Insertion-Sort e Quicksort: Analisi e confronto tra i due algoritmi di ordinamento

Luca Bindini

1 Introduzione

In questa relazione si andrà ad analizzare e confrontare due algoritmi di ordinamento: Insertion-Sort e Quicksort. Per entrambi gli algoritmi verranno eseguite più prove (10 per ciascuna dimensione) su array di dimensione crescente, arrivando fino alla dimensione massima (5000 elementi) aumentando gradualmente con uno step di 50 elementi.

2 Insertion-Sort

Insertion-Sort è un algoritmo di ordinamento iterativo in-place ovvero i valori sono ordinati sul posto (non è richiesta ulteriore memoria di appoggio). In questa prima sezione andremo ad analizzare il tempo di esecuzione del suddetto algoritmo al variare della dimensione e della particolare permutazione dei valori iniziali dell'array.

2.1 Tempo di esecuzione di Insertion-Sort

Il tempo di esecuzione di Insertion-Sort varia a seconda della permutazione dei dati iniziali, in particolare si distinguono i tre casi:

1. **Caso migliore** Il tempo di esecuzione è $T(n) = \Theta(n)$.
   Si verifica quando l'array iniziale è gia correttamente ordinato.

2. **Caso medio** Il tempo di esecuzione è $T(n) = \Theta(n^2)$.
   Si verifica quando l'array iniziale è permutato in maniera casuale.

3. **Caso peggiore** Il tempo di esecuzione è $T(n) = \Theta(n^2)$.
   Si verifica quando l'array iniziale è ordinato al contrario.

2.1.1 Caso Migliore

Per analizzare il caso migliore di INSERTION-SORT sono state eseguite varie prove con array ordinati di dimensione crescente. I risultati dei tempi di esecuzione sono visibili attraverso la tabella ed il grafico qui di seguito. Nel caso del grafico per ogni dimensione è stata fatta una media tra le varie prove.

Dim.Array	10	100	1000	5000
Prova 1	5.96046e-06	1.59740e-05	0.00013	0.00065
Prova 2	3.81469e-06	1.50203e-05	0.00013	0.00064
Prova 3	3.81469e-06	1.48023e-05	0.00013	0.00070
Prova 4	3.09944e-06	1.38342e-05	0.00013	0.00069
Prova 5	3.81469e-06	1.38282e-05	0.00013	0.00069
Prova 6	4.76316e-06	1.40666e-05	0.00014	0.00065
Prova 7	4.05311e-06	1.40766e-05	0.00013	0.00070
Prova 8	3.09944e-06	1.40645e-05	0.00013	0.00075
Prova 9	4.05311e-06	1.50203e-05	0.00018	0.00066
Prova 10	3.89765e-06	1.41348e-05	0.00021	0.00066

Tabella 1: Tempi di esecuzione di Insertion-Sort nel caso migliore espressi in secondi.

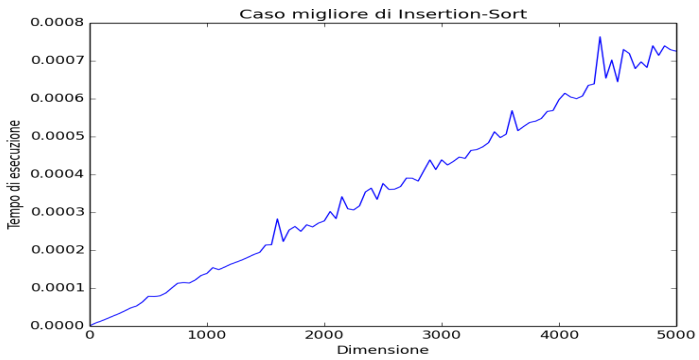

Figura 1: Grafico del tempo di esecuzione di Insertion-Sort nel caso migliore.

Come si evince dalla tabella e dal grafico il tempo di esecuzione cresce in modo lineare.

2.1.2 Caso Medio

Per analizzare il caso medio di Insertion-Sort sono state eseguite varie prove con array casuali di dimensione crescente. I risultati dei tempi di esecuzione sono visibili attraverso la tabella ed il grafico qui di seguito. Nel caso del grafico per ogni dimensione è stata fatta una media tra le varie prove.

Dim.Array	10	100	1000	5000
Prova 1	7.15255e-06	0.00030	0.03204	0.75846
Prova 2	6.91413e-06	0.00033	0.03175	0.78704
Prova 3	7.86781e-06	0.00031	0.03005	0.78067
Prova 4	8.10623e-06	0.00033	0.02951	0.75981
Prova 5	5.00679e-06	0.00025	0.03001	0.75864
Prova 6	4.05311e-06	0.00029	0.03744	0.75985
Prova 7	5.00679e-06	0.00028	0.03728	0.79267
Prova 8	6.19888e-06	0.00030	0.03121	0.78210
Prova 9	5.00673e-06	0.00041	0.03136	0.76599
Prova 10	5.02679e-06	0.00030	0.03069	0.80292

Tabella 2: Tempi di esecuzione di Insertion-Sort nel caso medio espressi in secondi.

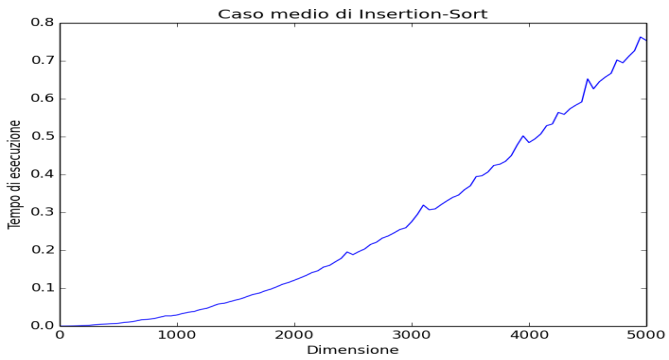

Figura 2: Grafico del tempo di esecuzione di Insertion-Sort nel caso medio.

Come si evince dalla tabella e dal grafico il tempo di esecuzione cresce in modo quadratico.

2.1.3 Caso Peggiore

Per analizzare il caso peggiore di INSERTION-SORT sono state eseguite varie prove con array ordinati al contrario di dimensione crescente. I risultati dei tempi di esecuzione sono visibili attraverso la tabella ed il grafico qui di seguito. Nel caso del grafico per ogni dimensione è stata fatta una media tra le varie prove.

Dim.Array	10	100	1000	5000
Prova 1	1.00135e-05	0.00070	0.06080	1.53612
Prova 2	8.10623e-06	0.00063	0.06184	1.52078
Prova 3	6.91413e-06	0.00070	0.06165	1.54564
Prova 4	7.86781e-06	0.00084	0.05937	1.53545
Prova 5	6.91413e-06	0.00063	0.06270	1.53101
Prova 6	7.86781e-06	0.00082	0.05882	1.55715
Prova 7	7.86781e-06	0.00061	0.06193	1.55118
Prova 8	7.15255e-06	0.00061	0.05924	1.59748
Prova 9	7.86781e-06	0.00064	0.06267	1.60986
Prova 10	6.91413e-06	0.00061	0.06040	1.53140

Tabella 3: Tempi di esecuzione di Insertion-Sort nel caso peggiore espressi in secondi.

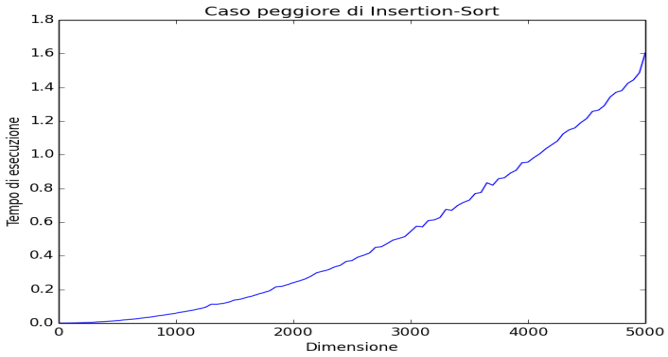

Figura 3: Grafico del tempo di esecuzione di Insertion-Sort nel caso peggiore.

Come si evince dalla tabella e dal grafico il tempo di esecuzione cresce in modo quadratico.

3 Quicksort

QUICKSORT è un algoritmo di ordinamento ricorsivo basato sul paradigma divide et impera. In questa prima sezione andremo ad analizzare il tempo di esecuzione del suddetto algoritmo al variare della dimensione e della particolare permutazione dei valori iniziali dell'array.

3.1 Tempo di esecuzione di Quicksort

Il tempo di esecuzione di QUICKSORT varia a seconda della permutazione dei dati iniziali, in particolare si distinguono i tre casi:

1. **Caso migliore** Il tempo di esecuzione è $T(n) = \Theta(nlgn)$.

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

3. **Caso peggiore** Il tempo di esecuzione è $T(n) = \Theta(n^2)$.

A differenza di INSERTION-SORT il caso medio di QUICKSORT è più vicino al suo caso migliore rispetto che a quello peggiore. Questo perché intuitivamente nella procedura di PARTITION se anche avessimo alcuni tagli sfavorevoli (ovvero i due sottoarray sono molto sbilanciati) si avrebbe comunque un costo logaritmico, quest'ultimo risultato è possibile dimostrarlo attraverso l'albero di ricorsione.

3.1.1 Caso Medio

Per analizzare il caso medio di QUICKSORT sono state eseguite varie prove con array casuali di dimensione crescente. I risultati dei tempi di esecuzione sono visibili attraverso la tabella ed il grafico qui di seguito. Nel caso del grafico per ogni dimensione è stata fatta una media tra le varie prove.

Dim.Array	10	100	1000	5000
Prova 1	1.21593e-05	0.00014	0.00199	0.01077
Prova 2	1.38282e-05	0.00013	0.00169	0.00921
Prova 3	1.50203e-05	0.00013	0.00173	0.00961
Prova 4	1.00135e-05	0.00013	0.00159	0.01095
Prova 5	1.09672e-05	0.00012	0.00186	0.00990
Prova 6	1.19209e-05	0.00013	0.00177	0.01019
Prova 7	9.05990e-06	0.00011	0.00256	0.01040
Prova 8	1.00135e-05	0.00021	0.00154	0.00979
Prova 9	2.40802e-05	0.00017	0.00163	0.00991
Prova 10	6.03199e-06	0.00013	0.00163	0.00973

Tabella 4: Tempi di esecuzione di Quicksort nel caso medio espressi in secondi.

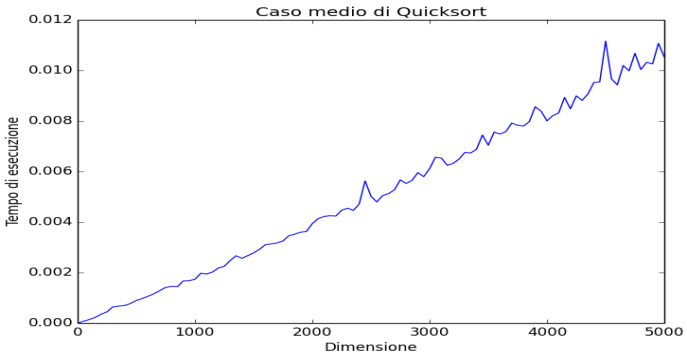

Figura 4: Grafico del tempo di esecuzione di Quicksort nel caso medio.

4 Confronto tra Insertion-Sort e Quicksort

Eseguendo l'algoritmo di QUICKSORT negli array dei casi migliori e peggiori di INSERTION-SORT si può notare come sostanzialmente essi siano nel caso peggiore di QUICKSORT in quanto per array ordinati (o ordinati al contrario) la procedura di PARTITION produrrà sempre lo sbilanciamento tra i due sotto-array più sfavorevole poiché non essendo una versione RANDOMIZED dell'algoritmo stesso viene sempre scelto come PIVOT l'ultimo elemento dei vari array.

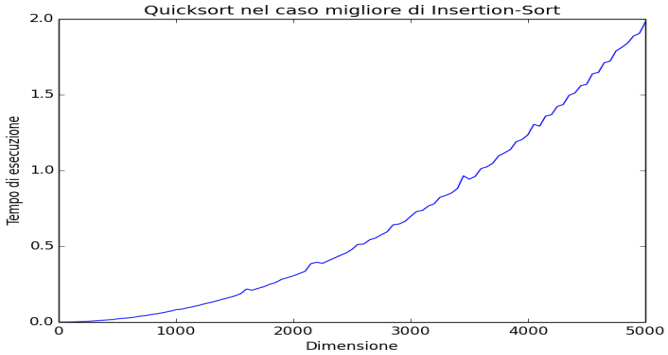

Figura 5: Grafico del tempo di esecuzione di Quicksort nel caso migliore di Insertion-Sort.

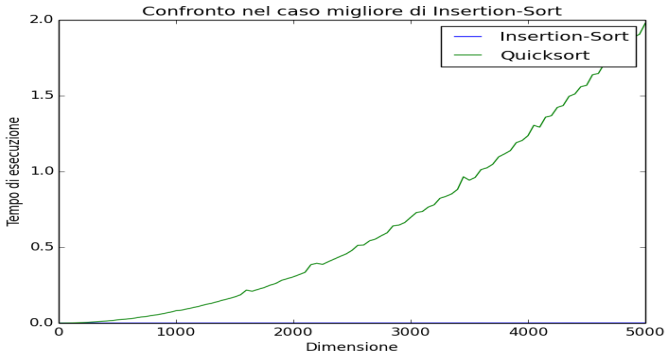

Figura 6: Confronto del tempo di esecuzione tra Quicksort e Insertion-Sort nel caso migliore di Insertion-Sort.

Come si può notare dai grafici qui di seguito nel caso peggiore di Insertion-Sort entrambi gli algoritmi hanno costo quadratico.

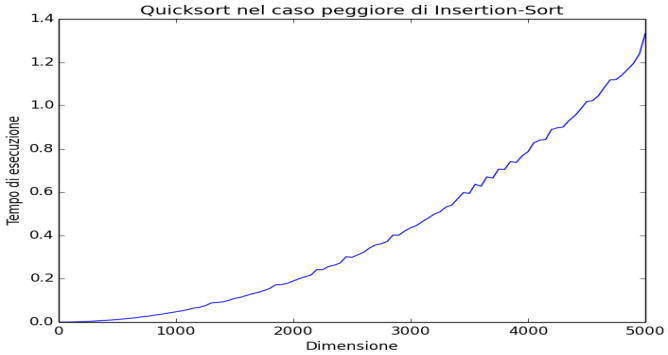

Figura 7: Grafico del tempo di esecuzione di Quicksort nel caso peggiore di Insertion-Sort.

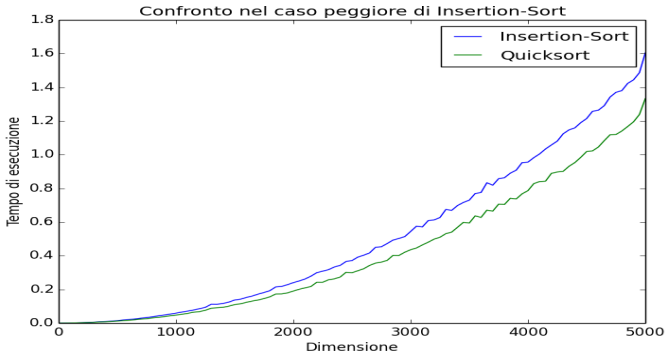

Figura 8: Confronto del tempo di esecuzione tra Quicksort e Insertion-Sort nel caso peggiore di Insertion-Sort.

4.1 Confronto tra i casi medi

Nel caso medio cioè con un array iniziale permutato casualmente è evidente come il costo $\Theta(nlgn)$ di QUICKSORT sia molto inferiore a $\Theta(n^2)$ di INSERTION-SORT.

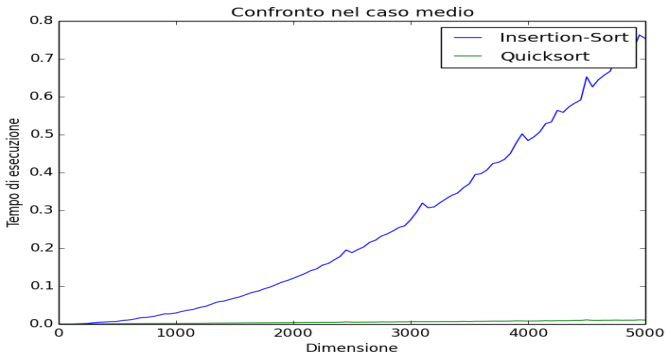

Figura 9: Confronto del tempo di esecuzione tra Quicksort e Insertion-Sort nel caso medio.

5 Conclusioni

Dopo i vari test si può constatare come QUICKSORT sia più veloce nella maggior parte dei casi, in quanto il caso medio è molto più vicino al caso migliore che a quello peggiore, al contrario di INSERTION-SORT.
[1]

[1] I test sono stati effettuati su una piattaforma con le seguenti specifiche:

1. Processore: 2,9 GHz Intel Core i5
2. Ram: 8 GB 2133 MHz LPDDR3
3. SO: Mac OS 10.14.5 Mojave