Considerazioni PRIMO ESERCIZIO

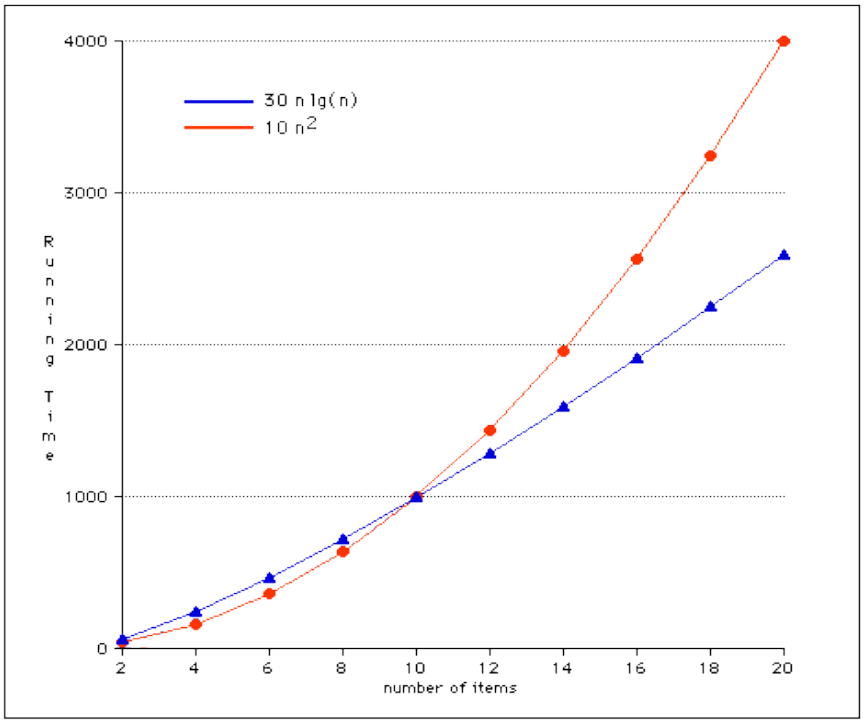
- Andrea Torella (matr. 912579)
- Agostino Messina (matr. 913813)

Alla base dell'uso dell'algoritmo "*Merge-BinaryInsertion Sort*" vi sono due principali osservazioni sull'analisi dell'efficienza degli algoritmi e la relativa notazione O-grande.

L'algoritmo "*merge sort*" è più lento di un semplice algoritmo di ordinamento con complessità $O(n^2)$ (come, ad esempio, l'algoritmo "*insertion sort*") per input di piccola dimensione.
Questo avviene nonostante "*merge sort*" abbia una complessità O(nlogn).
La risposta va ricercata sulla natura dell'analisi della complessità con notazione O-grande, Infatti quest'ultima concerne solo con il tasso di crescita della funzione presa in esame.
Ciò vuol dire che un algoritmo con complessità O(nlogn) sarà più veloce di un algoritmo con complessità $O(n^2)$ "col tempo", ovvero quest'ultimo può essere più veloce per input di piccola dimensione.

Per migliorare ulteriormente l'efficienza dell'algoritmo, si usa una versione modificata di *"insertion sort"* in cui la posizione, all'interno della sezione ordinata del vettore in cui inserire l'elemento corrente, è determinata tramite ricerca binaria.

Di seguito sono riportati i tempi dei due algoritmi in funzione della dimensione dell'input:



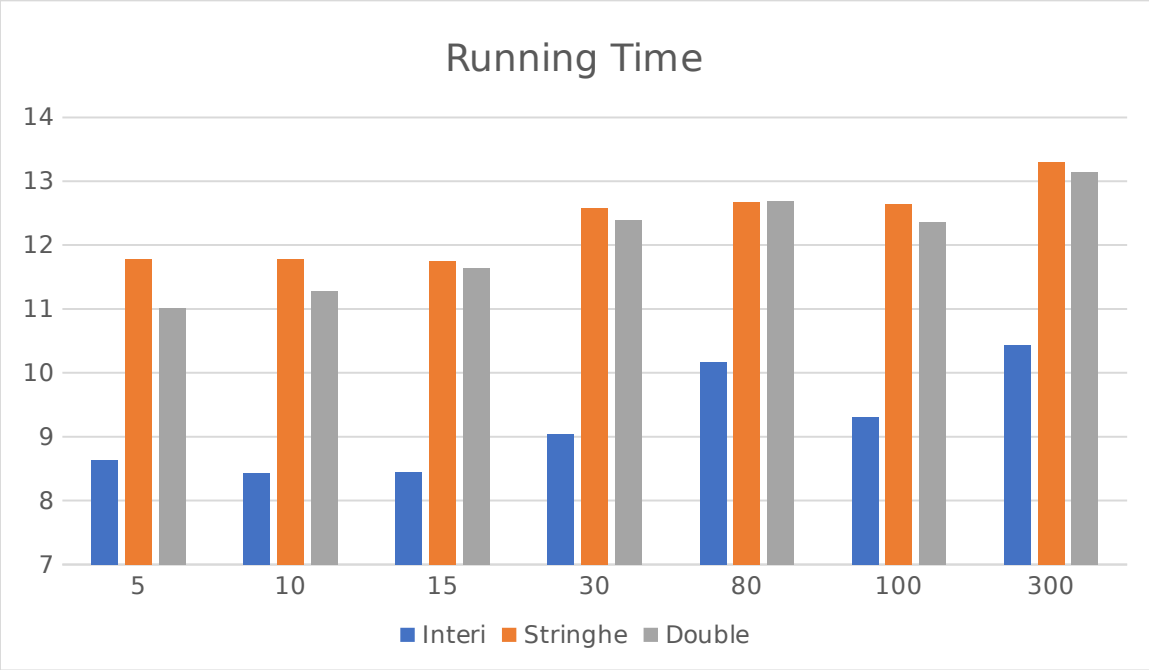
Come si evince dal grafico, il tempo impiegato dal "*binary insertion sort*" è minore per input piccoli (circa nell'intervallo [0, 10]) rispetto a "*merge-sort*".

Per input > 10 invece possiamo notare come la funzione relativa a “*binary insertion sort*” cresca in modo esponenziale e quindi molto più velocemente rispetto a quella relativa a “*merge-sort*”.

Dunque, un valore di k ottimale dovrà sicuramente essere scelto in funzione della dimensione dell’input e, in base a quanto visto teoricamente, sarà circa 10.

Verifichiamo, con delle prove sperimentali, se tutto ciò si riflette anche in un caso reale. Andiamo quindi ad eseguire l’algoritmo con valori di k diversi e misuriamo i tempi di esecuzione:

	5	**10**	**15**	**30**	**80**	**100**	**300**
Interi	8,638884	8,434494	8,453302	9,036321	10,176368	9,309481	10,429496
Stringhe	11,789869	11,777472	11,741521	12,578571	12,679888	12,639044	13,299001
Double	11,014853	11,284241	11,643765	12,391752	12,696121	12,358651	13,148835



I risultati sperimentali confermano quanto detto, si evince infatti che per k = 10 si ha il miglior risultato in termini di tempo : 8,434494 s (per ordinare in base al campo intero).
Al crescere di k possiamo vedere come la differenza in termini di tempo sia sempre più grande.