Yeury Galva Liriano

CSC 440

Assignment 6: Network Flow

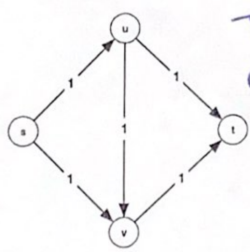
Due Tuesday, April 21^{st} by 3:30PM

You may work in small groups on this assignment, but the work you hand in must be your own. You may submit this in one of two ways:

- Write it up electronically (e.g. LaTeX) and upload a PDF to Gradescope.
- Scan your handwritten solution (you can use a scanning app for a smartphone, such as Evernote's free Scannable app or the Notes app in iOS 11) and upload a PDF to Gradescope.
- Solutions handed in on paper **will not be accepted**

Your full name:

1. (10 points) List all the minimum cuts in the following flow network (the edge capacities are the numbers on each edge)

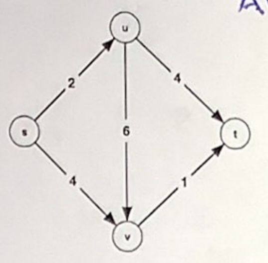


There's exactly 4 possible cuts in the graph. They are: a) {s}, {u, v, t}
b) {s, u}, {v, t}
c) {s, v}, {u, t} and
D) {s, u, v}, {t}

capacities of these 4 possible cuts are:
{s}, {u, v, t} => 2
{s, u}, {v, t} => 3
{s, v}, {u, t} => 2
{s, u, v}, {t} => 2
So, there are 3 minimun cuts in this graph.

They are:
i) {s}, {u, v, t}
ii) {s, v}, {u, t}
iii) {s, u, v}, {t}

2. (10 points) What is the minimum capacity of an (s, t) cut in the following network?



All possible cuts:

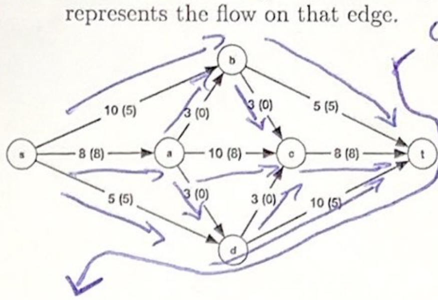
a> {s}, {u,v,t} => 2+6+1 = ~~10~~ 9

b> {s,u}, {v,t} => 2+~~2~~1 = ~~4~~3 ← minimum

c> {s,v}, {u,t} => 4+4 = 8

D) {s,u,v}, {t} => 2+6+~~4~~1 = ~~10~~ 9

3. (20 points) In the following network, a flow has been computed. The first number on each edge represents capacity, and the number in parentheses represents the flow on that edge.



a) Flow for s-a-c-t is 8
Flow for s-d-t is 8

Path	Flow
s-b-t	5
s-a-c-t	8
s-d-t	8

a. (10 points) What is the value of the flow that has been computed? Is it a maximum (s, t) flow? Total value of flow 5+8+8 = 21
The value computed for (s,t) flow i.e 21 is maximum because the capacities of edges is full and hence no other path of flow is possible.

b. (10 points) Find a minimum (s, t) cut on the network, state which edges are cut, and state its capacity.

Edge	Cut value/capacity
b-t	5
s-a	8
s-d	8

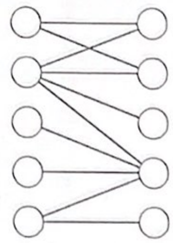
Following are steps to find all edges of the minimum cut.

1) Run Ford-Fulkerson algorithm and consider the final ~~graf~~ residual graph.

2) Find the set of vertices that are reachable from the source in the residual graph.

3) All edges which are from a reachable vertex to non-reachable vertex are minimum cut edges. Print all such edges.

4. (25 points) At the beginning of the semester, you implemented an algorithm for finding a perfect matching based on a preference list on a complete bipartite graph. Here, consider a different problem: given a bipartite graph that isn't necessarily complete, determine the maximal matching (and if that matching is perfect). Note that here there are no preference lists; there are only vertices and edges. So, we are not worried about *stability*. Consider the following bipartite graph.



- edges capacities are 1
- edges go from left to right
- Add source node s and sink node t

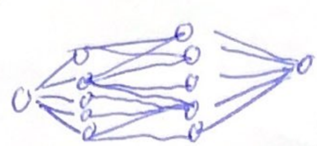
a. (15 points) How might you use the Ford-Fulkerson Maximum Flow algorithm to determine whether or not there is a perfect matching? Hints: flow networks need edge capacities, so you'll need to add some. And, flow networks use directed edges, so you'll need to make this a directed graph. You may need some other changes as well.

Since all edge capacities are 1, it would be useful to use F-F Alg to find max flow;

max flow essentially uses the same edges used in a maximal matching.

b. (10 points) On the example graph shown, what is the size of a maximum matching? Is it a perfect matching?

The max flow is 5 because all capacities set are 1, so essentially min cut would also be 5.



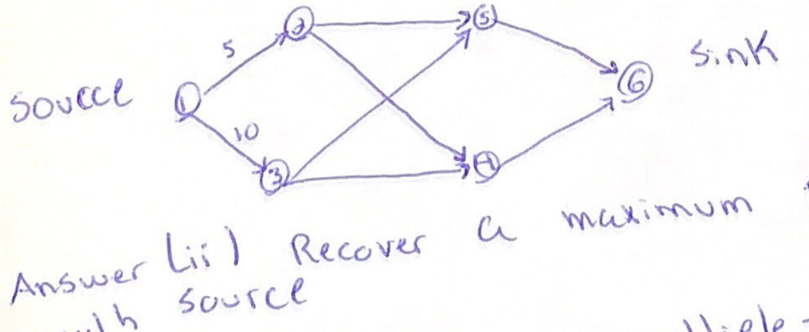
5. (25 points) Suppose we generalize the *max-flow* problem to have multiple source vertices $s_1, ...s_k \in V$ and sink vertices $t_1, ..., t_l \in V$. Assume that no vertex is both a source and a sink, the source vertices have no incoming edges, and sink vertices have no outgoing edges, and that all edge capacities are still integral. A flow is still defined as a nonnegative integer f_e for each edge $e \in E$ such that capacity constraints are obeyed on every edge, and conservation constraints hold at all vertices that are neither sources nor sinks. The value of a flow is the total amount of outgoing flow at the sources $\sum_{i=1}^{k} \sum_{e \in \delta^+(s_i)} f_e$. Prove that *max-flow* with multiple sources and sinks can be reduced to the single-source single-sink version of the problem. Specifically, given an instance of multi-source multi-sink *max-flow*, show how to (i) produce a single-source single-sink instance that lets you (ii) recover a maximum flow of the original multi-source. Sketch out a proof of correctness, and that your algorithms run in linear time (not counting the time required to solve *max-flow* on the single-source single-sink instance). Hint: consider adding additional vertices or edges to the graph.

Answer (i)

Mare-flow Problem : Single - Source Single - Sink (A, B, C)
Let assume that directed capacitated network with a Sink connecting a Source (ie, origin) node
(ie, destination) node

Set 'A' is the set of nodes in the network
Set 'B' is the set of directed Links (i, j)
Set 'C' is the set of capacities $C_{ij} \geq 0$ of the Links (i, j) $\in B$

then we can determine the maximum amount of flow that can be sent from the source node to sink node.



This is Max-Flow Problem for single-source and single-sink

Answer (ii) Recover a maximum flow of the original multi source

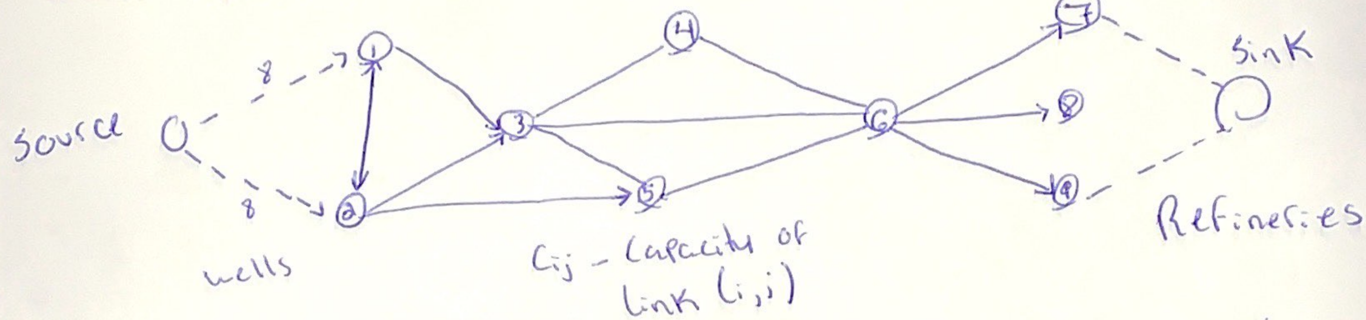
in fact, multiple-source, multiple-links Problems can be converted to a single source of single-sink problems. By using dummy source node.

5 → Continuation

Let assume that a dummy source node that is connected to the original to the original source nodes with infinite capacity links. at the same, a dummy sink node is connected with the original sink nodes

Space for problem 5

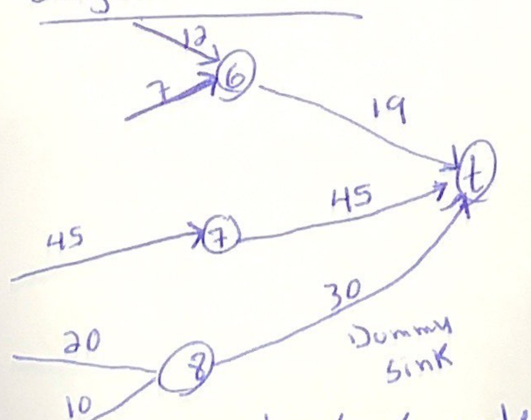
with infinite capacity links.



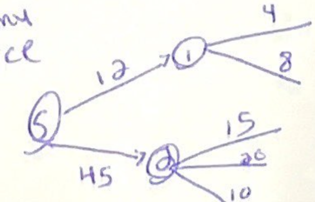
From the above figure, each outgoing link from the dummy source node to original source gets assigned a capacity, that is equal to the total capacity of the outgoing link. similarly, these same procedure is going on the multiple ~~sink~~ nodes and source nodes.

Each incoming sink from and original sink node to the dummy sink node gets assigned a capacity is equal to the total capacity of the incoming links to sink. As shown in figure ... original source nodes

original sink nodes



t = sink node



S = source node

6. (10 points) Describe a real-world problem, not yet discussed in class, that is amenable to *max-flow* or *min-cut*. How would you represent the problem in terms of max-flow or min-cut? Given the various asymptotic complexities of the algorithms for *max-flow* discussed so far, which algorithm might be most appropriate, and why?

Imagine there's a puddle in a yard that moves into a stream using Ford-Fulkerson algorithm, if somehow a min cut is found, it represents the max flow of water from puddle to stream. A min cut would essentially represent the puddle as some type of graph from puddle to stream... This kind of problem would be best to solve using Ford-Fulkerson algorithm.