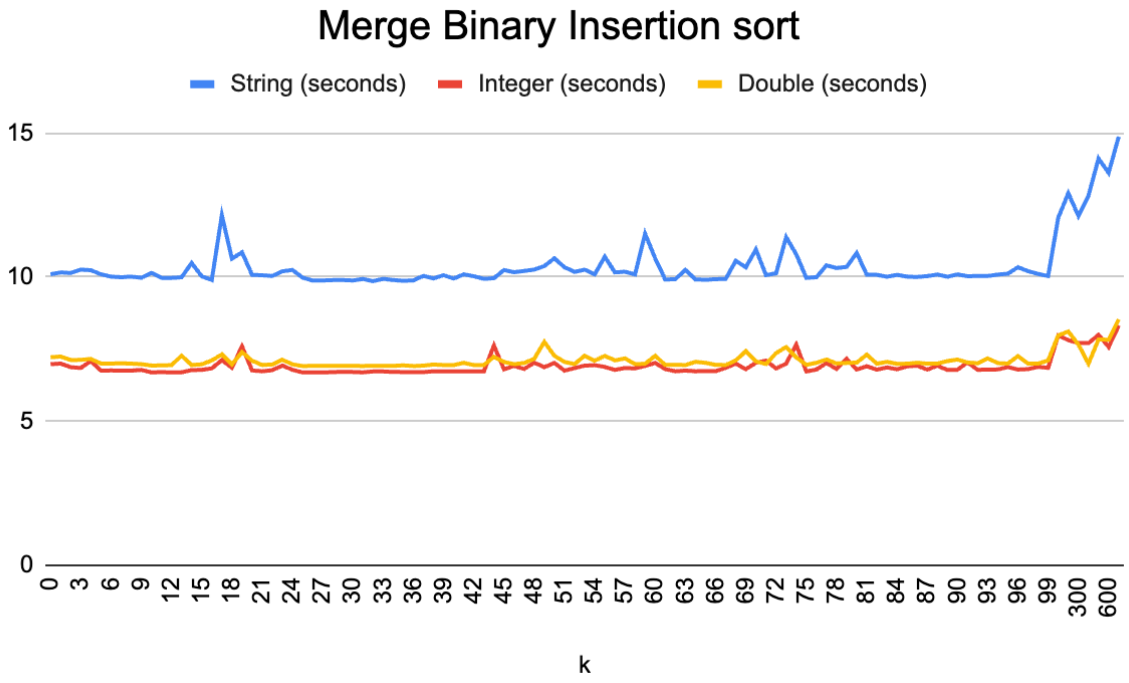
Merge Binary Insertion Sort



Nel grafico riportato qui sopra possiamo vedere come al variare di un parametro k, che decide quando viene utilizzato il binary insertion sort e quando il merge sort, varia il tempo di ordinamento del file csv dato in input al nostro algoritmo. I tempi in secondi sono stati calcolati per i tre parametri di ordinamento: stringhe, double e interi.

Come ci aspettavamo il tempo per ordinare il file rimane perlopiù stabile per i valori di k più bassi (0-99) e inizia a salire soltanto quando il valore di k passa la soglia del 99 continuando ad aumentare con l'avvicinamento di k a valori più grandi. Quindi i valori per ottenere un bilanciamento ottimale tre merge e binary insertion sort sono compresi tra 0 e 99.
Inoltre possiamo vedere una differenza del tempo di ordinamento anche in base al parametro scelto per ordinare nel caso dei double e degli interi tra i cinque e i dieci secondi mentre per le stringhe si sorpassa la soglia dei dieci, un risultato che era prevedibile visto il meccanismo con il quale si ordinano queste ultime.

I risultati ottenuti sono concordi con le nostre aspettative poiché l'algoritmo merge sort è efficiente nel caso di una grossa quantità di dati mentre è preferibile utilizzare il binary insertion sort per una quantità più modesta. La motivazione di questo risultato risiede nell'implementazione stessa dei due algoritmi; Il merge sort richiede operazioni di divisione e fusione di vettori, che utilizzano molte risorse con quantità di dati piccole annullando il vantaggio in termini di complessità, O(n log n), e aumentando l'overhead. Mentre il binary insertion sort con un implementazione più semplice è più adatto al trattamento di pochi dati.