Algoritmi e Strutture Dati

Foglio 3
13/03/2023

Esercizio 1. Applicate il Teorema dell'esperto per determinare limiti asintotici stretti per le seguenti ricorrenze:

1. $T(n) = 2T(n/4) + 1$
2. $T(n) = 2T(n/4) + \sqrt{n}$
3. $T(n) = 2T(n/4) + n$
4. $T(n) = 2T(n/4) + n^2$

Esercizio 2. Qual è il tempo di esecuzione di QUICKSORT quando tutti gli elementi dell'array A hanno lo stesso valore?

Esercizio 3. Considerate la variante di MERGE-SORT in cui una delle due chiamate ricorsive è sostituita da una chiamata a QUICKSORT. Qual è il tempo di esecuzione di questo nuovo algoritmo (nel caso peggiore)?

Esercizio 4. Disegnate l'albero di decisione per l'ordinamento di 3 interi corrispondente all'algoritmo MERGE-SORT.

Esercizio 5. Sia A un vettore di n interi distinti e ordinati. Vogliamo determinare se esiste un indice i tale che $A[i] = i$. Mostrare che è possibile fare ciò in $O(\log n)$ passi.

Esercizio 6. Siano dati un vettore A di n interi ed un intero k. Progettare un algoritmo efficiente che trova, se esiste, una coppia di indici i, j con $i \neq j$ tali che $A[i] + A[j] = k$.

Esercizio 7. Un vettore A di dimensione n ha un *elemento preponderante* se più della metà dei suoi elementi sono uguali. Dato un vettore di dimensione n, si vuole progettare un algoritmo efficiente in grado di stabilire l'esistenza di un elemento preponderante e, nel caso, di trovarlo. Gli elementi del vettore non sono necessariamente numeri e quindi non siamo in grado di fare confronti del tipo $A[i] > A[j]$ (possiamo pensare gli elementi del vettore come dei file .gif, per esempio). Tuttavia, siamo in grado di dire se $A[i] = A[j]$ in $O(1)$ passi. Fornire un algoritmo che risolva tale problema in $O(n \log n)$ passi.

Esercizio 8. Supponete che l'ultimo ciclo di COUNTING-SORT visto a lezione sia rimpiazzato da
for $j = 1$ **to** $A.length$
Dimostrate che il nuovo algoritmo opera ancora correttamente. Il nuovo algoritmo è stabile?

Esercizio 9. Descrivete un algoritmo che, dati n numeri interi in $[0, k]$, svolga un'analisi preliminare del suo input e poi risponda nel tempo $O(1)$ a qualsiasi domanda su quanti degli n interi ricadono nell'intervallo $[a, b]$. L'algoritmo deve impiegare un tempo $O(n + k)$ per l'analisi preliminare.

Esercizio 10. È possibile ordinare n interi in $[0, n^3 - 1]$ in tempo $O(n)$?