Comprehensive Examination Spring 2007 (Analysis of Algorithms)

1a. Explain why a $O(3^N)$ algorithm is worse than a $O(N^4)$ algorithm when you do not have any idea about the expected input problem instance-size N. [10]
1b. What is the maximum value (N) of the input size when you may choose the exponential algorithm over the other one? [10]

2. Set up a recurrence equation and solve it for the time-complexity of following algorithm fragment.

```
Algorithm Unknown (Input array A[], integer start, integer end, integer Z)
        Local integer X initialized to 1;
        If start = = end do
                X = X+1;
                Z = Z + X;
                Return Z;
        Else
                Return Z + unknown (A[], start+1, end, Z+1);
End algorithm.
```

Driver: Unknown (A[], 1, N, 1)
where N is the size of the array and the array index starts with 1. [20]

3. The following is a recurrence formula (for aligning sequences with gaps, you need not be concerned about the problem that the formula models). Write a Dynamic Programming algorithm for computing a[i,j] for the given i and j, where i and j are integers between 0 and a constant N>0.

a[i, 0]= -i, a[0, j]= -j,

a[i, j] = max{a[i-1,j]-1,   a[i, j-1]-1,   a[i-1, j-1] + p(i,j)} for both i and j >0, and for a given integer matrix p(i,j).

Analyze the time-complexity of the algorithm. [20]

4. Input to the following algorithm is a sorted array of integers (both the non-increasing and non-decreasing arrays). What is its output for each of the two cases of non-increasing and non-decreasing sorted list? Analyze the asymptotic time complexity. [20]

```
Algorithm Resorter( int [ ] a)
{
        int I=1, j=a.length; // the array is from 1 through a.length
        while (I<j)  {
                if (a[I] > a[j])
                        { int temp=a[I]; a[I]=a[j]; a[j]=temp;};
                I++;
                j--;
        };
}
```

5. Answer *true/false* for the following sentences (or explain if there is no such answer): [20]

a. All NP-hard problems are NP-complete problems.

b. The set of NP-complete problems is a subset of the NP-class of problems.

c. NP-complete problems cannot have polynomial algorithms.

d. In order to prove a problem X to be NP-hard one needs to develop a polynomial transformation from X to a known NP-hard problem.

e. 2-SAT is an NP-hard problem.

f. There exists a polynomial-time algorithm for finding the maximum spanning tree in a undirected and weighted graph.

g. *QuickSort* algorithm takes $O(n \log n)$ time for running on an already sorted array of size n with a pivot choosing policy from one end of the array.

h. This sentence (in question 5h) is false.

i. Suppose that G is a connected and undirected graph. If removing edge e from G disconnects the graph, then e is a tree edge in the depth-first search spanning-tree of G.

j. Suppose that e is a minimum weight edge of a weighted undirected graph G, and all the edge weights are distinct. Then e is always contained in the minimum spanning tree of G.