VE281 Programming Assignment One: Sorting

Project Report

Wei Yizhou 5133709269

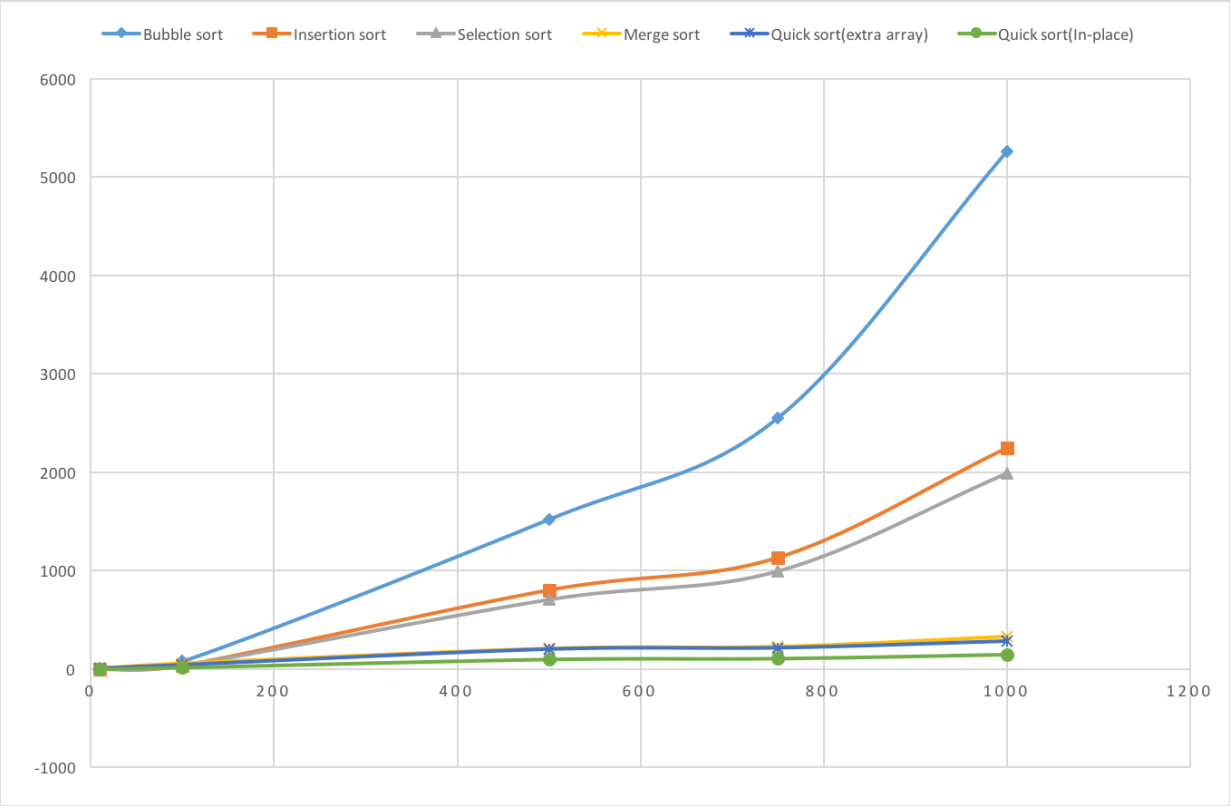

Figure1.Runtime versus Array size of each sorting algorithm

Project Design

In order to compare the runtime of each sorting algorithm with different array size, we first have to choose 5 different n representing the size of the array. We originally chose 10,100,1000,10000,100000 for the size but we soon found out that comparing to such large array size the difference of the runtime for each algorithm is so significant that the graph was not able to give effective information. Thus we use 10 100 500 750 1000 as the array size. Then we use mrand48() command to generate a random array with n elements and copy it for six times. Finally, we apply the six sorting algorithm to them and use clock() command to record the runtime.

Result

The comparison is a great success; the data we get from the program were close to our expectation. Bubble sort, Insertion sort and Selection sort used far more time than Merge sort and Quick sort when array size gets large. We also find that Bubble sort used almost twice as much time as Insertion sort and Selection sort. The reason is because that the Bubble sort can only switch the adjacent elements which take much more switch to get a element from last to the first. We also find that two different Quick sort have different runtime. Quick sort(in-place) is more efficient than Quick sort(extra array), because using extra array mean we have to do extra copy steps which takes more time. From this project, we learned the realization of each soring algorithm and get a deeper understanding about their efficiency.

Array size	**Bubble sort**	**Insertion sort**	**Selection sort**	**Merge sort**	**Quick sort(extra array)**	**Quick sort (In-place)**
10	9	2	2	7	7	2
100	80	39	36	57	42	17
500	1521	803	704	207	200	99
750	2580	1133	993	226	214	108
1000	5258	2243	1986	328	281	148

Table1. runtime data of each algorithm with different array size