# CS141 Fall 1999

## Final Exam

Monday 6 Dec, 1999

Lovingly prepared by:

Michalis Faloutsos

**Name:**

**SSN (4 last digits):**

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 read first all the questions and start from the questions you know best.

**In all questions, explain your answers** unless otherwise specified.

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.

Only students that are registered for CS141 can take this exam.

Good Luck and...Don't Panic!

| | |
|---|---|
| Q.1 | |
| Q.2 | |
| Q.3 | |
| Q.4 | |
| Q.5 | |
| Q.6 | |
| Q.7 | |
| Q.8 | |

1. [**1**] How does the instructor usually refer to your TA, Michael, in class? Hint: Probably a nice adjective. Make sure you write something.

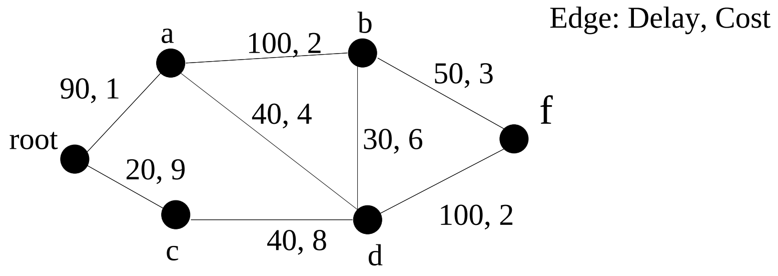
2. [**3**] Assume a recursive procedure that calls itself exactly $k$ times each time it runs as shown below in pseudo-code. Note that $i$ decreases by $k$ in the child-procedures.

```
void foo(int i)
   if i > 0  {
          repeat k times
           do foo(i-k)
           end
  }
}
```

   $k$ is a constant positive integer. Calculate the number of procedure calls $C(r)$ that the initial call foo($rk$) will create as a function of $r$ where $r$ is a positive integer.

   Hint: Induction may be a good idea. Drawing a picture of the procedure calls may help.

3. [**10**]

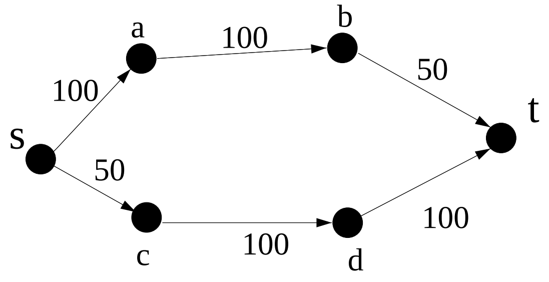


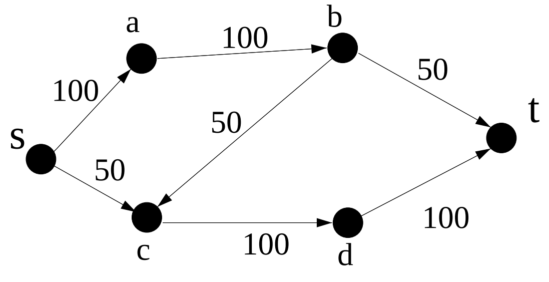
In the network given above, we are at the root and we want to establish a communication tree with **all** the other nodes. Each edge has a) a delay, and b) a cost value. The delay shows the delay in milliseconds for transferring data over the edge. The cost is the amount in sweet US dollars that we have to pay in order to include (lease) the edge in our communication tree.

(a) [**5**] Assume that we are only interested in minimizing the total cost (in $) of the tree that we want to create. Which algorithm should you use? Which is the tree with the minimum cost? Show the tree, its cost, and the order with which the nodes join the tree according to your algorithm.

(b) [**5**] Assume that we are only interested in minimizing the end-to-end delay (the delay from when you send the data until a friend receives it) between the root and each possible destination. Which algorithm should you use? Which is the tree that minimizes the end-to-end delays? Show the tree and the order with which the nodes join the tree according to your algorithm, and the delay that the delivery of data towards each destination (the delay of the path on the tree).

4. [**9**] Maximum-Flow.





(a) [**4**] In the two graphs above calculate the maximum-flow using the Ford-Fulkerson method for the given capacities. Show the augmenting path in each step, the amount of flow pushed, and the maximum-flow at the end.

(b) [**2**] Show what is the minimum cut in each graph, i.e. the sets $S, T$ such that $V = S \cup T$, $S \cap T = \emptyset$, and $s \in S$, $t \in T$. You may show it graphically if you want, but label your sets clearly. Comment on your answer.

(c) [**1**] Is there a relationship between a flow in the network and the capacity of the minimum-cut? Reminder: Capacity of min-cut $= \sum_{v \in S, w \in T} c(v, w)$.

(d) [**1**] Is there a relationship between the **maximum** flow and the capacity of the minimum-cut?

(e) [**1**] Explain the relationship between the min-cut and the max-flow intuitively (with 2-3 lines).

5. [**10**] Consider the following algorithm, Alg, for calculating the maximum weight spanning tree of an undirected weighted graph $G(V, E)$. For simplicity assume that each edge has a distinct weight, i.e. no weights are equal.

- Initialize tree $T$ to null.
- Sort the edges in order of decreasing weight, List.
- For each edge, $e$, in the order they appear in the List
  - if $e$ does not create a cycle with T, add $e$ in $T$

(a) [**3**] What is the time complexity of the algorithm? Assume that sorting $r$ elements takes $O(r \log r)$, and we check for cycles with breadth first search that for a graph with $r$ edges takes $O(r)$.

What is the complexity of a fast implementation where we select carefully where to check for cycles?

(b) [**7**] Prove that the algorithm finds correctly the maximum weight spanning tree.

Hint: Proof by contradiction may be a good idea. In case it helps, you may assume that trees you are comparing will differ by only one pair of edges (for *partial marks*).

6. [**6**] The Steiner tree problem. We want to find the minimum cost tree that spans a subset of nodes of an undirected weighted graph.

   (a) [**3**] Show an example where the Greedy algorithm will not find the optimal tree.

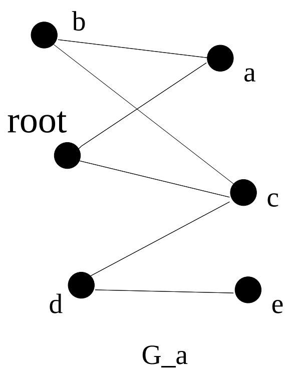
   (b) [**1**] Is there a case that the Naive algorithm finds the exact (optimal) solution? Give an example and explain.

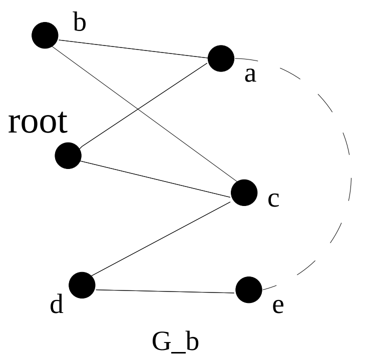
   (c) The competitive ratio of the Naive algorithm is tightly bounded by $M$, the number of participants in the tree. Explain whether the statements are "True" or "False" for **any** instance of the problem:

      i. [**1**] Naive will always find a tree $M$ times worse than the optimal tree.

      ii. [**1**] Naive will never find a tree that is more than $M$ worse than the optimal tree.

7. [**10**] Applications of Breadth First Search.





(a) [**4**] Execute Breadth First Search in the graphs above i) without the dotted edge, ii) with the dotted edge. Show the order with which the nodes join the tree in each case and the final tree only. Among children prefer nodes in alphabetical order.

Show clearly each generation of children where generation of a node is its distance from the root.

(b) [**2**] Which of the two graphs is bipartite? Show the edge that in your execution of the BFS shows that the graph is not bipartite. This is edge is most probably not the dotted one.

Reminder: 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.

(c) [**4**] Give an algorithm to detect whether a graph is bi-partite based on Breadth First Search. Explain why your algorithm will work, but a rigorous proof is NOT required.

8. [**4**] NP-completeness. Assume that the classes of problems NP-Complete (NPC) and Polynomial (P) do not intersect. An "efficient" solution takes polynomial time relative to the size of the input of the problem i.e the number of edges of a graph.

   Indicate whether the phrases below are true ("True") or false ("False"). Explain with one line your answer.

   (a) [**1**] An NPC problem can not be solved efficiently by any known algorithm.

   (b) [**1**] Even if I can solve one NPC problem efficiently, I may NOT be able to solve the other NPC problems efficiently.

   (c) [**1**] If I can polynomially reduce an NPC problem to a new problem X, then it is highly unlikely that an efficient algorithm for problem X exists.

   (d) [**1**] The meaning of NP-completeness is that I can not solve at all an NPC problem in the general case.