Programación 3
Año 2023

Práctico 11

Ejercicio 1 (Cota inferior para la complejidad de algoritmos de ordenamiento)**.** Consideremos un algoritmo de ordenamiento, S, que dado un arreglo A con n elementos (potencialmente todos distintos), entre los cuales existe una relación de orden total, reorganiza los elementos de A de forma que queden ordenados entre sí, es decir, si llamamos A' al resultado de ejecutar S sobre A, se cumple $A'[i] \leq A'[i+1]$ para todo i, $1 \leq i < n$. El algoritmo S determina el orden de los elementos de A exclusivamente a través de comparaciones entre elementos.

(a) Muestre que si los elementos de A son todos diferentes, el algoritmo S determina, implícitamente, una permutación P de posiciones $(1, 2, \ldots, n)$ que deja ordenados los elementos de A, es decir, $A[P(i)] < A[P(i+1)]$ para todo i, $1 \leq i < n$.

(b) Sea $C(n)$ una cota superior para la cantidad de comparaciones que realiza S para un arreglo de tamaño n. Acote superiormente la cantidad de permutaciones diferentes de $(1, 2, \ldots, n)$ que puede determinar S a partir de diferentes entradas A.

(c) Demuestre que el tiempo de ejecución de S es $\Omega(n \log n)$.
Sugerencia: Utilice la fórmula de Stirling para factorial

$$n! = \Theta\left(n^{n+\frac{1}{2}} e^{-n}\right) .$$

Ejercicio 2 (Bucket sort)**.** Una agencia de reparto tiene n registros con información de paquetes que debe entregar en Uruguay. Por motivos de logística, necesita ordenar estos registros ascendentemente por código de departamento, que es un número en el rango de 1 a 19.

(a) Dé un algoritmo para solucionar el problema en tiempo $O(n)$.
Sugerencia: Utilice un arreglo auxiliar de tamaño 19.

(b) ¿Puede implementar este algoritmo de forma que el orden de los elementos se determine exclusivamente realizando comparaciones entre códigos de departamento y el tiempo de ejecución sea $O(n)$? Describa el algoritmo y analice su tiempo de ejecución.
Sugerencia: En la solución de la parte (a) sustituya el arreglo auxiliar por una lista encadenada.

(c) ¿Cómo se relaciona el algoritmo de la parte (b) con la cota inferior $\Omega(n \log n)$ que estudiamos anteriormente? Detalle qué hipótesis usada en la demostración de esta cota no se cumple y explique dónde falla la demostración.

Ejercicio 3 (Cota inferior para la complejidad de algoritmos de búsqueda)**.** Sea $\mathcal{X}$ un conjunto con más de 2 elementos, entre los cuales existe definida una relación de orden total.

Consideremos un algoritmo de búsqueda, B, que dado un arreglo ordenado A con n elementos de $\mathcal{X}$, y un elemento $x \in \mathcal{X}$, devuelve la posición de una ocurrencia de x en A (o informa que A no contiene a x). El algoritmo B determina la posición de x en A exclusivamente a través de comparaciones de elementos de A, entre sí y con x.

Demuestre que el tiempo de ejecución de B es $\Omega(\log n)$.