

Problem 1: 15 Points

For how many steps will these algorithms run if started with an input n (a power of two)? Write recurrences and solve them.

```
Algorithm Mystery1( $n$ )
if  $n > 1$  do  $x = \text{Mystery}(n/2) + \text{Mystery}(n/2)$ 
for  $i=1$  to  $n$  do  $x = x + 1$ 
```

```
Algorithm Mystery2( $n$ )
if  $n > 1$  do  $x = \text{Mystery}(n/2) + \text{Mystery}(n/2)$ 
for  $i=1$  to  $n$  do
  for  $j=1$  to  $n$  do
     $x = x + 1$ 
```

Problem 2: 15 Points

A *bipartite graph* is an undirected graph whose vertices can be partitioned into two disjoint sets B (for Black) and R (for Red) such that there are no edges with both their endpoints in B and no edges with both their endpoints in R ; in other words, all edges have one endpoint in B and the other endpoint in R . We usually visualize bipartite graphs by putting all Black (B) vertices on the left and all Red (R) vertices on the right, and then edges always extend from left to right.

- (a) Argue that every tree is a bipartite graph by explaining how you can color its vertices Black and Red so that all edges extend between Black and Red vertices.
- (b) Give an algorithm that, on input an undirected graph $G(V, E)$ in its adjacency list representation, tests if $G(V, E)$ is bipartite. Your algorithm should have running time $O(|V| + |E|)$. You should argue about correctness and running time. Hint: Use DFS.

Problem 3: 15 Points

Suppose that you want to find the shortest path from the College of Computing to your house. In the graph model $G(V, E)$ of Atlanta, the node $s \in V$ represents the College of Computing, and the node $t \in V$ represents the house. The edges represent street blocks and nodes represent intersections. Each edge $e \in E$ has a length l_e representing the amount of time that it takes to travel that road segment. We assume that all streets and roads are two-way and it takes the same amount of time to traverse the road or street in each direction. In addition, each *node* $v \in V$ has a weight w_v representing the amount of delay incurred by going through that intersection. For example, if there is a long traffic light at an intersection, or if there is a coffee shop at which you cannot resist stopping, then that node may have a high weight. Give an efficient algorithm that accepts as input $G(V, E)$, $s \in V$, $t \in V$, and functions l and w as above, and returns the shortest path from s to t in G . You may assume that the graph is given in either adjacency list or adjacency matrix representation, but you should say which representation you are using. You should argue about correctness and running time. Hint: Begin with a reduction that converts $G(V, E)$ into a different graph $G'(V', E')$ such that G' has edge lengths l' only (and not vertex weights) and such that there are nodes s' and t' such that the shortest path from s' to t' in G' corresponds in a well-defined way to the shortest path from s to t in G .

Problem 4: 15 Points

In the all-pairs shortest-path algorithm we were concerned with the *total length* of the path between any two nodes. Suppose instead that we are concerned with the length of the *longest edge* on a path

between any two nodes. That is, the *bottleneck path* is defined to be the length of the longest edge in the path. Design an efficient algorithm to solve the all-pairs bottleneck path problem. That is, find the length of a bottleneck path between every pair of vertices. Assume that the input graph is given in its adjacency matrix representation and argue about the running time of your algorithm.

Problem 5: 15 Points

Let G be an undirected graph given in its adjacency matrix representation. Let Hamilton-Cycle be the following language: $\text{Hamilton-Cycle} = \{G(V, E) \mid G \text{ contains a cycle of length } |V|\}$. Let Hamilton-Path be the following language: $\text{Hamilton-Path} = \{G(V, E) \mid G \text{ contains a simple path of length } |V| - 1\}$. Using the fact that Hamilton-Path is NP-complete, show that Hamilton-Cycle is also NP-complete.

Problem 6: 15 Points

Argue that the class P is closed under union (if $L_1 \in P$ and $L_2 \in P$ then $L_1 \cup L_2 \in P$), intersection (if $L_1 \in P$ and $L_2 \in P$ then $L_1 \cap L_2 \in P$), concatenation (if $L_1 \in P$ and $L_2 \in P$ then $\{w \mid w = xy, x \in L_1, y \in L_2\} \in P$) and complementation (if $L \in P$ then $\Sigma^* - L \in P$). Argue that the class NP is also closed under union, intersection and concatenation. What can you say about the complements of languages in NP (that is, if $L \in NP$, what can you say about $\Sigma^* - L$)?

Problem 7: 10 points

If a friend of yours who is a science or engineer major, but not a computer science major, asks you, over coffee, “What is theoretical computer science?” “Why is 3500 a required course for CS majors?” what would you respond?