Relazione del 1° esercizio Laboratorio di Algoritmi e Strutture Dati

Zhao Xiao matr: 913828 mail: xiao.zhao@edu.unito.it

L'esercizio si compila con il commando <make>, e si usa il commando <./client "nome del file csv"> per avviare.

Durante l'esecuzione, viene richiesto di inserire il valore di k, e se si vuole ordinare in modo crescente. Al termine dell'esecuzione, viene chiesto se si vuole il risultato del ordinamento in .csv, nominato in <output_file.csv>.

In questo esercizio, ho fatto dei test con I valori di k da 0 fino a 3000 (fig 1, fig 2). Ho notato che quando k = 0, il tempo di esecuzione è simile al tempo di k = 100 per i String, mentre per Integer e Double si ha un tempo simile al k = 200.

In base agli esperimenti dimostro che, con k = 20~40, il tempo di esecuzione è ottimale. Per cui, si consiglia di scegliere il valore di k in questa area.

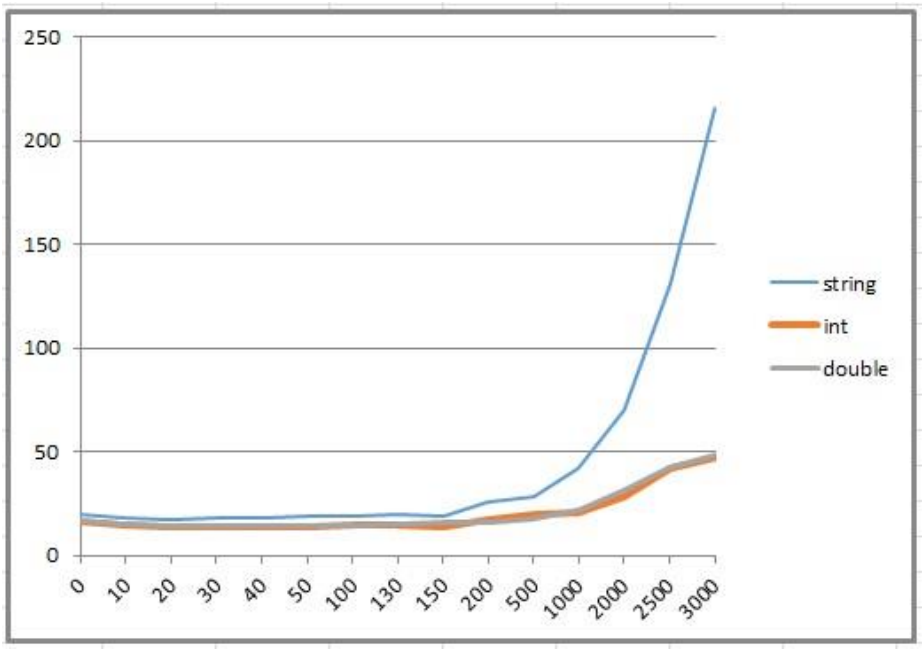
Con valori di k elevati, si rileva un tempo di esecuzione più lento perché insertion-sort non è adatto a gestire una dimensione troppo grande.

In base agli studi(fig 3), si dimostra che il tempo di esecuzione di ins-sort (O(n^2)) è minore di merge (O(n log n)) nei dimensioni piccoli. Questo succede perché la loro complessità temporale è simile, e la esecuzione dell'algoritmo ins-sort è più veloce di merge che deve fare ancora altri ricorsioni.

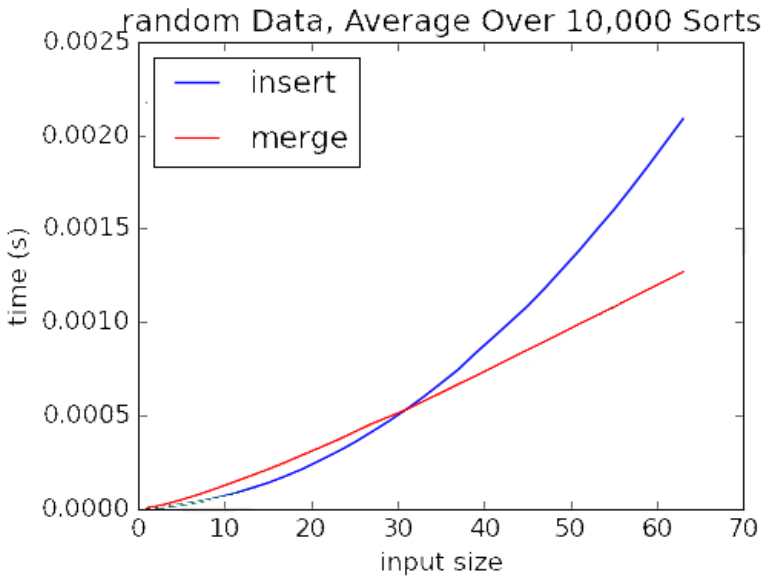
Quindi il ins-sort si dimostra migliore con una dimensione minore di circa 30, ma più la dimensione cresce, il tempo di esecuzione cresce in modo esponenziale. Questo fatto corrisponde con il risultato ottenuto dal mio progetto.

	A	B	C	D	E	F	G
1	k		string	int	double	total	
2	0		19.951031	16.859546	16.73924	53.54982	
3	10		17.919143	14.354553	15.343064	47.61676	
4	20		17.474824	14.182125	14.407116	46.06407	
5	30		18.21238	14.051663	14.365384	46.62943	
6	40		18.224038	13.995147	14.293527	46.51271	
7	50		18.881668	14.270332	14.506808	47.65881	
8	100		19.441148	14.416251	14.601685	48.45908	
9	130		19.854765	14.480574	14.830252	49.16559	
10	150		19.295413	14.131592	15.793495	49.2205	
11	200		26.138115	17.148144	16.041295	59.32755	
12	500		28.66457	19.972167	17.613015	66.24975	
13	1000		42.239543	20.488289	21.697488	84.42532	
14	2000		70.355982	28.105405	31.455277	129.9167	
15	2500		130.908033	42.301563	42.542936	215.7525	
16	3000		215.839909	47.369373	48.155527	311.3648	
17							

(figura 1)



(figura 2)



(figura 3)