University of Toronto at Scarborough
CSCC73 - Algorithm Design and Analysis

Assignment #6: Network Flow

Due: March 13, 2023 at 11.59pm This exercise is worth 5% of your final grade.

Warning: Your electronic submission on Gradescope affirms that this exercise is your own work and no one else's, and is in accordance with the University of Toronto Code of Behaviour on Academic Matters, the Code of Student Conduct, and the guidelines for avoiding plagiarism in CSCC73. Late assignments will not be accepted. If you are working with a partner your partners' name must be listed on your assignment and you must sign up as a "group" on MarkUs. Recall you must not consult **any outside sources except your partner, textbook, TAs and instructor**.

1. (10 marks) Suppose you have a flow network. We define a *critical* edge on the network to one that if we decrease the capacity of this edge the maximum flow is decreased. Give an efficient algorithm that finds a critical edge in a network. Be sure to explain the complexity of your algorithm and justify the correctness of your algorithm.

   Soln

   Run Ford Fulkerson to construct the residual graph. A critical edge must have flow equal to its capacity. Consider each edge (u, v) whose capacity is full in the residual graph and check if there is a u to v path in the residual graph (using DFS for example). If there is not, then (u, v) is a critical edge.

   The correctness follows from the following argument. If an edge is not full when max flow is achieved, it is possible to reduce it's capacity to the flow going through it without affecting the max flow. It is not sufficient however, to just find those edges whose capacity is maxed out as if there is an alternative route to send flow when reducing the capacity of that edge, then the edge is not critical. Therefore, consider an edge (u, v) that is at capacity after running Ford Fulkerson and for which there is no alternative path. This means that reducing this edge reduces the flow as there is no other route to send the flow.

   Alternatively, if one finds a min-cut by running Ford Fulkerson and then a BFS to find the set A of an (A, B) cut, then any edge (u, v) such that $v \in B$ and $u \in A$ is a critical edge as reducing the capacity of this edge reduces the value of the min-cut and by weak duality the value of the flow.

2. (10 marks) A vertex cover of an undirected graph $G = (V, E)$ is a subset of the vertices which touches every edge, ie., that is, a subset $S \subset V$ such that for each edge $\{u, v\} \in E$, one of both of u, v are in S.

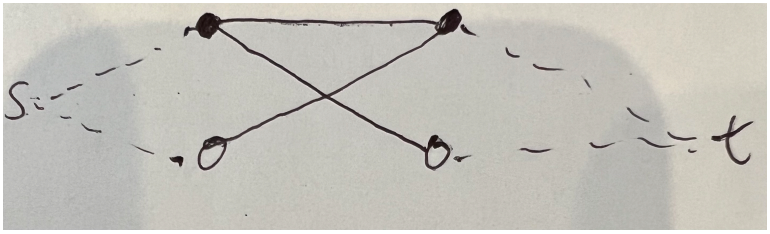
   Show that the problem of finding the minimum vertex cover in a bipartite graph reduces to maximum flow. HINT: Consider relating this problem to minimum cuts.

   Solution 1

   Suppose we set up a max flow for the bipartite graph $V = L \cup R$ (L and R are the two partitions of the vertex set) as we did to find a maximum matching. We construct a flow graph from G by adding

a vertex s adjacent with weight 1 to each vertex in L and similarly an edge from each vertex in R to a vertex t also with weight 1.

Consider the maximum matching found by constructing a maximum flow as done in class. Notice that in this case, we can construct a min vertex cover by taking one endpoint of each edge in the matching of the original graph G. Let (x, y) be an edge in the matching.When choosing an endpoint, we choose the one that belongs to another edge that is not in the matching (if possible). In other words, if (x, z) is an edge in G and z not the endpoint of an edge in the matching, we add x to our cover. Similarly for the symmetric case of y. What if both x and y belong to edges not in the matching with endpoints not belonging to matching edges? See figure below, then our matching was not maximal.



Therefore, we can select one of x or y for our vertex cover for each edge (x, y) in the matching and every edge of G will be covered and the number of vertices is equal to the size of the matching or max flow. Note that the vertex cover cannot be less than the size of a matching as for each vertex in a cover more than one edge can be covered, however a matching only overs two vertices at a time.

Solution 2

How does this relate to min-cuts? To use a minimum cut to find the vertex cover we set up our flow graph slightly differently. We make each edge between L and R have weight ∞. Now again, we construct a max flow and consider the residual graph. Notice that *none* of the cross edges between L and R can possibly be maxed out so will not belong to a min cut. This means all min cut edges must either be between s and a vertex of L or between t and a vertex of R.

If we select those vertices on edges in the min cut adjacent to s and those vertices adjacent to t we have exactly |min-cut| vertices. Do these vertices cover all the edges of our original graph G?

Suppose there is an edge (x, y) with no endpoints in the vertex cover (ie, no endpoints crossing the cut). That means that there must be an s, t path from $sxyt$ with residual capacity 1, contradicting that the max flow was achieved, therefore, every edge is covered by the vertices in our vertex cover.

3. (10 marks) Suppose you're looking at a flow network G with source s and sink t, and you want to express the intuition that some nodes are clearly on the "*source side*" of the main bottlenecks; some nodes are clearly on the "*sink side*" of the main bottlenecks; and some nodes are in the middle. However, G can have many minimum cuts, so we have to be careful in how we make this idea precise.

   - We say a node v is *upstream* if, for all minimum s, t-cuts (A, B), we have $v \in A$–that is, v lies on the source side of every minimum cut.
   - We say a node v is *downstream* if, for all minimum s, t-cuts (A, B), we have $v \in B$–that is, v lies on the sink side of every minimum cut.
   - We say a node is *central* if it is neither *upstream* nor *downstream*; *i.e.*, there is at least one minimum s, t-cut (A, B) for which $v \in A$, and at least one minimum cut (A^*, B^*) for which $v \in B^*$.

Give an algorithm that takes a flow network G and classifies each of its nodes as either upstream, downstream, or central. The running time of your algorithm should be within a constant factor of the time required to compute a *single* maximum flow. You should carefully explain why your algorithm works.

Soln We'll answer all three parts together. Consider the cut (A^*, B^*) found by performing BFS on the residual graph edges after computing a max flow F^* (recall that we only consider edges that have residual capacity > 0) after any maximum flow algorithm. We claim that a node v is *uptream* if and only if $v \in A^*$. If v is upstream, then it must belong to A^* since otherwise , it lies on the sink side of the minimum cut (A^*, B^*).

Conversely, suppose $v \in A^*$ were not upstream. Then there would be a minimum cut (A', B') with $v \in B'$. We will show this isn't possible.

- Consider the flow network G_2 obtained by adding an edge (v, t) with capacity 1 to G. Then the cut (A', B') has the same capacity in G as in G_2 since the added edge does not cross the cut, so min-cut(G_2) $\leq$min-cut(G).
- On the other hand, we can send an additional unit of flow by following an $s \to v$ path in F^* of G_2 using (v, t), so max-flow(G_2) $\geq$ max-flow(G) $+ 1 >$max-flow(G).
- By strong duality in G, min-cut(G_2) $\leq$ min-cut(G) $=$ max-flow(G) $<$max-flow(G_2), but this contradicts the weak duality in G_2. Thus all nodes in U must be upstream.

We can use a completely symmetric argument to show part (b). In particular, let B_* denote all those vertices that have a path in the residual graph to t and let $A_* = V - B_*$. Then (A_*, B_*) is a minimum cut and a node w is downstream if and only if $w \in B_*$. One way to find B_* would be to reverse the edges in the residual graph and again do a BFS but this time starting from t.

Therefore, we can use Ford-Fulkerson to find a maximum flow and the corresponding residual graph. Then, we use BFS to find each of A^* and B_*, the upstream and downstream nodes. All remaining nodes then are central. The complexity is bounded by the complexity of Ford-Fulkerson as BFS and reversing edges are both $O(n + m)$.