HEAP

Maria C. Torres Madroñero
Profesora asociada
Departamento de Ciencias de la Computación y la Información

Objetivos

- Implementar las estructuras de datos HEAP empleando arreglos
- Implementar soluciones algorítmicas empleando HEAPS

Recursos requeridos

- PC
- IDE para JAVA o Python – el estudiante deberá seleccionar un lenguaje de programación para el desarrollo de las prácticas de laboratorio

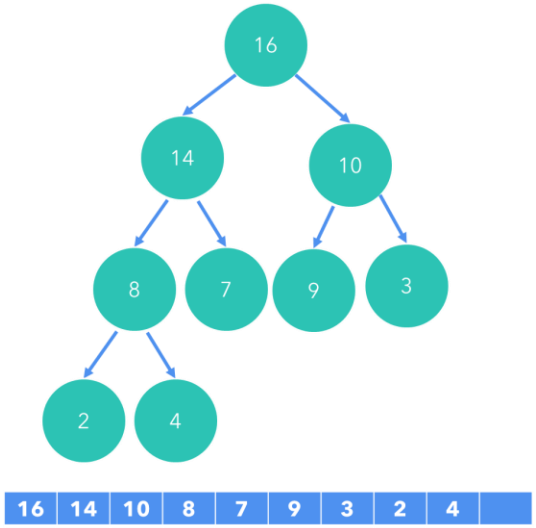
Actividades preliminares al laboratorio

- Lectura de la guía

Marco teórico

HEAP

Los HEAP o montículos binarios son una estructura de datos que permite fácilmente acceder a los valores mínimos o máximos de una colección. Su estructura se asemeja a un árbol binario, sin embargo, se representan usualmente mediante arreglos lineales. La figura muestra un ejemplo de un MAX_HEAP. Tenga en cuenta que en la memoria del pc lo que se mantiene es el arreglo (en color azul) de la figura.



Existen dos tipos de HEAP: el MAX-HEAP (mostrado en la figura) cumple la propiedad que todo nodo padre tiene una clave mayor o igual a la de sus hijos; por su parte, un MIX-HEAP cumple la propiedad que todo padre tiene una clave menor igual a la de sus hijos.

En el módulo de árboles binarios entenderemos mejor la relación padres e hijos. Por el momento, podemos asociar a cada nodo del HEAP un padre, excepto a la raíz que corresponde al valor que se encuentra en la posición 0 del arreglo. Si estamos trabajando con MAX-HEAP en esta posición encontraremos el valor más grande de la colección; si estamos trabajando con MIN-HEAP en esta posición se encontrará el valor mínimo de la colección. Cada padre tendrá un hijo izquierdo y un hijo derecho. Dado que en la memoria del PC lo que almacenamos es el arreglo, desde los índices es fácil calcular donde se encuentra el padre, hijo izquierdo e hijo derecho de un nodo.

El padre de un nodo i corresponde a $\lceil i/2 \rceil - 1$, el hijo izquierdo se encuentra en la posición $2i + 1$, y el hijo derecho en la posición $2i + 2$. Note que estas ecuaciones se encuentran diseñadas para lenguajes de programación que inician la indexación de un arreglo en 0.

Las aplicaciones más comunes de los HEAP son el algoritmo HEAPSORT (uno de los algoritmos más eficientes computacionalmente) y la implementación de colas de prioridad (PRIORITY QUEUE).

Implementación de heap

El diagrama de clase resume los principales elementos para la implementación de un HEAP usando un arreglo.

HEAP
- A[ ]: Object - size: int
+ HEAP(int capacity) + parent(int i): int + left (int i): int + right(int i): int + max-heapify(int i) + build-max-heap(B[ ]) + heap-sort()

En esta implementación mantenemos dos atributos: A[ ] corresponde al arreglo de objetos, estos objetos pueden ser de varios tipos, pero deben contener una clave (key) numérica que permita realizar la comparación de los datos dentro del HEAP; y size() corresponde al número de datos almacenados en el HEAP, size siempre será menor o igual a la longitud del arreglo.

Los métodos descritos en el diagrama de clase corresponden a los estudiados en clase, e incluyen:

- Parent(int i): retorna la posición del padre del nodo i
- Left(int i): retorna la posición del hijo izquierdo del nodo i
- Right(int i): retorna la posición del hijo derecho del nodo i
- Max-heapify(int i): Mantiene la propiedad del heap dado una posición i
- Build-max-heap(B[ ]): Construye un heap a partir de un arreglo no ordenado
- Heap-sort(): implementa el algoritmo de ordenamiento basado en un heap

Actividades

Problema 1 Implementación clase HEAP

En esta primera etapa del laboratorio, implemente la clase HEAP para un MAX-HEAP con los elementos descritos en el diagrama de clase previamente presentado. Se sugiere que la implementación del HEAP permita almacenar datos genéricos (tipo Object), pero las pruebas en este laboratorio la realizaremos usando valores enteros.

Problema 2 Implementación clase Priority Queue

En esta segunda etapa del laboratorio, diseñe e implemente una cola de prioridad empleando la clase HEAP del problema 1. Esta debe contener al menos las operaciones MAX-HEAP-INSERT, HEAP-EXTRACT-MAX y HEAP-MAXIMUM. La descripción en detalle de cada una de estas operaciones se encuentra en las lecturas de clase.

Problema 3 Uso de las clases HEAP y PrioriryQueue

Presente una prueba de todas las funcionalidades de las dos clases usando un arreglo de enteros generado de forma aleatoria con al menos 20 elementos.

Instrucciones de entrega

- La solución de los problemas debe desarrollarse en JAVA o Python. Los estudiantes tendrán la libertad de seleccionar el lenguaje de programación y plataforma para presentar la solución de los problemas.
- La solución debe emplear librerías nativas y se invita a los estudiantes a no usar código descargado de internet. Los laboratorios están diseñados para practicar los fundamentos teóricos; entre más código escriba el estudiante más fácil será su comprensión de los temas de clase.
- La solución se puede presentar en grupos de hasta 3 estudiantes.
- La solución de los problemas debe entregarse y sustentarse en el aula de clase o en la hora de asesoría a estudiantes. No se reciben soluciones por correo electrónico.