Algoritmi di ordinamento

Esercizio 3 - Febbraio 2019

Il candidato elenchi gli algoritmi di ordinamento che conosce, indicando per ognuno di essi se si tratta di un metodo iterativo o ricorsivo, e indicando per ognuno la complessità nel caso ottimo e nel caso pessimo.

1. Insertion Sort: È un algoritmo iterativo, che usa un ciclo per analizzare uno alla volta gli elementi del vettore e sposta di un posto verso destra gli elementi maggiori. Al caso ottimo (il vettore è già ordinato) ha complessità $\Omega(n)$, poiché l'algoritmo deve scorrere tutto il vettore per verificare che sia ordinato; al caso pessimo (il vettore è ordinato al contrario) ha complessità $O(n^2)$, poiché ogni iterazione dovrà scorrere e spostare ogni elemento prima di poter inserire il primo elemento.
2. Selection Sort: È un algoritmo iterativo, che usa un ciclo per selezionare l'elemento più piccolo di un vettore V[i…n], dopodiché scambia questo elemento con l'i-esimo. È composto da due cicli for che confrontano tra loro i numeri, e il tempo di calcolo è determinato dal numero di confronti effettuati, perciò il tempo di esecuzione non dipende dall'input ma dalla lunghezza del vettore. Per valutare la complessità dobbiamo analizzare i due cicli for: $\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c$ dove c è una costante e indica il costo

   $$\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} 1 = \sum_{i=1}^{n-1} (n - (i+1) + 1) = \sum_{i=1}^{n-1} (n - i) = \sum_{i=1}^{n-1} n - \sum_{i=1}^{n-1} i$$

   che equivale a $n(n-1) - \frac{n(n-1)}{2} = \frac{n(n-1)}{2} = \frac{n^2 - n}{2} \approx \frac{n^2}{2}$

   ovvero $\theta(n^2)$.
3. Bubble Sort: È un algoritmo iterativo che compara due elementi adiacenti e li scambia se sono nell'ordine sbagliato. L'algoritmo itera il processo finché non vengono più eseguiti scambi, ovvero finché la lista non è ordinata È un algoritmo molto inefficiente. L'algoritmo effettua $\frac{n^2}{2}$ confronti ed $\frac{n^2}{2}$ scambi (circa) nel caso

pessimo, dunque ha complessità $O(n^2)$. Nel caso ottimo (vettore ordinato) l'algoritmo si limita a scorrere il vettore senza effettuare scambi, e ha dunque complessità lineare $\Omega(n)$.

4. Merge Sort: È un algoritmo ricorsivo che divide il vettore in due parti, le ordina, poi le combina in maniera ordinata. Il Merge ha costo $O(n)$ perché ogni elemento viene confrontato con gli altri e viene spostato o meno, tuttavia viene spostato nella sua posizione definitiva al più una volta. Il MergeSort, chiama ricorsivamente sé stesso e usa Merge per unire i risultati: MergeSort richiama sé stessa due volte, e ogni volta su metà della sequenza in input (n/2), la complessità al caso pessimo è quindi $\Theta(n \log n)$

5. Quick Sort: È un algoritmo ricorsivo: si sceglie un perno e si divide l'array in due parti, la 1ª composta da elementi ≤ del perno, la 2ª da elementi $>$ del perno, in seguita si ordina ricorsivamente prima la prima parte, poi la seconda. Caso pessimo: abbiamo un pivot che genera uno sbilanciamento nella divisione in parti dell'array, così da dover poi effettuare n-1 confronti, poi n-2 confronti, n-3 ecc, cioè ogni volta il numero di confronti diminuisce sempre e soltanto di 1 elemento, fino alla fine delle chiamate. Dunque ho
$$n + c(n-1) + c(n-2) + c(n-3) + \ldots + 2c =$$
$$c(n + (n-1) + (n-2) + (n-3) + \ldots + 2) = c\left(\frac{n(n+1)}{2} - 1\right)$$
(-1 perché la somma parte da 2) che equivale a circa $\Theta(n^2)$. Il caso ottimo lo abbiamo invece quando l'algoritmo divide l'array in due sezioni perfettamente bilanciate, entrambe di dimensione *n/2*; in questo caso il tempo di esecuzione è $O(n \log n)$.