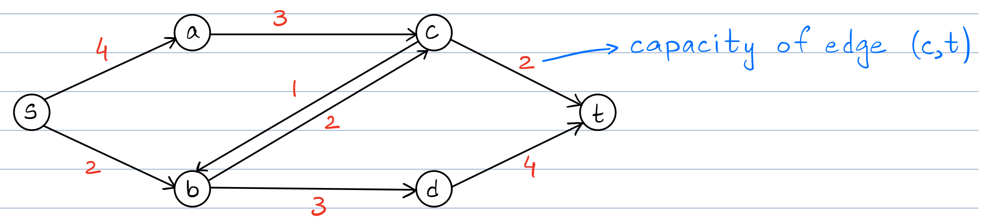
Network Flows

Given a directed Graph $G = (V, E)$, start node s, sink node t, each edge $e \in E$ has an associated non-negative capacity $c(e)$.

For all non-edges, capacity is 0.



Goal: Push max possible flow from s to t, subject to following constraints:

1. No flow on an edge can exceed capacity.

Capacity Constraint

For all $e \in E$, $f(e) \leq c(e)$

2. For all vertices, except s & t, flow in = flow out.

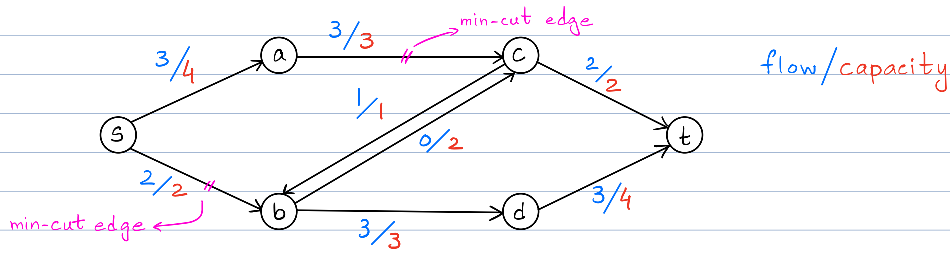
Flow Conservation

$$\text{For all } v \notin \{s,t\},\ \underbrace{\sum_{u \in V} f(u,v)}_{\text{flow in}} = \underbrace{\sum_{u \in V} f(v,u)}_{\text{flow out}}$$

Assume we send flow 2 across path s–b–c–t. This flow saturates the min capacity of the path.

However, it is a suboptimal choice since now we can only send flow 1 across path s–a–c–b–d–t.

The graph has max flow 5 given by



This flow saturates edges (a,c) & (s,b). Removing these edges disconnects s from t. In other words, these edges form an s-t cut of size 5.

So, max s-t flow $\leq$ capacity of min s-t cut

Max flow — Min Cut Theorem

Max s-t Flow = Capacity min s-t cut

Algorithm needs to find flow of value k & cut of capacity k.

Def 1

An s-t cut is a set of edges whose removal disconnects t from s.

OR

A partition of V into sets A & B, such that $s \in A$ & $t \in B$.
(Edges of cut are edges from A to B.)

Def 2 Capacity of a cut (A,B)
Sum of capacities of all edges going from A to B.

Def 3 For any edge (u,v), $f(u,v) = -f(v,u)$ (Skew-Symmetry)

1 flow on edge from u to v = -1 flow on edge from v to u.

This is a mathematical convenience. The back edge (v,u) need not exist in original graph.

If f & g are flows, then $h = f+g$ is given by

$$h(u,v) = f(u,v) + g(u,v) \quad \text{for all pairs } (u,v).$$

Ford Fulkerson Algorithm

Idea:

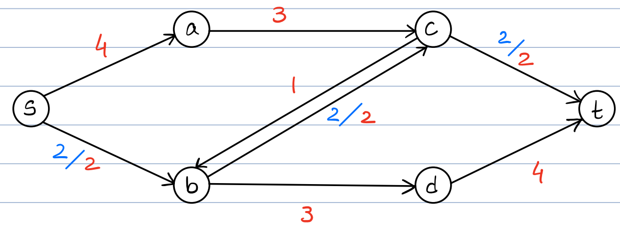
1. On an (s-t) path push max possible flow
2. Define residual capacities (maybe introduce back edges) on residual graph. Define?
3. Repeat till s & t become disconnected.

Why wouldn't we end up with a sub-optimal solution?

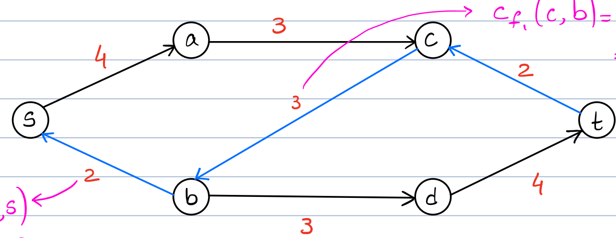
Def Given a flow f in graph G, residual capacity $c_f(u,v) = c(u,v) - f(u,v)$

Def Residual graph G_f is a directed graph with all edges of positive residual capacity. May include back-edges on original graph.

Example



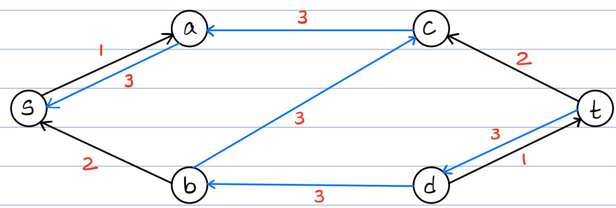
Sending a flow f_1 of 2 through s-b-c-t results in following residual graph G_{f_1} :



$c_{f_1}(c,b) = c(c,b) - f_1(c,b)$
$= 1 - (-2) = 3$

$c_{f_1}(b,s) = c(b,s) - f(b,s)$
$= 0 - (-2) = 2$

Next we send flow f_2 of 3 through s–a–c–b–d–t, resulting in residual graph G_{f_2} :



There no longer exists an s-t path in G_{f_2} , so we are done.

At each step found a new flow along an augmenting path & added to existing flow. So, max flow is $f_1 + f_2 = 5$.

Residual flow guarantees flow is legal. To implement, just use DFS.

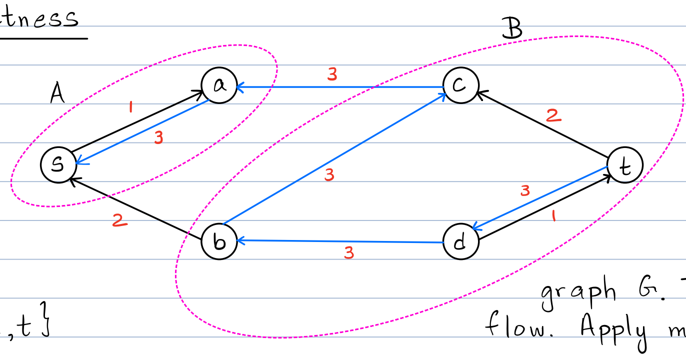
Need to prove complexity & correctness.

Complexity of F-F

Assume all capacities are integers. If algorithm finds a flow of F, then max possible iterations are F, since at each iteration, flow goes up by atleast one.

Thm On a graph G, with integer capacities, F-F takes $O(F(m+n))$.

Correctness



Note, final residual graph identifies a cut (A,B) with capacity 5 in original graph G. This matches the found flow. Apply max-flow min-cut Thm.

$A = \{s, a\}$
$B = \{b, c, d, t\}$

The final residual graph must have s & t disconnected, otherwise could increase flow from s to t, using existing path.

Let A be component containing s & B contains t.
Let c be capacity of the (A,B) cut in original graph.

Claim F-F finds flow of value c.