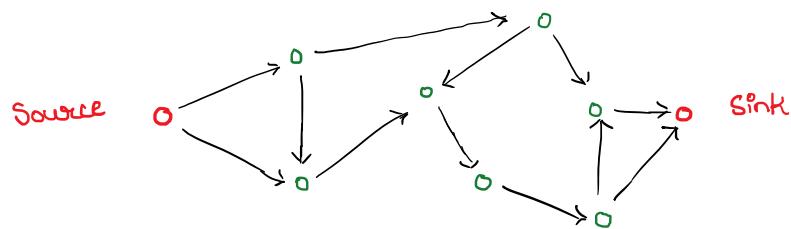


## Flow Networks

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.

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

### \* 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)  $\uparrow$  **flow constraint**

- The outwards flow for a node  $v$  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

$$\begin{aligned} |f| &= f^{\rightarrow}(s) + 0 \\ &= f^{\rightarrow}(s) + \sum_{v \in V \setminus \{s,t\}} (f^{\rightarrow}(v) - f^{\leftarrow}(v)) \xrightarrow{\text{O by flow constraint}} \\ &= \sum_{v \in V \setminus \{t\}} (f^{\rightarrow}(v) - f^{\leftarrow}(v)) \quad f^{\leftarrow}(s) = 0 \text{ by definition} \\ &= \sum_{v \in V \setminus \{t\}} f^{\rightarrow}(v) - \sum_{v \in V \setminus \{t\}} f^{\leftarrow}(v) \\ &= f^{\rightarrow}(V \setminus \{t\}) - f^{\leftarrow}(V \setminus \{t\}) \end{aligned}$$

①                    ②

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.

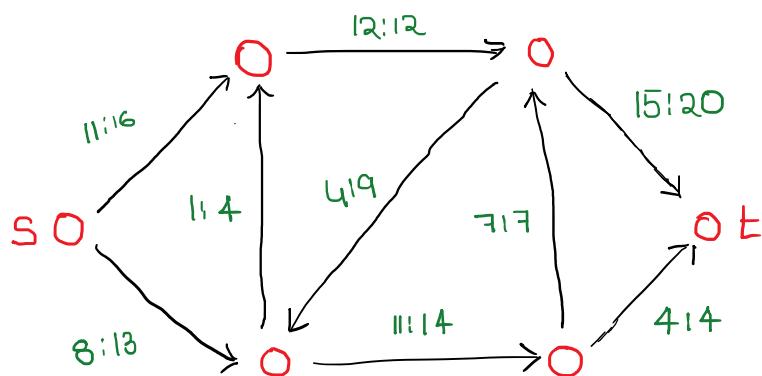
$$\rightarrow |f| = f^{\rightarrow}(v \setminus \{t\}) - f^{\leftarrow}(v \setminus \{t\}) = \underline{f^{\leftarrow}(t)}$$

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  $t$  as the sink. An  $(s,t)$ -cut is given by partitioning  $V$  into two sets,  $S$  and  $T$  which are mutually exclusive and exhaustive.

$$\begin{array}{ll} s \in S & S \cup T = V \\ t \in T & S \cap T = \emptyset \end{array}$$

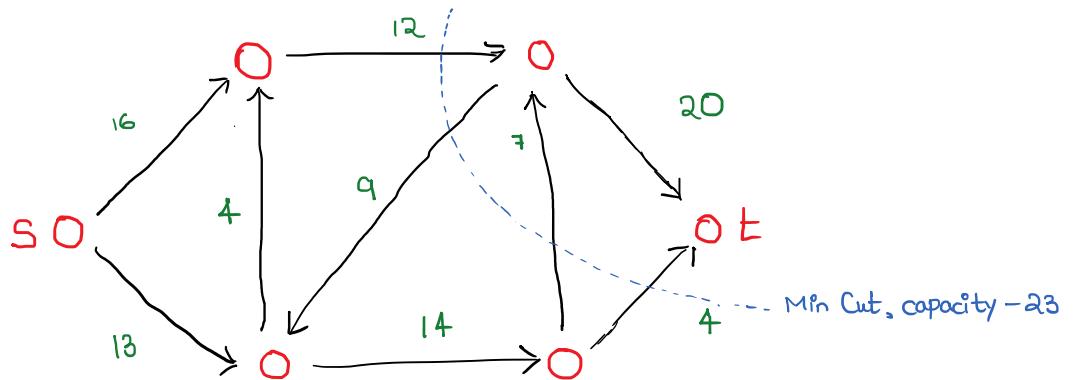
The capacity for an  $(S, T)$ -Cut  $(S, T)$  is given by:-

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

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

### - 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  $|f| \leq \text{capacity}(S, T)$ . The equality is achieved when the flow through edges  $S \rightarrow T$  is saturated and edges  $T \rightarrow S$  are avoided.

**Proof**

$$\begin{aligned}
 |f| &= f^{\rightarrow}(S) = f^{\rightarrow}(S) - f^{\leftarrow}(S) \\
 &\leq f^{\rightarrow}(S) && \text{as } f(u, v) \leq c(u, v) \\
 &\leq \sum_{\substack{u \in S, v \in T \\ (u, v) \in E}} f(u, v) \leq \sum_{\substack{u \in S, v \in T \\ (u, v) \in E}} c(u, v) && \text{by definition} \\
 &\leq \text{Cap}(S, T)
 \end{aligned}$$

$\downarrow$  def of  $\text{cap}(S, T)$

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

QED

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

- 1)  $f^*(s) = 0$  ————— T  $\rightarrow$  S edges should be ignored
- 2)  $f(u,v) = c(u,v)$  ————— S  $\rightarrow$  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.

### Ideal - Residual Graph

For a given flow network  $G(V,E)$  and  $c: E \rightarrow \mathbb{Z}$  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  $\pi$  be any path from  $s$  to  $t$ , with the minimum residual capacity along this path being given by  $\Theta(\pi, f)$ .

Consider the following function :-

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

$\text{Aug}(\pi, f) :$

$b \leftarrow \Theta(\pi, f)$

for every edge  $e \in \pi$ , 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, t)
end while
output f
  
```

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  $\text{Aug}(\pi, t)$  :-

$$|f'| = |f| + \text{Aug}(\pi, t) \quad \text{where } \text{Aug}(\pi, t) \in \mathbb{Z}^+$$



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

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

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

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.

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

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

- Maintaining  $G_f$  —  $O(m)$
- Finding path from  $s \rightarrow t$  —  $O(m+n)$
- Runtime of  $\text{Aug}(\pi, f)$  —  $O(n)$  as  $\pi$  can have almost  $n-1$  edges.

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

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

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

### 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 ;-

$$f \leq \text{cap}(A, B)$$

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

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

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

A direct contradiction to the definition of  $B$ .

$\therefore$  No such  $e$  exists

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

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

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

$\therefore$  No such  $e$  exists

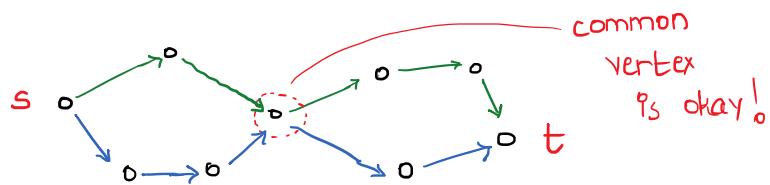
By weak duality, it has been proven that  $f$  is the maximum possible flow and also,  $(A, B)$  is the mincut of the given network.

Note that the time taken by the algorithm can be further reduced by cleverly choosing the path  $\pi$  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  $\pi$  and  $\pi'$  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.