InsertionSort VS Merge-Sort

Ilda Serjanaj

Aprile 2022

Indice

1 Introduzione al problema

2 Introduzione

Nella seguente relazione vengono presentati gli algoritmi di ordinamento Insertion-Sort e Merge-Sort e ne vengono comparate le prestazioni in funzione di vettori in input di dimensione crescente.

Per evidenziare al meglio l'andamento dei due algoritmi, i grafici si riferiscono a vettori di dimensione crescente fino a 4000 incrementandola di 10 ad ogni passo. Per valutarne la complessità cosidereremo l'ordinamento di tre vettori dati in input: il primo formato da dati random, il secondo da dati ordinati in ordine decrescente e l'ultimo in ordine crescente.

3 Insertion sort

L'algoritmo Insertion-Sort è efficente nell' odinamento di pochi elementi, esegue lo stesso funzionamento dell'ordinamento delle carte da gioco: parto con una mano vuota, prendo una carta per volta dal tavolo e la inserisco nella posizione corretta delle carte che ho in mano via via, confrontando la carta da inserire con le singole carte da destra verso sinistra in ogni momento le carte in mano sono ordinate. L'algoritmo ordina quindi i numeri in input **sul posto**: i numeri sono risistemati all'interno dell'array A senza richiedere ulteriore memoria aggiuntiva. Quando la procedura Insertion-Sort è completata, l'array A contiene la sequenza ordinata.

I tempi di esecuzione di Insertion-Sort nei tre casi sono:

Caso peggiore Si verifica quando gli elementi del vettore in input sono ordinati al contrario.
Il tempo di esecuzione di Insertion-Sort è $T(n) = \Theta(n^2)$

Caso medio Il tempo di esecuzione è $T(n) = \Theta(n^2)$

Caso migliore Si verifica quando gli elementi nel vettore sono già correttamente ordinati.
Il tempo di esecuzione è dato da $T(n) = \Theta(n)$

4 Merge-Sort

Merge-Sort segue il paradigma **Divide et Impera** ed 'e ricorsivo. Si divide il vettore dei dati in due parti (n/2), si richiama sui due vettori ricorsivamente fino a quando non si rimane con un solo elemento. Quest'algoritmo utilizza spazio di archiviazione aggiuntivo per l'ordinamento dei vettori: in totale usa 3 vettori (2 per memorizzare ogni metà, e la terza per l'elenco ordinato finale unendo le altre due, ogni vettore viene quindi ordinato in modo ricorsivo. Alla fine tutti i sotto-array ordinati vengono uniti. I tempi di esecuzione di Merge-Sort nei tre casi sono sempre uguali a $T(n) = \Theta(n \, lg \, n)$.

5 Confronto tra Insertion sort e Merge-Sort

5.1 Confronto nel caso medio

Come si può osservare in figura 1, nel caso di un array in input da ordinare composto da numeri casuali, rappresentante il caso medio di InsertionSort. Nel caso medio di Insertion-Sort, il tempo di esecuzione è asintoticamente uguale al tempo di esecuzione nel caso peggiore; ovvero una crescita quadratica in relazione alla dimensione dell'array in input da ordinare. Dunaque come si può vedere dalla figura 1è molto più veloce Merge-Sort.

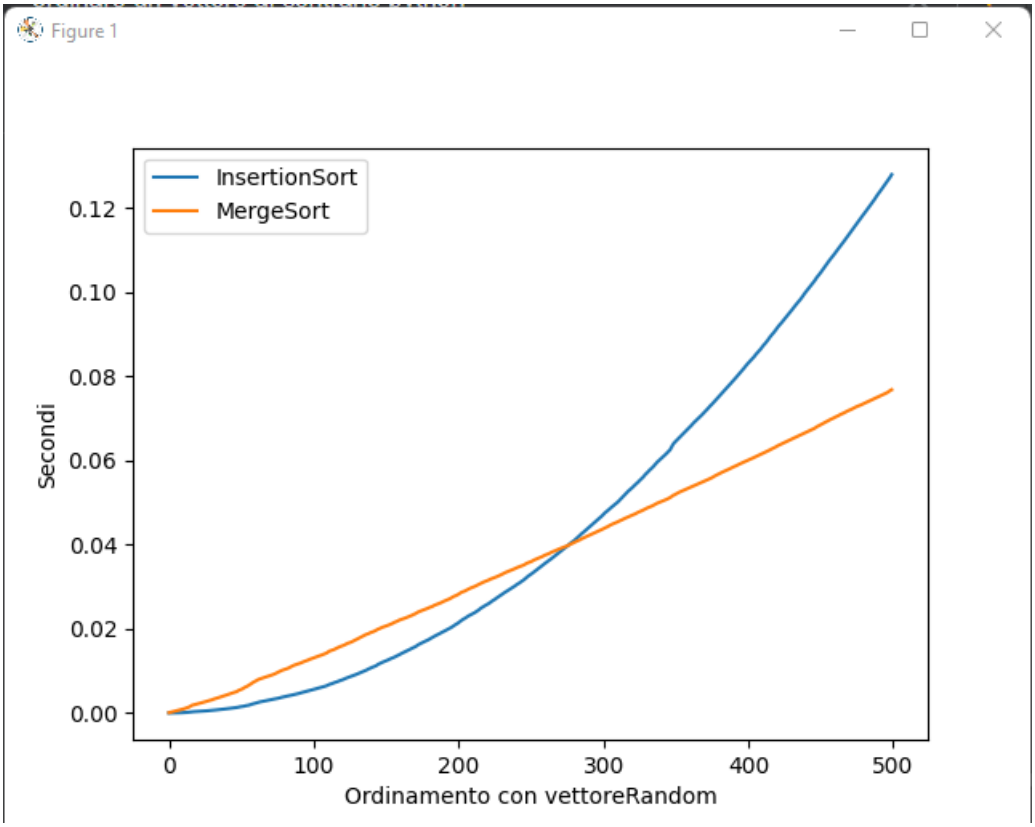

Figura 1: Confronto nel caso medio

5.2 Confronto nel caso peggiore per InsertionSort

Il peggiore di Insertion-Sort corrisponde al caso di un array ordinato al contrario. In questo caso il tempo di esecuzione dell'algoritmo cresce in modo quadratico rispetto alla dimensione dell'array. Come si può verificare dai grafici in figura 2, il tempo di esecuzione di InsertionSort nel caso di array ordinato al contrario cresce come $O(n^2)$ in funzione della dimensione n del vettore in input da ordinare risultando quindi essere, ancora una volta, più lento di Merge-Sort.

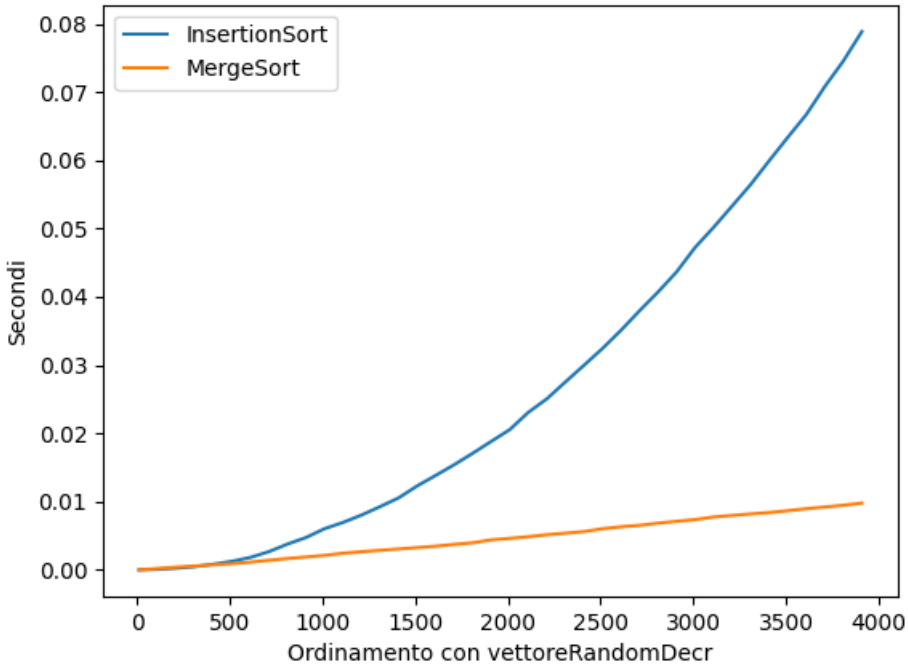

Figura 2: Confronto nel caso peggiore di Insertion-Sort

5.3 Confronto caso migliore InsertionSort

Il caso migliore di Insertion-Sort si verifica quando i valori sono già ordinati nel modo corretto. In tal caso il tempo di esecuzione dell'algoritmo cresce linearmente rispetto alla dimensione dell'array in input da ordinare, quindi come si vede anche dall'immagine 3, questa volta risulta più lento Merge-Sort.

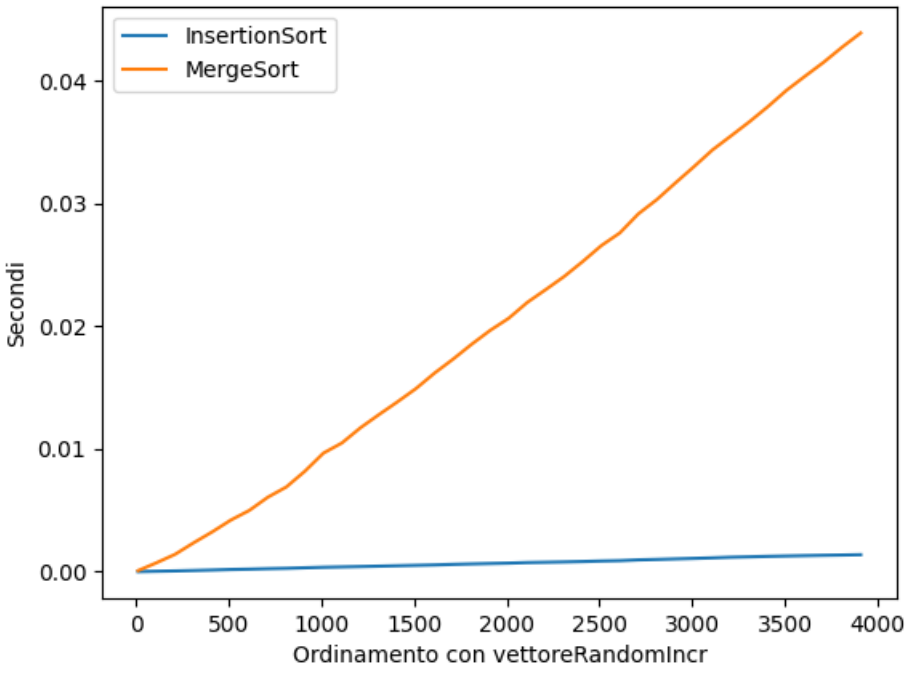

Figura 3: Confronto nel caso migliore di Insertion-Sort

6 Conclusioni

Tramite i vari test è stato possibile verificare i risultati teorici attesi. Come è stato possibile verificare dai vari esperimenti, la dimensione dell'array in input da ordinare nonché l'ordinamento iniziale dei valori influenzano il tempo di esecuzione dei due algoritmi. Si è dimostrato che Merge-Sort è più veloce in quasi tutti i casi in esame, poiché il suo caso medio, migliore e peggiore hanno la stessa complessità temporale, al contrario di Insertion-Sort il quale risulta più veloce sono nel caso di array in input già ordinato.

I test sono stati eseguiti su un Huawei D14 con processore 2,10 GHz AMD Ryzen 5 3500U, RAM 8 GB 2400 MHze sistema operativo Windows 11.