# CS570
Analysis of Algorithms
Fall 2007
Exam I

Name: ______________________
Student ID: ___________________

| | Maximum | Received |
|---|---|---|
| Problem 1 | 20 | |
| Problem 2 | 20 | |
| Problem 3 | 12 | |
| Problem 4 | 12 | |
| Problem 5 | 12 | |
| Problem 6 | 12 | |
| Problem 7 | 12 | |

Note: The exam is closed book closed notes.

1) 20 pts
Mark the following statements as **TRUE**, **FALSE**. No need to provide any justification.

**[ TRUE/FALSE ]**
A greedy algorithm is any algorithm that follows the heuristic of making the locally optimum choice at each stage with the hope of finding the global optimum.
**T**
**[ TRUE/FALSE]**
BFS can be used to find the shortest path between any two nodes in a weighted graph.
F
**[ TRUE/FALSE]**
DFS can be used to find the shortest path between any two nodes in a non-weighted graph.
F
**[ TRUE/FALSE]**
BFS can be used to test whether a graph is bipartite
T
**[ TRUE/FALSE ]**
If T is a spanning tree of G and e an edge in G which is not in T, then the graph T+e has a unique cycle.
T
**[ TRUE/FALSE ]**
Let T be a spanning tree of a graph G and e an edge of G which is not in T. For any edge f that is on a cycle in graph T+e, the graph T + e − f is a spanning tree.
T
**[ TRUE/FALSE ]**
Given a graph G(V,E) with distinct costs on edges and a set $S \subseteq V$, let (u, v) be an edge such that (u, v) is the minimum cost edge between any vertex in S and any vertex in V-S. Then, the minimum spanning tree of G must include the edge (u, v).
T
**[ TRUE/FALSE ]**
If f, g, and h are positive increasing functions with f in O(h) and g in Ω(h), then the function f+g must be in Θ(h).
F
**[ TRUE/FALSE ]**
If a divide and conquer algorithm divides the problem is half at every step, the log(n) factor related to the depth of the recursion tree will cause the algorithm to have a lower bound of Ω(n log(n))
F
**[ TRUE/FALSE ]**
Suppose that in an instance of the original Stable Marriage problem with n couples, there is a man M who is last on every woman's list and a woman W who is last on every man's list. If the Gale-Shapley algorithm is run on this instance, then M and W will be paired with each other.
T

2) 20 pts

a) Arrange the following in the order of big oh $4n^2$, $\log_2(n)$, 20n, 2, $\log_3(n)$, $n^n$, $3^n$, nlog(n), 2n, $2^{n+1}$, log(n!)

$2<\log_2(n)< \log_3(n)<2n<20n< \log(n!) <n\log(n)< 4n^2< 2^{n+1}<3^n< n^n$

b) Find the complexity of the following nested loop

```
sum = 0;
for (i=0; i<3; i++)
  for (j=0; j<n; j++)
    sum++;
```

O(n)

c) Find the complexity of the following code section

```
for (i=0; i<n; i++) {
  for (j=0; j<n; j++) {
    A[i] = random(n);
   } // assume random() is O(1)
  sort(A, n); // assume sort() is the fastest general sorting algorithm
}
```

$O(n^2)$

d) Find the complexity of the following function

```
int somefunc(int n) {
  if (n <= 1)
    return 1;
  else
    return somefunc(n-1) + somefunc(n-1);
}
```

$O(2^n)$

e) Find the runtime T(n) in the following recurrence equation:

$T(n) = 2T(n/4) + n^{0.51}$

$T(n)=O(n^{0.51})$

3) 12 pts
Suppose we are given an instance of the Shortest Path problem with source vertex s on a directed graph G. Assume that all edges costs are positive and distinct. Let P be a minimum cost path from s to t. Now suppose that we replace each edge cost $c_e$ by its square, $c_e^2$, thereby creating a new instance of the problem with the same graph but different costs.
Prove or disprove: P still a minimum-cost s - t path for this new instance.

The statement is FALSE
Consider the example:
Vertices: V={A, B, C, D}
Edges: (A->B)=100, (A->C)=51, (B->D)=1, (C->D)=51

Shortest path from A to D is A->B->D Path length=101
After squaring this path length become 100^2+1^2=10001
However, A->C->D has path length 51^2+51^2=5202<10001
Thus A->C->D become shortest path from A to D

4) 12 pts

Consider a long country road with houses scattered very sparsely along it. You want to place cell phone base stations at certain points along the road, so that every house is within four miles of one of the base stations. Give an efficient algorithm that achieves this goal, using as few base stations as possible.

Greedy approach

1) Start from the beginning of the road
2) Find the first uncovered house on the road
3) If there is no such a house, terminate this algorithm; otherwise, go to next line
4) Locate a base station at 4 miles away after you find this house along the road
5) Go to 2)

Proof:

Let the number of base stations used to cover the first n houses in our algorithm be G(n). For any other policy, denote the number of base stations used to cover the first n houses P(n). Optimality of our algorithm can be shown if G(n)<=P(n) for n=1, 2, 3, ….We prove this by induction

It is obvious that G(1)<=P(1)

Given G(n-1)<=P(n-1), let's consider G(n) and P(n)

If the n-th house is already covered by G(n-1) base stations, then G(n)=G(n-1)<=P(n-1)<=P(n). This completes the proof.

If the n-th house has not been covered by G(n-1) base stations, then G(n)=G(n-1)+1. If G(n-1)<P(n-1), then we have P(n)>=P(n-1)>=G(n-1)+1=G(n), which completes the proof.

Otherwise G(n-1)=P(n-1). In this case the n-th house must have note been covered by P(n-1) base stations in this other policy because we always make sure that our policy covers the longest distance from the beginning to the n-th base station. Thus P(n)=P(n-1)+1>=G(n). This completes the proof.

Running time O(n), where n is the number of houses

5) 12 pts
Given a sorted array *A* of distinct integers, describe an algorithm that finds *i* such that *A*[*i*] = *i*, if such an *i* exists. Your algorithm must have a complexity better than *O(n)*.

```
Function(A,n)
        {
        i=floor(n/2)
        if A[i]==i
                return TRUE
        if (n==1)&&(A[i]!=i)
                return FALSE
        if A[i]<i
                return Function(A[i+1:n], n-i)
        if A[i>i]
                return Function(A[1:i], i)
        }
```

Proof:
The algorithm is based on Divide and Conquer. Every time we break the array into two halves. If the middle element i satisfy A[i]<j, we can see that for all j<i, A[j]<j. This is because A is a sorted array of DISTINCT integers. To see this we note that A[j+1]-A[j]>=1 for all j. Thus in the next round of search we only need to focus on A[i+1:n]
Likewise, if A[i]>i we only need to search A[1:i] in the next round.
For complexity T(n)=T(n/2)+O(1)
Thus T(n)=O(log n)

6) 12 pts

Consider a max-heap implemented using pointers rather than an array, so that the root has pointers to two smaller heaps, all of whose elements are smaller than the root's element. Give an algorithm to find the smallest element in the heap, and argue that your algorithm always runs in O(n) time on a heap with n elements.

Min=value at the root
Run a BFS or DFG through the heap. Every time a new node is visited compare its value with Min, if it is smaller then Min=this value

Every node visited only once, thus O(n)

7) 12 pts
Find all possible stable matchings for the following table of preferences. The women are A,B,C,D and the men are a, b, c, d :

| | A | B | C |
|---|---|---|---|
| a | (1,3) | (2,2) | (3,1) |
| b | (3,1) | (1,3) | (2,2) |
| c | (2,2) | (3,1) | (1,3) |

The content of the box (a,A), which is (1,3), means that man a ranks woman A as his first choice and woman A ranks man a as her third choice.

First we can see that a, b, c all rank D as last choice, A, B, C all rank d as last choice. Thus d only matches with D, otherwise the man matching with D and the woman matching with d will prefer each other than d and D.
Now we only look at a, b, c and A, B, C we can see that all matches works among them. Thus stable matches include all matching among a, b, c and A, B, C and then (d, D).

Additional Space

Additional Space