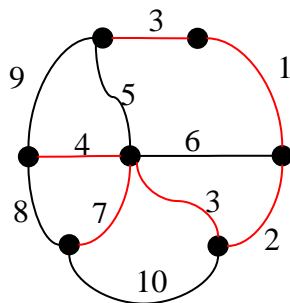


CS16, Spring 09, Practice Final Exam

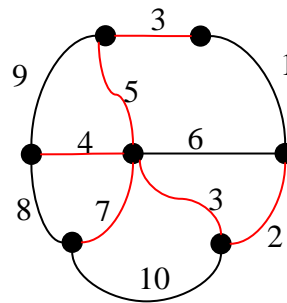
Note: The following problems are representative of the *kinds* of problems I might place on the exam, but you should not expect that the exam problems will be merely slight variations of these (although some of them may be).

Problem 1. We examined minimum weight spanning trees in class. Consider the problem of minimum *heft* spanning trees: the *heft* of a tree is the largest of its edge-weights. So while a min-weight spanning tree minimizes the SUM of the edge-weights, a min-heft spanning tree minimizes the MAX of the edge-weights.

There can be many minimum-heft spanning trees for a graph G. But as the following example shows, the minimum WEIGHT spanning tree is sometimes also a minimum HEFT spanning tree.



The min-weight spanning tree



One min-heft spanning tree

Either explain why the min-weight tree is always a min-heft tree, or give an example where it is not. You may assume that all edge weights are distinct, so there is a unique min-weight spanning tree.

Problem 2 Analysis. On the midterm you worked with n -bit binary numbers stored in a stack; in this problem, we'll store the numbers in an array of bits. And we'll make a rule that when we add 1 to the number $111\dots111$ (n ones in a row), we get $000\dots000$ (n zeros in a row), i.e., we "lose" the bit that's carried over to the $(n+1)$ -th position. We'll follow the convention that in an array x representing a number, $x[0]$ represents the "ones" bit, $x[1]$ is the "twos" bit, $x[2]$ is the "fours" bit, and so on. Here's code for incrementing a binary number

```
proc increment(Bit x[0..n])
  i = 0
  while (i < n)
    if x[i] is zero :
      x[i] = 1
      return x
    else
      x[i] = 0
      i = i+1 // "carry" the one to the next bit
  return x
```

- (a) What's the worst-case running time of "increment", in terms of n , the length of the binary bit-string? Explain briefly. You may assume that the only operation that "costs" anything is a bit-flip, i.e., an assignment of the form $x[i] = 0$ or $x[i] = 1$.
- (b) Suppose you start with some random bit string x of length n , and you increment it 2^n times, so that its final value is the same as its starting value. What's the average-case running time of "increment" during the course of this sequence of operations? (Again, do the analysis in terms of n). Hint: How many total bit-flips take place during the entire sequence of 2^n increments?

Problem 3. Running times. You have a choice of two algorithms to run on a graph $G = (V, E)$ with $n = |V|$ nodes and $k = |E|$ edges.

Algorithm P works like this:

```

for each node  $v$  in  $V$ 
  Let  $G' = (V', E')$  be the graph with  $v$ , and
    all edges incident on it, deleted
  Execute Prim-Jarnik on  $G'$  to find its minimum spanning tree
  Build a heap containing all edges of the minimum spanning
    tree, using their lengths as the heap-key
  Extract the edges from the heap in order, and do something
     $O(1)$  with each of them.

```

Algorithm Q works like this:

```

for each edge  $e$  in  $E$ 
  Let  $G' = (V', E')$  be the graph with  $e$  deleted
  Execute a breadth-first search on  $G'$  and record each node
    and its post number
  Find the node with the median post number, and do
    something  $O(1)$  with it.

```

Assume that in each algorithm, finding the graph G' can be done in $O(1)$ time.

- (a) As you know, the number k can be anywhere between 0 and about $n^2/2$. Suppose that k is about equal to n . Which algorithm do you prefer, and why?
- (b) Suppose k is about $n^2/2$. Which algorithm do you prefer, and why?

Problem 4. You are given a set of cities, along with the pattern of highways between them, in the form of an undirected graph $G = (V, E)$. Each stretch of highway e in E connects two of the cities, and you know its length in miles, $w_e > 0$. You want to get from city s to city t . There's one problem: your car can only hold enough gas to cover L miles. There are gas stations in each city, but not between cities. Therefore, you can only take a route if every one of its edges has length $w_e \leq L$.

(a) Given the limitation on your car's fuel tank capacity, show how to determine in linear time whether there is a feasible route from s to t .

(b) You are now planning to buy a new car, and you want to know the minimum fuel tank capacity that is needed to travel from s to t . Give an $O((|V| + |E|) \log |V|)$ algorithm to determine this.

Problem 5. In this problem, we examine *labeled binary trees*: each node v of a tree has a label, $v.label$, which is distinct from the labels of all other nodes in the tree. (You can think of the labels as letters of the alphabet, where we never re-use a letter within one tree). We'll say that tree S is a *subtree* of tree T if the following conditions hold:

- Every label on a node u of S appears as the label on some node u' in T .
- If the node v in S is the left child of the node u , then v' is a left descendant of u' in T (where a "left descendant" is either the left child, or the left child of the left child, or the left child of the left child of the left child, etc.); a similar rule applied when v is the right child of u : v' must be a right descendant of u' .

Note that this is trivially satisfied by the empty tree: it's a subtree of any tree (including the empty tree).

You'll develop an algorithm *subtree* to determine whether an given tree S is a subtree of another tree T .

- (a) Describe your algorithm conceptually in a few sentences. This is your best chance to make sure that the TA grading this problem understands what you're doing, so that even if your pseudocode has mistakes, you can get most of the credit for the problem.
- (b) Give pseudocode for your algorithm. Your code should return either TRUE (if S is a subtree of T) or FALSE (if it is not).

Note: it's possible to give quite a slow algorithm for this problem, or quite a fast one. For once, we don't care all that much. Of the 20 points for this problem, only 3 will be for efficiency. *Clarity*, however, is at a premium - a neat, well-explained, simple algorithm will get you almost full credit.