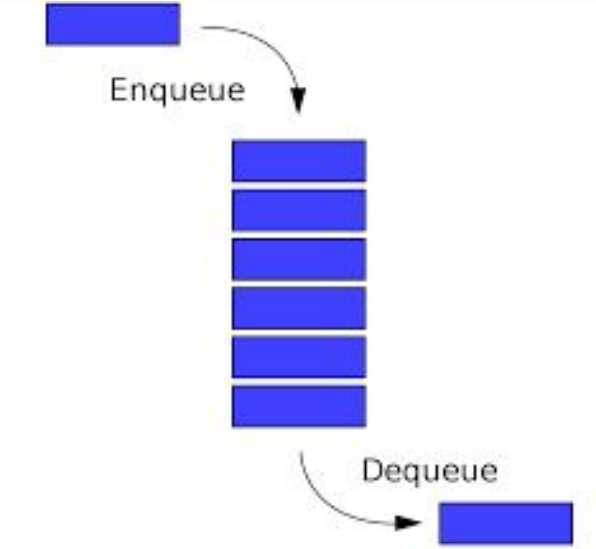
Cola de prioridad

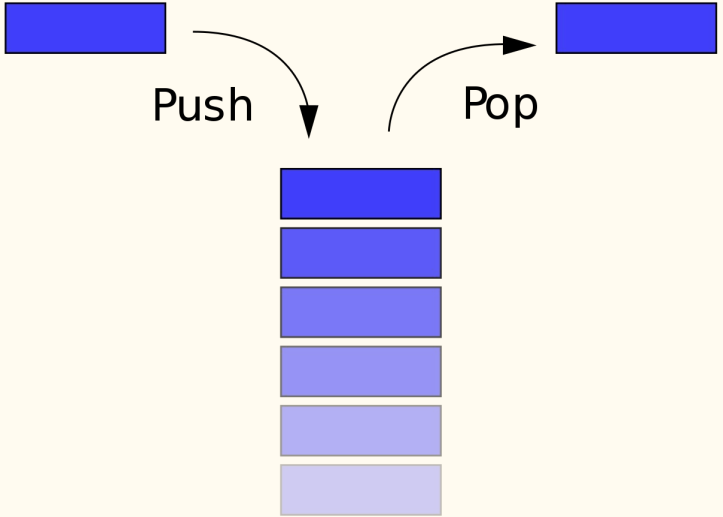
Donde no importa el orden de llegada

Introducción

Hasta ahora ...



Cola FIFO



Pila LIFO

A veces quien llega primero no tiene prioridad

A veces quien llega primero no tiene prioridad

Cola de prioridad

- Los elementos tienen arraigado una prioridad que define quién será el próximo en ser “atendido”.
- La definición mínima para este TAD es:

  - *// pre: la CP no está llena*
    // pos: inserta el elemento dentro de la CP
    void **insertar**(E elemento);

  - *// pre: la CP no está vacía*
    // pos: elimina el elemento con con mayor prioridad
    void **desencolar**();

  - *// pre: la CP no está vacía*
    // pos: retorna el elemento con mayor prioridad
    E **tope**();

  - *// pre: -*
    // pos: retorna si la CP está vacía
    bool **esVacia**();

  - *// pre: -*
    // pos: retorna si la CP está llena
    bool **estaLlena**();

Qué tiene más prioridad?

“Esto es prioridad 1”

“A partir de ahora, esto es la prioridad 1 millón”

Cola de prioridad - definiciones alternativas

// pre: la CP no está vacía
// pos: retorna el elemento con mayor prioridad y lo saca de la CP
E **desencolar**();

// pre: la CP no está vacía
// pos: retorna el elemento con mayor prioridad (no lo elimina)
E **tope**();

// pre: la CP no está vacía
// pos: elimina el elemento con mayor prioridad
void **desencolar**();
alternativo: void pop();

Implementaciones posibles?

Implementaciones posibles y sus órdenes

- ???
- ???
- ???
- ???

- *// pre: la CP no está llena*
  // pos: inserta el elemento dentro de la CP
  void **insertar**(E elemento);

- *// pre: la CP no está vacía*
  // pos: retorna el elemento con mayor prioridad y lo saca de la CP
  E **desencolar**();

- *// pre: -*
  // pos: retorna si la CP está vacía
  bool **esVacia**();

Implementaciones posibles y sus órdenes

- Lista
  - *// pre: la CP no está llena*
    // pos: inserta el elemento dentro de la CP
    void **insertar**(E elemento);
  - *// pre: la CP no está vacía*
    // pos: retorna el elemento con mayor prioridad y lo saca de la CP
    E **desencolar**();
  - *// pre: -*
    // pos: retorna si la CP está vacía
    bool **esVacia**();

Implementaciones posibles y sus órdenes

- Lista
- Lista ordenada

- *// pre: la CP no está llena*
  // pos: inserta el elemento dentro de la CP
  void **insertar**(E elemento);

- *// pre: la CP no está vacía*
  // pos: retorna el elemento con mayor prioridad y lo saca de la CP
  E **desencolar**();

- *// pre: -*
  // pos: retorna si la CP está vacía
  bool **esVacia**();

Implementaciones posibles y sus órdenes

- Lista
- Lista ordenada
- ABB

- *// pre: la CP no está llena*
  // pos: inserta el elemento dentro de la CP
  void **insertar**(E elemento);

- *// pre: la CP no está vacía*
  // pos: retorna el elemento con mayor prioridad y lo saca de la CP
  E **desencolar**();

- *// pre: -*
  // pos: retorna si la CP está vacía
  bool **esVacia**();

Implementaciones posibles y sus órdenes

- Lista
- Lista ordenada
- ABB
- AVL

- *// pre: la CP no está llena*
  // pos: inserta el elemento dentro de la CP
  void **insertar**(E elemento);

- *// pre: la CP no está vacía*
  // pos: retorna el elemento con mayor prioridad y lo saca de la CP
  E **desencolar**();

- *// pre: -*
  // pos: retorna si la CP está vacía
  bool **esVacia**();

Implementaciones posibles y sus órdenes

- Lista
- Lista ordenada
- ABB
- AVL

	Insertar	Mínimo	BorrarMín
Lista	O(1)	O(n)	O(n)
Lista ordenada	O(n)	O(1)	O(1)
ABB - AVL	O(lg n) cp	O(lg n)cp	O(lg n)cp
AVL	O(lg n)pc	O(lg n) pc	O(lg n) pc

- *// pre: la CP no está llena*
  // pos: inserta el elemento dentro de la CP
  void **insertar**(E elemento);

- *// pre: la CP no está vacía*
  // pos: retorna el elemento con mayor prioridad y lo saca de la CP
  E **desencolar**();

- *// pre: -*
  // pos: retorna si la CP está vacía
  bool **esVacia**();

Ver ejemplo de implementación con lista

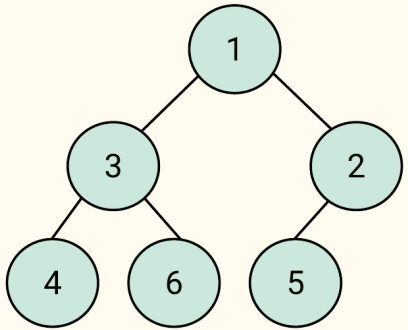
Heap

TAD CP

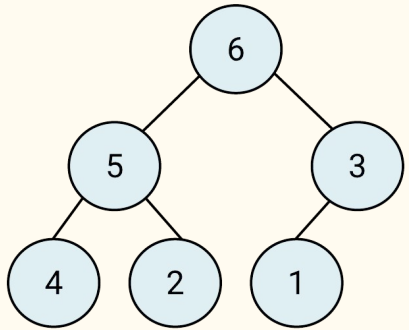
Heap

Heap - intro

- Los heaps es una **estructura de datos** que forman parte de la familia de los árboles.
- Se manejan bajo dos reglas/propiedades:
  - Propiedad estructural: es un árbol binario **completo**
  - Propiedad de orden: para cualquier nodo sus hijos son mayores o iguales (en caso de ser max heap serían menores o iguales)



Min heap

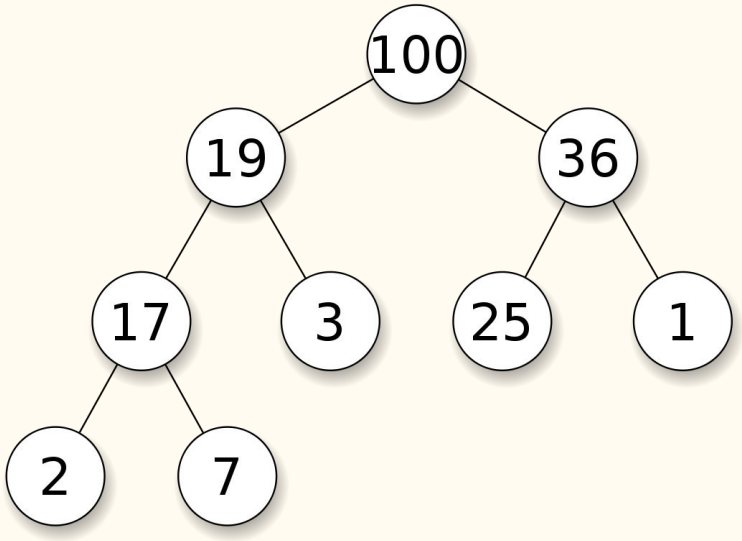


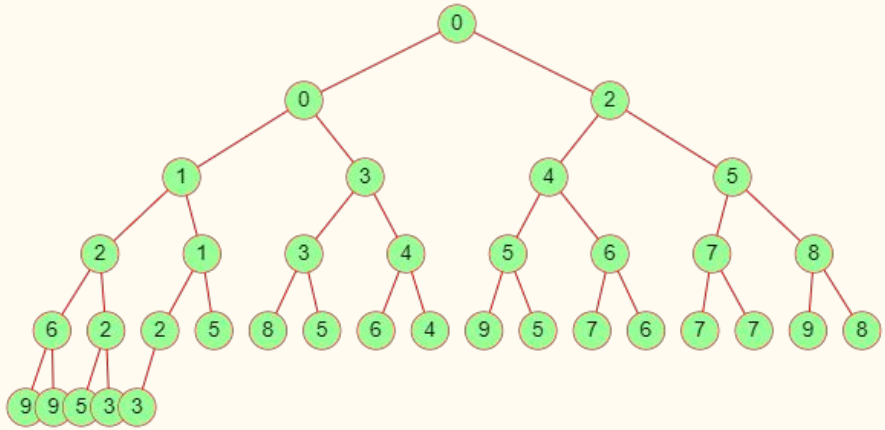
Max Heap

Nota: no confundir árbol binario con ABB

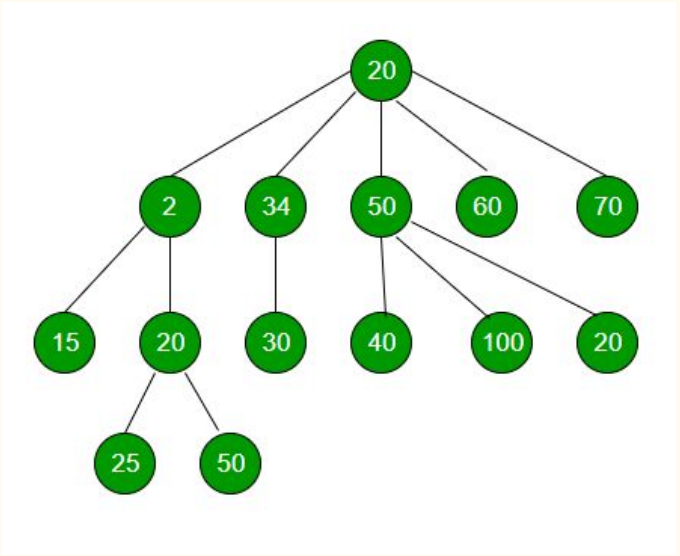
Es un árbol binario completo

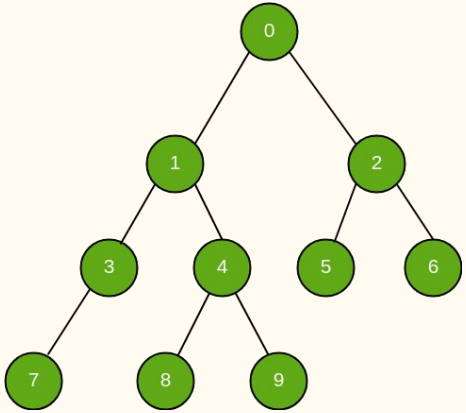
Un árbol binario es completo cuando todos los niveles están llenos, con la excepción del último, que se llena desde la izquierda hacia la derecha.





Es un árbol binario completo - **no** ejemplos

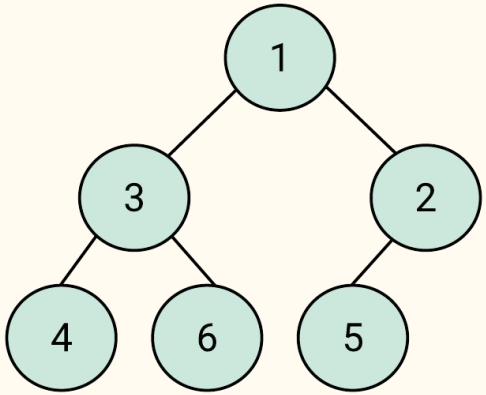




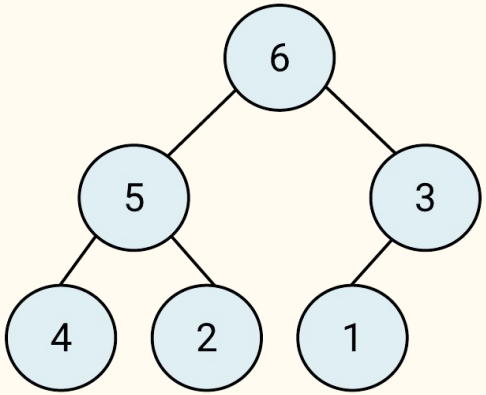
Propiedad de orden

Dependiendo del tipo de heap (min_heap o max_heap) se establece el orden.

Tener en cuenta que los hermanos (izq y der) **no** guardar relación entre ellos.



Min heap



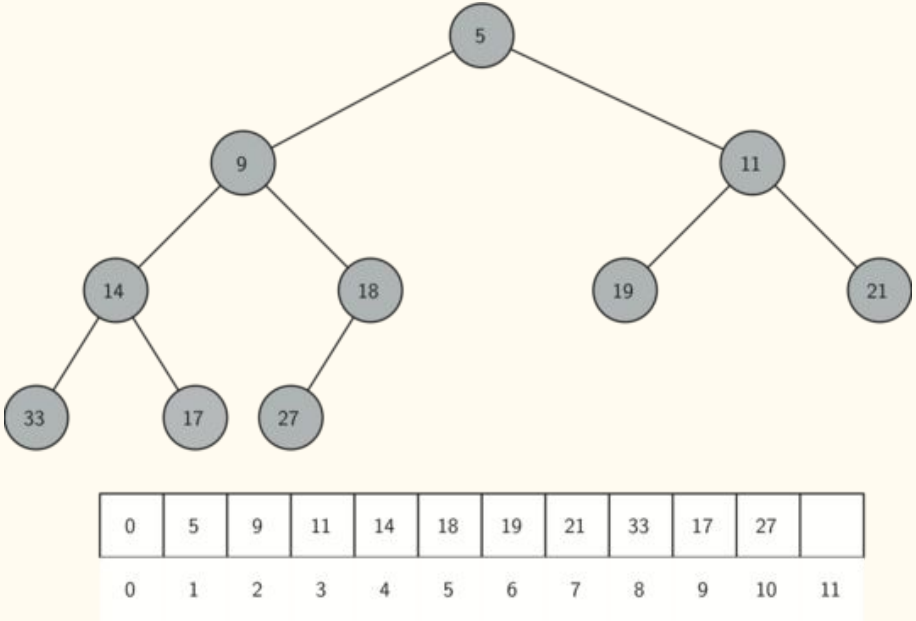
Max Heap

Dos caras de lo mismo

Dos caras de lo mismo

Debido a la propiedad estructural de los árboles completos es común representarlos en un array.

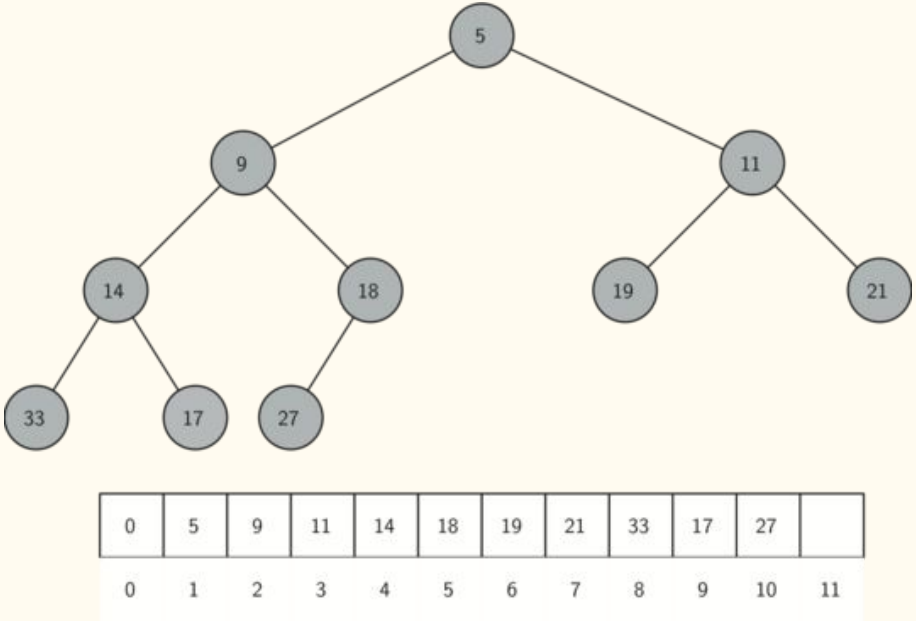
La raíz se encuentra (normalmente) en la posición 1. Mientras que el resto de los nodos se van insertando por nivel (y de izq a der).



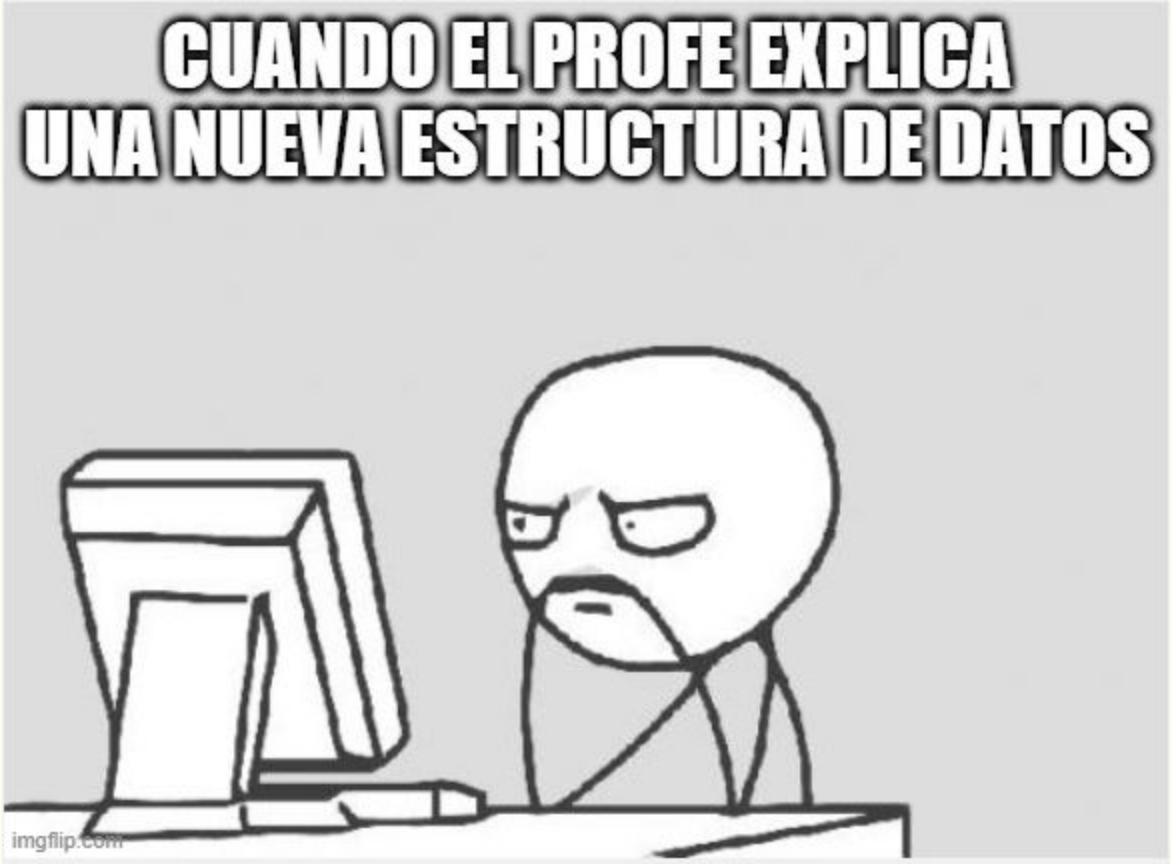
La navegación es fácil (y además bidireccional)

Dado un nodo cualquiera en la pos i, su hijo izquierdo se encuentra en i*2 y su hijo der en i*2+1.

Además se puede ir al padre de cualquier nodo con i/2.



Nota: debemos tener en cuenta los casos borde que veremos más adelante en el código.


CUANDO EL PROFE EXPLICA
UNA NUEVA ESTRUCTURA DE DATOS
imgflip.com

El “desequilibro”

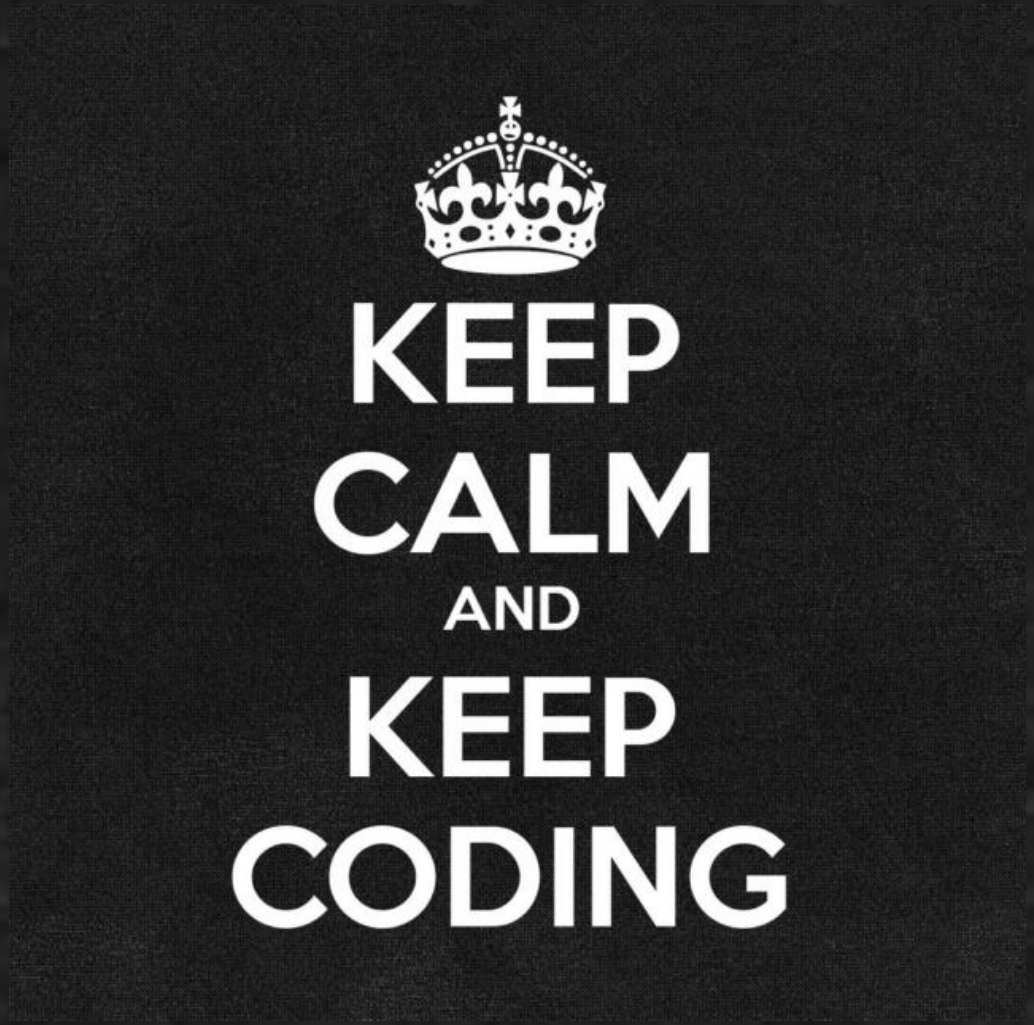
Las dos operaciones principales (insertar y borrarMin/borrarMax) crean la posibilidad de que se rompa al menos una de la reglas.

Cuando **insertamos**, lo hacemos en la última posición libre del array.
Problema -> puede que no cumpla la regla de orden
Solución -> encontrar la posición adecuada del nuevo elemento => **flotar**

Cuando **borramos el mínimo** dejamos la raíz del árbol libre (pos 1 del array).
Problema -> no cumple la regla de la estructura
Solución -> colocar momentáneamente el “último” elemento del heap y luego encontrar la posición adecuada => **hundir**

Ejemplo

https://www.cs.usfca.edu/~galles/visualization/Heap.html


KEEP
CALM
AND
KEEP
CODING

Órdenes

	insertar	borrarMin	obtenerMin
peor caso	???	???	???
caso promedio	???	???	???

Link de interes: https://web.archive.org/web/20160205023201/http://www.stats.ox.ac.uk/__data/assets/pdf_file/0015/4173/heapbuildjalg.pdf

Órdenes

	insertar	borrarMin	obtenerMin
peor caso	O(LogN)	???	???
caso promedio	**O(1)**	???	???

Link de interes: https://web.archive.org/web/20160205023201/http://www.stats.ox.ac.uk/__data/assets/pdf_file/0015/4173/heapbuildjalg.pdf

Órdenes

	insertar	borrarMin	obtenerMin
peor caso	O(LogN)	O(LogN)	???
caso promedio	**O(1)**	O(LogN)	???

Link de interes: https://web.archive.org/web/20160205023201/http://www.stats.ox.ac.uk/__data/assets/pdf_file/0015/4173/heapbuildjalg.pdf

Órdenes

	insertar	borrarMin	obtenerMin
peor caso	O(LogN)	O(LogN)	O(1)
caso promedio	**O(1)**	O(LogN)	O(1)

Link de interes: https://web.archive.org/web/20160205023201/http://www.stats.ox.ac.uk/__data/assets/pdf_file/0015/4173/heapbuildjalg.pdf

Puesta a punto

https://forms.office.com/r/TJfZKU6LGs

Cola de prioridad extendida

COLA DE PRIORIDAD EXTENDIDA

COLA DE PRIORIDAD

Cola de prioridad extendida

La CP extendida no es más que un definición de CP con más métodos, en los que se destaca:

- void **eliminar**(E el);
- void **cambiarPrioridad**(E el);
- void construirHeap(Iterador<E> *it);
- ...
- void vaciar();

Cola de prioridad extendida - eliminar

// pre: -
// pos: el elemento “elm” es eliminado de la CP
void **eliminar**(E elm);

1. ¿Cómo se hace?
2. ¿Qué orden tiene?

Cola de prioridad extendida - cambiar prioridad

Muchas veces la prioridad de un elemento pueden cambiar estando aún no procesado. Para ello, es útil “notificar” a la CP para que reacomode dicho elemento.

// pre: -
// pos: se actualiza la prioridad del elemento elm
void **cambiarPrioridad**(E elm);

1. ¿Cómo se hace?
2. ¿Qué orden tiene?

Cola de prioridad extendida - construir heap

Hay casos donde se desea construir un heap a partir de un Set de elementos. Normalmente una inserción uno por uno sonaría natural, pero existe otra forma que ofrece un mejor orden.

// pre: -
// pos: construye el heap a partir de todos los elementos del iterador
void **constuirHeap**(Iterador<E> *it);

Lectura e implementación a cargo del estudiante. Links de interés:

- Data Structures and Algorithm Analysis in C++ Mark Allen Weiss; Benjamin/Cummings Inc., 1994. Capítulo 6
- https://www.geeksforgeeks.org/building-heap-from-array/
- http://www.cs.umd.edu/~meesh/351/mount/lectures/lect14-heapsort-analysis-part.pdf

Ordenes CP extendida

	eliminar	cambiarPrioridad
peor caso	???	???
caso promedio	???	???

Ordenes CP extendida

	eliminar	cambiarPrioridad
peor caso	O(N) + O(LogN) = O(N)	???
caso promedio	???	???

Ordenes CP extendida

	eliminar	cambiarPrioridad
peor caso	O(N) + O(LogN) = O(N)	???
caso promedio	O(N) + O(LogN) = O(N)	???

Ordenes CP extendida

	eliminar	cambiarPrioridad
peor caso	O(N) + O(LogN) = O(N)	O(N) + O(LogN) = O(N)
caso promedio	O(N) + O(LogN) = O(N)	???

Ordenes CP extendida

	eliminar	cambiarPrioridad
peor caso	O(N) + O(LogN) = O(N)	O(N) + O(LogN) = O(N)
caso promedio	O(N) + O(LogN) = O(N)	O(N) + O(LogN) = O(N)

TAD CP
EXT.

Heap
+
Tabla de
hash

TAD CP
EXT.

Heap
+
Tabla de
hash

Cola de prioridad extendida - impl

La idea usar como estructura auxiliar una tabla de hash<T,int> donde guarde y mantenga la posición relativa de los elementos dentro de array. De esa manera reduciendo el tiempo de encontrar el elemento dentro del array.

eliminar:

1. Encontrar el elemento -> Tabla de hash
2. Sustituirlo y acomodar -> Heap (flotar+hundir)

cambiar prioridad:

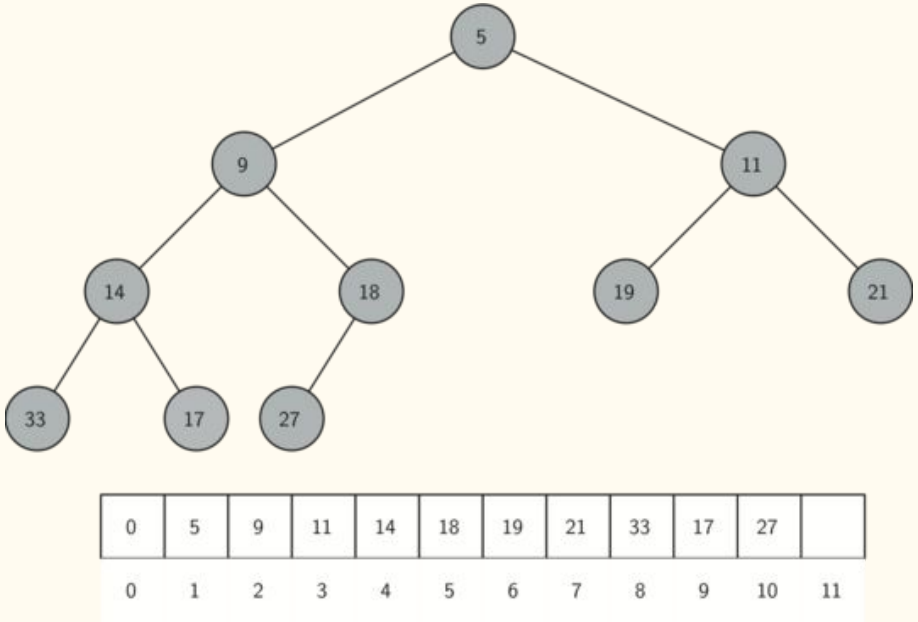
1. Encontrar el elemento -> Tabla de hash
2. Acomodar -> Heap (flotar+hundir)

Nota: cada vez que flotamos y hundimos los elementos DEBEMOS actualizar la tabla, ya que las posiciones van cambiando. Eso podría afectar los órdenes de insertar y borrar mínimo.

Cola de prioridad extendida - impl

KEY	VALUE
11	3
5	1
9	2
21	7
17	9

.
.
.



Ordenes CP extendida - Heap + Tabla de hash

	eliminar	cambiarPrioridad
peor caso	O(N) + O(LogN) = O(N)	O(N) + O(LogN) = O(N)
caso promedio	**O(1)** + O(LogN) = **O(LogN)**	**O(1)** + O(LogN) = **O(LogN)**