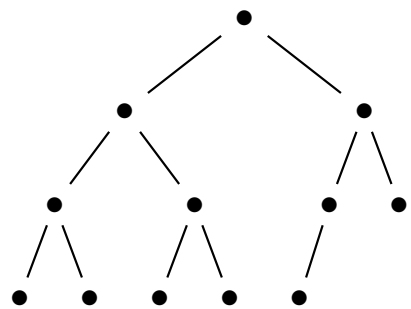
Heap: invariante

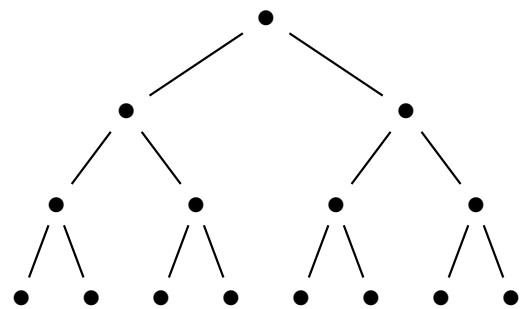
Un heap es un árbol binario con un invariante:

Forma

- Completo, salvo por el último nivel.
- Izquierdista.







Orden

- La raíz es el máximo.
- El invariante se cumple recursivamente para los hijos.

Heap: algoritmos

Máximo $O(1)$

- Está en la raíz del árbol.

Agregar $O(\log n)$

- Ubicar el elemento respetando la forma del heap.
- Mientras sea mayor que su padre, intercambiarlo con el padre. (*Sift up*).

Sacar máximo $O(\log n)$

- Reemplazar la raíz del árbol por el "último" elemento, respetando la forma del heap.
- Mientras sea menor que uno de sus hijos, intercambiarlo con el mayor de sus hijos.
(*Sift down*).

Heap: algoritmos

Ejemplo: insertar en secuencia $6, 4, 2, 9, 3, 8, 5$ y sacar el máximo.

Heap: algoritmos

Conj→cola (*heapify*) $O(n)$

- Armar un árbol con los elementos respetando la forma.
- Hacer *Sift down* para cada uno de los elementos, yendo "hacia atrás", desde el último hasta la raíz.

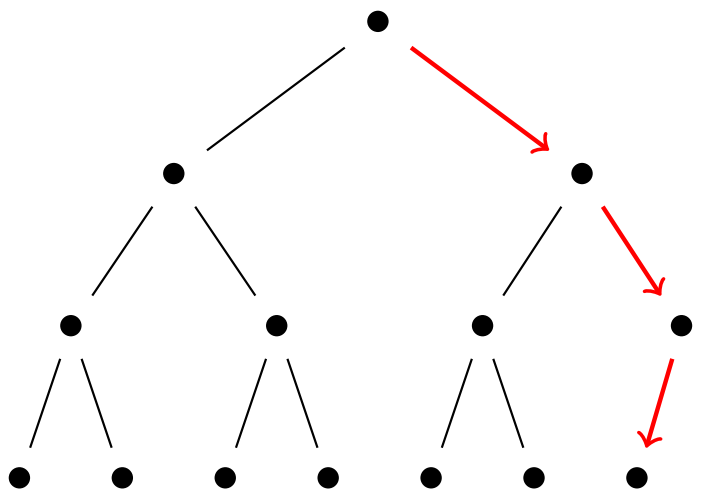
Ejemplo: *heapificar* la secuencia $6, 4, 2, 9, 3, 8, 5$.

Heap: técnicas de implementación

Técnica de implementación con punteros

Si el heap tiene n elementos, la posición del último se puede encontrar a partir de la representación en binario de n, ignorando el dígito 1 más significativo.

$n = 14 = (1110)_2 \quad \rightsquigarrow \quad$ [derecha, derecha, izquierda]



Heap: técnicas de implementación

Técnica de implementación con arreglos

Los elementos se pueden guardar en un arreglo de tamaño N.

Las siguientes funciones sirven para navegar el árbol:

$$\begin{aligned} \texttt{HIJO\_IZQ}(i) &= 2 * i + 1 \\ \texttt{HIJO\_DER}(i) &= 2 * i + 2 \\ \texttt{PADRE}(i) &= \lfloor \tfrac{i-1}{2} \rfloor \end{aligned}$$

Usando índices $0 \leq i < N$.