15-451 Algorithms, Fall 2012

Homework # 4 **Due: Wed-Fri, October 24-26, 2012**

Ground rules:

- This is an oral presentation assignment. You should work in groups of three. At some point before **Monday, October 22 at 11:59pm** your group should sign up for a 1-hour time slot on the signup sheet on the course web page.
- 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.

Problems:

1. **Fair carpooling.** The n employees of the Algo Rhythms music downloading service sometimes carpool to work together.[1] Say there are m days, and S_i is the set of people that carpool together on day i. For each set, one of the people in the set must be chosen to be the driver that day. Since people would rather not drive, they want the work of driving to be divided as fairly as possible. Your task in this problem is to give an algorithm to do this efficiently.

   The fairness criterion is the following: Say that person p is in some k of the sets, which have sizes $n_1, n_2, \ldots, n_k$, respectively. Person p should really have to drive $\frac{1}{n_1}+\frac{1}{n_2}+\cdots+\frac{1}{n_k}$ times, because this is the amount of resource that this person effectively uses. Of course this number may not be an integer, so let's round it up to an integer. The fairness criterion is simply that she should drive no more than this many times.

   For example, say that on day 1, Alice and Bob carpool together, and on day 2, Alice, Carl, and Dilbert carpool together. Alice's fair cost would be $\lceil 1/2 + 1/3 \rceil = 1$. So Alice driving both days would not be fair. Any solution except that one is fair.

   (a) Prove that there always exists a fair solution.

   (b) Give a polynomial-time algorithm for computing a fair solution from the sets S_i.

   Hint: Try to model the problem using network flow in such a way that part (a) falls out directly from the integrality theorem for network flow, and part (b) just follows from the fact that we can solve max flow in polynomial time. So, it all boils down to coming up with the right flow graph to model the problem.

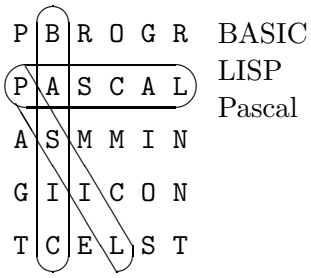
[1] Any relation to actual companies or bands of this name is purely coincidental.

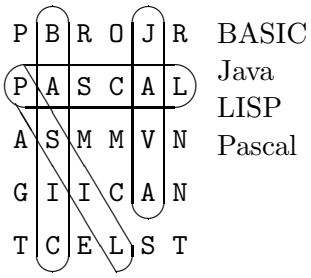
2. **Graduation.** Cranberry-Melon University has n courses.[2] In order to graduate, a student must satisfy several requirements. Each requirement is of the form "you must take at least k_i courses from subset S_i". The problem is to determine whether or not a given student can graduate. The hard part is that any given course cannot be used towards satisfying multiple requirements. For example if one requirement states that you must take at least two courses from $\{A, B, C\}$, and a second requirement states that you must take at least two courses from $\{C, D, E\}$, then a student who had taken just $\{B, C, D\}$ would not yet be able to graduate.

   Your job is to give a polynomial-time algorithm for the following problem. Given a list of requirements $r_1, r_2, \ldots, r_m$ (where each requirement r_i is of the form: "you must take at least k_i courses from set S_i"), and given a list L of courses taken by some student, determine if that student can graduate.

3. **Biconnectivity.** A solved word search problem consists of an n-by-n matrix of letters, along with m lines of letters that form words (call these lines for short). Two such lines only intersect at one letter. The sum of the lengths of all the lines is L.

   A set of lines is called *biconnected* if when any one of the lines is removed from the set, the remaining set of lines is still *connected*. By "connected" we mean that it's possible to get from any letter of any line to any other letter of any other line by walking along the lines. Removing a line simply means that we are not allowed to walk along that line of letters (the letters are still in the matrix). For example the word set on the left below is biconnected, but the one on the right is not.





   Give an algorithm that runs in time $O(L)$ time to determine if a solved word search problem is biconnected. The representation of the given solved word search problem is simply a list of lines, where each line is represented by a list of locations in the matrix. For example the line "PASCAL" in the problem above is represented as $((2,1),(2,2),(2,3),(2,4),(2,5),(2,6))$. (Of course you should try to make use of algorithms we have studied in this class.)

[2] Any relation to actual universities of similar name is purely coincidental. OK, that's a lie - it's not.