Universidad de los Andes *Marzo10, 2019*
Estructuras de Datos *ISIS 1206*
Daniel Yamin
Sebastian Puerto

Taller 5: Colas de Prioridad

1. **Análisis de Eficiencia:** Para el desarrollo del taller se implementaron dos colas de prioridad: a) basada en una Queue (MaxColaPrioridad) y b) basada en la estructura de Heap (MaxHeapCP). A continuación se muestra un resumen sobre la eficiencia de las dos implementaciones:

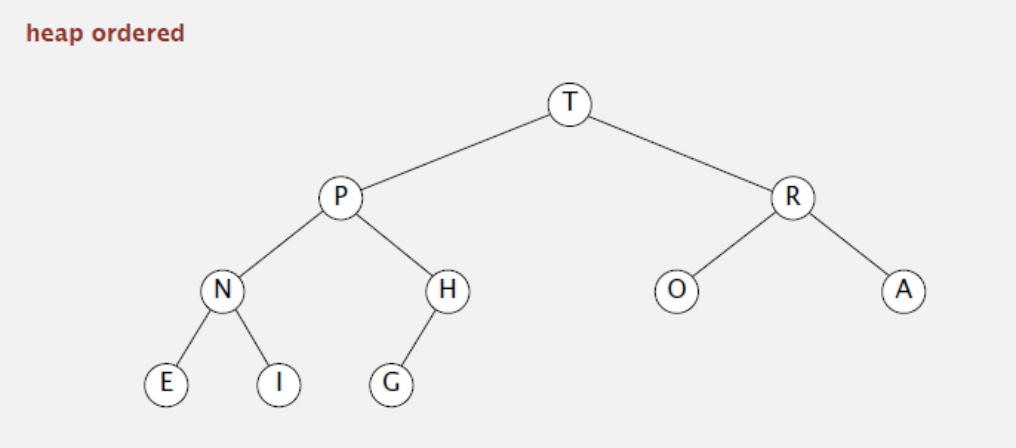
a. MaxColaPrioridad: En este caso, se debe tener en cuenta que debido a la implementación realizada, se tiene una cola ordenada puesto que al momento de agregar un nuevo elemento, se busca la posición en la que quede ir con el fin de obtener un orden descendente de las LocationVO. Entonces, las operaciones principales, tienen el siguiente orden de crecimiento:

- Insertar : O(N)
- DelMax: O(1)
- Max: O(1)

Puesto que se tiene un orden de los elementos, se sabe que el primero será el máximo y por esto las operaciones DelMax y Max tienen un orden constante e igual a 1. Sin embargo, para poder insertar un elemento se debe recorrer toda la cola y por eso la operación tiene orden de N.

Es importante notar que no se tiene un mejor o peor caso al agregar elementos puesto que sin importar el elemento a agregar, se debe recorrer toda la cola, siempre es esta una operación con orden de crecimiento igual a O(N). De igual forma, eliminar un elemento es siempre de la misma complejidad, con orden de crecimiento O(1).

b. MaxHeapCP: En este caso, se tiene una estructura Heap de ordenamiento como la que se muestra a continuación:



Donde cada nodo es mayor a sus hijos. Es importante resaltar que la implementación del ordenamiento Heap, se implementó en un arreglo dinámico. Ahora bien, a continuación se muestra el orden de crecimiento de las operaciones de la cola:

- Insertar : O(logN)
- DelMax: O(logN)
- Max: O(1)

Es importante mencionar los casos en los que dicho orden puede cambiar. Específicamente, el orden de crecimiento de la operación DelMax tendrá una complejidad de O(1) si todos los elementos de la cola tienen prioridades iguales o hay un conjunto muy pequeño de prioridades, pues al ubicar el último elemento de la lista al inicio la operación swim realizará un número constante de operaciones; en cualquier otra situación la operación DelMax tendrá el orden de O(log N) mencionado. Respecto a la operación de agregar, éste tendrá su mejor caso si el elemento a insertar es menor que todos los elementos del arreglo, la operación tendrá complejidad O(1); su peor caso ocurrirá en caso de que haya que recorrer este elemento por toda la heap, es decir cuando el elemento tiene una prioridad máxima, en cuyo caso la operación tendrá complejidad de O(log N).

En conclusión, teóricamente se tiene que la cola de prioridad con la estructura Heap es más eficiente que la cola basada en la estructura Queue. Más adelante se podrá concluir respecto a los resultados obtenidos de forma empírica.

2. Resultados: Tiempos de Ejecución.

Es importante aclarar pero los tiempos mostrados a continuación son un promedio de 5 tiempos obtenidos.

- Método Agregar:

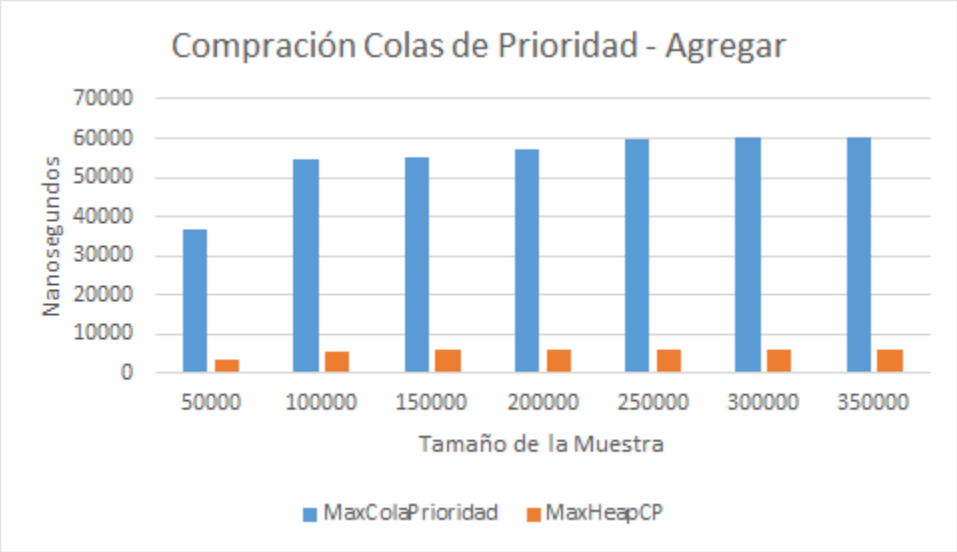
Tamaño Muestra	MaxColaPrioridad (nanosegundos)	MaxHeapCP (nanosegundos)
50000	36686	3253
100000	54609	5501
150000	55077	5891
200000	57351	6021
250000	59873	6083
300000	60023	6109
350000	60201	6119

- Método delMax:

Tamaño Muestra	MaxColaPrioridad (nanosegundos) (Tiempo Promedio)	MaxHeapCP (nanosegundos) (Tiempo Promedio)
50000	0	0
100000	0	0
150000	0	0
200000	0	0
250000	0	0
300000	0	0
350000	0	0

En primer lugar, es importante notar que teniendo en cuenta la eficiencia de ambos métodos para eliminar el máximo, puesto que se encuentran ordenados y se sabe de antelación cuál es el máximo, el tiempo es tan pequeño que se convierte 0 (despreciable) si se mide en nanosegundos y el sistema no deja medirlo en una unidad menor. Sin embargo, es posible pensar que, de ser posible calcularlo, el tiempo del método delMax de MaxColaPrioridad sería menor al de MaxHeapCP, ya que este proceso en nuestra implementación de cola requiere del manejo de pointers. En segundo lugar, se puede concluir que se cumple con la teoría expuesta

en el punto 1. La cola de prioridad Heap es mucho más eficiente en tiempo que la cola de prioridad Queue. Gráficamente, obtenemos el siguiente resultado, en el cual se muestra la comparación del tiempo del método agregar:



Donde se observa, la gran ventaja que tiene el método Heap.

3. Casos de Pruebas:

Para comprobar el buen funcionamiento de las colas de prioridad implementadas, se tuvieron en cuentas dos casos principales:

a. Caso Cola Vacía:

Es importante que si la cola este vacía funcione de una manera determinada. Específicamente, se creó un escenario (Escenario0) donde se tenía una MaxColaPrioridad y una MaxHeapCP, pero sin elementos. Se comprobó que el método esVacía retornara true, el método delMax retornara null y el método darNumElementos retornará 0. También se comprobó que se podían agregar satisfactoriamente elementos a dichas colas y finalmente se eliminaron comprobando el buen ordenamiento de los mismos.

b. Caso Cola con muchos elementos:

El segundo caso que se contempló fue el de una cola con 100 elementos (Escenario1). De la misma forma, se comprobaron los métodos esVacía (retornara False) y darNumElementos (retornara 100). Luego, se eliminaron todos los elementos de la lista, verificando que efectivamente se eliminará el mayor de todos en cada iteración, comprobando así el buen funcionamiento del método delMax y el ordenamiento de ambas colas. Finalmente, se insertó una nueva Location para verificar que el orden se mantuviera y, además, se insertaron varías

Location con un número de registros igual con el fin de comprobar el buen funcionamiento de las colas en caso de tener elementos con la misma prioridad.