CMPS 102: Homework #7

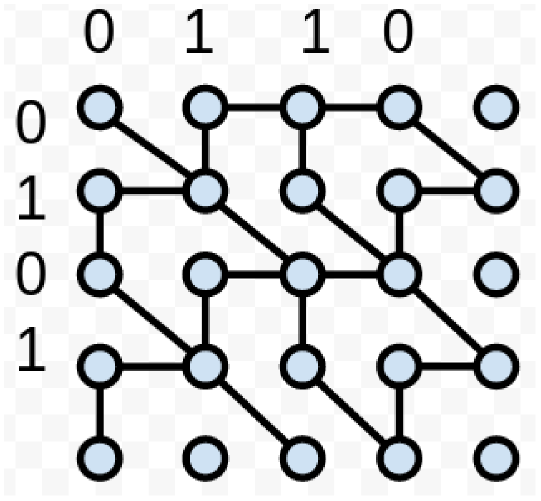
Due on Tuesday, May 26th, 2015

John Allard 1437547

Problem 1

1) Consider the longest common subsequence problem between two strings $x_1, ..., x_n$ and $y_1, ..., y_m$. Define a graph over the nxm grid (plus possibly some vertices around the edges) s.t. the longest common subsequence corresponds to the longest path in this graph. - Clearly describe the condition for the presence of an edge between two vertices on the grid.
- How should the edges be labeled?
- How do you find the longest path?
- Is this algorithm more efficient than the dynamic programming algorithm?

Answer : We will solve this problem using a directed graph over $m \times n$ vertices that are labeled according to their coordinates $(p, q) : 0 \leq p \leq m$ and $0 \leq q \leq n$. Edges are generated as follows : if letters p and q are the same, draw an edge from $V(p, q)$ to $V(p+1, q+1)$, aka a diagonal. If the letters are different, draw two edges from $V(p, q)$, one to $V(p+1, q)$ and the other to $V(p, q+1)$, aka down and right. The import part is this : edges that go diagonal are given a weight of 1, and edges that go right or down get a weight of 0. The longest path now doens't depend on the number of edges directly, but rather how many diagonal edges are along a path, since they contribute all of the weight.

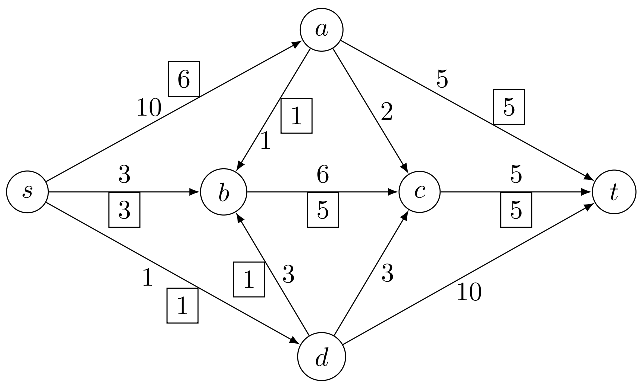


Graph of `0101` and `0110`.

Because all edges travel in either the positive x, positive y, or positive x and y directions, we know that this graph is acyclic. Because this graph is directed and acyclic, we can use a realtively fast algorithm (compared to the NP-hard algorithm for the longest-path on a general-graph case) to find the longest path. The algorithm contains two parts, making the graph and retreiving the longest path. To find the longest path, we have to find a topological sort of the graph.
This can be done with a modifed version of DFS that pushes vertices onto a stack, at the very end that stack will contain the topological sort of G. Once we have this sort, we can find the longest path using a simple algorithm.

Start by walking over all $m \times n$ vertices in the graph and generating edges. Each vertex can have at most two edges eminated from it, so each of the $m \times n$ operation are constant. Thus constructing the graph is $O(mn)$. Once the graph is constructed, DFS is run on it which runs in time $O(|V + E|)$, since $E \leq 2 * |V|$, DFS runs in time $O(|V|) = O(mn)$. DFS gives us our topoligical sort, which leads to the last portion of our algorithm, retreiving the longest path.
The algorithm in short walks along the topolical sort from the source vertex (which would be $V(0, 0)$ for the shortest-common-subsequence problem), and updates its neighbors where-ever it can by adding its own weight-sum to the weight of the incident edges and setting the neighbors weight-sum to this value.

Problem 2



1. What is the value of this flow? Is this a maximum (s, t) flow in this graph?
   Answer : The value for this flow is 10, as there is 10 units leaving the source and 10 units entering the drain. It is **not** a maximum (s, t) flow in this graph, I can find one that gives a flow of 11 by some clever redirection.
   s

2. Find a minimum $s - t$ cut in the flow network pictured in Figure 7.27[above], and also say what its capacity is.
   Answer : The minimum cut that I found is between the two following sets of vertices : (s, a, b) and (d, c, t). If you count the edges going into t, you get an incoming flow of 5+2+6+1=14. When you add up the outgoing flows, you get 3. So the total flow between the two cuts is $14 - 3 = 11$, which is the same as the maximum flow I discovered in part A of this problem.

Problem 3

Decide whether you think the following statement is true, give a short explanation. If it is false, give a counterexample.

Let G be an arbitrary flow network, with a source s, a sink t, and a positive integer capacity c_e on every edge e. If f is a maximum s-t flow in G, then f saturates every edge out of s with flow (i.e., for all edges e out of s, we have $f(e) = c_e$).

Answer : This statement is false, all I needed to do to confirm so was to look at a few examples from the textbook. If the above statement was true, it would vatly simplfiy the process of finding the maximum flow, as it would have to be equal to the sum of the capactities of the edges leaving the source s. A concrete counter-example is given below.

Problem 4

Decide whether you think the following statement is true or false. If it is true, give a short explanation. If it is false, give a counterexample.

> *Let G be an arbitrary flow network, with a source s, a sink t, and a positive integer capacity c_e on every edge e; and let (A, B) be a minimum s-t cut with respect to these capacities $\{c_e : e \in E\}$. Now suppose we add 1 to every capacity; then (A, B) is still a minimum s-t cut with respect to these new capacities $\{1 + c_e : e \in E\}$.*

Problem 5

Suppose you are given a direced graph $G = (V, E)$, with a positive integer capacity c_e on each edge e,... a maximum $s - t$ flow in G, defined by a flow value f_e on each edge
Now, suppose we pick a specific edge $e^* \in E$ and reduce its capacity by 1 unit. Show how to find a maximum flow in the resulting capacitated graph in time $\mathcal{O}(m + n)$, where m is the number of edges in G and n is the number of nodes.
Answer : Well, to start, we know that if the chosen edge e is not at capacity, then reducing its capacity by one will obviously not effect the max flow, as we didn't change any flow, just reduced the extra capacity of a single 'pipe'. Thus, if the edge isn't saturated, we don't have to do anything to find the max flow and thus we are done immediately. If, however, the given edge is not saturated :
If a given edge is not saturated, and we lower its capacity by one, we will need to forward this effect through the system. So take the decreased flow, and extrapolate that change through the inputs and outputs of that edge that had its capacity reduced. Once the effects of this change have propogated, we an iteration of FF algorithm on the new graph. FF will find an augmenting path and take it if it exists, which will find maximum-flow value for the new graph.

Finding if e is saturated can be done easily, propogating the effects of the reduction in capacity means we at most need to look at all m edges in the graph at most. Finally, running a single iteration of the Ford-Fulkerson runs in time $O(m + n)$, thus the total running time of the algorithm is $O(m + n)$.

Problem 6

Professor Adam has two children who, unfortunately, dislike each other. The problem is so severe that not only do they refuse to walk to school together, but in fact each one refuses to walk on any block that the other child has stepped on that day. The children have no problem with their paths crossing at a corner. Fortunately both the professor's house and the school are on corners, but beyond that he is not sure if it is going to be possible to send both of his children to the same school. The professor has a map of his town. Show how to formulate the problem of determining whether both his children can go to the same school as maximum-flow problem.s

Answer : To start, the house of the professor represent the source node, as it is where the children leave from when they go to school. The sink node is the school, as it absorbs both students. We also know that the school and home are on street corners, which means there are at least two streets to leave the house from and at least 2 streets to arrive at the school from, which means at minimum we know those two nodes work. We then have to determine if we can find two seperate paths that both leave the home and meet up at the school, without ever crossing eachother along the way. To formalize it in terms of our problem, we let each street corner represent a vertex. Both children can be at the same vertex at the same time. The streets represent the edges of our graph, and each edge has a capactity of 1 because it can at most carry only one of the two children. Our task is then to see what the maximum-flow of this graph is. If it is 0 or 1, then both kids with be forced to take the same edge at some point, which means they will walk on the same street which doesn't work. Thus if maximum-flow is ≤ 1 the boys can't go to the same school. If, however, the maximum flow is ≥ 2, both students can go to school together since this means there are two paths with unique edges between the home (source) and the school (sink).