Assignment 8 – Sorting

Write pseudo-code not Java for problems requiring code. You are responsible for the appropriate level of detail. For all the questions in this set, assume you are working in arrays.

1. How many comparisons and interchanges (in terms of file size n) are performed by Simple insertion sort for the following files:

i) A sorted file
ii) A file that is sorted in reverse order (that is, from largest to smallest)
iii) A file in which x[0], x[2], x[4]... are the smallest elements in sorted order, and in which x[1], x[3], x[5]... are the largest elements in sorted order, e.g. [3 14 5 15 9 18 11 19].

2. How many comparisons and interchanges (in terms of file size n) are performed by Shell Sort using increments 2 and 1 for the following files:

i) A sorted file
ii) A file that is sorted in reverse order (that is, from largest to smallest)
iii) A file in which x[0], x[2], x[4]... are the smallest elements in sorted order, and in which x[1], x[3], x[5]... are the largest elements in sorted order, e.g. [3 14 5 15 9 18 11 19].

3. Determine the number of comparisons (as a function of n and m) that are performed in merging two ordered files a and b of sizes n and m, respectively, by the merge method presented in the lecture, on each of the following sets of ordered files:

a. m=n and a[i] < b[i] < a[i+1], e.g. a=[6, 9, 12, 15, 29, 37] and b = [8, 10, 14, 25, 33, 45]

b. m=n and a[n] < b[1], e.g. a =[2, 5, 9] and b = [12, 14, 16]

a[i] refers the value in position i of file a, etc.

4. Determine the number of comparisons (as a function of n and m) that are performed in merging two ordered files a and b of sizes n and m, respectively, by the merge method presented in the lecture, on each of the following sets of ordered files:

a. m=n and a[n/2] < b[1] < b[m] < a[(n/2)+1],

e.g. a = [2, 5, 7, 55, 61, 72] and b =[9, 15, 17, 21, 29, 46]

b. m=1 and b[1] < a[1]
c. m=1 and a[n] < b[1]

a[i] refers the value in position i of file a, etc.

For questions 5 – 8, compare the efficiency of using sequential search on an ordered table of size n and an unordered table of the same size for the key *Item*:

5. **If no record with the key *Item* is present**

6. **If one record with the key *Item* is present and only one is sought.**

7. **If more than one record with the key *Item* is present and it is desired to find only the first**

8. **If more than one record with the key *Item* is present and it is desired to find them all.**

9. Let's sort using a method not discussed in class. Suppose you have *n* data values in in array *A*. Declare an array called *Count*. Look at the value in *A[i]*. Count the number of items in *A* that are smaller than the value in *A[i]*. Assign that result to *count[i]*. Declare an output array *Output*. Assign *Output[count[i]] = A[i]*. Think about what the size of *Output* needs to be. Is it *n* or something else? Write a method to sort based on this strategy.

10. Analyze the cost of the sort you wrote in the previous problem. What is the impact of random, ordered, or reverse ordered data?