Name : Ansari M.Saeem M.Saleem

Uid : 2019430001

Subject : AAC

Expt no : 1

Aim : Write a program to randomly generate 800k elements and compare time complexity for Insertion sort, Heap sort and Merge sort.

Aim:

Write a program to randomly generate 800k elements and compare time complexity for Insertion sort, Heap sort and Merge sort.

Objective:

- Implementation of Insertion sort, heap sort and merge sort in java/C/ C++ /python programming language.
- Implementation of File handling concepts in java/C/C++/python programming language.
- Analysing and comparison time complexity for Insertion sort, heap sort and merge sort.

Methodology:

- Eight lakh elements are randomly generated and write it into a file.
- The Data set is then separated into 16 files of fifty thousand elements and stored into txt files.
- Each 16 files of 50k elements is sorted and create 16 files of sorted elements.
- A pair sets of 50k elements files is merged then sorted and stored into 8 files of 100k elements.
- Process is repeated for files containing 100k elements and then further repeated for files containing 200k elements and then for files containing 400k and finally resulting in a single file of 800k elements.
- All sorting techniques are applied on all above steps and time required to sort the file is store into sheet.

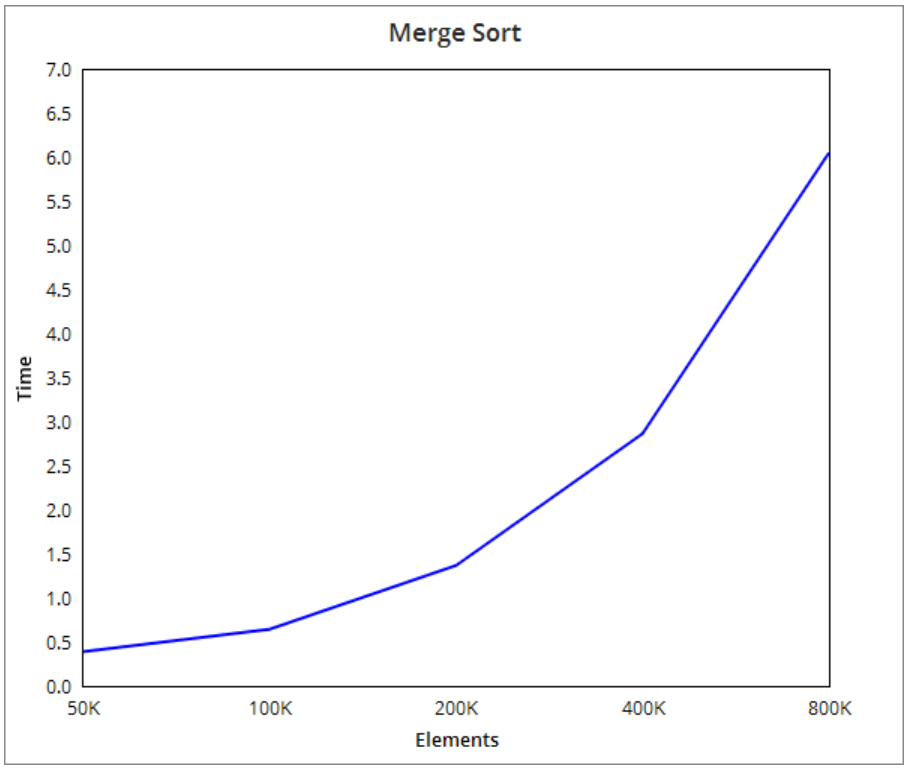
Implementation:

- Random number is generated using randint() function from random libray of python.
- To store ,split and merge files ,file handling concept of python is used.
- All sorting techniques are implimented in python.
- Time required to sort each file is store into data frames and then into sheets using pandas.

Result:

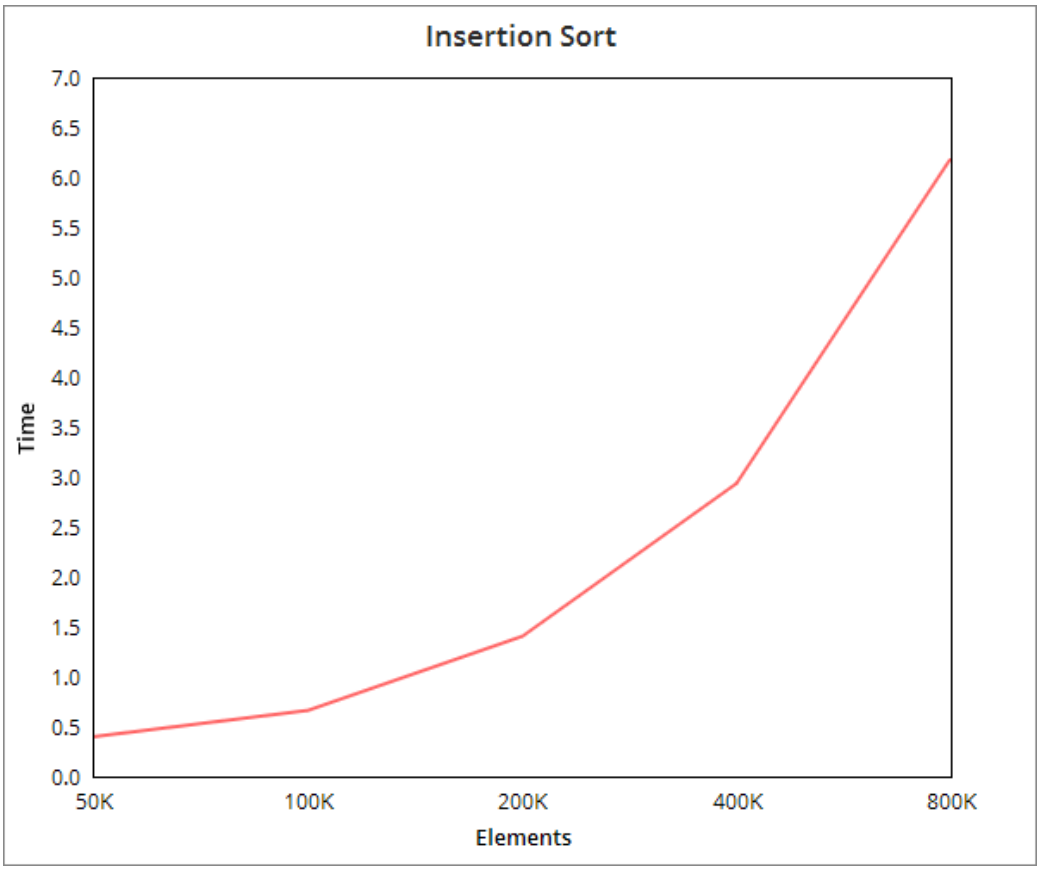
Merge:

	50K	100K	200K	400K	800K
	0.37852	0.64437	1.35133	2.84833	6.05686
	0.37535	0.6414	1.36854	2.88262	
	0.37192	0.64136	1.36208		
	0.43118	0.64346	1.38823		
	0.37315	0.64311			
	0.37153	0.64374			
	0.36902	0.6416			
	0.38407	0.64178			
	0.37188				
	0.48489				
	0.37431				
	0.41668				
	0.38125				
	0.37221				
	0.3692				
Average	0.38834	0.6426	1.36754	2.86548	6.05686



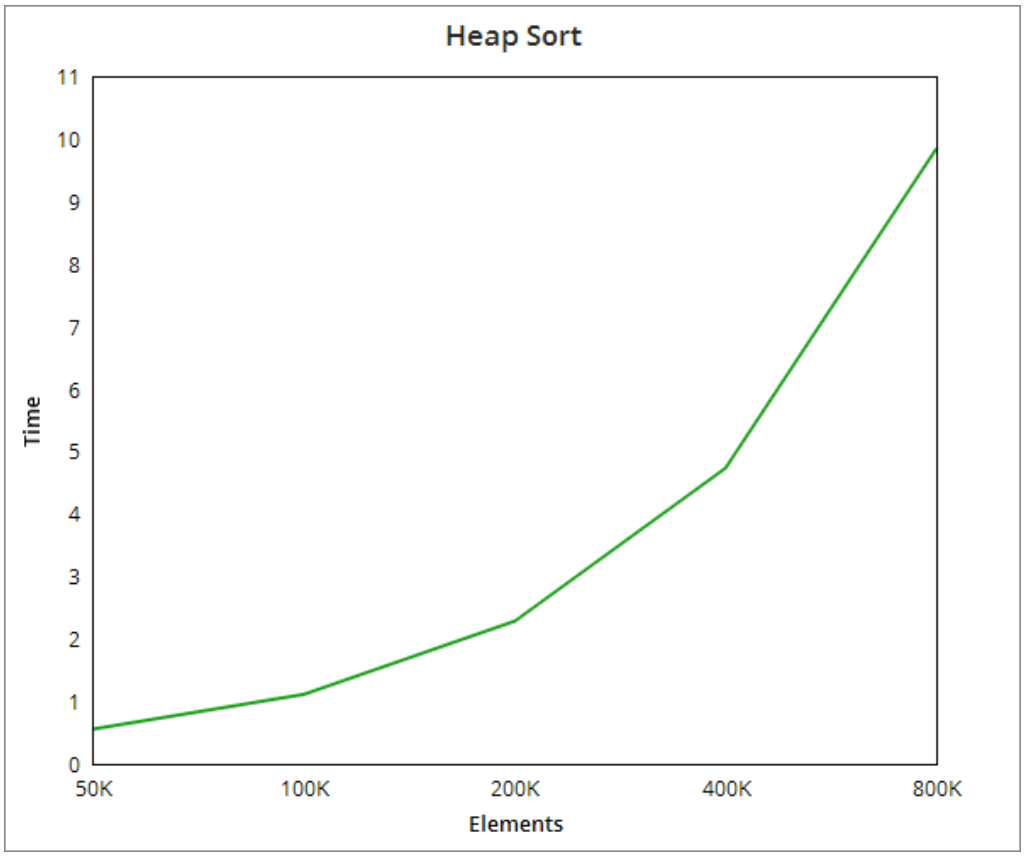
Insertion sort:

	50K	100K	200K	400K	800K
	0.38267	0.66239	1.38747	2.92122	6.19811
	0.38498	0.6609	1.40481	2.95518	
	0.38491	0.65959	1.39812		
	0.44091	0.662	1.42893		
	0.3844	0.66173			
	0.3811	0.66216			
	0.37867	0.66114			
	0.39415	0.66006			
	0.38137				
	0.495				
	0.38398				
	0.43786				
	0.39196				
	0.38221				
	0.37868				
	0.38155				
Average	0.39777	0.66125	1.40483	2.9382	6.19811

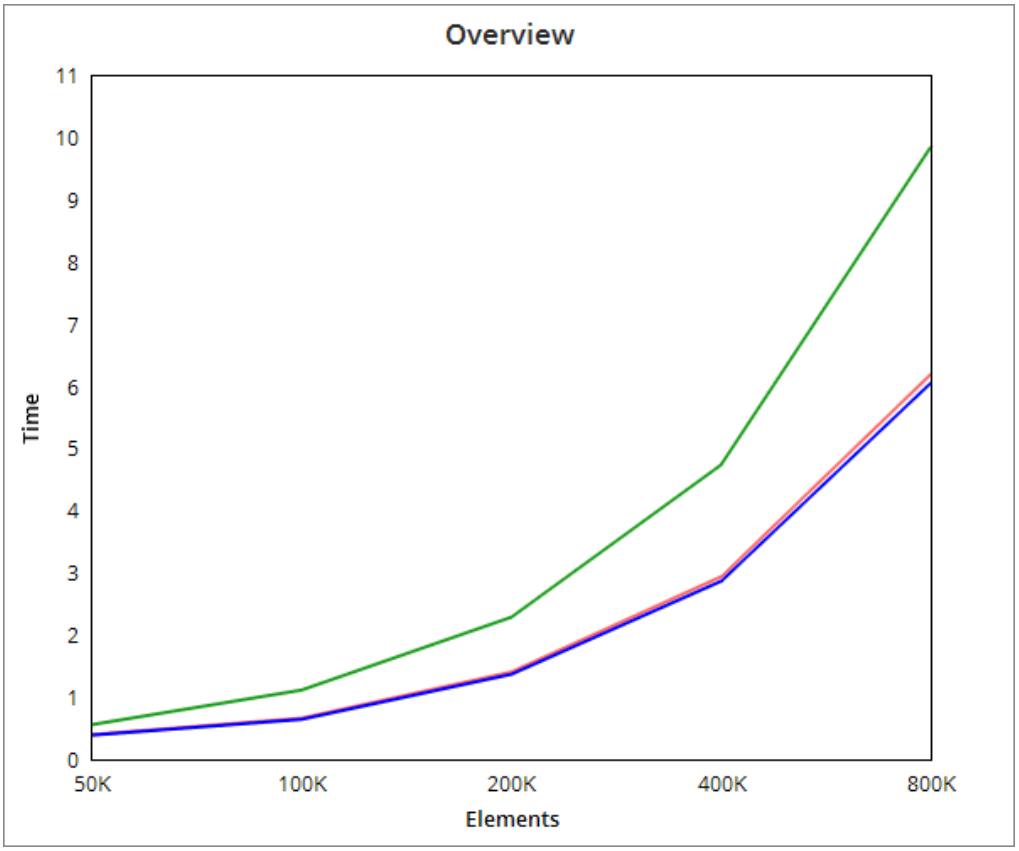


Heap Sort:

	50K	100K	200K	400K	800K
	0.55303	1.10744	2.27481	4.76279	9.85128
	0.55049	1.11409	2.29961	4.71086	
	0.58854	1.11768	2.29142		
	0.55301	1.1163	2.27489		
	0.54667	1.11553			
	0.54633	1.11549			
	0.54411	1.10891			
	0.55037	1.10497			
	0.54486				
	0.55849				
	0.54917				
	0.64276				
	0.54736				
	0.54655				
	0.54215				
	0.54191				
Average	0.55661	1.11255	2.28518	4.73682	9.85128



Overview Analysis:



Conclusion:

In this experiment, we come to know that merge sort and insertion sort required almost same time.On the other hand heap sort required slightly more time as compare to them.Therotically merge sort and heap sort should perform well but this not happen because from second iteration onwards we are passing two sorted array where merge sort and heap sort performs in its worst case.