**Assignment 10 – Sorting - Searching Ordered Data**

*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 ].**

1. For a sorted file there are O(n) comparisons, and 0 interchanges
2. For a list of length 2, there would be one comparison and one interchange. For a list of length 3 there would be 3 comparisons and 3 interchanges. Extrapolating, for a list of length n there would be ½ \* n^2 comparisons and ½ \* n^2 interchanges, which approaches O(n^2) complexity.
3. For a list of this style, a list of length 3 would require 3 comparisons and 1 interchange. A list of size 5 would require 7 comparisons and 3 interchanges. Extrapolating, a list of length n would require ¼ n^2 comparisons and 1/8 n^2 interchanges, which still approaches O(n^2) complexity.
4. **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 ].**

**[8, 7, 6, 5, 4, 3, 2, 1]**

**[7, 8, 5, 6, 3, 4, 1, 2]**

1. For a sorted file there are n/2 + n = 3n/2 comparisons and 0 swaps
2. For a list in reverse order, using the increment of 2 there would be n/2 comparisons and n/2 swaps. For the increment of 1, the first two items would require 1 comparison and 0 swaps. A list of length 4 would require 6 comparisons and 4 swaps. A list of size 6 would require 15 comparisons and 12 swaps. Expanding, a list of size n would require 2\*n^2 comparisons and 2 \* n^2 – n/2 swaps.
3. Using the increment of 2, there would be n/2 comparisons and 0 swaps. This is not a great increment for this list. After the increment of 2, it would require the same number of comparisons and interchanges as the insertion sort, which was ½ n ^ 2 and ½ \* n ^ 2.
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:**
   1. **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]**
   2. **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.**

1. Each insert into a sorted array (except the last one) will require a comparison. Therefore, the number of comparisons = n + m – 1.
2. In this example, you will only need comparisons for inserting the indices of array a into a sorted array. Inserting array b will not require any comparisons. Therefore the number of comparisons is n.
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:**
   1. **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]**

* 1. **m=1 and b[1] < a[1]**
  2. **m=1 and a[n] < b[1]**

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

1. For insert into a sorted array, all values in a and b will have a comparison except the last half of array a. Therefore the number of comparisons is m + n/2.
2. For this example, there will need to be only one comparison. The index in b is inserted first, then there are no values left in b to compare to a. So the number of comparisons is 1.
3. For this example, you will need to compare every value in a to the value in b. So the number of comparisons is n.

**For questions 5 – 6, 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 *target*:**

1. **a) If no record with the key *target* is present.**

**b) If one record with the key *target* is present and only one is sought.**

1. The search will take O(n).
2. The search will take a maximum time of O(n), a minimum time of O(1) and an average time of n/2 (technically O(n)).
3. **a) If more than one record with the key *target* is present and it is desired to find only the first one.**

**b) If more than one record with the key *target* is present and it is desired to find them all.**

1. The search will take a maximum time of O(n), a minimum time of O(1), and an average time of n/1+b, where b is the number of times the target appears in the list. As b -> n, the search time approaches O(1).
2. The search will take a maximum time of O(n), a minimum time of O(1), and an average time of n\*(b/(b+1)), where b is the number of times the target appears in the list. As b -> n, the search time approaches O(n).