15-451 Algorithms, Fall 2010

Homework # 4 **due: Tuesday-Friday, October 26-29, 2010**

Ground rules:

- This is an oral presentation assignment. You should work in groups of **exactly** three. At some point before **Monday, October 25 at 11:59pm** your group should sign up for a 1-hour time slot on the signup sheet on the course web page. By signing up with fewer than three group members you are giving course staff permission to place unpaired students in your group.
- Each person in the group must be able to present every problem. The TA/Professor will select who presents which problem. The other group members may assist the presenter.
- You are not required to hand anything in at your presentation, but you may if you choose. If you do hand something in, it will be taken into consideration (in a non-negative way) in the grading.

Problems:

(28 points) 1. **2Implication** Consider the following problem, which we will call 2Implication. In 2Implication, you are given a set of *implications* of the form $a \Rightarrow b$ where a and b are literals, *i.e.* Boolean variables or negations of Boolean variables. You want to find a way to assign to each of the variables the value **true**, or the value **false**, such that *all* the implications evaluate to **true**. Note that (**false** $\Rightarrow b$) evaluates to **true** for all values of the literal b, and (**true** $\Rightarrow b$) evaluates to the value of b. For example, here is an instance of 2Implication:

$$(\neg x_1 \Rightarrow \neg x_2) \wedge (x_1 \Rightarrow \neg x_3) \wedge (\neg x_1 \Rightarrow x_2) \wedge (x_3 \Rightarrow x_4) \wedge (x_1 \Rightarrow x_4)$$

It is possible to make this instance evaluate to **true** by setting x_1 and x_4 to **true**, and x_2 and x_3 to **false**.

(a) Are there any other variable assignments that make this 2Implication formula true? If so, give them all. (This problem is a warmup, your group will not need to present your solutions to this problem. However, we strongly doing the warmup problems.)

(b) Give an instance of 2Implication, with four variables, such that there is no variable assignment that makes the formula true. (This problem is a warmup, your group will not need to present your solutions to this problem. However, we strongly doing the warmup problems.)

Given an instance I of 2Implication, with n variables and m implications, we can construct a directed graph $G_I = (V, E)$ as follows.

- G_I has $2n$ nodes, one for each variable and its negation.

- G_I has $2m$ edges: for each implication $(a \Rightarrow b)$ of I (where a, b are literals), G_I has an edge from a to b, and an edge from $\neg b$ to $\neg a$.

(c) Use this construction to represent the instance of 2Implication given above. (As an additional warmup use this construction to represent the instance you designed for part (b). You are not required to present this extra step, but we think you would be very helpful.)

(d) Show that if G_I has a strongly connected component containing both x and $\neg x$ for some variable x, then there is no variable assignment that makes I true.

(e) Now show the converse of part (d). In other words, show that if none of G_I's strongly connected components contain both a variable and its negation, then there is a variable assignment, such that the instance I evaluates to true.

(f) Use your answers to the previous parts to give a linear time algorithm for solving 2Implication i.e., an algorithm that gives the variable assignment if it exists, or concludes that there is no variable assignment that can make the instance true.

(28 points) 2. **CAN HAS CANDY?** You are a strategic student and would like to give a candy to one of the 5 TAs as a favor. We, the generous TAs, have heard your voice and decided to help you out on your mission. We are kind enough to give you access to a candy generation system. Unfortunately, it is not just a push-button system and it has its own unique limitations.

Here is the detailed description of the candy generation system. The entire generation process is divided into n tasks. There are m machines at your disposal, each of which does the exact same job. For each task T, you are also given the number of jobs u_T it needs. Now, T can use multiple machines to get all the u_T jobs done. Also, it can use any given machine multiple times. But a single job can not be divided among multiple machines! If that were everything, life would have been simple! Unfortunately, there are two caveats. After a task T uses a machine M for some (given) $b_{T,M}$ times, M will reject any more requests from T. Also, machine M wears out after using it for c_M times and will be useless for any task thereafter. A candy can be generated only if all the n tasks can be completed.

To sum, you have n tasks to perform, m machines to use and the restrictions $b_{T,M}$ and c_M as described above. Assume that all the quantities, viz. n, m, u_T, $b_{T,M}$, c_M for all possible T and M, are non-negative integers.

Now, you want to decide if this beautiful system is of any use to you, i.e. will the system be able to generate a candy for you? In particular, give an algorithm to decide if a candy can be generated, from the given information, by reducing it to an appropriate *network flow* problem.

Last but not the least, here is an algorithm which you might be using to give the candy (if you are fortunate enough to generate one using the above system) to one of us based on the rumors we have heard. We think you love LOLCODE...lol!

```
HAI
CAN HAS CANDY?
```

```
I HAS A TA
IM IN YR LOOP
   UP TA!!1
   IS TA BIGGER THAN 5? KTHX
   HAS TA GIVEN ME FULL CREDIT?
    AWSUM THX
      GIVE TA 451-MADE-CANDY
      KTHX
    O NOES
      TRY NEXT TA
IM OUTTA YR LOOP
KTHXBYE
```

(28 points) 3. **Construction Chaos** The mayor of Smthirogla has decided that the intersections for the roads in his city are too run down. Thus, he wants to perform construction on them. However, the mayor is on a tight budget. Thus, he only wants to choose a select few intersections to perform construction on. Moreover, the construction on an intersection blocks all the roads intersecting at that intersection. Thus if the construction results in a complete separation of one part of the city from the other, the citizens are just screwed! Furthermore, the city construction workers are lazy and they will not bother to create an alternate path.

Therefore, the goal of the mayor is to choose intersections on which to perform construction that will allow citizens to still reach every point of the city. Assume that at the start of this scenario, any citizen can go from one point of the city to another by traversing the city roads (no intersections have been blocked yet).

We can view this problem as the intersections being vertices, and the roads being edges. Using this model, answer the following problems.

(a) Is it always possible to perform construction on at least one intersection without splitting the city into unreachable parts? Prove or give a counterexample.

(b) Develop an algorithm that runs in $O(V + E)$ which finds (if any) an intersection that the mayor cannot perform construction on. In essence, if the mayor performs construction on that intersection, parts of the city will be unreachable to some citizens. Find such an intersection, or determine that such an intersection does not exist in $O(V + E)$.

(c) The mayor has chosen an intersection to perform construction on! He has made sure that the blockage will not split the city in two. However, previous to this problem, he had a map that calculated the shortest paths from his office, to every intersection in the city. Note: the mayor actually has the *shortest paths* for each pair of nodes s-t and not just the shortest path lengths, marked on his map. Now that the intersection is gone, the map has changed! Can the mayor update his map efficiently to have the new shortest paths, without re-running a single-source shortest path algorithm from his office?

Basically, the mayors map is a shortest path tree from his office. If one node is deleted, does he need to recreate the entire map from scratch to update the shortest path tree?

(16 pts) 4. **The Curveball:** After your team present your solution to each of the above problems we reserve the right to make small changes to the problem, and ask how the answers would change. You will be expected to answer these questions on the fly. We are not going to tell you what these variations are in advance, that would defeat the purpose. Your best strategy is to make sure that everyone on your team *understands* each problem and its solution. If you have a good *understanding* of the solution then these variations should not be too difficult to answer.