Ing. de Sistemas e Informática
Estructuras de Datos y Algoritmos
Jorge Mario Londoño P.

Ejercicios de Repaso

Montículos, Tablas de símbolos

Montículos (Binary heaps)

1. Proponer la implementación de las operaciones insertar, remover el máximo utilizando:

- Una cola
- Una pila

Para cada una de ellas indicar su tiempo (en número de comparaciones) y compararlo con la implementación en montículo.

2. Utilizando el resultado del punto 1, indicar porque no es eficiente mantener una variable auxiliar *máximo* que se actualice cada que se inserta un item. La variable sería muy efectiva para obtener el mayor valor.

3. Se tienen una colección de puntos *(x,y,z)* en el espacio. Dar un algoritmo para encontrar los 10 puntos más cercanos al origen.

4. En alguna aplicación se requiere una *cola de prioridad max/min*, es decir una cola que permita insertar elementos y remover eficientemente tanto el mayor como el menor elemento. Proponer una implementación eficiente de la *cola de prioridad max/min*.

Búsqueda

1. En clase se consideraron dos implementaciones básicas de la búsqueda: Con listas no ordenadas y con arreglos ordenados.

a. Explorar una tercera alternativa: Listas ordenadas. Indicar la implementación de los métodos `get`/`put` (Nota: put debe preservar el orden de las llaves en la lista).

b. Evaluar la eficiencia de esta nueva alternativa y compararla con la que se tiene para los otros dos casos.

2. Las tablas de símbolos ordenadas (búsqueda binaria) permiten implementar la

interfaz de tabla de símbolos ordenada.

a. Dar implementaciones de los métodos `floor`, `delete` para la implementación en arreglos ordenados (búsqueda binaria).

b. Dar implementaciones de los métodos `floor`, `delete` para las listas ordenadas (ver problema anterior).

c. Comparar la eficiencia de estas dos operaciones con base en el número de comparaciones.

3. Estrategias de borrado lazy/eager.

En clase se consideraron ambas estrategias como alternativas para la implementación del método `delete` de la tabla de símbolos.

a. Dar implementaciones de las dos estrategias para la tabla de símbolos no ordenada basada en lista simple. Estimar su desempeño de peor caso y de caso medio. Nota: Mantener la consistencia del método size en cualquiera de los casos.

b. Dar implementaciones de las dos estrategias para la tabla de símbolos ordenada basada en arreglos. Estimar su desempeño de peor caso y de caso medio.

4. Listas ordenadas dinámicas

Se desea un implementación de la tabla de símbolos ordenada utilizando listas doblemente enlazadas, con la característica adicional que las llaves de la tabla pueden cambiar en tiempo de ejecución.

a. Dar una implementación básica de la tabla de símbolos ordenada (`get`,`put`,`select`,`rank`,etc) utilizando una lista doblemente enlazada.

b. Definir un método para cambiar la llave asociada a un valor:

```
boolean changeKey(Key oldkey, Key newkey)
```

El método changeKey devuelve verdadero si `oldkey` está en la ST y cambia el valor de la llave a newkey. Al cambiar la llave, se debe cambiar la posición del nodo en la lista para mantenerla ordenada, lo cual se puede lograr intercambiando con los vecinos izquierdos/derechos tantas veces como sea necesario.

c. Bajo que situación se daría el mayor número posible de intercambios del método changeKey. Cuál sería el orden tilde del peor caso?

d. Estimar el número de intercambios en el caso medio para la operación changeKey.

Árboles de Búsqueda Binarios

Se tiene un árbol binario de búsqueda vacío. Se hacen operaciones put con la siguiente secuencia de llaves:

19, 2, 13, 7, 8, 6

a. Indicar el árbol resultante y su altura.

b. Se realiza la operación `get(9)`. Indicar cuantas comparaciones se realizan y el valor retornado.

c. Se realiza la operación `put(5,v)`. Indicar cuantas comparaciones se realizan y el árbol resultante.