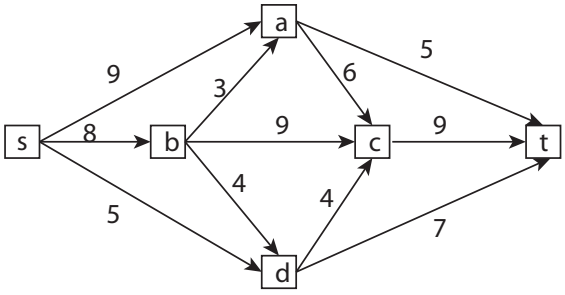
CSC373 Summer'22
Tutorial 4 Solutions
June 9-16, 2022

Q1 Ford Fulkerson

Consider the following network:



(a) Compute a maximum flow in this network using the Ford-Fulkerson algorithm. For each iteration, write down the augmenting path, its bottleneck/residual capacity, and the value of the flow at the end of the iteration.

(b) Consider the cut $X_0 = (S = \{s, b, c, d\}, T = \{a, t\})$. Identify all forward and all backward edges across X_0. Compute the capacity of X_0.

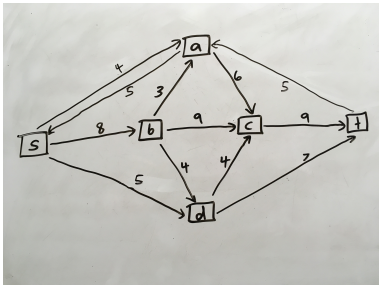
(c) Find a cut in the network whose capacity is equal to the value of the flow you computed in part (a). (This provides a guarantee that your flow is indeed maximum.) Use the idea outlined in the proof of correctness of the Ford-Fulkerson algorithm.

Solution to Q1

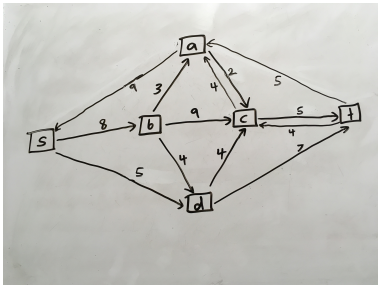
(a) The iterations are as follows.

Iteration	Augmenting Path	Residual Capacity	Value of Flow
1	(s, a, t)	5	5
2	(s, a, c, t)	4	9
3	(s, b, c, t)	5	14
4	(s, d, t)	5	19
5	(s, b, d, t)	2	21

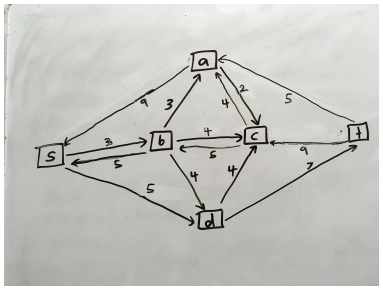
The residual graph at the end of each iteration is shown in the figure below. After the fifth iteration, we are done as there are no more directed paths from s to t in the final residual graph.



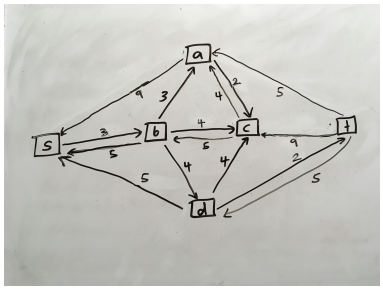
(a) After Step 1



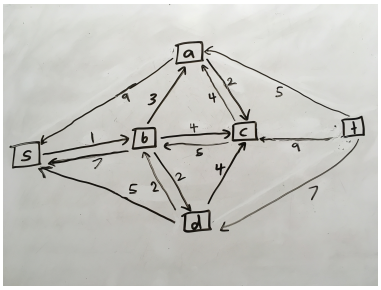
(b) After Step 2



(c) After Step 3



(d) After Step 4



(e) After Step 5

(b) The forward edges across X_0 are (s, a), (b, a), (c, t), and (d, t). The backward edge across X_0 is (a, c). The capacity of X_0 is $9 + 3 + 9 + 7 = 28$.

(c) From looking at the final residual graph in the figure above, the set of vertices reachable from s are $\{s, a, b, c, d\}$. Hence, the cut $(\{s, a, b, c, d\}, \{t\})$ must be a minimum cut. Its capacity is $5 + 9 + 7 = 21$, which matches the value of the maximum flow computed in part (a).

Q2 Graph Modifications

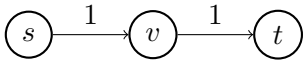
In this problem, we will consider what happens to the maximum flow when the flow network G is modified slightly.

(a) TRUE/FALSE: In any network G with integer edge capacities, there always exists an edge e such that increasing the capacity of e increases the maximum flow value in G.

(b) Suppose we are given a network G with n nodes, m edges, and integer edge capacities, and we are also given a flow f in G of maximum value. We now increase the capacity of a specific edge e by one. Give an $O(m + n)$ time algorithm to find a maximum flow in the updated network.

Solution to Q2

(a) FALSE. Consider the following network.



The maximum flow in this network is 1, but clearly increasing the capacity of any single edge would

not increase the maximum flow as the capacity of the other edge would still serve as a bottleneck.

(b) The algorithm operates as follows.

- Construct the residual graph G_f of flow f in the original network.
- If it already contains a forward edge for e, then f is still a maximum flow.
- If it does not, add a forward edge for e with capacity 1.
- If the updated residual graph contains a path P from s to t, then augmenting a unit flow along this path gives a new maximum flow f', otherwise f is still a maximum flow.

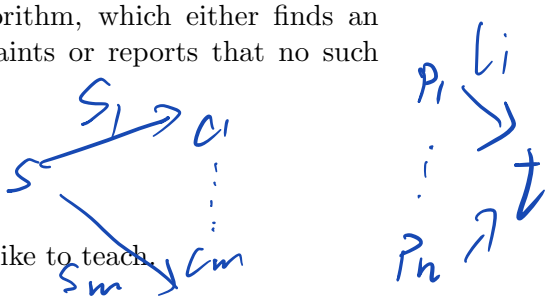
The key observation is that the Ford-Fulkerson algorithm can start from any given flow and augment it to a maximum flow. The only way the maximum flow can increase by increasing the capacity of a single edge e is if G_f does not have a forward edge for e, and by adding such an edge, a new s-t path is obtained. Further, since we are increasing the capacity of e by one, the added forward edge will also have capacity 1. Thus, when we find an s-t path, its bottleneck capacity will be 1. Thus, augmenting a unit flow along this path will remove the forward edge for e, and thus leave no further s-t paths in the residual graph.

The running time is linear because constructing the residual graph and then finding a path from s to t in this graph both take linear time.

Q3 Teaching Assignment

Suppose there are m courses: $c_1, \ldots, c_m$. For each $j \in \{1, \ldots, m\}$, course c_j has s_j sections. There are n professors: $p_1, \ldots, p_n$. For each $i \in \{1, \ldots, n\}$, professor p_i has a teaching load of ℓ_i and likes to teach the subset of courses $A_i \subseteq \{c_1, \ldots, c_m\}$.

Your goal is to use the network flow paradigm to design an algorithm, which either finds an assignment of professors to courses satisfying the following constraints or reports that no such assignment exists.



- Each professor p_i must be assigned exactly ℓ_i courses.
- Each course c_j must be assigned to exactly s_j professors.
- No professor should be be assigned a course that they do not like to teach.
- No professor can teach multiple sections of the same course.

(a) Describe your full algorithm. That is, describe the network flow instance created (nodes, edges, and edge capacities) and how your algorithm uses a maximum flow in this instance to determine if a valid assignment of professors to courses exists, and output one if it does.

(b) Prove that your reduction is correct. That is, prove that there exists a valid assignment of professors to courses if and only if your algorithm finds one.

(c) What is the worst-case running time of your full algorithm if you use the naïve Ford-Fulkerson algorithm to solve the network designed in part (a)?

Solution to Q3

(a) Create a flow network instance with the set of vertices $\{s, t, p_1, \ldots, p_n, c_1, \ldots, c_m\}$ and the following edges:

- (s, p_i) with capacity ℓ_i, for each p_i;
- (c_j, t) with capacity s_j, for each c_j;
- (p_i, c_j) with capacity 1, for each p_i and c_j such that p_i prefers to teach c_j (i.e. $c_j \in A_i$).

The full algorithm is as follows.

1. Compute a maximum flow f in the above instance via the Ford-Fulkerson algorithm. Note that it will return an integral flow.
2. If $f(s, p_i) = \ell_i$ for each professor p_i and $f(c_j, t) = \ell_j$ for each course c_j, then a feasible assignment exists. Assign professor p_i to teach course c_j whenever $f(p_i, c_j) = 1$.
3. Otherwise, report that a feasible assignment does not exist.

(b) We will say that an integral flow f in the above network is "saturated" if $f(s, p_i) = \ell_i$ for each professor p_i and $f(c_j, t) = \ell_j$ for each course c_j. To prove that the algorithm is correct, we will establish a 1-1 correspondence between saturated integral flows f and feasible assignments obtained; specifically, the correspondence will be the following: professor p_i teaches course $c_j \iff f(p_i, c_j) = 1$. We will prove this in both directions separately. (Note: In this problem, it is possible to directly establish an "if and only if" argument to prove both directions together, but this needs to be done carefully. In general, it will be safer to prove both directions separately.)

Saturated integral flow ⇒ valid assignment: First, take any saturated integral flow f. Construct the corresponding assignment according to the above correspondence. We want to prove that this is a valid assignment. Note that because f is saturated, each p_i has an incoming flow of ℓ_i and each c_j has an outgoing flow of s_j. Since the edges from professors to courses have unit capacity and the flow is integral, each p_i has ℓ_i outgoing edges carrying a unit flow (and the rest carrying no flow) and each c_j has exactly s_j incoming edges carrying a unit flow (and the rest carrying no flow). Hence, by our correspondence, we have that each p_i is assigned to teach a single section each in exactly ℓ_i courses of their liking and each course is assigned exactly s_j professors, which implies that the assignment is valid.

Valid assignment ⇒ saturated integral flow: Conversely, take any valid assignment and construct a saturated flow f according to the above correspondence. By the construction, it is clear that f is integral. Further, since each professor p_i is teaching a single section each in ℓ_i courses and each course is assigned s_j professors, each p_i has an outgoing flow of ℓ_i and each c_j has an incoming flow of s_j. Hence, setting $f(s, p_i) = \ell_i$ for each p_i and $f(c_j, t) = s_j$ for each c_j satisfies the flow conservation constraints, generating a valid and saturated flow.

Thus, we have established that our algorithm, which checks the existence of a saturated integral flow and converts it to a valid assignment using the above correspondence, is correct.

(c) This flow network instance has $m+n+2$ vertices and $m+n+\sum_{i=1}^{n}|A_i| \leq m+n+m\cdot n$ edges. The sum of capacities of edges leaving s is $\sum_{i=1}^{n}\ell_i \leq \sum_{i=1}^{n} m = n\cdot m$.[1] Hence, the worst-case running time of the naïve Ford-Fulkerson algorithm is $O(n^2\cdot m^2)$, which is polynomial in the input length.

[1]Technically, we are not given that $\ell_i \leq m$ for each i, but it is clear that this is necessary for a feasible solution to exist. Hence, we can reasonably imagine that we will be given an input satisfying this constraint. Alternatively, you can add an explicit preprocessing step in the algorithm, which returns "No feasible solution exists" if it is violated.