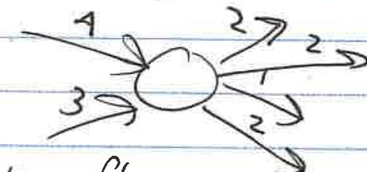
Max Flow Min Cut Theorem

A flow network is a graph, directed and weighted with two special vertices, the source and the sink, s and t respectively. Each edge has a positive capacity $c(u,v)$. A flow in a graph is an assignment to the edges such that $0 \le f(u,v) \le c(u,v)$ and the flow is conserved. The flow into any vertex equals the flow out of any vertex.

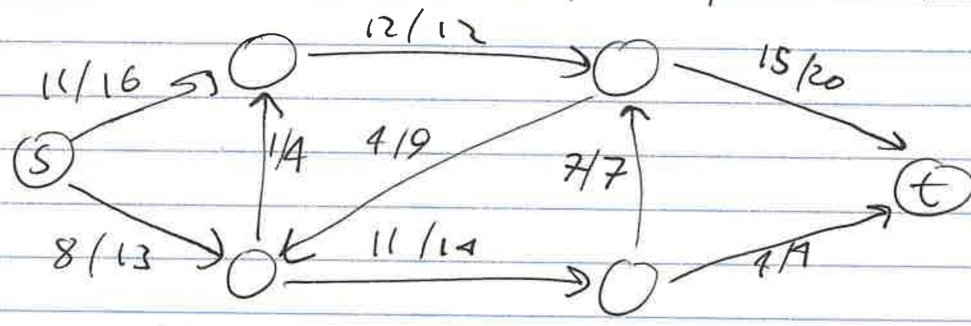
$\forall u \quad \sum f(v,u) = \sum_{v} f(u,v)$

except for the source and sinks.

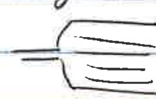


The source s has no incoming flow and the sink t has no outgoing flow.

We are concerned with two problems, maximum flow, and minimum cut.

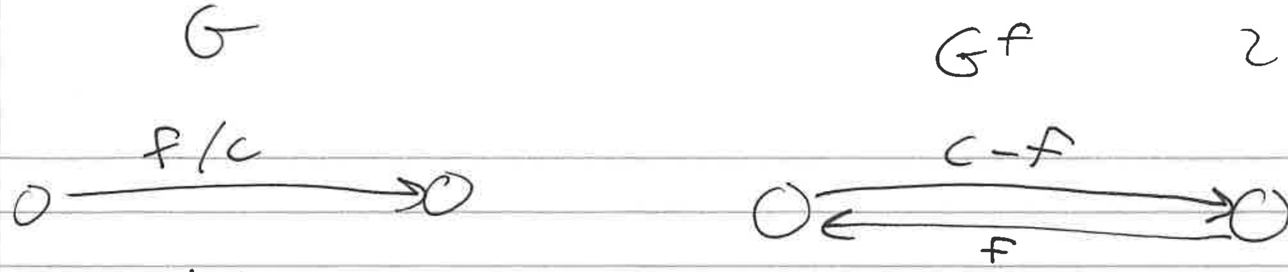


maxflow Consider the following flow network. We are concerned with the max amount we can flow from s to t. This is max flow, although there is this edge of 14, only 13 can make it there



Mincut The cut of a flow network is a partition S and $T = V - S$ with $s \in S$ and $t \in T$ where you add all edges $S \to T$ and subtract edges $T \to S$. The minimum cut is the smallest of all possible cuts.

These seemingly unrelated problems actually are very related! We will soon prove that the max flow = min cut in any such flow network. Then we wish to compute the max flow using a greedy method, but observe that maximizing the flow f may involve decreasing the flow among ~~certain~~ edges individually. We model this by keeping track of an extra data structure called the residual graph G^f.

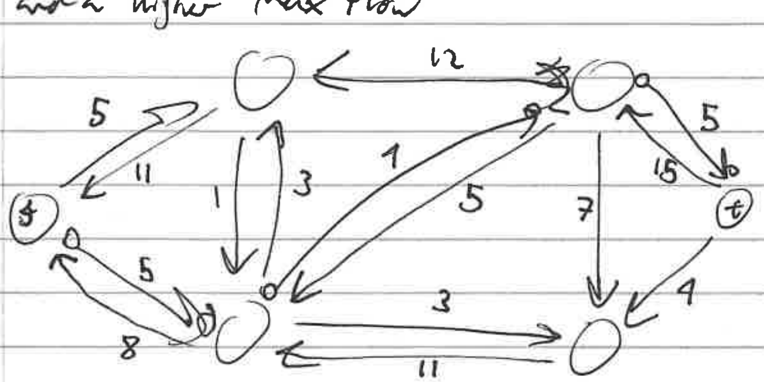


we would like to BFS from s to t but allow for backtracking in case we went too greedy. Instead we BFS on G^f instead of G. G^f has forward edges of the remaining capacity and a backwards edge of the flow. If an edge has weight zero we may not draw it. We can model choosing to decrease the flow by taking this backwards edge. Observe that if there is an s-t path in G^f, then we may increase the flow. This gives us the ford fulkerson method

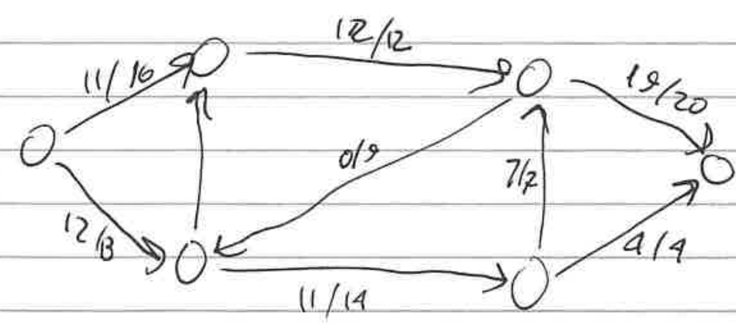
```
def ff(G)
    initialize Gf = G
    f = zero
    while Gf has (s-t) path
        compute bottleneck off path as b
        augment path in Gf by b (c-f -> c-(f+b), f -> f+b)
```

note that ford fulkerson isn't really an algorithm with a defined runtime, there are varying implementations. we take our previous flow, compute G^f, find a path, and a higher max flow

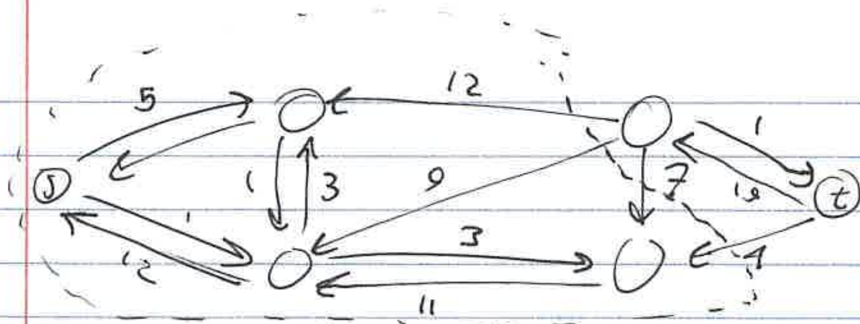
G^f



bottleneck is 4. so add four to graph



from 19 → 23! flow increased? is it maximum? lets compute the G^f again.

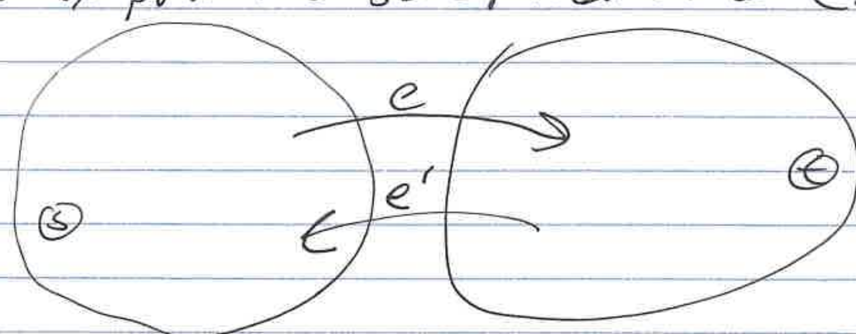


$f_{max} \Rightarrow$ G^f no (s,t) path

If we call explore on s, note that t is not reachable, so the algorithm terminates. Why is it maximum though?? We prove that if f is maximum then G^f has no more augmenting paths. Suppose for the sake of contradiction that f was maximum but G^f did have a path, with bottleneck b. In G, (not G^f), we may add b to each edge in the path, giving a new flow of f+b, contradicting that f was the maximum.

Next, note that any flow is less than every cut. So the max flow $\leq$ min cut.
It is enough for us to show then that if f is maximum, then $|f| = (S,T)$ cut for some cut. Since f is max, G^f has no s,t path. Take $S =$ explore(G^f, s) $T = V - S$.

explore on G^f
cut on G



Since G^f has no s-t path, the values of edges $e = 0$. In G, the edges e must be at full capacity since there is no path s-t in G^f. So the edges of full capacity are equal to the cut of G.

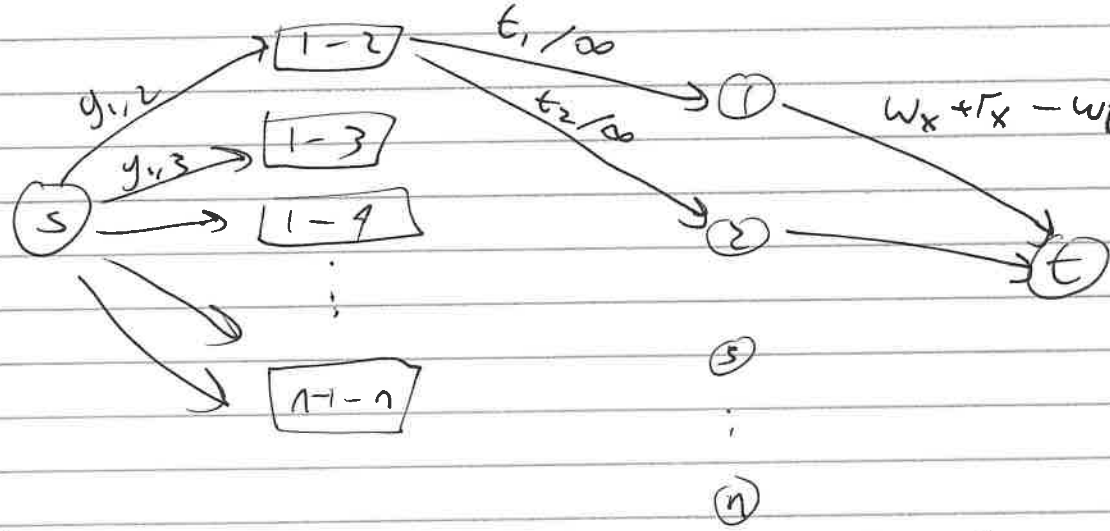
To determine if a flow is maximum, find an equivalent minimum cut?

Many problems which have nothing to do with roads and packages can be modeled as max flow. Suppose you are a baseball fan, and want to know if your team will be eliminated from playoffs. Its not as simple as calculating if all teams lose games and your team wins them all. As every loss is someone elses win

Team 1	11	games left	1
Team 2	11		1
Team 3	8		3

⋮

even if team 3 wins all 3, if the game between team 1 and 2 is against each other, one will have 12 points, so $8+3 < 12$, can't come in first.



let $g_{i,j}$ be the number of games left between i, j. g_{ij} is from s to node $[i-j]$. each $[i-j]$ only has two outgoing edges of unbounded capacity, but by flow conservation $t_1 + t_2 = g_{1,2}$. Each possible flow corresponds to a combination of games. we set our last layer to determine if a team can score more points than us, team x. we have w_x wins and r_x remaining, so we can get as much as $w_x + r_x$. If $w_x + r_x < w_i$ ~~$\forall i$~~, then x is eliminated, so set capacities to be $w_x + r_x - w_i$