Esercizio k fuori posto

Si consideri un array di n numeri inizialmente ordinato in ordine crescente, ma nel quale k valori sono stati diminuiti per cui risultano "fuori posto".

Si scriva un algoritmo che riordina l'array in O(n+klg k), non necessariamente in loco.

Esercizio k fuori posto

Si consideri un array di n numeri inizialmente ordinato in ordine crescente, ma nel quale k valori sono stati diminuiti per cui risultano "fuori posto", cioè tali che l'array non risulta ordinato. Si scriva un algoritmo che riordina l'array in O(n+klg k), non necessariamente in loco.

L'idea è di copiare in un array d'appoggio i k elementi fuori posto, per poi ordinarli con il mergesort, con un costo θ(k lg k) nel caso peggiore, e infine fondere l'array ordinato così ottenuto con gli elementi rimanenti dell'array di partenza, questa operazione è lineare nel numero degli elementi da fondere, θ(n), quindi si ha un costo θ(n + k lg k), nel caso peggiore.
Qualche dettaglio in più è necessario per realizzare l'idea:

1. **come si individuano gli elementi fuori posto**
2. **come realizzare la fusione**

individuare i fuori posto

Come si individuano gli elementi fuori posto in un array A come quello descritto nella traccia?

La prima condizione cui si pensa è che è fuori posto ogni elemento A[i] tale che A[i-1] > A[i], visto che l'array è ordinato in ordine crescente, che è corretta se i k elementi diminuiti non sono consecutivi.
Infatti se A[j] e A[j-1] sono stati diminuiti entrambi può capitare che A[j-1] ≤ A[j] ma che in realtà siano entrambi fuori posto:
Per esempio se A = [15,18,60,80,100] e si diminuiscono 3 elementi portando 18 a 9, 60 a 10 e 100 a 5, si ha l'array A= [15,9,10,80,5], e il confronto a due a due porterebbe a individuare come fuori posto solo 9 e 5, mentre anche 10 deve essere riposizionato nell'array ordinato finale.

individuare i fuori posto

La soluzione è di portarsi dietro il più grande elemento trovato fino a quel momento nella scansione, ogni elemento più piccolo o uguale di quello trovato tra gli elementi successivi viene considerato fuori posto, e quindi memorizzato in un array B, mentre il successivo elemento più grande diventa il nuovo elemento di confronto. Il tempo di questo passo è in θ(n).

Per esempio se A = [15,18,60,80,100] e si diminuiscono 3 elementi portando 18 a 9, 60 a 10 e 100 a 20, si ha l'array A= [15,9,10,80,20].
Si considera una variabile FuoriPosto che si inizializza a 15, per cui 9 viene copiato in B, poi anche 10, poi FuoriPosto diventa 80 e quindi anche 20 viene memorizzato in B
Se invece venissero diminuiti 15 a 3, 60 a 20 e 100 a 5, ottenendo l'array A = [3,18,20,80,5] verrebbe costruito B = [5], la variabile FuoriPosto verrebbe inizializzata a 3, poi aggiornata a 18, a 20 e infine a 80, così che il 5 viene individuato come fuori posto.

realizzare la fusione

A questo punto si ha il vettore degli elementi fuori posto e si può ordinarlo con il mergesort. Questo array avrà al più dimensione k, perché qualche elemento diminuito potrebbe comunque non essere fuori posto.

Come realizzare la fusione? Il modo più semplice e anche meno oneroso in termini di memoria aggiuntiva è di copiare in un array C, che avrà almeno n - k elementi, gli elementi non fuori posto durante la scansione precedentemente descritta per trovare i fuori posto.
L'array C risulterà ordinato.
A questo punto si realizza la fusione in A dei due arrays B e C, con un costo θ(n) sia in tempo che in spazio.

Nota. B e C verranno inizializzati come array di k e n elementi rispettivamente e bisognerà aggiungere in coda un valore maggiore di ogni elemento presente inizialmente in A, e bloccare il processo di fusione quando n elementi risultano sistemati in A.