CSCI 570 - Spring 2022 - HW 8 Solution

Due March 16, 2022

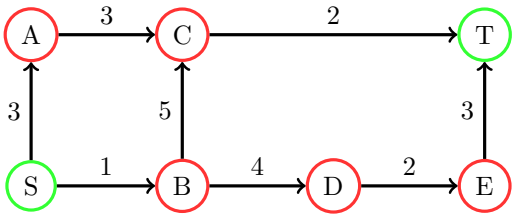
Graded

1. The following graph G has labeled nodes and edges between it. Each edge is labeled with the 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.

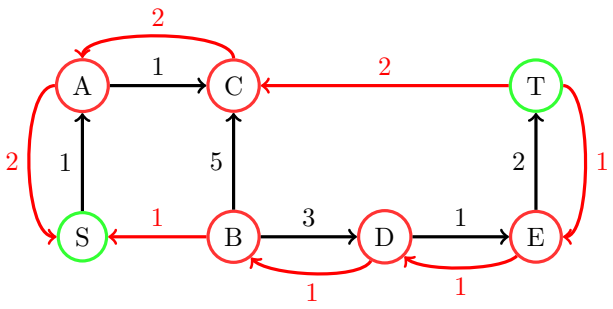
   (b) What is max-flow value?

   (c) What is the min-cut?

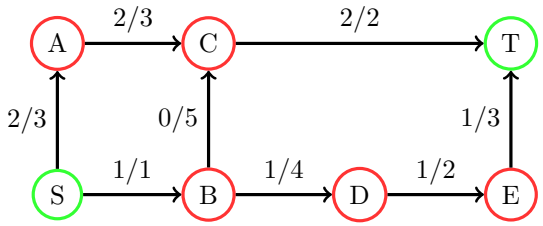


Solution:

(a) Final Residual Graph



Final Flow graph (not required to be drawn)



(b) The max flow is 2+1=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)

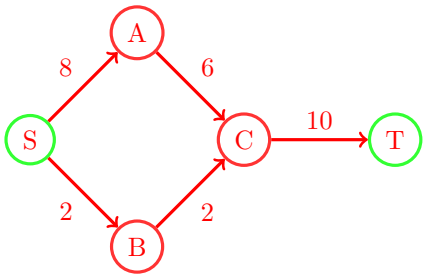
- 5 pts: Correct Final Residual graph.
- −2 pts: Incorrect edge capacity.
- 2 pts: Correct max flow value.
- 3 pts Correct min cut sets.

2. The following statements may or may not be correct. For each statement, if it is correct, provide short description. If it is incorrect, provide a counter-example to show that it is incorrect.

   (a) In the capacity-scaling algorithm, an edge e can be present in the residual graph $G_f(D)$ in at most one D-scaling phase.

   Solution:

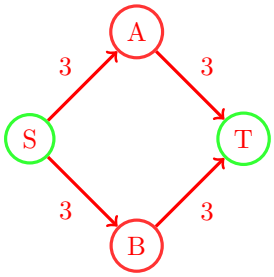
   False.
   Counter Example:
   In the following counter example graph G, the edge (C, T) is present in two D-scaling phases.



   (b) Suppose the maximum (s,t)-flow of some 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 value at most f + 1.

   Solution:

   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.



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

Solution:

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

Rubric (15 pts)

- 5 pts For each sub-question
  - 2 pts: Correct True/False
  - 3 pts: Correct description or counter example

3. 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 $u_e = 1$ 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. Give a polynomial-time algorithm to solve this problem. In other words, you should find a set of edges F ⊆ E so that $|F| = k$ and the maximum st flow in the graph G' = (V, E-F) is as small as possible. Give a polynomial-time algorithm to solve this problem.

   Follow up: If the edges have more than unit capacity, will your algorithm guarantee to have minimum possible max-flow?

Solution:

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.
- 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 that s-t cut, decreasing max-flow to 0 and disconnecting 's' and 't'. Else if, g > k, then remove 'k' edges from g, 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 minimum possible max-flow.

Rubric (15 pts)

- 12 pts: Correct Algorithm proposal with polynomial time.
- 3 pts Correct answer for non-unit edges.

4. USC students return to in person classes after a year long interval. There are k in-person classes happening this semester, c_1, c_2, ..., c_k. Also there are n students, s_1, s_2, ..., s_n attending these k classes. A student can be enrolled in more than one in-person class and each in-person class consists of several students.

   (a) Each student s_j wants to sign up for a subset p_j of the k classes. Also, a student needs to sign up for at least m classes to be considered as a full time student. (Given: $p_j \geq m$) Each class c_i has capacity for at most q_i students. We as school administration want to find out if this is possible. Design an algorithm to determine whether or not all students can be enrolled as full time students. Prove the correctness of the algorithm.

   (b) If there exists a feasible solution to part (a) and all students register in exactly m classes, the student body needs a student representative from each class. But a given student cannot be a class representative for more than r (where $r < m$) classes which s/he is enrolled in. Design an algorithm to determine whether or not such a selection exists. Prove the correctness of the algorithm. (Hint: Use part (a) solution as starting point)

Solution: Part (a)

We can solve the problem by constructing a network flow graph and then running Ford–Fulkerson algorithm to get the max flow. If the max flow is equal to nm, then all students can be enrolled as full time students.

The setup of the graph is described below.

(a) Place the n students and k classes as vertices in a bipartite style graph. Connect each student s_j with all the classes in p_j using edges with capacity of 1.

(b) Place a sink vertex which is connected to all the classes (or alternatively students). Also, place a source vertex which is connected to all the students (or alternatively classes).

(c) Connect all s_j student vertices to the source (or sink) vertex with capacity of m.

(d) Connect all c_i class vertices to the sink (or source) vertex with capacity of q_i.

Claim: The enrollment is feasible if and only if there exists a max flow of nm.

Proof of Correctness

Forward Claim: The max flow is nm if the problem has a feasible solution.

Proof: If there exists a feasible solution for the problem, this means that all the in-person classes has at max of q_i students and each student was

able to sign up for at least m classes. By the law of conservation, the outward flow at the student vertex must be equal to the incoming flow to student vertex. Therefore, all the edges from source to the student vertices will be saturated with value of m, as each of the student was able to sign up for his/her classes. Similarly all the incoming flow at student vertex will be distributed to some of the out going edge with value of 1. This total flow from student vertices to the class vertex will be nm (again, by the law of conservation). And that is will eventually flow to the sink vertices with edges of max capacity q_i. Therefore, if the problem has a feasible solution the the max flow will be nm.

Backward Claim: The problem has a feasible solution if the max flow is nm.

Proof: If the problem has a max flow of nm. This means that along any path from source to sink vertex there is a student who was able to sign up for a particular class. Simply, we just have to follow the flow along a path.

Because the flow is max flow, this means that each of the student vertices receive a flow of m and the outward flow at each of the student vertices will be exactly m. Thus, the student will be able to sign up for at least m classes, which satisfies the constraint of the problem. Also, by the construction of network flow each class can accommodate at most q_i, which satisfies another constraint as the flow along the edge will be at most q_i (that is less than or equal to the capacity). Hence, the problem has a feasible selection if the max flow is nm.

Solution: Part (b)

We can solve the problem by starting from the solution from part (a). We will modify the constructed network flow graph slightly and then will run Ford–Fulkerson algorithm to get the max flow. If the max flow is equal to the number of classes k, then a selection exists otherwise no such selection exists.

The setup of the graph is described below.

(a) Use the same nodes as constructed in Part (a) of the solution.

(b) The course selections (edges between students and courses) for each student will be reduced to what is available in the solution from part (a) (our feasible solution). i.e. we will remove some edges between students and courses that they did not get to sign up for.

(c) The capacity on student to class edges will same remain same (i.e. 1).

(d) Assign 1 as the capacity from classes to sink (or alternatively source) vertex. (As there can only be 1 student representative from each class.)

(e) Assign r capacity for edges between source (or alternatively sink) and student vertices.

Claim: The selection is feasible if and only if there exists a max flow of k.

Proof of Correctness

Forward Claim: The max flow is k if the problem has a feasible selection.

Proof: If there exists a feasible selection for the problem, this means that all the in-person classes had a valid selection and each student was at max selected for r selections. By the law of conservation, the outward flow at the class vertex must be equal to the incoming flow to class vertex. Therefore, all the edges from sink to the class vertices will be saturated with value of 1, as each of the class had exactly 1 selection. This means that, k was the inward flow at class vertices. (law of conservation). Similarly, the number of selections per student will have the maximum value of r classes. And the total flow from source vertex to the student vertices will be k (again, by the law of conservation). Therefore, if the problem has a feasible solution the the max flow will be k.

Backward Claim: The problem has a feasible selection if the max flow is k.

Proof: If the problem has a max flow of k. This means that along any path from source to sink vertex there is a selection of student from a class. Simply, we just have to follow the flow along a path.

Because the flow is max flow, this means that all the class vertices receive a flow of 1 and the outward flow at the class vertices will be exactly 1 (and hence the flow is k). Thus, only 1 selection of student per class is made, which satisfies the constraint of the problem. Therefore, each class has selection of 1 student only. Also, by the construction of network flow each student can only be selected for at max r classes, which satisfies another constraint as the flow along the edge will at most r (that is less than or equal to the capacity). Hence, the problem has a feasible selection if the max flow is k.

Rubric Part A (20 pts)

- 12 pts: Correct construction of the the network flow graph.
- -4 pts: For each incorrect edge weight.
- 8 pts Accurate Proof of correctness.

Rubric Part B (20 pts)

- 12 pts: Correct construction of the the network flow graph.

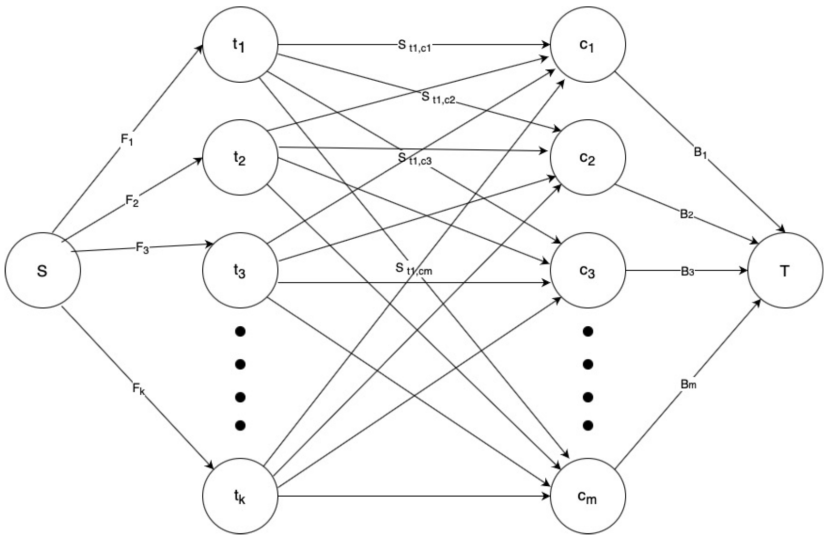
- -4 pts: For each incorrect edge weight.
- 8 pts Accurate Proof of correctness.

5. A tourist group needs to convert their USD into various international currencies. There are n tourists t_1, t_2, ..., t_n and m currencies c_1, c_2, ..., c_m. Tourist t_k has F_k Dollars to convert. For each currency c_j, the bank can convert at most B_j Dollars to c_j. tourist t_k is willing to trade as much as S_{kj} 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 Rupees, up to 500 of his USD for Japanese's Yen, and up to 200 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 the requests (if it is possible). Also, construct and draw the network flow graph, with source, sink appropriate nodes, and edge capacities.

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

Solution:



(a) **Network Flow Construction and Algorithm:**

We will solve the problem, by first constructing a network flow graph. Then running Fork-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 edges 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 ie. 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 weights as B_j, which is the maximum limit of that particular currency c_j that the bank can convert to 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) **Claim:**

The problem has a solution if and only if the max flow through the constructed graph is $\sum_k F_k$, ie. All the tourists are able to exchange their specified USD 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 max-flow $= \sum_k F_k$ 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 S to each tourist t_i is saturated. Thus, we get the max-flow in the graph from S to T is $\sum_k F_k$.

Proof in Backward Direction

Let us assume max-flow exists and thus prove that a valid assignment exists.

Let us assume we have max-flow in the above constructed graph. The max-flow $= \sum_k F_k$. From the above constructed Network flow graph we can see that, if max-flow $= \sum_k F_k$ exists, then it means all the edges are saturated, which in turn means all the tourist's were able to convert their currencies. Hence it is proved that a valid assignment exists if a max-flow $= \sum_k F_k$ exists.

Rubric (20 pts)

For part (a) - 10 pts

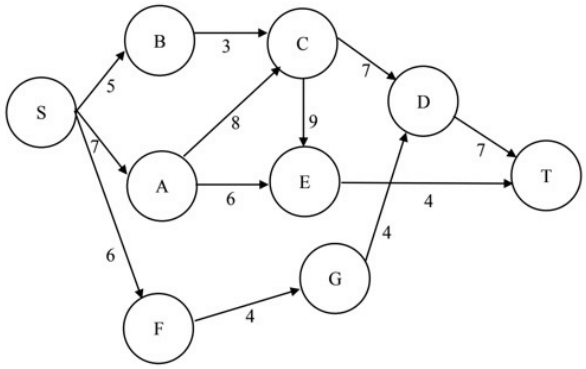
- 10 pts: Correct Construction of Network Flow graph.

- -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.

6. (20 pts) Perform two iterations (i.e. two augmentation steps) of the scaled version of the Ford- Fulkerson algorithm on the flow network given below. You need to show the value of Δ and the augmentation path for each iteration, and the flow f and G_f (Δ) after each iteration. (Note: iterations may or may not belong to the same scaling phase)



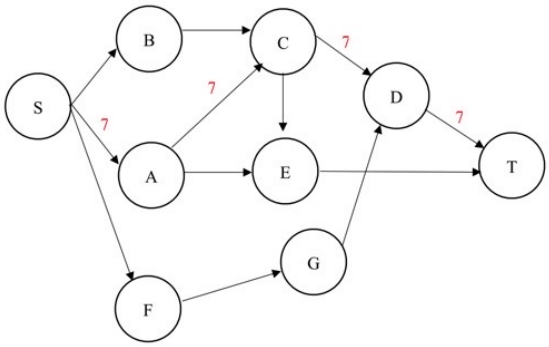
(a) i. Give the value of Δ and the augmentation path
ii. Show the flow after the first iteration
iii. Show the G_f (Δ) after first iteration

(b) i. Give the value of Δ and the augmentation path
ii. Show the flow after the second iteration
iii. Show the G_f (Δ) after second iteration

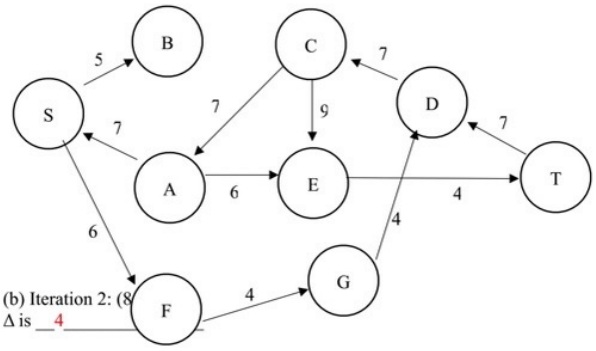
(c) Can the choice of augmentation paths in the scaled version of Ford-Fulkerson affect the number of iterations? Explain why

Solution:

(a) i. Δ : 4; Augmentation path : SACDT (there are few different choices of augmentation path)
ii. Flow after first iteration :

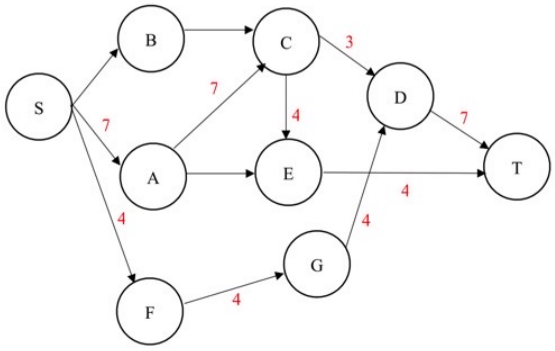


iii. G_f (Δ) after first iteration :

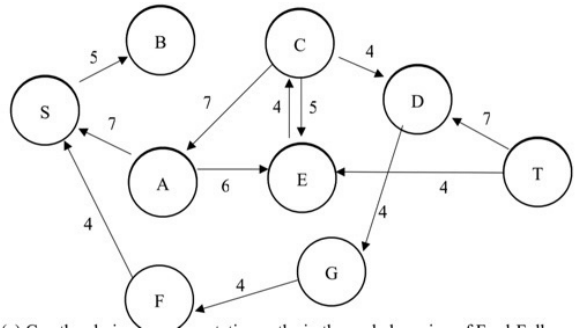


(b) i. Δ : 4; Augmentation path : SFGDCET (there are few different choices of augmentation path)

ii. Flow after second iteration :



iii. G_f (Δ) after second iteration :



(c) Yes, for example, if we had chosen paths with bottleneck values of 4 (SAET and SFGDT) in the first two iterations, then we should have needed more than two iterations to get to max flow. But with the choice of augmentation paths given above, we where able to get to max flow in 2 iterations. Or the argument could be: Yes, there are still different choices for augmentation paths within a given scaling phase. The different augmentation paths could have different bottleneck values. This can affect how quickly the value of flow goes up and therefore affect how many iteration we will have to perform.

Rubric (20 pts)

Parts a) + b)

- Computing the Delta values: 3+3 = 6
- Aug path + flow: 2+2 = 4
- G_f (Δ): 3+3 = 6
  (-1 for 1 edge missing/wrong. -2 for more.)

Part c)
0 If the answer is No. If Yes, 1 for the answer, 3 for the explanation. Partial credit 1/2 if the explanation is incomplete/unclear.

Ungraded

7. A vertex cover of an undirected graph $G = (V, E)$ is a subset of the vertices $A \subseteq V$ such that for every edge $e \in E$, at least one of its incident vertices is in A (that is every edge has at least one of its ends in A). Design an algorithm that takes a bipartite graph G' and a positive integer k, and decides if G' has a vertex cover of size at most k.

Solution:

Partition the vertices of the graph into left vertices and right vertices, that is $V = L \cup R, L \cap R = \phi$ and every edge has one end in L and the other in R.

We construct a network $\bar{G'}$ as follows. Add a source s, a sink t and the following edges;

(i) directed edges of capacity 1 from s to every vertex in L ,

(ii) directed edges of capacity 1 from every vertex in R to t

(iii)make every edge in $\bar{G'}$ directed from L to R and of infinite capacity.

Let $(S, \bar{S'})$ be a cut of $\bar{G'}$ of finite capacity. Without loss of generality assume that $s \in S$. We claim that we can construct a vertex cover of G' of size equal to the capacity of the cut.

Define $A := (L - S) \cup (R \cap S)$.

Edges that are incident on $L - S$ are covered by $L - S$. The only edges left to check are the ones with an end in $L \cap S$. The edges that are incident on $L \cap S$ have their other end in S and are hence covered by $R \cap S$. Thus A is indeed a vertex cover.

Edges between L and R have infinite capacity and hence cannot be in the cut. The only edges crossing the cut are either incident on s or incident on t. The number of edges of the form (s, u) where $u \in \bar{S}$ equals $|L - S|$. The number of edges of the form (v, t) where $v \in S$ is $R \cap S$. Hence the size of A equals the capacity of the cut $(S, \bar{S})$ and our claim is true.

We now claim the converse. That is, given a vertex cover A of G', we can construct a cut of $\bar{G'}$ of capacity equal to the size of A.

Define the cut $S := \{s\} \cup (L - A) \cup (R \cap A)$.

For an edge $(u, v) \in E$, if $u \in S$ then $v \in S$ (since A is a vertex cover). Thus edges in E cannot cross the cut.

The number of edges of the form (s, u) where $u \in \bar{S'}$ equals $|L \cap A|$. The number of edges of the form (v, t) where $v \in S$ equals $|R \cap A|$. Hence the capacity of the cut is $|L \cap A| + |R \cap A|$ which is exactly the size of A. Hence, the converse follows.

We have thus proven that the size of the min-vertex cover of G' equals the capacity of the min-cut of $\bar{G}'$. The capacity of the min-cut of $\bar{G}'$ can be computed using the Ford-Fulkerson algorithm (say the capacity is m). If $k < m$, output NO, otherwise output YES.