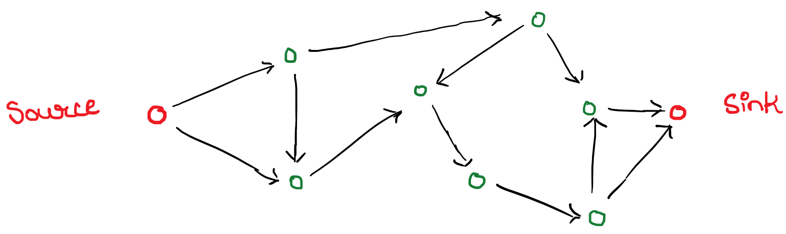
Flow Networks

Suppose that a Network consists of a Source, a Sink, and these two nodes are connected via switches. The bandwidth of each connection isn't the same. At the same time, each of the switch does cut-through switching without any Queue capacity.



This system can be modeled using a directed graph with two special nodes, the source and the sink.

Let the graph be $G(V, E)$, source be s and sink be t.

- By definition, no edge terminates at s and no edge begins at t.

Each edge has a capacity $c(e)$, which is the maximum possible capacity of that edge. We define $f(e)$, as the capacity of the edge in use currently.

$$\forall e \in E, \quad 0 \leq f(e) \leq c(e)$$ Capacity Constraint

* Flow

Following the example, each bridge has a data stream flowing through it. As no data can be accumulated, Inward flow = Outward flow

(Similar to Kirchoff's law from electricity) flow constraint

- The outwards flow for a node is represented by $f^{\rightarrow}(v)$, and is given by:-

$$f^{\rightarrow}(v) = \sum_{\substack{u \in V \\ (u,v) \in E}} f((u,v))$$

The value of inwards flow is similarly defined, and is represented as $f^{\leftarrow}(v)$.

- Value of the flow $|f|$

The total amount of data "flowing" through the network. It is given by $|f|$, and is calculated as shown.

Lemma

$$|f| = \sum_{\substack{v \in V \\ (s,v) \in E}} f(s,v) = f^{\rightarrow}(s) = \sum_{\substack{v \in V \\ (v,t) \in E}} f(v,t) = f^{\leftarrow}(t)$$

Proof

From above, we can see that

$$|f| = f^{\rightarrow}(s) + 0$$

$$= f^{\rightarrow}(s) + \sum_{v \in V \setminus \{s,t\}} \left(f^{\rightarrow}(v) - f^{\leftarrow}(v)\right) \rightarrow 0 \text{ by flow constraint}$$

$$= \sum_{v \in V \setminus \{t\}} \left(f^{\rightarrow}(v) - f^{\leftarrow}(v)\right) \qquad f^{\leftarrow}(s) = 0 \text{ by definition}$$

$$= \sum_{v \in V \setminus \{t\}} f^{\rightarrow}(v) - \sum_{v \in V \setminus \{t\}} f^{\leftarrow}(v)$$

$$= \underset{①}{f^{\rightarrow}(V \setminus \{t\})} - \underset{②}{f^{\leftarrow}(V \setminus \{t\})}$$

New notation!

Notice that all edges appear in ① as t has no flow leaving it. However, the edges "supplying" t would not be present in ②. These edges are left after subtraction.

$\rightarrow |f| = f^{\rightarrow}(V \setminus \{t\}) - f^{\leftarrow}(V \setminus \{t\}) = f^{\leftarrow}(t)$

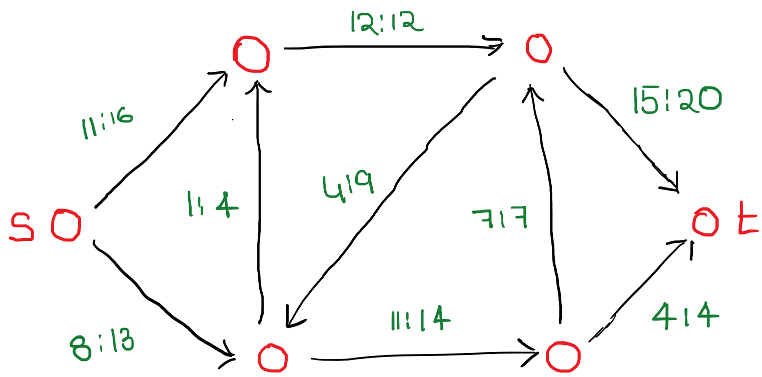
QED

* Flow Network notation

The network is represented as a directed acyclic graph, with designated source and sink nodes. Each edge is labelled as

$f(e) : c(e)$.

Capacity constraints and flow constraints have to be satisfied.



We would like to know what the maximum possible flow in this network is. The following concept is introduced for this.

* An (s,t)-cut :-

Let the flow network be given by $G(V,E)$ with s as the source and t is the sink. An (s,t)-cut is given by partioning V into two sets, S and T which are mutully exclusive and exhaustive.

$$s \in S \qquad S \cup T = V$$
$$t \in T \quad , \quad S \cap T = \phi$$

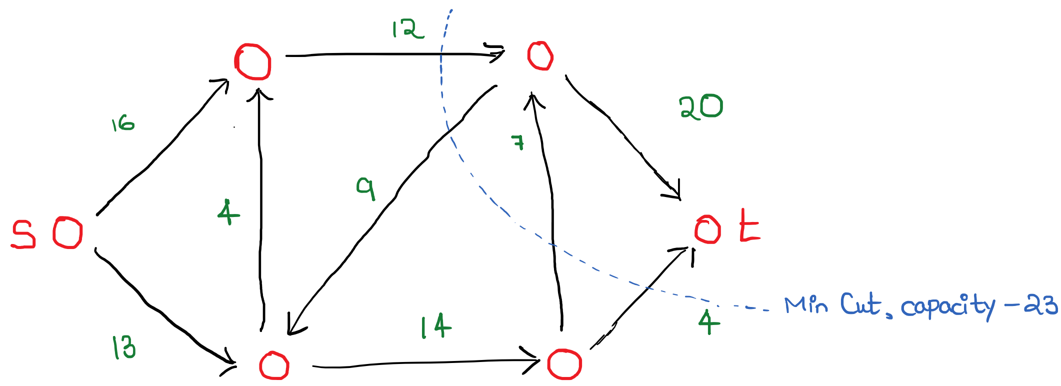
The capacity for an (s,t)-Cut (S,T) is given by :-

$$cap(S,T) = \sum_{\substack{u \in S,\, v \in T \\ (u,v) \in E}} c(u,v)$$

Notice how only one direction is considered, $S \to T$.

- Mincut problem

Given a flow network, we wish to find the cut of the graph which has the least possible capacity. For example,



There is an inherent relationship between max-flow and min-cut classes of problems. The below lemma hints at what this could be.

Lemma
Weak duality

Consider a flow network G. For any (s,t)-Cut (S,T) over this G, the value of $|f| \leq capacity(S,T)$. The equality is acheived when the flow through edges $S \to T$ is saturated and edges $T \to S$ are avoided.

Proof

$$|f| = f^{\to}(S) = f^{\to}(S) - f^{\leftarrow}(S)$$

$$\leq f^{\to}(S)$$

as $f(u,v) \leq c(u,v)$ by definition

$$\leq \sum_{\substack{u \in S,\, v \in T \\ (u,v) \in E}} f(u,v) \leq \sum_{\substack{u \in S,\, v \in T \\ (u,v) \in E}} c(u,v)$$

def of cap(S,T)

$$\leq cap(S,T)$$

$\leq \operatorname{cap}(S,T)$

QED

Also notice that inequalities are obtained by ignoring $f^{\leftarrow}(S)$ and taking $f(u,v) \leq c(u,v)$. Therefore, the equality holds if :-

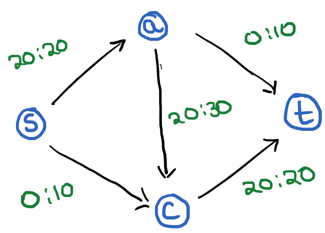
1) $f^{\leftarrow}(S) = 0$ —— T→S edges should be ignored
2) $f(u,v) = c(u,v)$ —— S→T edges should be saturated

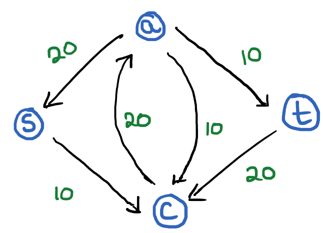
We define a few more concepts and ideas before tackling the max-flow problem. These will feel random at first, but the reasons will become clear as we move ahead with the solution.

Idea 1 – Residual Graph

For a given flow network $G(V,E)$ and $c: E \to \mathbb{Z}$ defined with a value of flow $|f|$ in the network, the residual Graph G_f is defined as follows :-

- Nodes in G_f are the same as nodes in G
- If an edge e in G has $f(e) < c(e)$, then a similar edge with capacity $c(e) - f(e)$ is present in G_f. This is called the Forward Edge.
- If an edge e in G has $f(e) > 0$, then a reverse edge with capacity $f(e)$ is present in G_f. This is called the Backwards Edge.





Atleast one and atmost two edges are added into the residual graph for every edge in G.

$$\# \text{ of edges in } G_f \leq 2 \times \# \text{ of edges in } G$$

Idea 1.2 Augmenting paths

For a residual graph G_f, let π be any path from s to t, with the mimum residual capacity along this path being given by $\theta(\pi, f)$.

Consider the following function :-

if $e = (u,v)$ then $e^{-1} = (v,u)$

```
Aug(π, f) :
    b ← θ(π, f)
    for every edge e ∈ π, do
        if e is a forward edge, then
            increase f(e) in G by b
        else
            decrease f(e⁻¹) in G by b
        endif
    endfor
```

That is, we're modifying the original graph by using the information from the residual graph. The reasoning will be made clear shortly.

* ## Ford-Fulkerson's Algorithm

We state the algorithm below. The proof for the algorithm shall follow.

```
for Edge e ∈ E, set f(e) = 0          ——— Initialisation
compute G_f
while path π from s to t exists in G_f
      f ← Aug(π, f)
end while
output f
```

Initialisation

Need to prove correctness

Although this algorithm seems very elegant, we have yet to prove the correctness and the time complexity. As a start, we first show that the algorithm terminates.

1) Termination of the algorithm

We first make an assumption -

All flow values and capacities are integral —— (1)

At every iteration, let f' be the updated value of flow. Notice that by the definition of $\mathrm{Aug}(\pi, f)$:-

$$|f'| = |f| + \mathrm{Aug}(\pi, \theta) \quad \text{where } \mathrm{Aug}(\pi, \theta) \in \mathbb{Z}^+$$

because flows belong to $\mathbb{Z}^+$

It can be seen that the flow in network has an upper limit, which is denoted by C_{max}.

$$|f| \leq C_{max} = \sum_{\substack{v \in V \\ (s,v) \in E}} c(s,v)$$

As the value of flow keeps increasing, and there's an upper limit;

This implies that the algorithm must terminate.

2) Time Analysis

The value of flow starts at 0, and can increase upto C_{max} by a minimum of 1 at every iteration.

$\Rightarrow$ Max. number of iterations = C_{max}

Assume that $|V| = n$ and $|E| = m$. Then for each iteration :-

- Maintaining G_f — $O(m)$
- Finding path from $s \to t$ — $O(m+n)$
- Runtime of Aug(π, f) — $O(n)$ as π can have atmost $n-1$ edges.

$\Rightarrow$ Total time is bounded by $O(C_m \cdot (m+n))$

For connected graphs, we usually assume that $m \gg n \Rightarrow m+n \approx m$

$\therefore$ Runtime is bounded by $O(C_m |E|)$

3) Correctness proof

Let the final value of flow obtained from the algorithm be f. Let the corresponding residual graph be G_f. The algorithm terminates when no path exists from s to t. Define two sets A, B as follows :-

A – Vertices reachable by s in G_f

B – Vertices not reachable by s in G_f

Notice that A, B form a cut as the sets partition the vertex set into two. By Weak duality discussed earlier ;-

$$f \leq cap(A,B)$$

We shall prove that all edges from $A \to B$ are saturated, and all edges from $B \to A$ are ignored.

(i) Let $\exists e \in E$ from $A \to B$ st $f(e) < c(e)$

This means that a forward edge exists in G_f from $A \to B$.

A direct contradiction to the definition of B.

$\therefore$ No such e exists

(ii) Let $\exists e$ from $B \to A$ such that $f(e) > 0$

Means that a reverse edge from $B \to A$ exists.

$\Rightarrow$ forward edge from $A \to B$ $\Rightarrow$ contradiction !

$\therefore$ No such e exists

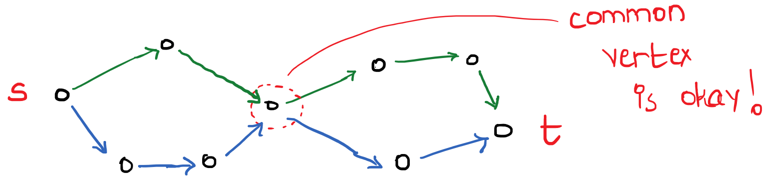
By weak duality, it has been proven that f is the maximum possible flow and also, (A,B) is the mincut of the given network.

Note that the time taken by the algorithm can be further reduced by cleverly choosing the path π taken by the algorithm. One of the heuristics discussed by Bellman and Ford was to choose the shortest path from s to t. However, other heuristics are also present.

We shall now look at a few problems which can be converted into the max-flow problem via clever manipulation.

1) Number of disjoint paths

Two paths π and π' are said to be disjoint if no common edge exists between them. Given a graph $G(V,E)$ we would like to find the number of disjoint paths from s to t. Note that the following paths are disjoint as well.



G'

Define a flow network with the same graph G and $c(e) = 1, \forall e \in E$. For this network, we state and prove the following lemma.

Lemma Number of disjoint edges in G is equal to the maximum flow possible in G', provided the flow values in G' are integral.

Proof As the flow value is an integer, the value of $f(e) = 0$ or 1 for any edge. We interpret $f(e) = 0$ as that edge not being a part of any disjoint path, and $f(e) = 1$ means that edge belongs to a path in the set of disjoint paths.

Both directions of the implication can be proven by contradiction quite easily. This concludes our proof.

1.2) Network Connectivity problem

Given a graph $G(V,E)$ and $s,t \in V$, we would like to find the smallest set $F \subseteq E$ such that no path exists between s,t in the graph $G'(V, E \setminus F)$.

It can be quite clearly seen that the least value of $|F|$ would be equal to maximum number of disjoint paths. Therefore, this problem can also be converted to max-flow, albeit indirectly.

2) Maximum bipartite Matching

A bipartite graph has two sets which portion the vertex set. All edges in the graph are between these sets.

Similarly, a matching is a subset of edges such that no vertex is shared by two edges belonging to the subset.

Given an undirected bipartite graph $G(V,E)$ with $V = X_1 \cup X_2$ we wish to find the largest possible matching.

We construct a flow network G' from G as follows:-

* For $u \in X_1$ and $v \in X_2$, if $(u,v) \in E$ then add a directed edge (u,v) in G'
* For a new node s, add edges from s to every node in X_1.
* For a new node t, add edges from every node in X_2 to t.
* Set capacity of every edge as 1 in G'.

For this flow network, we state and prove the following lemma.

Lemma Let G' be the flow network constructed for a bipartite graph G. For this graph, a matching M is possible iff a flow in G' with $|f| = |M|$ is possible.

Proof

Forward direction

Forward direction is trivial. Let X_1' be a set of vertices such that:-

$$X_1' \subseteq X_1 \text{ such that } \forall u \in X_1' \ \exists v \in X_2 ,\ (u,v) \in M$$

Similarly, $X_2' \subseteq X_2$ such that $\forall v \in X_2' \ \exists u \in X_1 ,\ (u,v) \in M$

The flow through an edge is 1 if it belongs to the matching. Also, edges from s to X_1' and X_2' to t are also 1.

It can be seen that $|f| = |M|$.

Backward direction

Given G' with a flow $|f|$, we need to create a matching M such that $|M|$.

We state the following:-

$$(u,v) \in M \text{ iff } u \in X_1 ,\ v \in X_2 \text{ and } f(u,v) = 1$$

By definition, $|f| = |M|$. It can be seen that M must be a matching.

Assume M wasn't a matching.

$\Rightarrow \exists v \in X_1$ which has two edges. (e_1 and e_2)

$\Rightarrow$ By construction of M, $f(e_1) = f(e_2) = 1$

However, only one edge is directed towards v in G'

$\Rightarrow f^{\rightarrow}(v) \neq f^{\leftarrow}(v)$ — contradiction.

$\therefore$ M is a matching.

QED