

Network Flows - Final, sketch of solutions

Average 85.67, high 95!

1). (a). A FEEDBACK EDGE SET of an undirected graph $G = (V, E)$ is a subset of edges $E' \subseteq E$ whose removal results in an acyclic graph $G' = (V, E \setminus E')$. Given a graph with positive edge weights $w_{ij} > 0$, describe a polynomial time algorithm that finds a minimum weight feedback edge set. Recall, this problem is NP Complete for directed graphs.

The feedback edge set is a set of edges to be removed so that the result is a tree. Since we want to remove the smallest weight edge set, this is equivalent to finding the set of edges *not* to be removed of max weight, namely, the max spanning tree, which can be computed in polynomial time.

(b). For a graph $G = (N, E)$ define $conn(G)$ the number of connected components of G . Prove that $conn(G) + m \geq n$ for every graph G . $|N| = n$, and $|E| = m$.

For each connected components, we have $m_i \geq n_i - 1$, since each connected component has a spanning tree. Adding over all i , we get $\sum_i (m_i + 1) \geq \sum_i n_i$, and $\sum_i m_i = m$, $\sum_i n_i = n$, $\sum_i 1 = conn(G)$.

2). We say that an undirected graph has a *strongly connected orientation* if its edges can be directed so that the resulting graph is strongly connected (meaning that for every two nodes i, j there is a directed path from i to j and from j to i). A *bridge* in a graph G is an edge whose removal disconnects G .

Prove: A graph $G = (V, E)$ has a strongly connected orientation if and only if G is connected and has no bridge. (Ideally, your proof should be constructive. It should describe an efficient algorithm that gives a strongly connected orientation, if one exists).

Clearly the condition is necessary. If the graph is not connected, then nodes in different connected components will not have a directed path connecting them regardless of how the edges are oriented. Also, if the graph has a bridge, say (i, j) then without loss of generality say the edge is directed from i to j , there is no directed path from j to i .

One such algorithm: Find the 2 connected components of the graph. Note that by our assumption that no bridges exist, each 2 connected component must be more than a single edge. We show how to direct the edges of each such 2 connected component to get that component strongly connected: Do DFS and direct the tree edges down, and non tree edges up. This works, for the following reason: The resulting graph will have a path from the root of the DFS tree to every node i . Also, since the graph under consideration is 2-connected, using the non tree arc (and possibly some tree arcs) from every node i there is a directed path to the root. The subtree rooted at i has a "back edge" to a node between i and the root (a node of level less than i 's level) say node j . If $j \neq$ the root, then the subtree rooted at j has a back edge to a node with level less than j . This process can be continued until we find a directed path from i to the root. Now since the oriented graph has a path from the root to every node and back, then there is also a path from every node u to every node v : Go from u to the root and then from the root to v . This is a directed walk that can be shortcut into a directed path.

Another approach: Since there are no bridges, every node has degree at least 2, and so the graph has a cycle. Orient the cycle to become a directed cycle. Now you know that all the nodes of this cycle are strongly connected. Shrink these nodes into a single node and repeat. Note that no bridges are created by this process.

3). Let $G = (N, R, B)$ be a connected (multi) graph, where N are the nodes, R is a set of red edges and B is a set of blue edges. Suppose that for every node $i \in N$ we have that $deg_R(i) = deg_B(i)$, where $deg_R(i)$ is the degree of node i in $G(R) = (N, R)$ (and $deg_B(i)$ is the degree of node i in $G(B) = (N, B)$).

(a). Prove that G has an Euler cycle. The graph is connected (given) and for each node, $deg(i) = deg_R(i) + deg_B(i) = 2deg_R(i)$ so each degree is even. By Euler's Theorem, an Euler cycle exists.

(b). Prove that G has an Euler cycle in which the edges of R and B alternate (i.e., when traversing the cycle, we walk along a red edge then blue then red etc.).

Use the proof for existence of Euler cycle: Partition the edges into a set of cycles each of which alternates between red and blue edges. This can be done by starting at an arbitrary node, and following a path that alternates red and blue edges until a node is repeated. Since at each node the number of touching red edges is equal to the number of blue edges, we will not “run out” of either colour before the other colour.

Now, sew the cycles together, one by one at common nodes in such a way that colours continue to alternate. This is possible, because at a node i which 2 cycles contain, each cycle has both a red and blue edge touching the node. Say the first cycle enters node i on a red edge, you then detour and follow the second cycle starting at node i and exiting it on the blue edge. At the end of this second cycle, node i must be re-entered on the red edge, and then continue along the first cycle, leaving i on the blue edge.

Comment: several people described only the first step, finding an alternating cycle, without explaining how to do so while also using all the edges. If one arbitrarily follows edges (in an alternating way) one may get a cycle that does not contain *all* the edges. You can use this method to partition the edges into alternating cycles, but then have to somehow merge the cycles together. Otherwise, one should explain carefully how to make sure that the edge following approach does not close a cycle before covering all the edges.

Another incomplete solution was to say that we can follow alternating edges starting at an arbitrary node until one finds an Euler cycle. Note that unless special care is taken, the cycle may not contain all the edges.

4). We are given a directed graph $D = (N, A)$ with (nonnegative) upper bounds on the flow and nodes $s, t \in N$. An arc is *upward critical* if increasing the capacity on this arc (strictly) increases the max flow from s to t . An arc is *downward critical* if decreasing the capacity on this arc (strictly) decreases the max flow from s to t .

(a). Does every network have an upward critical arc? No. Consider the graph on 3 nodes, s, a, t , arcs (s, a) and (a, t) each of capacity 1. Increasing the capacity of one of the arcs will not change the max flow from s to t .

(b). Describe an algorithm to identify all upward critical arcs. Your algorithm should be faster than solving m max flow problems! An arc (p, q) is upward critical if there exists a flow augmenting path from s to p , and a flow augmenting path from q to t . After finding a max flow, (which can be done in $O(n^3)$) do another labelling to find all nodes S that are reachable by an augmenting path from s . (This set clearly does not include t , or else this would not be a max flow.) By a similar labelling from t , find all nodes T that have an augmenting path to t . Each of these labellings can be done in linear time. Now an arc (p, q) is upwards critical if and only if $p \in S$ and $q \in T$. This second step takes linear time so that is less than the first (max flow) step.

Another (not as good) option is to check for every saturated arc (i, j) whether there exists a flow augmenting path s to i and j to t , which can be done in linear time, resulting in an $O(m^2)$ algorithm, which could be slower than the max flow part of $O(n^3)$.

(c). Does every network have a downward critical arc? Yes. Any arc that belongs to a min cut is downward critical, since decreasing its capacity decreases a min cut, and so also the max flow. Note: An arc that is saturated is not always downward critical, if it is not in a min cut.

(d). Is the set of upward critical arcs the same as the set of downward critical arcs? No. Same example as in part (a). Neither arc is upward critical, both both are downward critical.

5). For each of the following prove or give a counterexample:

(a). Given a graph G with distinct edge costs, the edge of minimum cost in some cycle must be part of the minimum spanning tree. False. Consider the graph on nodes a, b, c, x, y, z , with edges $(a, x)(x, b)(b, y)(y, c)(c, z)(z, a)$ all of cost about 1, (close to 1 but distinct) and edge (a, b) of cost 10, edge (b, c) of cost 11, and (c, a) of cost

12. Edge (a,b) has smallest cost in the cycle a-b-c-a, but is not in any MST, which will contain 5 out of the 6 edges of cost about 1.

(b). Given a graph G with distinct edge costs, the shortest path between s and t is unique. False. Consider the graph on nodes s,a,t with edges (s,a) of cost 1, (a,t) of cost 2 and (s,t) of cost 3. There are two different shortest paths from s to t .

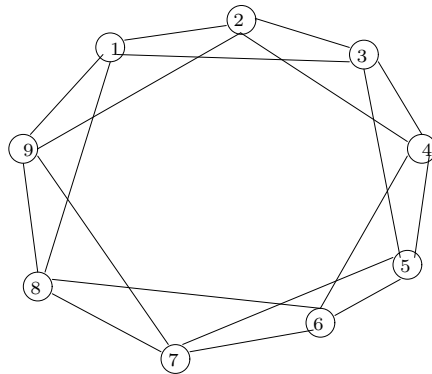
(c). Given a directed graph G with distinct capacities, the minimum cut (S, \bar{S}) is unique. False, see example of part (a) problem 4.

(d). Given a directed graph G with nonnegative arc lengths, and a node s . The tree of shortest paths from s remains the same if a constant $B > 0$ is added to the lengths of all arcs (s, i) (for all nodes i). True. Every path from s will have its cost increased by the same constant, B and therefore the same paths remain shortest.

(e). Given a directed graph G with nonnegative arc lengths, and a node s . The tree of shortest paths from s remains the same if for some node $j \neq s$, and some constant $B > 0$ is added to the lengths of all arcs (j, i) (for all nodes i). False. Consider the graph with nodes s,i,j,t and arc (s,t) of cost 4, arcs (s,i) (i,j) (j,t) of cost 1. The shortest path is $s-i-j-t$. If we increase the cost of (i,j) by 3 (or any $B > 1$) then the new shortest path will be $s-t$.

(f). Given a min cost flow problem on directed graph G with nonnegative arc costs, and capacities, the min cost flow x^* remains the same if a constant B is added to all edge costs. False. Consider the same graph in part (e) in which we want to send 1 unit of flow from s to t . At the original costs, we send the flow on path $s-i-j-t$. If all costs are increased by a constant, say 2, then we will instead send the flow on path $s-t$.

6). Given a Hamilton tour on a set of n nodes, $\{(v_i, v_{i+1}) \mid i = 1, \dots, n\}$, The *double tour* is the set of edges of the tour and their shortcuts $\{(v_i, v_{i+2}) \mid i = 1, \dots, n\}$ (all indices are modulo n). See the example below for 9 nodes.



The DOUBLE TOUR PROBLEM (DTSP) is: Given a graph G with nonnegative costs on the edges, find a Double Tour of minimum cost. We assume our graph is complete, undirected and that the costs satisfy the triangle inequality.

Describe an approximation algorithm for the DOUBLE TOUR PROBLEM. You should clearly describe an algorithm that runs in polynomial time, then prove that the Double Tour has cost at most some constant times the cost of an optimal Double Tour, for all instances of the problem. You may use any of the approximations of TSP discussed in class.

A 1.5 apx for TSP yields a 2.25 apx for the corresponding double tour problem. Use the apx TSP as your double tour cover. Its cost is the cost of the tour plus cost of short cuts. By triangle inequality, the shortcuts cost at most 2 times the cost of the tour, so the solution has cost at most 3 times APX(TSP) which is at most 4.5 times OPT(TSP). Since $\text{OPT(TSP)} \leq \text{OPT(DTSP)}$ this gives us a 4.5 apx.

We can do better by noticing that an optimal solution to DTSP contains two disjoint Hamilton cycles. For $n = \text{odd}$ we have the tour and the shortcuts form a second tour. For $n = \text{even}$, we have the tours $1, 3, 5, \dots, n-1, n, n-2, n-4, \dots, 4, 2, 1$, and the tour $2, 3, 4, \dots, n-2, n-1, 1, n, 2$. So $\text{OPT}(\text{DTSP}) \geq 2\text{OPT}(\text{TSP})$, and therefore $\text{APX}(\text{DTSP})/\text{OPT}(\text{DTSP}) \leq 4.5/2 = 2.25$.