CSCI 270 - Spring 2023 - HW 8

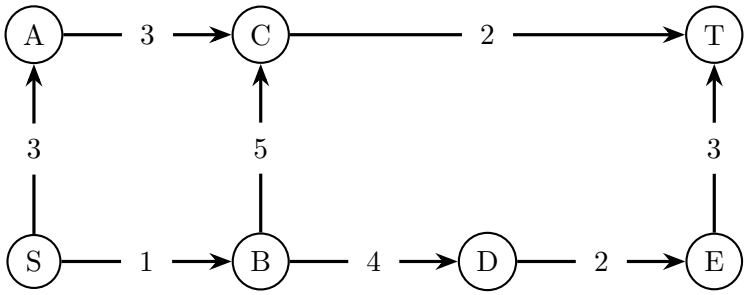
Due 1: March 15, 2023 (You can get credits and feedback before midterm)
Due 2: March 22, 2023 (You can get credits and feedback after midterm)

1. (10pts) The following graph G has labeled nodes and edges between it. Each edge is labeled with its capacity.

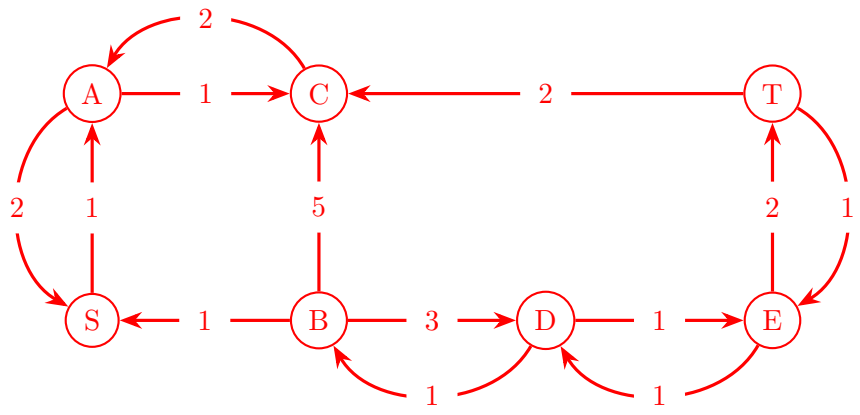
   (a) Draw the final residual graph G_f using the Ford-Fulkerson algorithm corresponding to the max flow. Please do NOT show all intermediate steps.

   (b) What is the max-flow value?

   (c) What is the min-cut?



(a) Final Residual Graph



(b) The max flow is 3.

(c) The min cut is $\{S, A, C\}$ and $\{B, D, E, T\}$. (Quick check - This is correct because all the edges to and from the min cuts sets are either saturated or have no no-flow.)

Rubric (10 pts)

- 5 pts: Correct Final Residual graph.(-2 pts: Incorrect edge capacity.)
- 2 pts: Correct max flow value.
- 3 pts: Correct min cut sets.

2. (15pts) Determine if the following statements are true or false. For each statement, briefly explain your reasoning.

   (a) In a flow network, the value of flow from S to T can be higher than the maximum number of edge disjoint paths from S to T. (Edge disjoint paths are paths that do not share any edge)

   (b) For a flow network, there always exists a maximum flow that doesn't include a cycle containing positive flow.

   (c) If you have non-integer edge capacities, then you cannot have an integer max flow.

   (d) Suppose the maximum s-t flow of a graph has value f. Now we increase the capacity of every edge by 1 . Then the maximum s-t flow in this modified graph will have a value of at most $f + 1$.

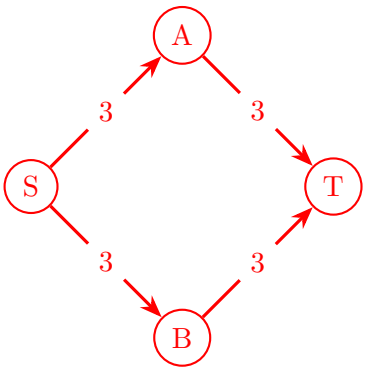
(e) If all edges are multiplied by a positive number k, then the min-cut remains unchanged.

(a) True, consider the case when edges have capacity greater than 1.

(b) True, we can always remove the flow from such a cycle and still get the same flow value.

(c) False, consider a graph with source s, sink t and two nodes a and b. Let's say there is an edge from s to both a and b with capacity 0.5 and from both a and b to t with capacity 0.5 , then the max flow for this graph is 1 , which is an integer.

(d) False. Counter-Example: In the following graph, the maximum flow has value $f = 3 + 3 = 6$. Increasing the capacity of every edge by 1 causes the maximum flow in the modified graph to have value $4 + 4 = 8$



(e) True. The value of every cut gets multiplied by k, thus the relative-order of min-cut remains the same.

Rubric (15 pts) For each question:

- 1 pt: Correct choice
- 2 pt: Valid reasoning for the answer

3. (15pts) You are given a flow network with unit-capacity edges. It consists of a directed graph $G = (V, E)$ with source s and sink t, and $c_e = 1$ for every edge e. You are also given a positive integer parameter k. The goal is delete k edges so as to reduce the maximum s-t flow in G by as much as possible. In other words, you should find a subset of edges $F \subseteq E$ such that $|F| = k$ and the maximum s-t flow in the graph $G' = (V, E \backslash F)$ is as small as possible. Give a polynomial-time algorithm to solve this problem and briefly explain its correctness.

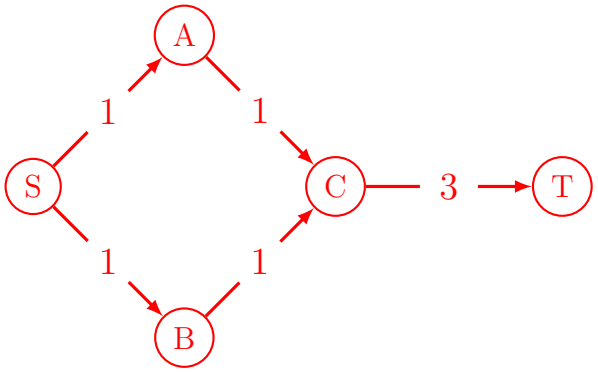
Follow up: If the edges have more than unit capacity, will your algorithm produce the smallest possible max-flow value?

Algorithm:

- Assume the value of max-flow of given flow network $G(V, E)$ is g. By removing k edges, the resulting max-flow can never be less than $g - k$ when $|E| \geq k$, since each edge has a capacity 1 and removing each edge reduces the max-flow value by at most 1.
- According to max-flow min-cut theorem, there is an s-t cut with g edges. If $g \leq k$, then remove all edges in this s-t cut, decreasing max-flow to 0 and disconnecting s and t. Else if $g > k$, then remove k edges from this cut, and create a new cut with $g - k$ edges.
- In both the cases, whether the max-flow is 0 or $g - k$, the max-flow cannot be decreased any further thus giving us the maximum reduction in flow.
- The algorithm has polynomial run-time. It takes polynomial time to compute the minimal-cut and linear time in k to remove k edges.

If all the edges don't have unit capacity, removing k edges from min-cut in the above mentioned way does not guarantee to have the smallest possible max-flow value.

Note: The following is the counter-example when $k = 1$. The algorithm proposed above will remove (S,A) (or (S,B)), which ends up reducing the max-flow from 2 to 1. But if we remove (C,T), the max flow will be 0.



Rubric (15 pts)

- 8 pts: Correct algorithm proposal with polynomial time.
- 4 pts: Correct explanation for algorithm.
- 3 pts: Correct example for the follow up.

4. (20 pts)A tourist group needs to convert their USD into various international currencies. There are n tourists $t_1, t_2, \ldots, t_n$ and m currencies $c_1, c_2, \ldots, c_m$. Each tourist t_i has F_i Dollars to convert. For each currency c_j, the bank can convert at most B_j Dollars to c_j. Tourist t_i needs to trade S_{ij} of his Dollars for currency c_j. (For example, a tourist with 1000 dollars might be willing to convert up to 200 of his USD for Indian Rupees, up to 500 of his USD for Japanese Yen, and up to 300 of his USD for Euros). Assume that all tourists give their requests to the bank at the same time.

   (a) Design an algorithm that the bank can use to satisfy all the requests (if it is possible). To do this, construct and draw a network flow graph, with appropriate source and sink nodes, and edge capacities.

   (b) Prove your algorithm is correct by making a claim and proving it in both directions.

   (a) Network Flow Construction and Algorithm: We will solve the problem, by first constructing a network flow graph and then running Ford-Fulkerson or any other Network Flow Algorithm to get the assignments.

   Construction:

   - Insert two new vertices, source node S and sink node T.
   - Connect all the tourists t_i with source node S assigning edge weight equal to F_i. Here, F_i stands for the maximum USD tourist t_i can exchange.
   - Then, connect all tourists t_i to all the currencies available i.e., c_j with each edge having weight S_{ij} which is the t_i tourist's limit to exchange his USD for a particular currency c_j.
   - Connect currencies c_j with sink node T, with edge weight B_j, which is the maximum limit of that particular currency c_j that the bank can convert from USD.

   In the graph constructed this way run Ford-Fulkerson or any Network Flow Algorithm from source S to sink T, the algorithm will return assignments, if it exists, that can solve the requests while following all the proposed constraints.

   (b) Below, we define $S_{\text{total}} = \sum_j \sum_i S_{ij}$ for simplicity.

   Claim: The problem has a solution if and only if the max flow through the constructed graph is S_{total}, i.e., all the tourists are able to exchange their USD as they want to while following all the constraints.

   Proof: We have to prove the claim in both directions:

- **Proof for Forward Direction:** Let us assume there is a valid assignment and prove that a max-flow of value S_{total} exists.

  Assume there is a valid assignment such that the bank can satisfy all the tourist's requests for conversion while following all the constraints. This means that all the outgoing edges from source each tourist t_i to every currency node is saturated. Thus, we get the maxflow in the graph from S to T is S_{total}.

- **Proof for Backward Direction:** Let us assume we have max-flow in the above constructed graph. The max-flow = S_{total} From the above constructed Network flow graph we can see that, if max-flow = S_{total} exists, then it means all the edges between tourists and currencies are saturated, which in turn means all the tourists were able to convert their currencies. Hence it is proved that a valid assignment exists if a max-flow = S_{total} exists.

Rubric (20 pts)

For part (a) 10pts

- 10 pts: Correct Construction of Network Flow graph. (-2 pts: For each incorrect edge weight/node. (source and sink can be reversed))

For part (b) 10 pts

- 4 pts: Correct Claim.
- 3 pts: Correct Forward Proof.
- 3 pts: Correct Backward Proof.

5. (15 pts) You have successfully computed a maximum $s - t$ flow f for a network $G = (V; E)$ with integer edge capacities. Your boss now gives you another network G' that is identical to G except that the capacity of exactly one edge is decreased by one. You are also explicitly given the edge whose capacity was changed. Describe how you can compute a maximum flow for G' in O(|V| + |E|) time.

   Say the edge (u, v) the edge that has lost capacity, then

   - Find a path in G from s to v on edges that are carrying some flow (BFS or DFS O(|V| + |E|))
   - Find a path in G from u to t on edges that are carrying some flow (BFS or DFS O(|V| + |E|))
   - Reduce flow by unit on edges that are on the path $s \to u \to v \to t$. Since all these edges are carrying some flow with the capacities, the results will be a new

valid flow f' (But not necessarily a max flow.)

Here, we run an iteration of Ford-Fulkerson algorithm and get the max-flow ($O(|E|)$). We note that Ford-Fulkerson algorithm does not run more than one iteration because otherwise G' will hold more flow than G does, which is impossible.

Rubric (15 pts)

- 10 pts: Correct algorithm.
  - 5pts: The part of finding a unit flow
  - 5pts: Run an interation of Ford-Fulkerson algorithm
- 5 pts: Correct reasoning for the complexity $O(|V| + |E|)$.