Estructuras de Datos (2022-1)
Laboratorio 6

Jaime Ignacio Ansorena Carrasco
Matricula: 2020401497

06 de junio de 2022

Ejercicios

1. **Análisis de complejidad**

   a) PriorityQueueUnsorted::push(int n)
   Insertar en el vector posee una complejidad algorítmica $\boldsymbol{O(1)}$

   b) PriorityQueueUnsorted::top()
   Obtener el elemento mínimo requiere iterar a través del vector por lo que posee una complejidad algorítmica $\boldsymbol{O(n)}$

   c) PriorityQueueUnsorted::pop()
   Eliminar el elemento mínimo requiere iterar a través del vector por lo que posee una complejidad algorítmica $\boldsymbol{O(n)}$

   d) PriorityQueueHeap::push(int n)
   Insertar en el heap esta determinado por la altura del mismo y por el algoritmo de upheap, por lo que posee una complejidad algorítmica $\boldsymbol{O(log\, n)}$

   e) PriorityQueueHeap::top()
   Obtener el elemento mínimo posee una Complejidad Algorítmica $\boldsymbol{O(1)}$

   f) PriorityQueueHeap::pop()
   Eliminar el elemento mínimo esta determinado por la altura del mismo y por el algoritmo de downheap, por lo que posee una complejidad algorítmica $\boldsymbol{O(log\, n)}$

2. **Análisis experimental**

Analizando experimentalmente los algoritmos de Selection Sort y Heap Sort implementados se obtiene la siguiente tabla con los tiempos de ejecución de cada uno :

n	Selection Sort (s)	Heap Sort (s)
10	0.0000080000	0.0000040000
100	0.0001670000	0.0000360000
1000	0.0136440000	0.0002920000
2000	0.0499610000	0.0006480000
3000	0.1121880000	0.0010020000
4000	0.1966240000	0.0013440000
5000	0.3131540000	0.0017560000
6000	0.4393470000	0.0021840000
7000	0.5960230000	0.0025280000
8000	0.7833440000	0.0028950000
9000	0.9994280000	0.0032910000
10000	1.2514370000	0.0038890000

Cuadro 1: Análisis experimental de algoritmos

Se construyen los siguiente gráficos comparativos :

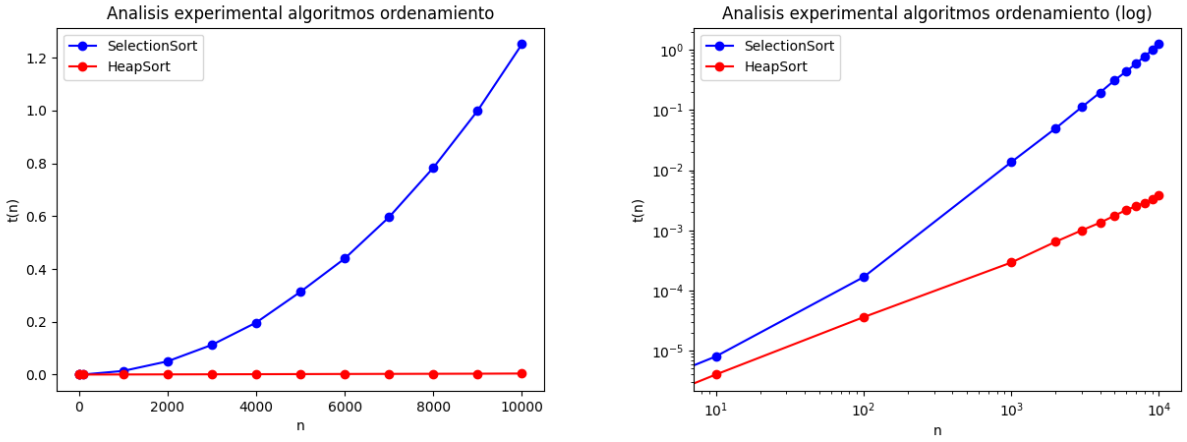

Figura 1: Algoritmos Selection Sort y Heap Sort

3. Analizando la tabla y los gráficos se concluye que el algoritmo de Selection Sort es ineficiente ya que requiere dos iteraciones anidadas para ordenar el vector, por lo que su complejidad algorítmica es de $\boldsymbol{O(n^2)}$.

   En cambio el algoritmo de Heap Sort es considerablemente mas rápido ya que esta determinado por la altura del árbol, por lo que su complejidad algorítmica es de $\boldsymbol{O(n \log n)}$.