Data Structure & Algorithm Homework 2

Zhuohang Li

Q1.

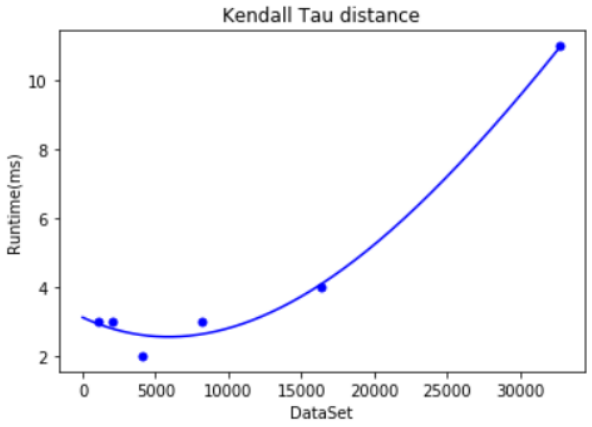
#Comparisons	data0.1024	data0.2048	data0.4096	data0.8192	data0.16384	data0.32768
InsertionSort	1023	2047	4095	8191	16383	32767
ShellSort	3061	6133	12277	24565	49141	98293

#Comparisons	data1.1024	data1.2048	data1.4096	data1.8192	data1.16384	data1.32768
InsertionSort	265553	1029278	4187890	16936946	66657561	267966668
ShellSort	3061	6133	12277	24565	49141	98293

The performance of insertion sort is sensitive to input. The best case is given a sorted array, there are only n-1 compares and no exchanges. The worst case is given a reverse sorted array, there will be $\frac{n^2}{2}$ compares and $\frac{n^2}{2}$ exchanges. In the average case, there are $\frac{n^2}{4}$ compares and $\frac{n^2}{4}$ exchanges. Insertion sort is slow for large number of data due to the exchange operation only involves adjacent elements. This could be really bad for example when the smallest element is at the end of the array, then n-1 exchanges will be needed to get it in place. Shell sort however can produce partially sorted array that can be efficiently sorted eventually by insertion sort.

Q2.

Data Set	Distance	Runtime(ms)
1024	264541	3
2048	1027236	3
4096	4183804	2
8192	16928767	3
16384	66641183	4
32768	267933908	11



The idea is to use the reverse-mapped array ainv[]which satisfies ainv[a[i]]=i, and then use b[] as the key of mapping to get a new array: bnew[i]=ainv[b[i]]. The number of inversions of the new array is the Kendall Tau distance between a[] and b[]. To calculate the number of inversions, consider merge sort.

Every time when we merge two arrays together, if the right element is smaller than the left one, then merging them together will decrease the number of inversions by exact the number of the remaining left array. For example when merging [3 5 7] and [2 4 6], 3 is greater than 2, therefore by putting 2

before 3 we are decreasing the number of inversions by 3. So we can get the number of inversions by keeping track of merge sort.

The overall complexity of this algorithms is determined by the merge sort part, which is nlog(n).

Q3.

Since this set of data is already sorted, insertion sort it again will only cost n-1 compares and 0 exchanges, which is fasted among sorting algorithms we have learnt so far. Another way of doing this is to utilize the fact of knowing every element in this data set by making n assignments instead of sorting. The complexity will be O(N).

Q4.

#Comparisons	0.1024	0.2048	0.4096	0.8192	0.16384	0.32768
Basic MergeSort	5120	11264	24576	53248	114688	245760
Bottom Up MergeSort	5120	11264	24576	53248	114688	245760

#Comparisons	1.1024	1.2048	1.4096	1.8192	1.16384	1.32768
Basic MergeSort	8954	19934	43944	96074	208695	450132
Bottom Up MergeSort	8954	19934	43944	96074	208695	450132

When the array length is pow of 2, the two merge sort algorithms are just making function calls in different orders, so they have exactly the same number of comparisons which is between $\frac{1}{2}n\log(n)$and $n\log(n)$.

Q5.

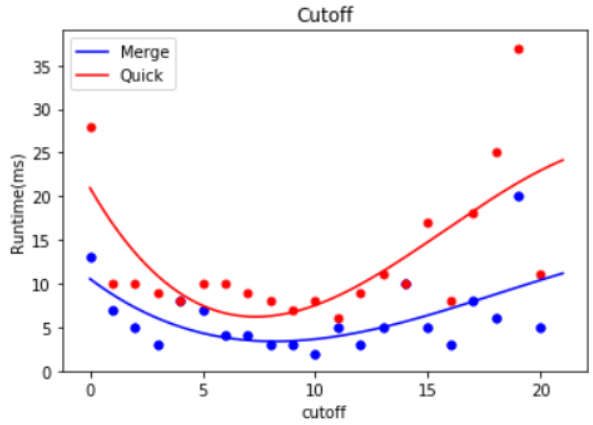
Basic Merge Sort		
DataSet	number of comparisons	Runtime(ms)
0.1024	5120	4
0.2048	11264	1
0.4096	24576	1
0.8192	53248	4
0.16384	114688	20
0.32768	245760	32

Basic Merge Sort		
DataSet	number of comparisons	Runtime(ms)
1.1024	8954	2
1.2048	19934	1
1.4096	43944	2
1.8192	96074	6
1.16384	208695	24
1.32768	450132	36

Quick Sort using Median-of-three		
DataSet	number of comparisons	Runtime(ms)
0.1024	12253	7
0.2048	27324	2
0.4096	58727	2
0.8192	127490	16
0.16384	272816	17
0.32768	568280	58

Quick Sort using Median-of-three		
DataSet	number of comparisons	Runtime(ms)
1.1024	12373	1
1.2048	26172	2
1.4096	56656	3
1.8192	130209	7
1.16384	268224	19
1.32768	578508	25

The complexity of merge sort is O(nlogn). The complexity of quick sort, depending on the input, can vary from O(nlogn) to O(n^2).



As shown in the diagram above, when cutoff=7, continue to increase the cutoff value will increase instead of decrease the runtime.

Q6.

The match-up is:

(5) (6) (1) (4) (3) (8) (2) (7)

Reference:

Algorithms 4th edition, by Robert Sedgewick & Kevin Wayne