RMIT University Vietnam

COSC2451

Assignment 1

Nguyen Phuc Thinh, Trinh Luan, Cao Phi Hung

8/3/2013

Contents

The Test Result:.......... 2

Test Results analization: 3

Notes:.......... 4

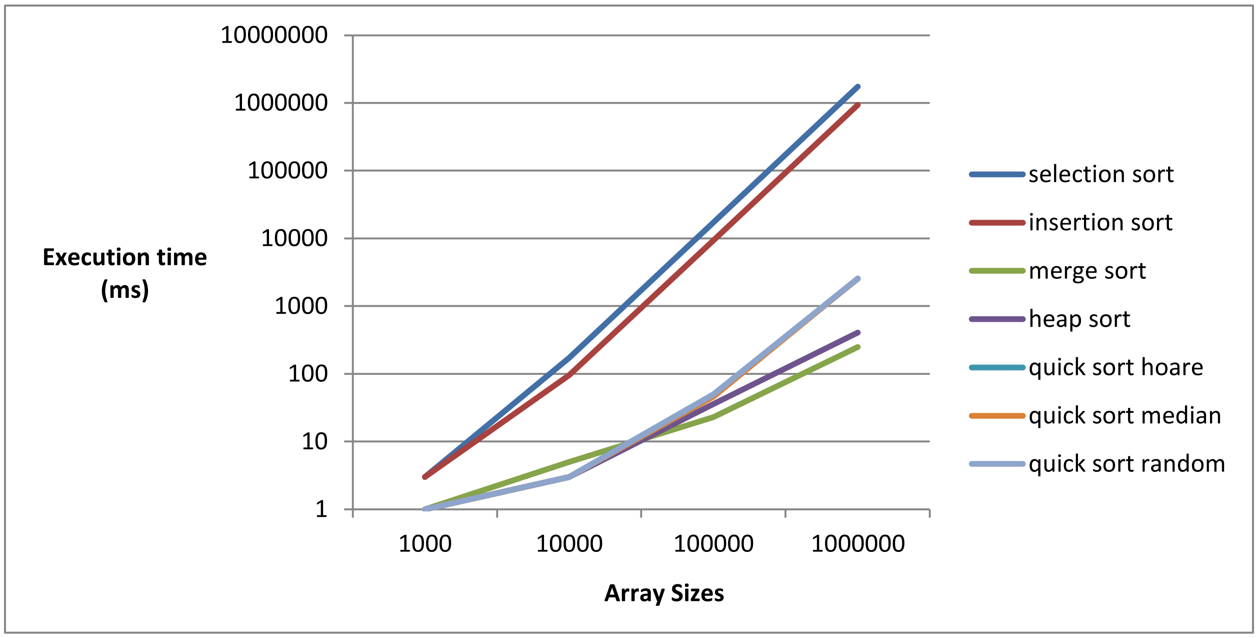
References: 5

The Test Result:

The complete execution time (in ms) after running pt_gen_timings is presented in the following table (the nbits is set to 10 when generating the random arrays, so the values in the array will vary between 0 and $2^{10} - 1$):

Array size	1000	10000	100000	1000000
selection sort	3	173	17290	1751810
insertion sort	3	96	9356	938930
merge sort	1	5	23	250
heap sort	1	3	36	406
quick sort hoare	1	3	50	2540
quick sort median	1	3	46	2570
quick sort random	1	3	50	2576

The following graph visualizes the information from the table above:



Test Results analization:

Best performance function: The function that performs the best is merge_sort, which has an execution time of merely 250 ms for an array of size 1.000.000.
Worst performance function: The function that performs the worst is selection_sort, which has an execution time of 1.751.810 ms (around 30 minutes) for an array of size 1.000.000.

Explanation:

- Merge Sort applies the method "divide and conquer", which divides the array in many smaller sub-arrays, sort them, and then merge them back into the original array. This approach has the complexity of O(n log n).
- Selection Sort starts with finding the smallest element in the array and put it at the beginning, then it has to go through all the elements in the array to determine which element is the smallest among the rest of the array. Typically, fiven an array of n elements, it has to go and "select" the element n times, and for each time, it has to go through roughly n elements of the array. This results in the complexity of O(n^2), which is drastically larger than merge sort's.

Unexpected results: Except for the Segmentation fault (core dumped) error due to the team members' trivial mistakes in transfering the pseudo-code into C code, no unexpected result has been recorded during the completion of the assignment.

Notes:

The team is aiming for the competition. Thus, functions in the pt_sorting.c file are constantly renewed every few days in hoping to make the sort functions run faster. Some functions are rewritten by the team member who is in charge of them, some by the other team members who, by chance, found a better solution when doing research. An example of such function is the quick_sort_hoare, initially coded by Nguyen Phuc Thinh (s3372765) and had an execution time of 25.000 ms. After working on the quick_sort_median and quick_sort_random, Cao Phi Hung (s3372748) found out that his version of quick_sort has an execution time of only 2.500 ms. Seeing the wide gap between the 2 execution time, the team decided to reject the old quick_sort_hoare, and replace it with the later version that runs much faster. The old quick_sort_hoare is then stored in the file called pt_sorting_rejected.

References:

- Insertion Sort
http://en.wikipedia.org/wiki/Insertion_sort

- Merge sort
http://www.codenlearn.com/2011/10/simple-merge-sort.html
http://www.vogella.com/articles/JavaAlgorithmsMergesort/article.html

- Heap sort
http://www.roseindia.net/java/beginners/arrayexamples/heapSort.shtml
http://www.code2learn.com/2011/09/heapsort-array-based-implementation-in.html

- Quick sort (hoare, median and random)

http://eternallyconfuzzled.com/tuts/algorithms/jsw_tut_sorting.aspx