CS141 Fall 2000

Midterm Exam

Lovingly prepared by:

Michalis Faloutsos

Name:

SSN (4 last digits):

READ THIS INSTRUCTIONS CAREFULLY.

You have to answer all problems. The points for each question are in brackets.

You are allowed to have one-page (two-sides) of notes with you.

A good tactic is to start from the questions you know best. You may want to browse through all them first. Don't spend too much time on one question.

In all questions, explain your answers unless otherwise specified.

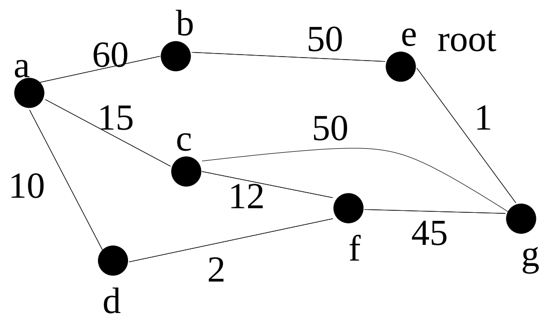
Your hand-writing should be readable and your writing understandable. If the graders can not decipher an answer, it's your loss.

Use other paper as scrap-paper, but try to reply in the space below each question and the back of the same page. If additional space is needed, indicate it clearly on the given hand-out.

Good Luck and...Don't Panic!

Q.1	
Q.2	
Q.3	
Q.4	
Q.5	

1. [**15**]



In the following weighted undirected graph, consider node "e" as the root and execute

(a) [**5**] Breadth First Search

(b) [**5**] Depth First Search

(c) [**5**] Prim's algorithm for the Minimum Spanning Tree

Show the execution of the algorithms by answering the two questions in parallel:

a) Show the order with which nodes are visited or selected and explain why are each node is selected in that order.

b) Show the gradual growth of the tree that each algorithm creates every time a new node is visited (added).

Attention: In case of multiple candidate nodes, choose nodes in alphabetical order. E.g. if I am in a and b, c, d are my unvisited adjacent nodes, I will pick b first.

2. [**5**]

What is the complexity of the Breadth First Search algorithm if we assume that we represent the graph with an adjacency matrix? The queue Q is a simple first-in-first-out linked list.

Discuss the complexity of each of the three phases (1,2,3) of the algorithm described in the pseudocode below.

Reminder: Adjacency matrix is an NxN matrix with zeroes and ones depending on whether there is an edge or not.

```
BFS pseudocode

1 Initialize all nodes to white

2 begin with the root, make it grey, add it in queue Q

3 while Q not empty
    remove head of Q: u
    for each, w, adjacent nodes of u
        if w is white
        then
          paint w grey and put it in Q
          w's father is set to u
        end
    end
    paint u black
  end while
```

3. [**10**] A bipartite graph is a graph $G(V, E)$ such that $V = B \cup C$ and $B \cap C = \emptyset$, and edges exist only between nodes of B and C i.e. $(u, w) \in E$ means that $u \in B$ and $w \in C$ or vice versa.

   A Hamiltonian Cycle is a simple cycle that passes from all the nodes of the graph **exactly once**.

   Prove that if a bipartite undirected graph has an odd number of nodes (in total), then it does not have a Hamiltonian Cycle.

4. [**10**] Consider the class newTree that maintains the information of an N-size tree in the form of an array Father:

```
class newTree {
  int *Father;
  int N;
  int root;
  int NONE = -1;

  newTree(int size, int r){
    root = r;
    N = size;
    Father = new int[N];
    for(i=0; i< N; i++)
        Father[i] = NONE;   // initialize
  }

}
```

Write the following methods for this class and express the complexity in terms of the number of nodes N and the maximum tree depth D.

You can use auxiliary fields and methods if you want.

a)[2] Write a method father(int u) that returns the father of a given node.

What is the complexity of this method?

b) [4] Write a method depth(int u) that returns the depth of node u in the tree (root has zero depth, its children have 1 etc).

What is the complexity of this method?

c) [4] Write a method degree(int u) that returns the degree of the node.

What is the complexity of your method?

Note: all the information for the tree is in the Father array. We **do not have** a typical graph representation available.

5. [**5**] Assume a recursive procedure that calls itself at least 1 and up to k times as shown below in pseudocode. Note that, in the children procedures, i is decreased by two.

```
void silly(int i)
   if i > 0  {
   choose x randomly between [1, k]    // uniformly distributed
        repeat x times
           do silly(i-2)
           end
  }
}
```

(a) [**1**] Best case: Show what is the **minimum** number of procedure calls that the call silly(n) can create in total ($n > 0$)- not including itself but including all its children, grand-' children, etc.

(b) [**2**] Worst case: Show what is the **maximum** number of procedure calls that the call silly(n) can create ($n > 0$) in total.

(c) [**2**] In this question, let us consider only the immediate children-procedures of one silly() procedure. Let us define the **cost** of each procedure call to be equal to 2^x where x the number of children of this procedure.

i) What is the min, max and average number of (immediate) procedure calls that a procedure makes?

ii) What is the min, max and average cost of a procedure?

Hint: a) A graphic representation of the function and its children may help you think about the problem.

b) If x is uniformly distributed in $[a, b]$ the average value of x is $(a + b)/2$.