Heapsort

Guillermo Palma

Universidad Simón Bolívar
Departamento de Computación y T.I.

CI-2612: Algoritmos y Estructuras II

Plan

Definiciones

Estructura de datos Árbol

Estructura de datos que es una representación de un árbol jerárquico, posee un valor como raíz (nodo raíz) y subárboles de hijos que se derivan de un nodo padre.

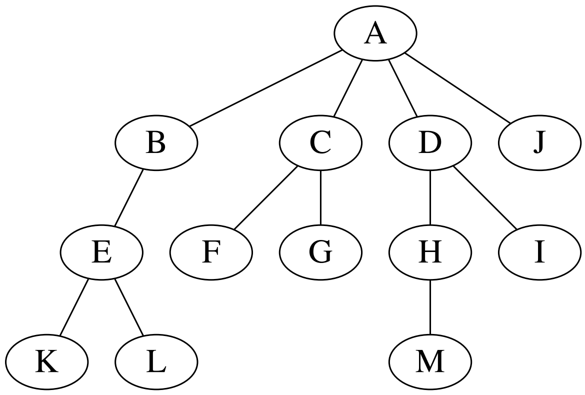

Figura: Ejemplo de la Estructura de datos Árbol

Estructura de datos Árbol

- A es nodo raíz
- A, B, C, D, E, y H son nodos padre
- K, L, F, G, M, I y J son nodos hojas
- B, C, D y J son hijos de A
- H y I son hijos de D
- ...

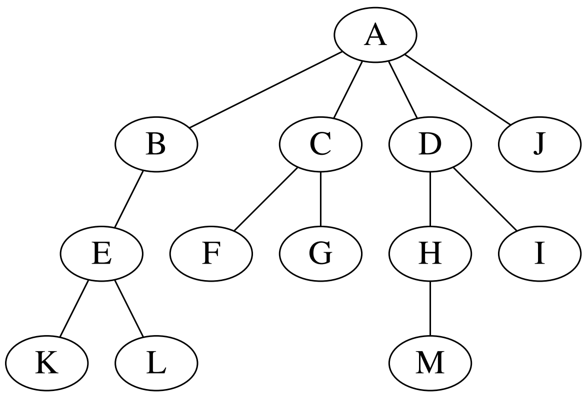

Figura: Ejemplo de la Estructura de datos Árbol

Definiciones

Altura de un nodo

El número de lados del camino más largo desde un nodo determinado hasta un nodo hoja.

Nivel de un nodo

El número de nodos del camino más largo desde el nodo raíz hasta el nodo determinado, sin contar el nodo raíz.

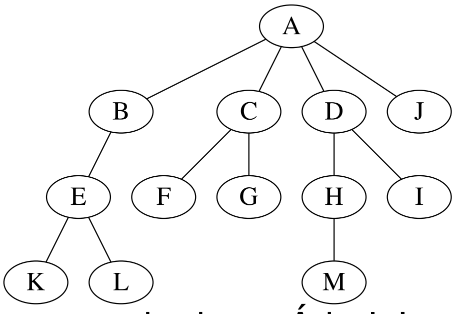

Figura: Ejemplo de la Estructura de datos Árbol. La altura de la raíz A es 3. El nivel del nodo I es 2

Definiciones

Árbol Binario

Es un tipo de estructura de datos Árbol, en la cual cada nodo padre tiene a lo sumo dos nodos hijos.

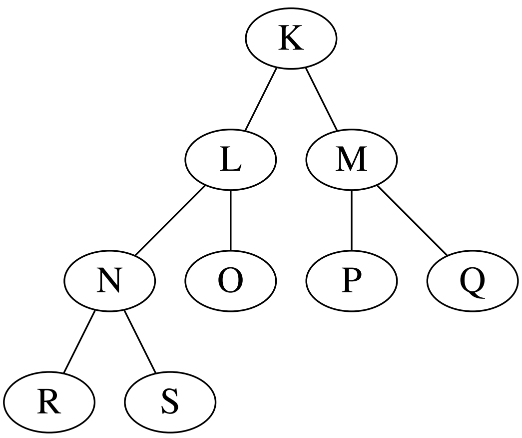

Figura: Ejemplo de un Árbol Binario

Definiciones

Heap

Es una estructura de árbol binario que almacena una colección de claves y tiene las siguientes dos propiedades:

- Todas las hojas en el mismo nivel y todas los nodos internos tienen grado 2, excepto posiblemente por el último nivel, el cual es construido de izquierda a derecha.
- Propiedad del Heap:
  - Para un Max-heap la clave de un nodo x es menor o igual a la clave del padre, esto es $Parent(x) \geq x$
  - Para un Min-heap la clave de un nodo x es mayor o igual a la clave del padre, esto es $Parent(x) \leq x$,

Max-heap

Para todos los nodos x, excepto la raíz, se cumple que $Parent(x) \geq x$

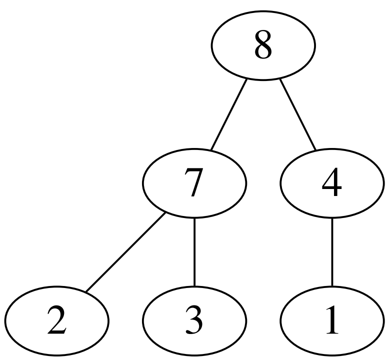

Figura: Ejemplo de un Max-heap

Min-heap

Para todos los nodos x, excepto la raíz, se cumple que $Parent(x) \leq x$

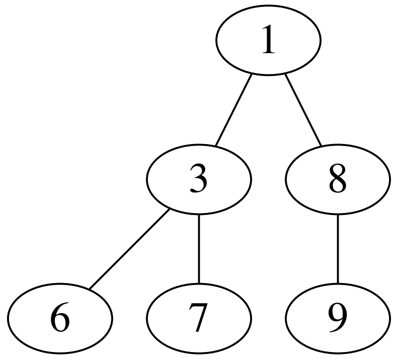

Figura: Ejemplo de un Min-heap

Representación de un Heap

- Se puede representar como un arreglo
- La raíz es $A[1]$
- Padre de $A[i] = A[\lfloor i/2 \rfloor]$ ($Parent\ (i) = \lfloor i/2 \rfloor$)
- Hijo izquierdo de $A[i] = A[2i]$ ($LEFT(i) = 2i$)
- Hijo derecho de $A[i] = A[2i+1]$ ($RIGHT(i) = 2i+1$)
- Altura del heap $A \leq length(A)$

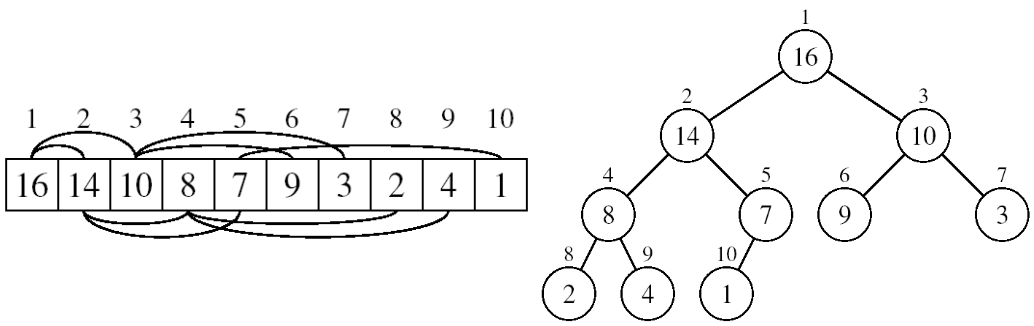

Figura: Ejemplo de un Max-heap. Tomado de [1]

MAX-HEAPIFY

- Procedimiento que mantiene las propiedades de un Heap
- Sea un nodo *i* más pequeño que su hijo:
    - Los subárboles izquierdo y derecho de *i* son Max-heaps
    - Intercambia con el hijo más grande
    - Mover la clave hacia bajo del heap
    - Continuar hasta que no haya ningún nodo sea más pequeño que su hijo

Ejemplo de MAX-HEAPIFY

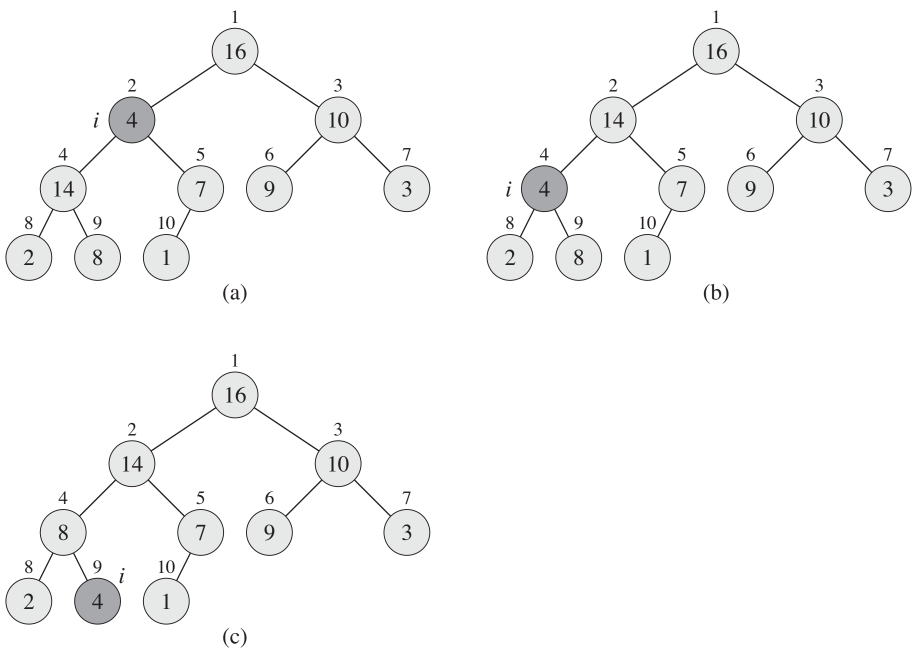

Figura: Llamada `MAX-HEAPIFY(A, 2, 10)`. a) *A*[2] viola la propiedad del Heap. b) *A*[4] viola la propiedad del Heap. c) Se cumple la propiedad del Max-heap. Tomado de [1]

Procedimiento MAX-HEAPIFY

Procedimiento MAX-HEAPIFY(A, i, n)

inicio
- $l \leftarrow$ `LEFT`(i) ;
- $r \leftarrow$ `RIGHT`(i) ;
- **si** $l \leq n$ *y* $A[l] > A[i]$ **entonces**
  - *largest* $\leftarrow l$;
- **en otro caso**
  - *largest* $\leftarrow i$;
- **si** $r \leq n$ *y* $A[r] > A[largest]$ **entonces**
  - *largest* $\leftarrow r$;
- **si** *largest* $\neq i$ **entonces**
  - `SWAP`$(A[i], A[largest])$;
  - `MAX-HEAPIFY`$(A, largest, n)$;

Tiempo del peor caso de MAX-HEAPIFY

- Se recorre el camino más largo de la raíz a la hoja
- En cada nivel se hace dos comparaciones
- O(Altura del heap), esto es $O(\log n)$

Construyendo un Max-heap

- Se convierte un arreglo no ordenado en un Max-heap
- Los elementos del subarreglo $A[(\lfloor n/2 \rfloor + 1)..n]$ son hojas
- Se aplica MAX-HEAPIFY a los elementos entre 1 y $\lfloor n/2 \rfloor$

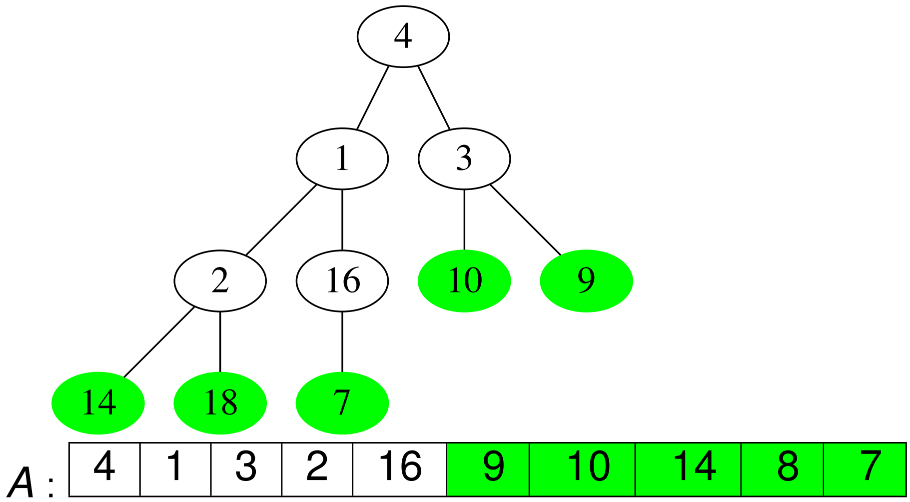

Figura: Árbol para construir un Max-heap. Las hojas del árbol están en verde

Procedimiento para construir un Max-heap

Procedimiento BUILD-MAX-HEAP(A)

inicio

$n \leftarrow$ `length(`A`)` ;

para $i \leftarrow \lfloor n/2 \rfloor$ ***decrementando hasta 1*** **hacer**

MAX-HEAPIFY(A, i, n) ;

Ejemplo para construir un Max-heap

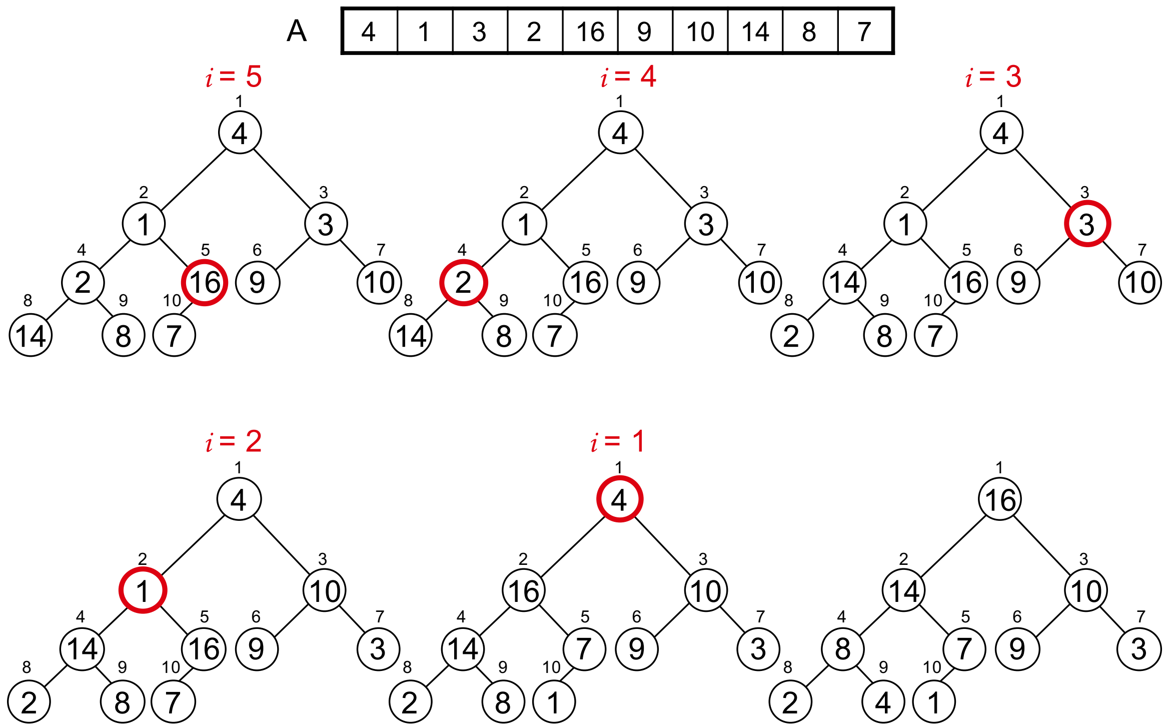

Figura: Llamada a `BUILD-MAX-HEAP(A)`. Tomado de [1]

Tiempo del peor caso de BUILD-MAX-HEAP

- Se aplica $n/2$ veces el procedimiento MAX-HEAPIFY ($O(\log n)$)
- Por lo tanto es $O(n \log n)$. Este límite superior, aunque correcto, no es asintóticamente ajustado.
- Tarea: ver que en el peor caso BUILD-MAX-HEAP es $O(n)$

Heapsort

- Algoritmo de ordenamiento que usa las estructuras y propiedades de los Heaps
- Idea del Algoritmo:
  - Se construye un max-heap de un arreglo inicial
  - Intercambia la raíz (clave más grande) con el nodo con la menor clave
  - Se descarta el último intercambiado por decrementar el tamaño del heap
  - Se llama a MAX-HEAPIFY
  - Continuar el proceso hasta que solo quede un único nodo

Procedimiento Heapsort

Procedimiento HEAPSORT(A)

inicio

  `BUILD-MAX-HEAP`(A) ;

  $n \leftarrow$ `length`(A) ;

  para $i \leftarrow n$ ***decrementando hasta*** 2 **hacer**

    `SWAP`($A[1], A[i]$) ;

    `MAX-HEAPIFY`($A, 1, i-1$) ;

Ejemplo de ejecución de Heapsort

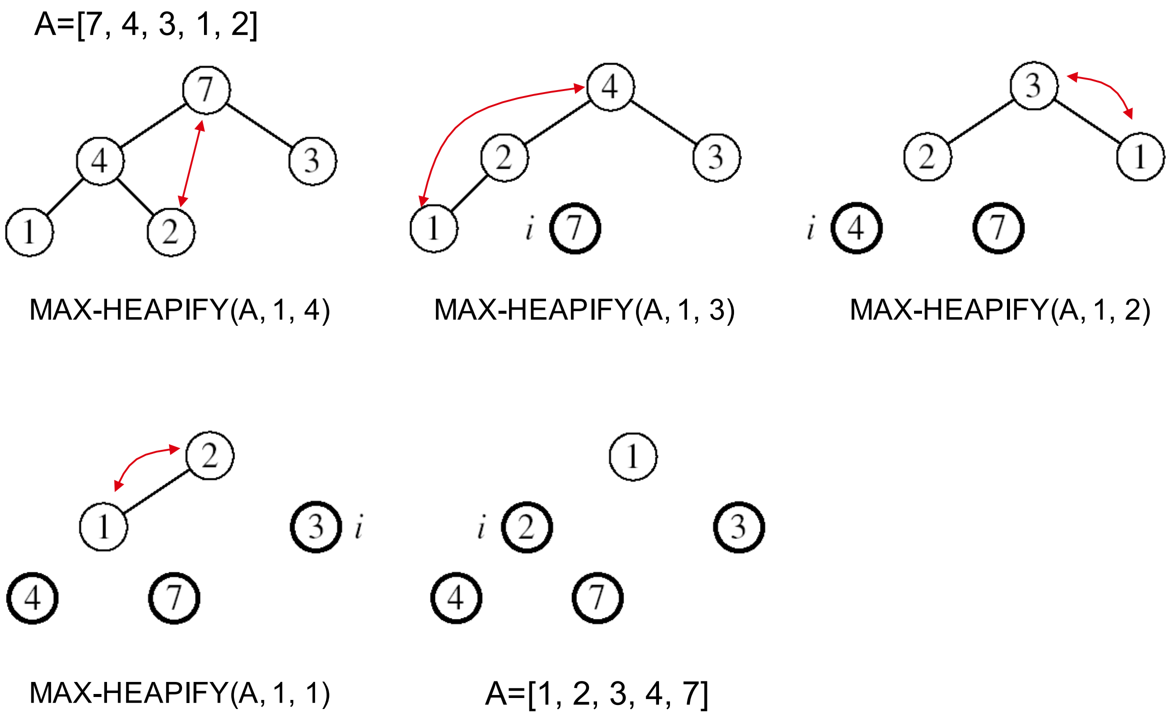

Figura: Llamada a `HEAPSORT(A)`. Tomado de [1]

Tiempo del peor caso de Heapsort

- BUILD-MAX-HEAP es $O(n)$
- Se aplica $n - 1$ veces el procedimiento MAX-HEAPIFY ($O(\log n)$)
- Por lo tanto Heapsort es $O(n \log n)$

Referencias

T. Cormen, C. Leirserson, R. Rivest, and C. Stein.
Introduction to Algorithms.
McGraw Hill, 3ra edition, 2009.