CSC206 Algorithms and Paradigms
Spring 2017
Assignment 3 Report

Student Name: Akshay Poosarla Student ID: 218904037

Part1: Algorithm Comparison

Report the results of your sorting experiments in the tables below. All times should be in milliseconds.

		10000	100000	1000000
Insertion Sort	Sorted	0 ms	2 ms	5 ms
	Nearly Sorted	0 ms	1ms	6ms
	Reversely Sorted	68ms	6622 ms	318751
	Random	34ms	1414 ms	156427

		10000	100000	1000000
MergeSort	Sorted	3ms	23ms	115ms
	Nearly Sorted	2ms	12ms	100ms
	Reversely Sorted	2ms	12ms	111ms
	Random	2ms	24 ms	205ms

		10000	100000	1000000
QuickSort	Sorted	4 ms	18ms	76ms
	Nearly Sorted	1ms	7ms	70ms
	Reversely Sorted	2ms	9ms	65ms
	Random	2ms	11ms	131 ms

		10000	100000	1000000
STLSort	Sorted	0ms	2ms	7ms
	Nearly Sorted	1ms	25ms	66ms
	Reversely Sorted	0ms	2ms	2ms
	Random	1ms	61ms	115ms

Discussion

Discuss the following points:

Quicksort leads to stack overflow error when we chose pivot as last element in the array(No randomization) it is because each level the elements in the array is reduced by only one so for 1000000 as it leads to stack overflow.Assymptotic complexity will be theta(n-square) which is the worst case of quick sort.

1. For each data type (sorted, nearly sorted, reversely sorted, random), which algorithm has the best performance? Explain why?
   Execution time = Number of operations/speed

Execution time will be less when number of operations are less

Sorted: Insertion sort

For sorted list insertion sort takes theta (n) time. As the list is sorted the inner loop won't get executed and outer loop has n iterations. So number of operations are less.

Nearly Sorted: Insertion sort

For nearly sorted list Insertion sort has the best performance .No of executions of inner loop will be more than sorted list, but less than n. So it will be Omega (n) and Big O (n square). But when you compared with other algorithms the best case among them is Theta (N) which is more compared to insertion sort

Reversely Sorted: STL sort

STL Sort is best among the given sorts

Random: Quick Sort

Quick sort and Stl sort running times are almost similar .Choosing right pivot for quick sort is important for quick sort.

2. How does the performance of each algorithm change when we change the input type? Explain why?

   Insertion Sort:

   Sorted list: since it is sorted list no of executions of inner loop is zero so outer loop iterates till n. Insertion sort gives it best performance for this kind of input

   Partially Sorted List: No of executions of the inner loop is more than zero but less than theta (Nlogn).performance is better than reversely sorted and random input type

   Reversely Sorted: Number of executions of the inner loop is more when compared to previous input types. Insertion sort input is worse for this input type when compared with other sorts. Number of operations are more. (N square times)

   Randomly Sorted: Performance of this input will in between partially sorted and reversely sorted number of inner loop executions is between numbers of inner loops of partially sorted and reversely sorted. Operations lies between operations of partial and reverse sorted list (half of the reversely sorted).

   MergeSort: Merge sort performance won't be affected much by the type of input as it divides the problem into equal smaller problems.

   Quicksort: Sorted, Reversely Sorted, Nearly sorted when compared to above one random input numbers are slow.Performace mainly depends on selecting the pivot element. Selecting the right pivot element is key for performance of quick sort so input type doesn't change the performance but random input are slower than other types.

   STLSort: Sorted, Nearly sorted, reversely sorted, Random for all these types stl is more optimized and execution time will be lesser when compared with other sorting techniques mentioned here.

3. How does the performance of each algorithm change when we change the input size? Explain why?
   Insertion Sort: Insertion sort gives the best performance when the size is small .As the input size increases the performance of the insertion sort degrades because number of comparisons of the inner loop increases. So number of operations also increases.
   But number of comparisons of the inner loop is less (sorted data) even though the size increases the performance won't degrade.
   Merge Sort: it requires more space complexity than other sorts. As the input size increases copying elements into temporary array and then sorting it might take more execution time and that leads in the degrade in performance. Number of instruction increases as the input size increases.
   Quicksort : As the input size increases execution time also increases choosing the right pivot element for the partition helps us in getting better results irrespective of the size.
   STLSort: stl sort is inbuilt sort which works fine for any input size

4. Do you have any other observations or insights?
   Insertion sort works well for smaller inputs.
   Quick sort has less space complexity than merge sort. And by taking right pivot quick sort performs better than merge sort.

Part2: Quicksort Focused Study

Report the results of your focused study of Quicksort in the table below.

	Recursion Depth	**Execution Time (ms)**
Simple Random	53	127ms
Median of Three	39	169ms
with InsertionSort	37	110
Your Best Result (Extra Work)	37	99

Simple Random: In this simple random method by choosing a random number and swapping with the last index. Recursion depth differs based on the random element taken. This reduces the probability of taking largest or smallest number as pivot element

Comment lines 294 to 298 and 300- 301

Median of three: choosing three Random values from array and considering median value as pivot.Recurssion depth is less when compared to random as we are choosing the median of three numbers.

Comment lines 294 to 299 and 301

With Insertion Sort: As we know insertion sort works pretty good for the smaller inputs so this helps us in reducing the recursion depth and execution time.

Comments lines 300 and 301
Extra Credit:

I have tried different ways to optimize the array
First method: used three way partition to split the array by choosing two elements. This is slight variation as the base of log changes to nlog2 to nlog2 and much to my surprise this is giving larger times for randomly sorted elements.

There are two ways to optimize the quick sort

1) Choose the best pivot element for the array
2) Reduce the recursion depth

In order to choose the best pivot element I have taken the median of high low and mid value And as insertion sort is better for smaller arrays I have called insertion sort for size less than 40.I was able to reduce the around 25 ms.

Comment lines 299 and 300
And Inaddition to this when I am searching the techniques to enhance the quick sort I have come across the dualpivotselection element and tried to implement it.This technique is giving very good results and while researching about this I got to know that this is the technique which is being used in the inbuilt sort method. So I cannot use that ☹

System Specifications:
Operating system : windows 8 ,64 bit
Processor: Intel core i5
Ram :12 gb
Cores: 2