Relazione Esercizio 1

Primo uso: Insertion Sort e Merge Sort

Premessa: Nello scrivere questa relazione abbiamo omesso il calcolo dei tempi di caricamento degli array, e abbiamo considerato solamente il tempo impiegato dall'algoritmo per ordinare l'array passatogli.

Nell'analisi dei tempi di ordinamento degli algoritmi ***Insertion Sort*** e ***Merge Sort***, abbiamo considerato i 5 seguenti casi:

1. Ordinamento di un array vuoto
2. Ordinamento di un array con gli elementi già ordinati in ordine crescente
3. Ordinamento di un array un gli elementi ordinati in modo casuale
4. Ordinamento di un array con gli elementi già ordinati in ordine decrescente
5. Ordinamento basato sull'algoritmo:
    1. Nel caso dell'*Insertion Sort*, ordinamento dei primi 10 elementi dell'array *integers.csv*
    2. Nel caso del *Merge Sort*, ordinamento di tutti gli elementi (20'000'000 in totale) dell'array *integers.csv*

In seguito a diverse prove, abbiamo ottenuto i seguenti risultati, riportati in forma tabellare:

	Insertion Sort	**Merge Sort**
Caso 1	Frazione di secondo	Frazione di secondo
Caso 2	Frazione di secondo	Frazione di secondo
Caso 3	Frazione di secondo	Frazione di secondo
Caso 4	Frazione di secondo	Frazione di secondo
Caso 5	**Frazione di secondo ***	17-25 secondi

(*) Non abbiamo ritenuto opportuno tentare di **far ordinare l'array da 20'000'000 elementi all'*Insertion Sort*,** poiché solo per ordinare 100'000 (1/200 della dimensione di *integers.csv*) elementi ha impiegato un tempo superiore ai due minuti e mezzo (poiché si tratta di un algoritmo con complessità $O(n^2)$), mentre il *Merge Sort* non ha mai impiegato (per via della sua complessità O(n log n)) più di 25 secondi per ordinare tutti i 20'000'000 elementi.

Secondo uso: trova la somma di due elementi

Per quel che riguarda la seconda funzione, abbiamo progettato un algoritmo che prende in input un array T di N interi e un secondo array S di M interi, che verifica se in T vi sono due interi la cui somma è un elemento di S.

Questo algoritmo, ordina prima di tutto l'array A utilizzando l'algoritmo *Merge Sort*, dopodiché un ciclo verifica che S contenga almeno un elemento che equivalga alla somma di due interi contenuti in T. Questo ciclo ha, nel peggiore dei casi, una complessità O(M), e per via dell'algebra di O-grande, tutto l'algoritmo si ritrova ad avere una complessità totale di

$$\Omega(1) * \Omega(N \log N) = \Omega(N \log N) \text{ nel migliore dei casi}$$
$$O(M) * O(N \log N) = O(N \log N) \text{ nel peggiore dei casi}$$

Di conseguenza, questo algoritmo ha la stessa complessità del *Merge Sort*, che nel caso medio si ritrova ad avere una complessità di $\Theta(k \log k)$ (con k = numero di elementi dell'array da ordinare).