

# Homework 5

Due April 21

## Problem 5.1

We have three containers whose sizes are 10 pints, 7 pints, and 4 pints, respectively. The 7-pint and 4-pint containers start out full of water, but the 10-pint container is initially empty. We are allowed one type of operation: pouring the contents of one container into another, stopping only when the source container is empty or the destination container is full. We want to know if there is a sequence of pourings that leaves exactly 2 pints in the 7- or 4- pint container.

- (a) Model this as a graph problem: give a precise definition of the graph involved and state the specific question about this graph that needs to be answered.
- (b) What algorithm should be applied to solve the problem?

## Problem 5.2

The police department in the city of Computopia has made all streets one-way. The mayor contends that there is still a way to drive legally from any intersection in the city to any other intersection, but the opposition is not convinced. A computer program is needed to determine whether the mayor is right. However, the city elections are coming up soon, and there is just enough time to run a linear-time algorithm.

- (a) Formulate this problem graph-theoretically, and explain why it can be solved in linear time.
- (b) Suppose it now turns out that the mayor's original claim is false. The mayor next claims something weaker: if you start driving from city hall, navigating one-way streets, then no matter where you reach, there is always a way to drive legally back to the city hall (which is situated at an intersection). Formulate this weaker property as a graph-theoretic problem, and show how it too can be checked in linear time.
- (c) Suppose this statement also turns out to be false, so the mayor is indicted for lying to the public, and must go from city hall to the courthouse. The mayor wants to take the route with the shortest length (where each segment of road  $e$  has length  $l_e > 0$ ). However, if there are several shortest routes of the same length, the mayor wants to go through as few intersections as possible, since an angry demonstration of citizens might appear at each intersection. Give an efficient algorithm to find the shortest path to the courthouse that goes through as few intersections as possible.

**Problem 5.3**

Suppose a CS curriculum consists of  $n$  courses, all of them mandatory. The prerequisite graph  $G$  has a node for each course, and an edge from course  $v$  to course  $w$  if and only if  $v$  is a prerequisite for  $w$ . Find a linear-time algorithm that works directly with this graph representation, and computes the minimum number of semesters necessary to complete the curriculum. You may assume that a student can take any number of courses in one semester.

**Problem 5.4**

There is a network of roads  $G = (V, E)$  connecting a set of cities  $V$ . Each road in  $E$  has an associated length  $l_e$ . There is a proposal to add one new road to this network, and there is a list  $E'$  of pairs of cities between which the new road can be built. Each such potential road  $e' \in E'$  has an associated length. You may assume that  $E$  and  $E'$  are disjoint. As a designer for the public works department you are asked to determine the road  $e' \in E'$  whose addition to the existing network  $G$  would result in the maximum decrease in the driving distance between two fixed cities  $s$  and  $t$  in the network. If no such road decreases the driving distance, say so. Give an efficient algorithm for solving this problem.

**Problem 5.5**

You are given a graph  $G = (V, E)$  with positive edge weights, and a minimum spanning tree  $T = (V, E')$  with respect to these weights; you may assume  $G$  and  $T$  are given as adjacency lists. Now suppose the weight of a particular edge  $e \in E$  is modified from  $w_e$  to a new value  $\hat{w}_e$ . You wish to quickly update the minimum spanning tree  $T$  to reflect this change, without recomputing the entire tree from scratch. There are four cases. In each case give a linear-time algorithm for updating the tree.

- (a)  $e \notin E'$  and  $\hat{w}_e > w_e$ .
- (b)  $e \notin E'$  and  $\hat{w}_e < w_e$ .
- (c)  $e \in E'$  and  $\hat{w}_e < w_e$ .
- (d)  $e \in E'$  and  $\hat{w}_e > w_e$ .

**Problem 5.6**

Give an algorithm that takes as input a directed simple graph with positive edge lengths, and returns the length of the shortest cycle in the graph (or indicates that the graph is acyclic). Your algorithm should take time at most  $O(|V|^3)$ .

### Problem 5.7

*NOTE: This will be a rather difficult problem. Make sure you do not spend all your time on this problem, and leave no time for the other problems.*

On the planet WeeBoo, there's a new breed of lemmings. At a certain time each year, groups of lemmings line up and try to destroy themselves by marching off a cliff. We'll draw this situation as follows:

```
**_*_**|
ED C BA
```

The lemmings are the asterisks; blanks are spaces between lemmings, and the vertical bar is the cliff. If all the lemmings took one step to the right, we'd have this:

```
_**_*_*|
ED C B
```

because lemming A would have fallen off the cliff.

Once per minute, each lemming either moves one step to the right, or remains where it is. One way for the lemmings to kill themselves off is for everyone to march to the right at every step; since the leftmost lemming is 6 steps from the cliff, it'll take seven units of time before all the lemmings are gone. It'll look like this:

```
**_*_**|
ED C BA
```

```
_**_*_*|
ED C B
```

```
--**_*_|
ED C
```

```
---**_*|
ED C
```

```
----**_|
ED
```

```
-----**|
ED
```

```
-----*|
E
```

```
-----|
```

We can tally up the number of lemming-minutes of existence above by simply counting the number of letters:  $5 + 4 + 3 + 3 + 2 + 2 + 1 = 20$  lemming-minutes.

These lemmings, however, live on the planet WeeBoo, and are somewhat unusual: if two of them collide, they explode and both disappear. So if at time one, lemmings A and D remain still, the picture looks like this:

```
**_*_**|
ED C BA
```

```
----*__|
      C
```

```
-----*_|
          C
```

```
-----*|
          C
```

```
-----|
```

This sequence has only 9 lemming-minutes. It's possible to get even fewer: lemmings B and D remain still at the first minute, and lemming B remains still at the second minute as well:

```
**_*_**|
ED C BA
```

```
----**_|
      CB
```

```
-----|
```

which gives a total of just 7 lemming minutes.

Here's the problem: given an initial configuration of lemmings (read left to right, where a "1" means a lemming and a "0" means "no lemming"), where the configuration has length  $k$  and there are  $n$  lemmings (our example would be 1101011, with  $k = 7$  and  $n = 5$ ), write an algorithm to determine how the lemmings should behave to die off as fast as possible (i.e., to minimize the number of lemming-minutes before they're all gone).

You do not need to tell us your algorithm. Instead, apply it to the sequence

```
**_*___*__*_**_*___*___*_*|
LK..J...I.HG.F...E.DCB.A
```

and tell us the number of lemming minutes you come up with. (You could also tell us which lemmings remain still at each time; we'll assume all the others take a step).

Hint: it may be easiest to answer this by first developing an algorithm, and then coding it up in some language you like. It should run in a reasonable big-O

time (i.e., not exponential in either  $n$  or  $k$ ) if you want to get an answer in time to hand it in.