CSC373 Week 4 Notes

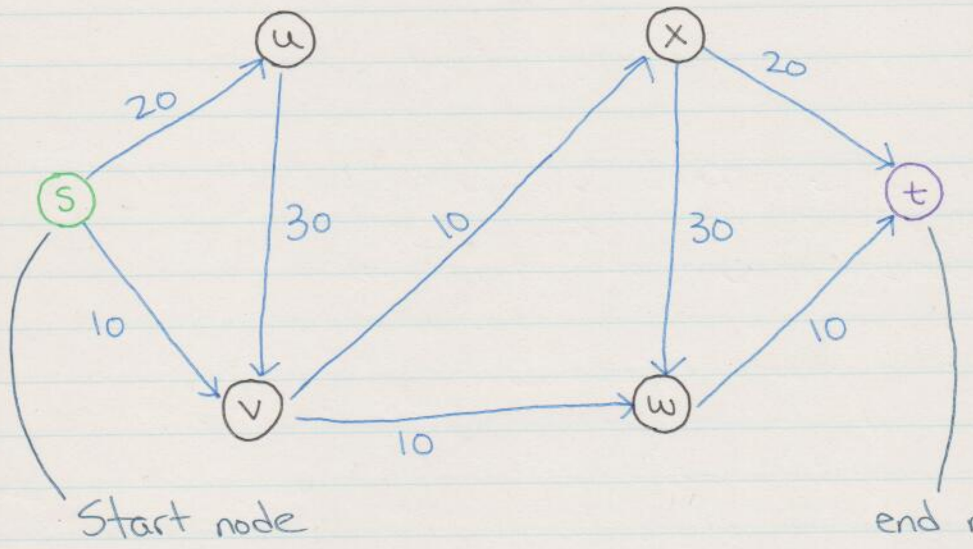
Introduction to Network Flow:

- Input:
  1. A directed graph $G = (V, E)$.
  2. Edge capacities that are non-negative.
  3. A source node, s, and a target node, t.

- Output: Maximum "flow" from s to t.

- Assumptions:
  1. No edges enter s
  2. No edges leave t
  3. Each edge capacity is non-negative (≥ 0)

- E.g.



Note: Treat the edge capacities like the max amount of stuff/weight that can go on the edge at a single time.

- The flow is the amount of stuff/weight carried over an edge.

- The flow is always between 0 and the capacity of the edge, inclusive.

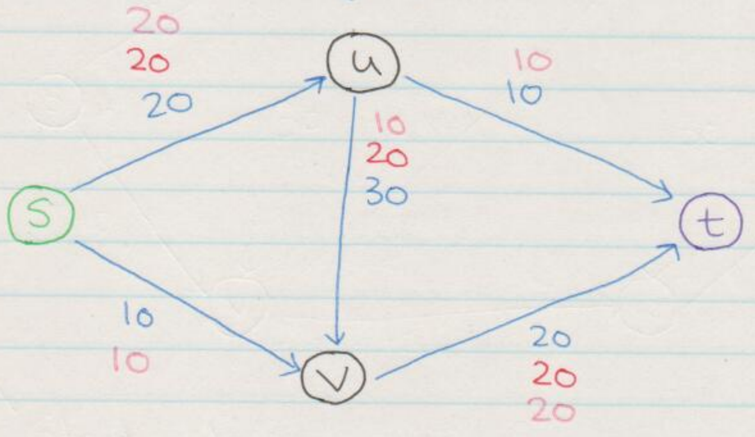
  I.e. $0 \le f(e) \le c(e), \; \forall e \in E$

  $\uparrow$ flow $\qquad \uparrow$ edge capacity

- For all nodes except for s and t, the sum of the flow entering that node equals the sum of the flow leaving it.

  I.e. $\forall v \in V \setminus \{s,t\}, \; \sum_{e \text{ entering } v} f(e) = \sum_{e \text{ leaving } v} f(e)$

- Consider the graph below:



I will use red to denote the numbers for the first example and pink for the second example.

Blue numbers mean the max capacity of that edge.

In the first example, we put 20 on the edge (s,u), then 20 on the edge (u,v) and 20 on the edge (v,t). However this is not the most optimal solution.

Note: The most optimal soln is the min between the total capacity of all the edges flowing from s or the total capacity of all the edges flowing into t.

In my second example, we do get an optimal solution. This is because the total amount flowing into t is 30, which is the same as both the total capacity of edges flowing out of s or the total capacity of edges going into t.

- We can use a residual graph to reverse bad decisions.

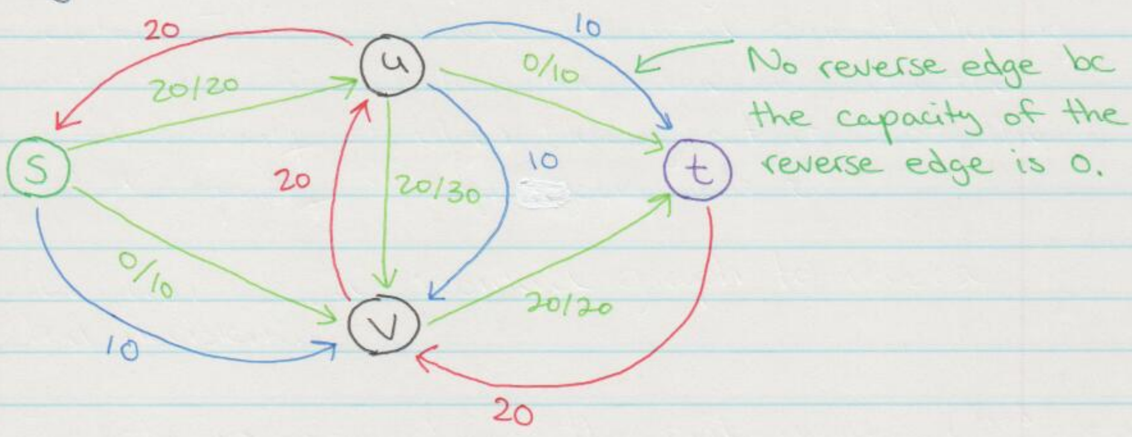
- In the example above, the pink numbers (2nd soln) was the residual graph.

Residual Graph:
- Let c(e) denote the capacity for edge e.
- Let f(e) denote the flow of edge e.
- Suppose the current flow is f.
  Let G_F be the residual graph of f.
  G_F has the same vertices as f
  For each edge e=(u,v) in f, G_F has at most 2 edges:
    1. Forward Edge
       - e = (u,v)
       - Capacity = c(e) − f(e)
       - How much more flow we can send on this path

2. Reverse Edge
   - $e^{rev} = (v,u)$
   - capacity = $f(e)$
   - This is the max reverse flow. Since the original flow along edge e is $f(e)$, the max we can send back is $f(e)$.

Note: We only add edges where the capacity is greater than 0.

- E.g.



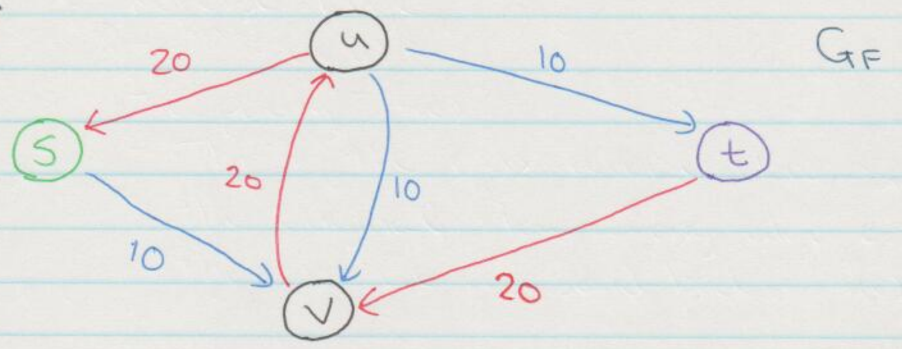
The red lines are the reverse edges and the blue lines are the forward edges.

Notice that some edges only have a forward edge or a reverse edge. This is bc the other edge has a capacity of ≤0, so we don't add it.

- Let P be an S-T path in G_F.
- Let bottleneck(P, F) be the smallest capacity across all edges in P.
- We can augment f by sending bottleneck(P, F) units of flow along P.

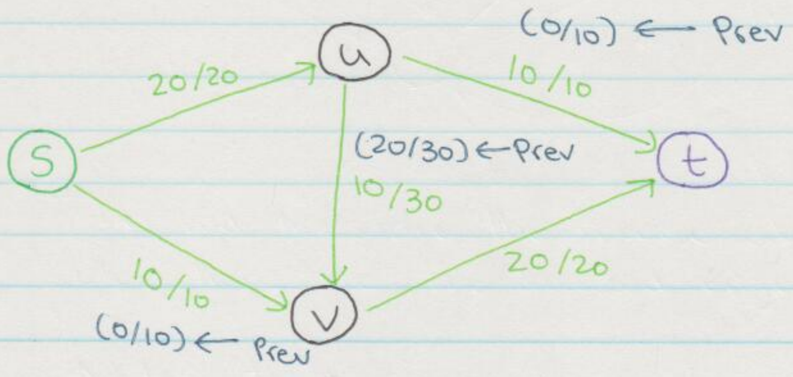
Note: Sending x units of flow along P means:
1. For each forward edge in P, inc the flow by x.
2. For each reverse edge in P, dec the flow by x.

- E.g.



We see that (s,v) and (u,t) are bottlenecks as they have the smallest capacity (10). Hence, we will increase the flow by 10 along all forward edges and decrease the flow by 10 along all reverse edges.

New flow f:



We added/increased the flow from (s,v) and (u,t) by 10 and decreased the flow from (u,v) by 10.

- Now, we'll prove that the new flow is valid:
  1. Capacity Constraints:
     - If we increase flow on e, we can do so by at most the capacity of the forward edge e in G_F which is $c(e) - f(e)$.

       $$f(e) + (c(e) - f(e)) = c(e) \leftarrow \text{Max new flow}$$

     - If we decrease flow on e, we can do so by at most the capacity of the reverse edge e^{rev} in G_F, which is $f(e)$. Hence, the new flow is at least $f(e) - f(e)$ or 0.

  2. Flow Conservation:
     - Each node except for s and t has 2 incident edges.

     - If they are both forward/both reverse, that means one is incoming, one is outgoing. The flow is increased on both or decreased on both. The net flow is 0.

     - If one is forward and the other reverse then we get both incoming or both outgoing. Flow is increased on one and decreased on the other. The net flow is 0.

Ford - Fulkerson Algorithm:

```
Max Flow (G):

    // Initalize
    set f(e) = 0 for all e in G

    // while there's an s-t path in GF:
    While P = FindPath (s, t, Residual (G, f)) != None:
          f = augment (f, P)
          UpdateResidual (G, f)

    return f
```

- Running time: $O((m+n) \cdot C)$

The max flow / max num of augmentations is at most $C = \sum_{e \text{ leaving } s} c(e)$

For path P in G_F, because bottleneck $(P, f) \geq 1$, each augmentation increases flow by at least 1. That's why the max flow is at most $\sum_{e \text{ leaving } s} c(e)$.

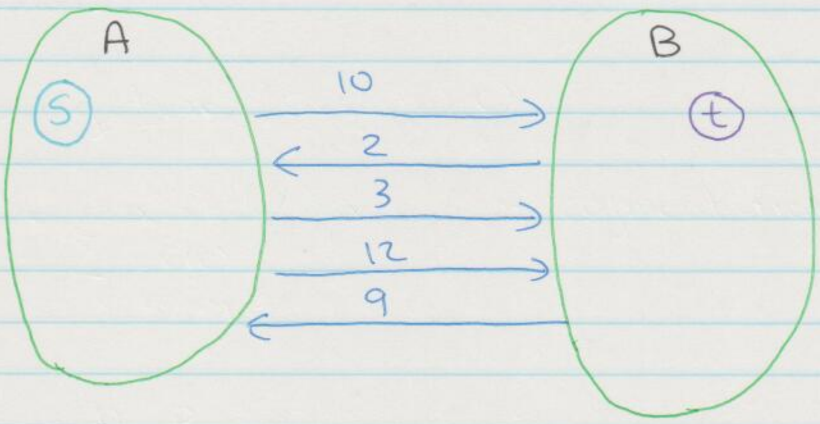
The time it takes to perform an augmentation is $O(m+n)$. This is bc G_F has n vertices and at most 2m edges. So finding P, computing bottleneck (P, f) and updating G_f takes $O(m+n)$ time.

Proof of Optimality:

- Let (A, B) be an s-t cut. This means we partition the nodes into 2 groups, A and B. Furthermore, $s \in A$ and $t \in B$.
  $A \cup B = V$
  $A \cap B = \emptyset$

- Let cap(A, B) be the sum of the capacities of the edges leaving A.

- E.g.



$\text{Cap}(A, B) = 10 + 3 + 12$
$= 25$

- Thm: For any flow f and any s-t cut (A, B),
  $v(f) = f^{out}(A) - f^{in}(A)$

  Note: $v(f) = f^{out}(s) = f^{in}(t)$

  Proof:
  Edges that connect nodes in the same partition cancel each other out.
  Only edges that connect nodes in both partitions can deliver the flow to t.
  I.e. Consider nodes x and y s.t. they're in the same partition. Suppose the flow from x to y is e. That means y has to "release" e amount of flow. If it's to other nodes in the same partition, it doesn't directly lead to t, so it cancels out.
  Assume only x goes into y.

So, we're only interested in the edges that span both partitions.

Therefore, we need to sum the capacity of the edges that go from A to B and subtract the capacity of the edges that go from B to A.

We need to do the subtraction because the edges that go from B to A "return" some of the flow.

- Thm: For any flow f and any s-t cut (A,B), $v(f) \le cap(a,b)$

Proof:

$$\begin{aligned} v(f) &= f^{out}(A) - f^{in}(B) \\ &\le f^{out}(A) \\ &= \sum_{e \text{ leaving } A} f(e) \\ &\le \sum_{e \text{ leaving } A} c(e) \\ &= Cap(A,B) \end{aligned}$$

Hence, $\max_{f} v(f) \le \min_{(A,B)} cap(A,B)$

I.e. The max value of any flow $\le$ min capacity of any s-t cut

Implications:

1. Max flow = Min cut
2. Ford-Fulkerson generates max flow

- Thm: Ford-Fulkerson returns / finds the max flow

Proof:
f = flow returned by F-F
A^* = nodes reachable from s in G_F
$B^* = V \setminus A^*$

Note: We look at the residual graph G_F but define the cut in G

Note: The F-F algo terminates when there are no paths from s to t. (See pg. 7)

Claim: (A^*, B^*) is a valid cut.
$s \in A^*$ by definition
$t \in B^*$ because there's no path from s to t in G_F. Hence, $t \notin A^*$

Consider the following:

1. Each edge (u,v) going out of A^* in G must be Saturated $(c(e) = f(e))$. Otherwise, G_F would have its own forward edge (u,v) and $v \in A^*$.

   Recall: If edge (u,v) in G has a flow less than its capacity, then in G_F, there will be a forward edge with capacity $c(e) - f(e)$.

2. Each edge (v,u) coming into A^* in G must have 0 flow. Otherwise G_F would have its reverse edge (u,v) and then $v \in A^*$.

E.g. Suppose $A \in A^*$ and $B \in B^*$

$$\begin{tikzcd} \boxed{A} \arrow[r, "5/10"] & \boxed{B} \end{tikzcd}$$
G

, then in G_F:

Forward Edge
$$\begin{tikzcd} \boxed{A} \arrow[r, "5/10"] & \boxed{B} \end{tikzcd}$$

This means we can reach B from S, which means $B \in A^*$, contradicting our

$$\begin{tikzcd} B \arrow[r, "5/10"] & A \end{tikzcd}$$

G

, then in G_F: Reverse Edge

$$\begin{tikzcd} A \arrow[r, "5/10"] & B \end{tikzcd}$$

This means that we can reach B from S, contradicting our assumption.

Based on 1 and 2, we conclude:

$f^{out}(A^*) = cap(A^*, B^*)$

$f^{in}(A^*) = 0$

$\therefore\ v(f) = f^{out}(A^*) - f^{in}(A^*)$

$= cap(A^*, B^*)$