Prueba de oposición - Algoritmos 2016

Brian Bokser

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

16 de octubre de 2016

Contenidos

Introducción

Enunciado

La cantidad de parejas en desorden de un arreglo $A[1 \ldots n]$ es la cantidad de parejas de posiciones $1 \leq i < j \leq n$ tales que $A[i] > A[j]$.
Dar un algoritmo que calcule la cantidad de parejas en desorden de un arreglo y cuya complejidad temporal sea estrictamente mejor que $O(n^2)$ en el peor caso.
***Hint**: Considerar hacer una modificacion de un algoritmo de sorting.*

¿ Que algoritmo de Sorting podemos utilizar?

Algunos de los algoritmos que vemos en la materia, con sus complejidades son:

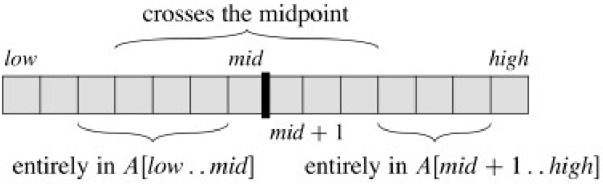
- Selection sort $\rightarrow$ O(n)
- Insertion sort $\rightarrow$ $O(n^2)$
- Counting sort $\rightarrow$ $O(n)$
- Heap sort $\rightarrow$ $O(nlogn)$
- Quick sort $\rightarrow$ $O(n^2)$
- Merge sort $\rightarrow$ $O(nlogn)$

A priori podemos descartar a los que tienen complejidad $O(n^2)$, y a Counting Sort, porque necesita que los elementos esten en un rango acotado. Nos quedan Heap sort y Merge sort, y sospechamos de este último porque, ¡Utiliza la técnica de D&Q!

Queremos subproblemas

Notamos la siguiente propiedad: Si una pareja i, j de las buscadas existe, hay 3 casos:

- i,j estan en el [1..n/2]
- i,j estan en el rango [(n/2) + 1 .. n]
- i en [1..n/2] y j en el otro.



(a)

Como i < j, i no puede estar en [(n/2) + 1 .. n] si j esta en el otro rango.

Lo recién planteado nos acerca aún más al merge sort, ya que podemos resolver dividir el arreglo en dos mitades, y resolver recursivamente en cada una de ellas. Esto nos daría i, j en el primer y segundo caso.
¿Y los del tercero?