Algorithms
Computer Science 140 & Mathematics 168
Spring 2012
Homework 9b
Due Tuesday, March 27

The second exam in this course will be a 3 hour and 15 minute takehome exam (15 minutes to calmly read the exam and 3 hours to take it). It will be available at the CS office on Tuesday, April 17 and will be due at the beginning of class on Thursday, April 19. There will be no class on Tuesday the 17th. More details on the content and structure of the exam coming soon!

On this assignment, as usual, you may appeal to any results that we've proved in class without reproving them. If you wish to use any such result, just state the result that you're using.

1. **[20 Points] Headbook and Clinked In!** Social networking sites, like Headbook and Clinked In, distribute their databases over multiple servers. Each person with an account at Clinked In can be viewed as a vertex and relationships are represented by edges. Clinked In would like to divide the vertices between two servers in such a way that the number of friendships between vertices on different servers is minimized. (The reason for this is that each time we have to follow a link to a friend on the other server, that involves a slow query across the network.)

   Given an *undirected* graph G, a *partition* of the graph is a division of the vertices into two sets A and B such that every vertex is in exactly one of A and B. The only constraint on A and B is that neither set can be empty. The *crossing number* of the partition is the number of edges with one endpoint in A and one endpoint in B. Your job is to find an algorithm that finds a partition with the smallest possible crossing number.[1]

   (a) Describe your algorithm for finding a partition with minimum crossing number. (You may use existing algorithms to help!)

   (b) Prove that your algorithm is correct. (Be careful here. If your algorithm is designed carefully, your completely rigorous proof can be quite short. If not, it might be long and complicated. Short is better!)

   (c) Derive the running time of your algorithm. It must be polynomial in the number of vertices and edges in the graph.

[1] In practice, we generally want to add the constraints that the two sets have similar size, but we'll ignore that here. In addition, we'd generally like to partition over more than two sets because Clinked In has thousands of servers rather than just two. Nonetheless, this problem provides us with a good first start. This is a real problem that Mudd alums working at social networking companies are currently researching!

2. **[25 Points] Napquest!** Napquest provides its users with the ability to find a shortest path from a given starting point to a destination point. Because the graphs (maps) in general contain cycles but all of the edge weights are positive, they currently use the standard implementation of Dijkstra's Algorithm to find shortest paths. Recall that our implementation of Dijkstra's Algorithm used a heap and had running time of $O(m \log n)$.

   Napquest realizes that they need to provide the fastest possible response time to their clients and they would therefore like to improve the efficiency of their implementation of Dijkstra's Algorithm. They notice that all of the edge weights in their maps are integers in the range from 1 to W, where W is some integer constant. They believe that this extra constraint on the edge weights may be exploited to achieve an even faster implementation of Dijkstra's Algorithm. This is where you come in!

   (a) Under the assumption that all edge weights are positive integers in the range from 1 to W, describe an implementation of Dijkstra's Algorithm to compute the shortest path lengths from a source vertex to all other vertices in the graph in time $O(Wn + m)$.

   (b) Now, we'll do even better! Again under the assumption that all edge weights are positive integers in the range from 1 to W, describe an implementation of Dijkstra's Algorithm to compute the shortest path lengths from a source vertex to all other vertices in the graph in time $O((m+n) \log W)$. (*Note:* You may find it useful to observe - and prove - that the possible set of distinct values for the labels of the unlocked vertices is relatively small. Then, you can use an appropriate data structure to get your desired running time.)

3. **[25 Points] Space Shuttle Profit Optimization!**

   After successful stints at Millisoft, Hurts Car Rental, the brokerage firm of Weil, Proffet, and Howe, Clinked In, and Napquest, you've been hired by NASA to help optimize their next generation space shuttle program.

   On a given mission, NASA will consider a set of experiments that industrial sponsors would like to conduct (for example, "Does zero-gravity compromise the integrity of Spam?"). Let $E = \{E_1, E_2, \ldots, E_m\}$ denote the set of experiments under consideration. Let p_j denote the amount of money the sponsor will pay to conduct experiment E_j. The experiments use a set $I = \{I_1, I_2, \ldots, I_n\}$ of instruments to conduct the experiments. For each experiment E_j, let R_j be the subset of I that contains all of the instruments needed to conduct experiment E_j. Notice that this allows a single instrument to be used in multiple experiments. The instruments are potentially large and heavy and the cost of taking instrument I_k is c_k dollars. Your job is to determine which experiments should be performed in order to maximize the net revenue, which is the total income from the performed experiments minus the total cost of all instruments carried. Amazingly, this problem can be solved using **network flow**!

We'll construct a network flow problem as follows: The network contains a source vertex, s, vertices $I_1, I_2, \ldots, I_n$, vertices $E_1, E_2, \ldots, E_m$, and a sink vertex t. For each instrument I_k there is a directed edge from s to vertex I_k with capacity c_k (the cost of taking this instrument). For each experiment E_j there is an edge from vertex E_j to t with capacity p_j (the payment for this experiment). Finally, if instrument I_k is in set R_j (meaning that instrument I_k is needed for experiment E_j) then there is a directed edge from vertex I_k to vertex E_j with **infinite** capacity.

(a) First, try this. Consider a situation in which there are three experiments E_1, E_2, E_3 which will bring in 10, 6, and 6 dollars, respectively. Also there are four instruments I_1, I_2, I_3, and I_4 which cost 3, 2, 5, and 7 dollars, respectively to take on the shuttle. Experiment E_1 requires instruments I_1 and I_2, experiment E_2 requires instruments I_1 and I_3, and experiment E_3 requires instruments I_3 and I_4. Using brute-force, enumerate all seven possible combinations of experiments that could be taken and determine which combination is the most profitable. What is this combination and what is the net revenue?

(b) For the example problem above, construct the corresponding network flow problem. Find the maximum flow in the network (show your residual graphs at each step) and then find the corresponding cut whose capacity is equal to this flow.

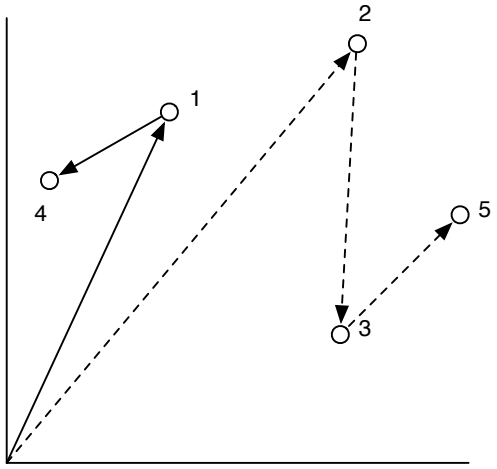
(c) Let's let S denote the vertices that are on the same side of the above cut as vertex s and let T denote the vertices that are on the same side of the cut as t. What do you notice about the instruments and experiments that are in the set T?

(d) Now, let τ be the sum of the payments that NASA would receive for taking all of the possible experiments. In this case, $\tau = 10 + 6 + 6 = 22$. From τ, subtract the capacity of the cut you found above. Surprise! What is this number and how does it appear to relate to this problem?

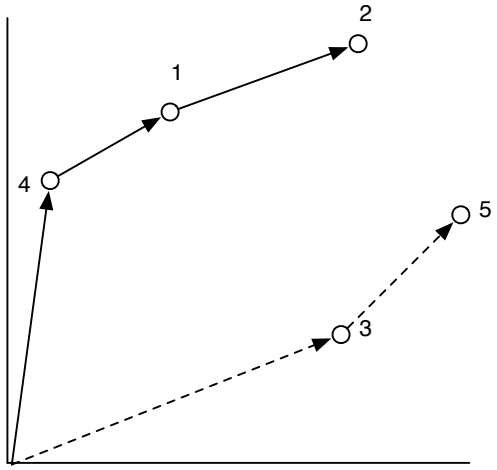
(e) Finally, we're ready to generalize all of this into an efficient algorithm for solving the profit maximization problem in general. Assume that we've set up a network flow problem corresponding to a given set of experiments and instruments. Show that for any cut with finite total capacity, if an experiment E_j is in T (the side of the cut containing vertex t), then all of the instruments used in this experiment must **also** be in T.

(f) Now argue that the maximum net revenue that can be achieved is simply the total sum τ of the payments that would be received for taking all of the experiments minus the capacity of the minimum cut.

(g) Now just summarize all of this by describing the algorithm, step-by-step, for finding the maximum net revenue, given a set of experiments, instruments, and the corresponding payments and costs. Give a careful derivation of the worst-case running time of the algorithm assuming there are m experiments and n instruments.



A legal (but not necessarily optimal) solution



An illegal solution because one rider visits 4 and then 1, which is out of order!

4. **Wheel Deliver! (Optional 20 Point Bonus Problem)** At this point in the course, you have the fundamental algorithm skills to solve new problems. Thus, bonus problems from here on out will simulate the kind of challenges that you might be presented in an interview or on the job - namely, I'll give you a problem and you'll need to decide what algorithmic tools to use to solve it. The bonus problems may or many not use the tools that we've seen that week. Here's the bonus problem for this week!

   Two East Dorm students, we'll call them Anne and Stan, have started a new unicycle-based meal delivery business called "Wheel Deliver" that works like this: Residents of Claremont call between 5 and 5:30 PM and order food for delivery. Anne and Stan go to Hoch-Shanahan Dining Hall, pick up the food, and deliver the food on their unicycles.

   Let n denote the number of customers that call on a given evening and let $p_1, \ldots, p_n$ denote the locations of these customers, sorted in the order in which the customers called: p_1 is the location of the first caller to place an order and p_n is the location of the last caller to place an order. Let p_0 denote the location of Hoch-Shanahan, where Anne and Stan begin their delivery. Let $d(i, j)$ denote the distance between p_i and p_j.

   Anne and Stan split up the delivery locations (not necessarily evenly) so that each order is delivered by exactly one of them. In addition, "Wheel Deliver" has a fairness rule that goes like this: If customers i and j where $i < j$ are assigned *to the same unicycle rider* then the meal must be delivered to customer i (at point p_i) before customer j (at point p_j) even if that makes the trip longer. Note that if i and j, $i < j$, are assigned to different riders then we don't care which of those two get their food first. **Subject to these constraints, the objective is to minimize the total distance travelled by the two unicycle riders.**

Here's your task:

(a) Describe your algorithm in clear English or pseudo-code.

(b) Derive the running time of your algorithm (it must be polynomial in the number of points).

(c) Prove the correctness of your algorithm. That is, prove that it minimizes the total distance travelled by the two unicycle riders.