Algorithms   Fall 2004   Graduate Comprehensive Exam

1. Set up the recurrence equation for asymptotic time complexity of the following algorithm and solve it for the usual theta function.

```
Algorithm Little (int array A[], int start, int end)
begin
if end = = start do
        return              // null
else    Little (A, start+1, end);

end algorithm.
```

2. The following is an example of a 0-1 Knapsack problem where the profit has to be maximized by picking up the unbreakable objects in a knapsack with limited capacity: Objects {(2 lbs, $10), (4 lbs, $2), (2 lbs, $5), (3 lbs, $6)}, Knapsack limit 10 lbs. A Dynamic Programming algorithm utilizes the following formula for computing the optimal profit:

$P(I, m) = P(I\text{-}1, m)$ when $w_I > m$, and

$P(I, m) = \max\{P(I\text{-}1, m), P(I\text{-}1, m - w_I) + p_I\}$ when $w_I \leq m$

$P(I, m)$ is the optimal profit for the first I objects with variable knapsack limit $m \leq 10$ lbs, w_I and p_I are the respective weight and profit of the I-th object.

For all m and I values, $P(I, 0) = P(0, m) = 0$.

Briefly describe the dynamic programming algorithm.

3. The following algorithm takes *any* sorted array of integers (both the non-decreasing and non-increasing arrays) as its input. What is its output in each case of non-decreasing and non-increasing sorted list? What is the algorithm's asymptotic time complexity?

```
Algorithm Unknown( 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--;
        };
}
```

4. There are three columns in a variable-length page and the following articles are to be placed in the columns such that the page length is minimal. Articles have no pre-assigned ordering for placement on the page, and they may not be split across the columns. The list of the lengths of the articles in inches is {3, 5, 1, 7, 10, 6, 2, 3}. Mention which greedy algorithm would you run for the problem? Step through that algorithm over the above list. [Hint: some scheduling algorithm.]

5. Answer *true/false* for the following sentences (answer on the question paper):
All NP-hard problems are NP-complete problems.
The set of NP-complete problems is a subset of the NP-class of problems.
It has been proved that NP-complete problems cannot have polynomial algorithms.
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.
2-SAT is an NP-hard problem.