

CS375 Weekly Assignment 9

Due April 29nd 10PM ET

A general note for CS375: When writing up your homework, please write neatly and explain your answers clearly, giving all details needed to make your answers easy to understand. Graders may not award full credit to incomplete or illegible solutions.

You are encouraged to discuss the problems with your classmates, **but your submission should be your own**. When writing your solutions up, you should not be looking at any other student's submission.

Clear communication **is the point**, on every assignment. In general in CS375, unless explicitly specified otherwise, answers should be accompanied by explanations. Answers without explanations may not receive full credit. Please feel free to ask me any questions about explanations that might come up!

Necessary acknowledgements: many of these questions are reproductions or adaptations of questions of Eric Aaron's devising. Others are from our course texts as listed on the syllabus.

Recall the MAXFLOW problem from class:

MAXFLOW

Input: A graph $G = (V, E)$ with vertices V and directed edges E with capacities $c(u, v) \geq 0$ for each edge (u, v) in E and two marked vertices s and t (we'll assume there are no edges into s or out of t).

Output: The maximum flow on the graph.

Constraints:

1. For each vertex v other than s and t it holds that $\sum_u f(u, v) = \sum_w f(v, w)$:
ie, the flow into all vertices equals the flow out.
2. For each edge (u, v) in E it holds that $f(u, v) \leq c(u, v)$.

In order to define the Ford-Fulkerson algorithm in class, we defined the residual graph of G with respect to f as $G_f = (V, E_f)$ where $c_f(u, v) = c(u, v) - f(u, v)$, $c_f(v, u) = f(u, v)$, and E_f is all the edges with positive residual capacity c_f (note that this may remove some edges from E and may include some reverse edges).

We defined the Ford-Fulkerson algorithm as follows:

Ford-Fulkerson($G = (V, E)$, s , t):

 initialize $f(u, v) = 0$ for all (u, v) in E

 while there exists a path P from s to t in G_f :

 let (u, v) be the edge of minimal residual capacity in P

 Increase the flow f by $c_f(u, v)$ for every edge in P .

We also considered the following problem:

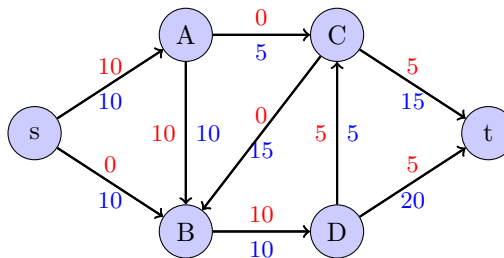
MINCUT

Input: A graph $G = (V, E)$ with vertices V and directed edges E with capacities $c(u, v) \geq 0$ for each edge (u, v) in E and two marked vertices s and t (we'll assume there are no edges into s or out of t).

Output: The minimum s - t cut in the graph, where an s - t cut is a partition of the vertices of G into two sets S and T such that s is in S and t is in T . The value of an s - t cut (S, T) is defined as $\sum_{u \in S} \sum_{v \in T} c(u, v)$, the sum of the capacities of edges going from S to T .

and showed that the maximum flow on a graph is equal to the minimum cut. In fact, we showed that technically the FORDFULKERSON algorithm was capable of determining the maximum flow and minimum cut, effectively designing an algorithm for both problems simultaneously.

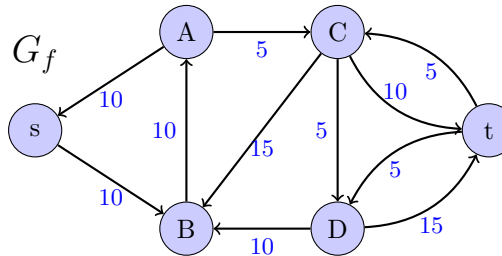
- (a) Consider the following graph with capacities marked in blue, on which I've already pushed some flow (in red):



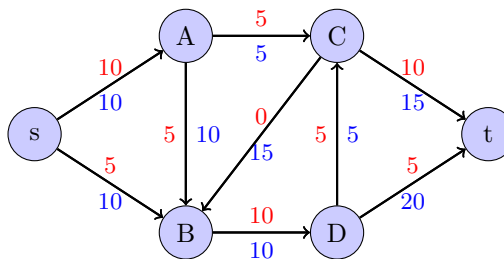
Using this as a starting point (so don't reset the flow to 0), enter the while loop in the FORDFULKERSON algorithm, trace through what happens, and determine the max flow.

- (b) Identify an S-T cut with cut-capacity equal to the maximal flow.

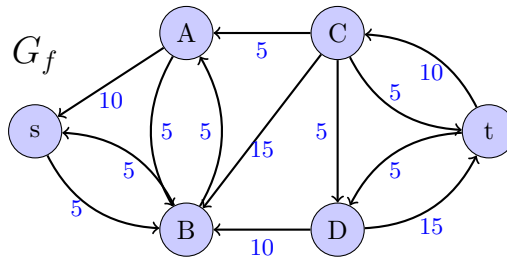
Solution: First, here's our residual graph:



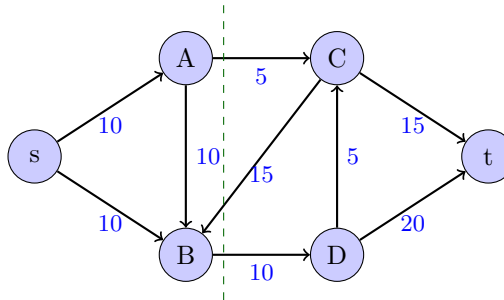
The path (s, B, A, C, t) is a path from s to t in G_f whose minimal capacity is 5: so we will augment our flow in G along this path by 5, giving us the following flow:



Our new residual graph is as follows:



Since we can see there's no path from s to t , Ford-Fulkerson terminates. From class, we know that this must be a maximum flow. To find the corresponding min cut, consider the following cut:



This corresponds to the cut (S, T) where $S = \{s, A, B\}$ and $T = \{C, D, t\}$: the edges *from* S to T only include (A, C) and (B, D) , the capacities of which are 5 and 10 respectively, giving us a cut of value 15. Since we have a cut of value 15 and a flow of value 15, we know that the cut is minimal and the flow is maximal.

2. Consider the following problem:

CIRCULATION

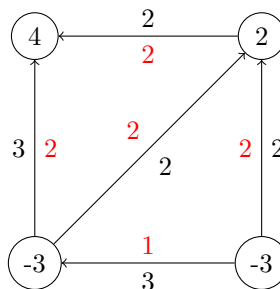
Input: A directed graph G where each edge $e = (u, v)$ has an integer capacity $c(u, v)$ and each vertex v has a *demand* d_v .

Output: TRUE if there is a way to set flow values on $f(u, v)$ to handle the demands of each vertex, otherwise FALSE.

Constraints:

- The flow on each edge is at most the capacity: $f(u, v) \leq c(u, v)$ for all edges (u, v) .
- The demands on each vertex v are satisfied: $\sum_u f(u, v) - \sum_w f(v, w) = d_v$.

As an example:



The bottom left node has a flow of 1 entering and a flow of 4 leaving, satisfying its demand of -3 .

Show that $\text{CIRCULATION} \leq_p \text{MAXFLOW}$.

Solution: Given an input G for CIRCULATION, we will compute an equivalent input for MAXFLOW. To make an input for MAXFLOW, we need to make a graph with capacities and two marked vertices s and t : a bunch of that was already given to us as input (G). The only thing G is missing are s and t ; so to make our input $\tilde{G}$ for MAXFLOW, we will

- (a) add two vertices s and t to G
- (b) add an edge from s to every vertex v whose demand is negative with a capacity of $|d_v|$
- (c) add an edge from every vertex u whose demand is positive to t with a capacity of d_u

By our construction, there is a valid circulation flow on G if and only if the maximum flow f^* on $\tilde{G}$ satisfies $f^* = \sum_{v:d_v < 0} |d_v|$ and $f^* = \sum_{u:d_u > 0} d_u$ (basically, that the maximum flow is equal to both the total *supply* of flow and *demand* of flow).

First, suppose that there is a valid flow f for the circulation graph G . Then we can transform that flow into a flow $\tilde{f}$ on $\tilde{G}$ by pushing $|d_v|$ flow out of every s into each vertex v of negative demand and pushing d_u from u to t for every vertex u of positive demand. Consider the net flow through each vertex other than s and t :

- First, think about the vertices v whose demand was negative. Since the circulation was valid, we know that $\sum_w f(w, v) - \sum_w f(v, w) = d_v$ for the flow in G . But in $\tilde{G}$, we've just increased the flow into v by $|d_v|$, so for $\tilde{f}$ we get $\sum_w \tilde{f}(w, v) - \sum_w \tilde{f}(v, w) = d_v + |d_v| = 0$.
- Next, look at the vertices u whose demand was positive. Since the circulation was valid, we know that $\sum_w f(w, u) - \sum_w f(u, w) = d_u$ for the flow in G . But in $\tilde{G}$, we've just increased the flow leaving u by d_u , so for $\tilde{f}$ we get $\sum_w \tilde{f}(w, u) - \sum_w \tilde{f}(u, w) = d_u - d_u = 0$.
- We didn't touch the vertices v of demand 0, so the net flow through them is still 0.

As a result, the net flow through each vertex is 0, which means $\tilde{f}$ is a valid flow for the MAXFLOW problem. Not only that, every edge out of s and every edge entering t is fully saturated with flow, so $\tilde{f}$ is definitely a max flow with value $\sum_{v:d_v < 0} |d_v| = \sum_{u:d_u > 0} d_u$.

On the other hand, if the maximum flow $\tilde{f}$ in $\tilde{G}$ has value $\sum_{v:d_v < 0} |d_v| = \sum_{u:d_u > 0} d_u$, then we know every edge out of s and every edge into t is fully saturated. So consider the flow f in G calculated by copy-pasting the flows from the corresponding edges in $\tilde{G}$. Like above, let's think about the net flow through each vertex again:

- First, think about the vertices v whose demand was negative. Since the flow was valid in $\tilde{G}$, we know that $\sum_w \tilde{f}(w, v) - \sum_w \tilde{f}(v, w) = 0$ in $\tilde{G}$. But in G , we've just decreased the flow into v by $|d_v|$, so for f we get $\sum_w f(w, v) - \sum_w f(v, w) = 0 - |d_v| = d_v$.
- Next, look at the vertices u whose demand was positive. Since the flow was valid in $\tilde{G}$, we know that $\sum_w \tilde{f}(w, u) - \sum_w \tilde{f}(u, w) = 0$ in $\tilde{G}$. But in G , we've just decreased the flow leaving u by d_u , so for f we get $\sum_w f(w, u) - \sum_w f(u, w) = 0 - d_u = d_u$.
- We didn't touch the vertices v of demand 0, so the net flow through them is still 0.

As a result, the flow in f satisfies the circulation constraints and we get a valid circulation flow in G .

Letting n be the number of vertices in G and m be the number of edges, note that our construction of $\tilde{G}$ takes $O(n)$ time to produce (assuming we just alter G - if we need to make a new graph object, then this would become $O(n + m)$ time since we have to iterate over the entire original graph), since we are just making two vertices s and t and connecting them to the vertices of G . The number of vertices in $\tilde{G}$ is $n + 2$ and the number of edges is $m + n'$ where $n' \leq n$ is the number of vertices of non-zero demand. Hence, if we have an algorithm for MAXFLOW with a runtime of $f(n, m)$ then our induced algorithm for CIRCULATION has a runtime of $O(n) + f(n + 2, m + n')$.

3. Consider the following problem:

LOWERBOUNDEDCIRCULATION

Input: A directed graph G where each edge $e = (u, v)$ has an integer capacity $c(u, v)$ and an integer lowerbound $\ell(u, v)$, and each vertex v has a *demand* d_v .

Output: TRUE if there is a way to set flow values on $f(u, v)$ to handle the demands of each vertex, otherwise FALSE.

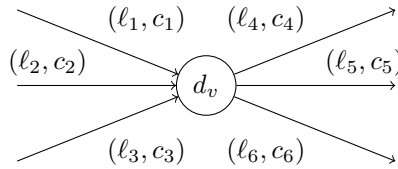
Constraints:

- The flow on each edge is at least the lowerbound and at most the capacity:
 $\ell(u, v) \leq f(u, v) \leq c(u, v)$ for all edges (u, v) .
- The demands on each vertex v are satisfied: $\sum_u f(u, v) - \sum_w f(v, w) = d_v$.

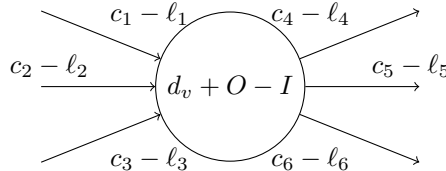
- (a) Show that $\text{LOWERBOUNDEDCIRCULATION} \leq_p \text{CIRCULATION}$, which in turn means that $\text{LOWERBOUNDEDCIRCULATION} \leq_p \text{MAXFLOW}$.
- (b) Suppose I have an algorithm for MAXFLOW with a runtime of $f(n, m)$ where n is the number of vertices and m is the number of edges in the graph given as input. Using the reductions you've determined so far, what would be the resulting runtime of the induced algorithm for LOWERBOUNDEDCIRCULATION?

Solution: Given a graph G for LOWERBOUNDEDCIRCULATION we'll compute an equivalent graph $\tilde{G}$ as input for CIRCULATION. The only difference between these two problems as inputs are the lowerbounds: in order for a flow to be valid in G , each edge's flow must be at least the lowerbound. Since the regular CIRCULATION problem doesn't have these lower bounds, we need to convert the input. One way to do this is to 'start' with a flow of the lowerbounds on every edge, then update the demands and capacities to ask "what do we need to change the flow by to handle the constraints".

So imagine we put our magnifying glass on G and found a vertex that looks something like the following:



Let's ponder what this picture should look like in $\tilde{G}$. First, since we're assuming each edge already has a flow of the given lowerbounds, the 'residual' capacity of each edge should really just be $c_i - \ell_i$. For the demand of the vertex, we're assuming that $I = \sum_{i=1}^3 \ell_i$ flow is already flowing into v and $O = \sum_{i=4}^6 \ell_i$ flow is already flowing out of v : this means that the *remaining* demand of v is $d_v + O - I$. So our updated picture for $\tilde{G}$ should look like:



More generally, to determine the capacities and demands in $\tilde{G}$:

- For each edge (u, v) , define its capacity $\tilde{c}(u, v) = c(u, v) - \ell(u, v)$
- For each vertex v , define its demand as $\tilde{d}_v = d_v + \sum_w \ell(v, w) - \sum_u \ell(u, v)$

To show that this works:

- If the graph G has a valid flow f that satisfies the constraints of LOWERBOUNDEDCIRCULATION, then consider the flow $\tilde{f}$ defined by $\tilde{f}(u, v) = f(u, v) - \ell(u, v)$. By validity of the flow f , it must be the case that $f(u, v) \geq \ell(u, v)$ for all edges (u, v) , so $\tilde{f}(u, v) \geq 0$. Since we've set the capacity $\tilde{c}(u, v) = c(u, v) - \ell(u, v)$, we also get that $\tilde{f}(u, v) = f(u, v) - \ell(u, v) \leq c(u, v) - \ell(u, v) = \tilde{c}(u, v)$. Thus our flow $\tilde{f}$ doesn't break any rules of CIRCULATION for any of the edges - but we still need to make sure the demands are satisfied. Since f was assumed valid, we know that $\sum_u f(u, v) - \sum_w f(v, w) = d_v$ for every vertex v in G . In $\tilde{G}$, we get that

$$\begin{aligned} \sum_u \tilde{f}(u, v) - \sum_w \tilde{f}(v, w) &= \sum_u (f(u, v) - \ell(u, v)) - \sum_w (f(v, w) - \ell(v, w)) \\ &= \sum_u f(u, v) - \sum_w f(v, w) + \sum_w \ell(v, w) - \sum_u \ell(u, v) \\ &= d_v + \sum_w \ell(v, w) - \sum_u \ell(u, v) \\ &= \tilde{d}_v \end{aligned}$$

So every vertex's demand is also satisfied by this flow! So we've found a flow that satisfies all the constraints for CIRCULATION.

- (This direction is going to look exactly like the above, but in reverse) On the other hand, suppose I have a flow $\tilde{f}$ for $\tilde{G}$ that satisfies the constraints for CIRCULATION. Consider the flow f defined on G by setting $f(u, v) = \tilde{f}(u, v) + \ell(u, v)$. Since $\tilde{f}(u, v) \geq 0$, this immediately gives that $f(u, v) \geq \ell(u, v)$. Additionally, since $\tilde{f}(u, v) \leq \tilde{c}(u, v) = c(u, v) - \ell(u, v)$, we get that $f(u, v) = \tilde{f}(u, v) + \ell(u, v) \leq \tilde{c}(u, v) + \ell(u, v) = c(u, v) - \ell(u, v) + \ell(u, v) = c(u, v)$, so the flow on each edge in G satisfies the rules of LOWERBOUNDEDCIRCULATION. Next we'll check the demands: note that

$$\begin{aligned} \sum_u f(u, v) - \sum_w f(v, w) &= \sum_u (\tilde{f}(u, v) + \ell(u, v)) - \sum_w (\tilde{f}(v, w) + \ell(v, w)) \\ &= \sum_u \tilde{f}(u, v) - \sum_w \tilde{f}(v, w) + \sum_u \ell(u, v) - \sum_w \ell(v, w) \\ &= \tilde{d}_v + \sum_u \ell(u, v) - \sum_w \ell(v, w) \\ &= d_v + \sum_w \ell(v, w) - \sum_u \ell(u, v) + \sum_u \ell(u, v) - \sum_w \ell(v, w) \\ &= d_v \end{aligned}$$

so the demand for each vertex is also met! As a result, f is a valid flow for G in the LOWERBOUNDEDCIRCULATION problem.

At this point, we can conclude that there is valid flow on $\tilde{G}$ for the CIRCULATION problem exactly when there is a valid flow on G for the LOWERBOUNDEDCIRCULATION problem, so these inputs are indeed equivalent.

In the construction of $\tilde{G}$, we didn't add any new vertices nor any new edges. However, we did have to iterate over every edge and vertex. Updating each edge's capacity only takes constant time, but each updated demand for each vertex requires iterating over the edges its incident to. So this means creation of the graph $\tilde{G}$ takes $O(n^2 + m)$ time¹. To plug it into an algorithm for MAXFLOW, we need to take $\tilde{G}$ and apply the transformation of the previous problem's algorithm, which takes another $O(n + m)$ time (and adds 2 vertices and $O(n)$ edges). As a result, the total runtime of the algorithm for LOWERBOUNDEDCIRCULATION induced by our combined reduction is $O(n^2 + m) + f(n + 2, m + O(n))$.

¹this can be improved, but I'd be okay with this as an answer