Design and Analysis of Algorithms

CS 302

Homework # 3

Q1) Comparison of Sorting algorithms

In this question, you are going to implement several sorting algorithms and compare their performance. The performance comparison will be based on time. You are free to use any language. However, since the assignment tries to compare the algorithm, the programming language, and the runtime environment (computer, operating system, compiler) have to be identical for all algorithms.

Sorting algorithms

1. Insertion Sort
2. Merge Sort
3. Quick Sort (Pivot is element stored in last index of array)
4. Quick Sort2 (Pivot is selected by using median of three random values)
5. Heap Sort

Data Size

1. 10
2. 100
3. 1000
4. 10000
5. 100000
6. 1000000

You should run sorting algorithms on following 4 types of integer data.

1. Completely random.
2. Sorted
3. Almost sorted: 90% of the items are in increasing order, but 10% of randomly-chosen items are random.
4. Reversed: sorted items are in reverse order.

Measure running time of each sorting algorithm using some library for clock time. In order to get good estimate of running time you should each sorting algorithm 5 times and record the average running time. You should repeat this for every size of input.
Create table for results and comparison of different sorting algorithms. Create separate table for each type of input. Each table should have running time of sorting and input size.

Table 1: Random data
Table 2: Sorted Data
Table 3: Almost sorted
Table 4: Sorted Data in reverse

Q2) Insertion Sort on small array instead in Merge Sort

Although merge sort runs in O (n lg n) worst-case time and insertion sort runs in $O(n^2)$ worst-case time, the constant factors in insertion sort can make it faster in practice for small problem sizes on many machines. Thus, it makes sense to ***coarsen*** the leaves of the recursion by using insertion sort within merge sort when sub problems become sufficiently small. Consider a modification to merge sort in which n=k sub lists of length k are sorted using insertion sort and then merged using the standard merging mechanism, where k is a value to be determined.

Implement this modified version of merge sort and report the optimal value of k such that when input size is smaller than or equal to k then it is more efficient to use insertion sort for sorting smaller sub lists.
You are required to conduct experiments and give your analysis in form of a report. For each experiment, generate the data set of n integers randomly and find the optimal value of k. Perform these experiments for n=1000, 10000, 50,000 and 100,000. In each experiment compute the running time of standard as well as modified merge sort for different values of k. In order to make your analysis more precise, conduct each experiment five times and record the average running time.

Submission

Submit complete code and report containing plots as zip file on slate.