Clase Práctica 6 - Ordenamiento

Juan Ignacio Elosegui

Junio 2024

Ejercicio 1

- "Ordenar una secuencia ya ordenada" No es necesario usar ningún algoritmo de ordenamiento previamente visto, ya que la secuencia ya está ordenada. Si quisiéramos ordenarla se podría resolver en O(1).

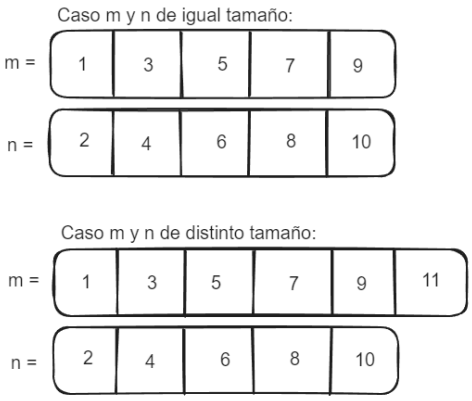
- "Insertar `k` elementos en su posición en una secuencia `v` ordenada, con `k` significativamente mas chico que `v.size()`" Se puede usar **Insertion Sort**, porque queremos insertar cada uno de los `k` elementos en su posición correcta. Vamos a tener una complejidad de O($k \cdot |v|$), porque en el peor caso, podríamos llegar a recorrer toda la secuencia `v` hasta encontrar la posición correcta de cada elemento. También se podría usar una versión de **Insertion Sort combinado con un ABB** para hacer la tarea que se nos pide. Encontrar la posición para insertar un elemento de `k` tiene una complejidad de O($k \cdot \log |v|$), y la inserción en sí tiene una complejidad de O($|v|$).

- "Encontrar los `k` elementos más chicos de la secuencia `v`, con `k` significativamente más chico que `|v|`" Primero nos convendría ordenar la secuencia `v` con **Selection Sort**, sólo que usando la siguiente modificación: cuando la parte que ya está ordenada tiene el tamaño de `k`, devolver ese vector, que contendrá los `k` elementos más chicos de `v`. Esto tiene una complejidad de O($k \cdot |v|$) para el peor caso, ya que va a iterar sólo `k` veces sobre la secuencia.

59	7	388	41	2	280	50	123
2	7	388	41	59	280	50	123
2	7	41	388	59	280	50	123
2	7	41	388	59	280	50	123

Acá sabemos que $|v| = 8$ y que $k = 3$ (es decir, queremos los 3 elementos más pequeños de la secuencia). Va a terminar el Selection Sort cuando la subsecuencia verde alcance un tamaño de 3.

- "Dadas dos secuencias ordenadas, devolver una secuencia que contenga sus elementos ordenados" Acá nos serviría usar sólo la función **Merge** que se usa comúnmente en el algoritmo de *sorting* de Merge Sort. Si tenemos una secuencia `m` y otra secuencia `n`, ambas ordenadas, vamos a tener una complejidad de O($|m| + |n|$) en el peor caso.



En ambos casos, se cumple el requerimiento de complejidad en el peor caso.

- "Encontrar los `k` elementos más grandes de una secuencia `v`, con `k` significativamente más chico que `|v|`" Se puede usar de vuelta el **Insertion Sort**, que sigue el mismo lineamiento que en el ítem 3, pero queremos devolver la porción de la secuencia que falta. Hay que hacer `k` iteraciones para encontrar los `k` elementos más grandes. En cada iteración, encontramos el máximo de la porción de la secuencia que no ha sido ordenada y lo colocamos en su posición correcta.

- "Ordenar una secuencia `v` en el que sus elementos están desordenados en a lo sumo `k` posiciones, con `k` significativamente más chico que `|v|`)" Podemos usar de vuelta **Insertion Sort**, pero sin modificaciones, donde su complejidad es de $O(k \cdot |v|)$ en el peor caso.

Ejercicio 2