

---

# CSS.201.1 ALGORITHMS

*Instructor: Umang Bhaskar*

*TIFR 2024, Aug-Dec*

---

SCRIBE: SOHAM CHATTERJEE

SOHAMCHATTERJEE999@GMAIL.COM

WEBSITE: SOHAMCH08.GITHUB.IO

# CONTENTS

## CHAPTER 1

### MAXIMUM FLOW

### PAGE 3

1.1	Flow	3
1.2	Ford-Fulkerson Algorithm	4
	1.2.1 Max Flow Min Cut	6
	1.2.2 Edmonds-Karp Algorithm	8
1.3	Preflow-Push/Push Relabel Algorithm	8

# Maximum Flow

## 1.1 Flow

Suppose we are given a directed graph  $G = (V, E)$  with a source vertex  $s$  and a target vertex  $t$ . And additionally for every edge  $e \in E$  we are given a number  $c_e \in \mathbb{Z}_0$  which is called the capacity of the edge.

### Definition 1.1.1: Flow

An  $s - t$  flow is a function  $f : E \rightarrow \mathbb{R}_0$  which satisfies the following:

- ①  $\forall e \in E, f(e) \leq c_e$
- ②  $\forall v \in V \setminus \{s, t\}, \sum_{e \in \text{in}(v)} f(e) = \sum_{e \in \text{out}(v)} f(e)$

Also the value of a flow  $f$  is denoted by  $|f| := \sum_{e \in \text{out}(s)} f(e)$ .

Before proceeding into the setup and the problem first we will assume some things

**Assumption.** •  $\text{in}(s) = \emptyset$  i.e. there is no edge into  $s$ .

•  $\text{out}(t) = \emptyset$  i.e. there is no edge out of  $t$ .

• There are no parallel edges

### Lemma 1.1.1

For any flow  $f$ ,  $|f| = \sum_{e \in \text{in}(t)} f(e)$

**Proof:** We have for every edge  $e \in E$ ,  $\exists v \in V$  such that  $e \in \text{in}(v)$  and  $\exists u \in V$  such that  $e \in \text{out}(u)$ . Hence we get

$$\sum_{e \in E} f(e) = \sum_{v \in V} \sum_{e \in \text{in}(v)} f(e) = \sum_{v \in V} \sum_{e \in \text{out}(v)} f(e) \implies \sum_{v \in V} \left[ \sum_{e \in \text{in}(v)} f(e) - \sum_{e \in \text{out}(v)} f(e) \right] = 0$$

Now we know  $\forall v \in V \setminus \{s, t\}, \sum_{e \in \text{in}(v)} f(e) = \sum_{e \in \text{out}(v)} f(e)$ . Therefore we get

$$\sum_{v \in V} \left[ \sum_{e \in \text{in}(v)} f(e) - \sum_{e \in \text{out}(v)} f(e) \right] = 0 \implies \sum_{v \in \{s, t\}} \left[ \sum_{e \in \text{in}(v)} f(e) - \sum_{e \in \text{out}(v)} f(e) \right] = 0 \implies \sum_{e \in \text{out}(s)} f(e) - \sum_{e \in \text{in}(t)} f(e)$$

Hence we have  $|f| = \sum_{e \in \text{in}(t)} f(e)$ . ■

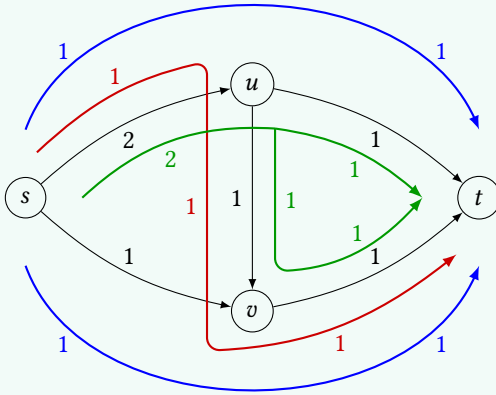
## MAX FLOW

**Input:** A directed graph  $G = (V, E)$  with source vertex  $s$  and target vertex  $t$  and for all edge  $e \in E$  capacity of the edge  $c_e \in \mathbb{Z}_+$

**Question:** Given such a graph and its capacities find an  $s - t$  flow which has the maximum value

**Example 1.1.1**

Consider the following directed graph with capacities:  $V = \{s, t, u, v\}$ ,  $c_{s,u} = 2, c_{s,v} = c_{u,t} = c_{v,t} = c_{u,v} = 1$ . Firstly the following function:  $f' : f'(s, u) = 2 = f(u, t)$ . It is not a flow since  $f(u, t) = 2 > 1 = c_{u,t}$ . Now we define three different flow functions:



- $f$ :  $f(s, u) = f(u, v) = f(v, t) = 1$  and otherwise 0. Therefore  $|f| = 1$
- $g$ :  $g(s, u) = g(u, t) = 1, g(s, v) = g(v, t) = 1$  and otherwise 0. Therefore  $|g| = 2$
- $h$ :  $h(s, u) = 2, h(u, t) = h(u, v) = h(v, t) = 1$  and otherwise 0. Therefore  $|h| = 2$

Notice here  $g$  and  $h$  has the maximum flow value.

## 1.2 Ford-Fulkerson Algorithm

### Definition 1.2.1: Residual Graph

Given a directed graph  $G = (V, E)$  and capacities  $C_e$  for all  $e \in E$  and an  $s - t$  flow  $f$  the residual graph  $G_f = (V, E_f)$  has the edges with the following properties:

- ① If  $(u, v) \in E$  and  $f(u, v) > 0$  then  $(v, u) \in E_f$  and  $c_{v,u}^f = f(u, v)$ . Such an edge is called a *backward* edge.
- ② If  $(u, v) \in E$  and  $f(u, v) < c_{u,v}$  then  $(u, v) \in E_f$  and  $c_{u,v}^f = c_{u,v} - f(u, v)$ . It is called *forward* edge.

### Algorithm 1: FORD-FULKERSON

**Input:** Directed graph  $G = (V, E)$ , source  $s$ , target  $t$  and edge capacities  $C_e$  for all  $e \in E$

**Output:** Flow  $f$  with maximum value

```

1 begin
2   for  $e \in E$  do
3      $f(e) = 0$ 
4   while  $\exists s \rightsquigarrow t$  path  $P$  in  $G_f$  do
5      $\delta \leftarrow \min_{e \in P} \{c_e^f\}$  for  $e = (u, v) \in P$  do
6       if  $e$  is Forward Edge then
7          $f(u, v) \leftarrow f(u, v) + \delta$ 
8       else
9          $f(u, v) \leftarrow f(v, u) - \delta$ 

```

We call one iteration of the While loop at line 4 *Flow Augmentation*.

**Lemma 1.2.1**

At any iteration the  $f'$  obtained after the flow augmentation of the flow  $f$  is a valid flow

**Proof:** At any iteration let  $P$  be the path from  $s \rightsquigarrow t$  and  $\delta = \min_{e \in P} c_f(e)$ . Let  $f'$  be the new function such that for each  $(u, v) \in P$  if  $(u, v)$  is forward edge in  $G_f$  then  $f'(u, v) = f(u, v) + \delta$  and if  $(u, v)$  is backward edge in  $G_f$  then  $f'(v, u) = f(v, u) - \delta$  and for other edges  $e \in E \setminus P$ ,  $f'(e) = f(e)$ .

Now since  $\delta = \min_{e \in P} c_f(e)$ ,  $c_f(e) \geq \delta$  for all  $e \in P$ . Hence if  $(u, v)$  is backward edge then  $(v, u) \in E$  and  $c_f(u, v) = f(u, v)$ . Hence  $f'(v, u) = f(v, u) - \delta \geq 0$ . Therefore for all  $e \in E$ ,  $f'(e) \geq 0$ .

Now first we will show  $f'(e) \leq c_e$  for all  $e \in E$ . If  $(u, v) \in P$  is a forward edge then  $(u, v) \in E$  and  $c_f(u, v) = c_{u,v}f(u, v)$ . Therefore  $f'(u, v) = f(u, v) + \delta \leq f(u, v) + c_{u,v} - f(u, v) = c_{u,v}$ . Now if  $(u, v) \in P$  is a backward edge then  $(v, u) \in E$  and  $c_f(u, v) = f(u, v)$ . Therefore  $f'(u, v) = f(u, v) - \delta \leq f(u, v) \leq c_{u,v}$ . For other edges  $e \in E \setminus P$ ,  $f'(e) = f(e) \leq c_e$ . Therefore  $f'(e) \leq c_e$  for all  $e \in E$ .

Now we will prove for all  $v \in V \setminus \{s, t\}$ ,  $\sum_{e \in in(v)} f'(e) = \sum_{e \in out(v)} f'(e)$ . If  $v$  is not in the path  $P$  in  $G_f$  then,  $f'(e) = f(e)$  for all edges  $e \in in(v) \cup out(v)$ . Hence the condition is satisfied for such vertices. Suppose  $v$  is in the path  $P$ . Then there are two edges  $e_1$  and  $e_2$  in  $P$  which are incident on  $v$ . If both are forward edges or both are backward edges then one of them is in  $in(v)$  and other one is in  $out(v)$ . WLOG suppose  $e_1 \in in(v)$  and  $e_2 \in out(v)$  we have

$$\sum_{e \in in(v)} f'(e) = \sum_{e \in in(v) \setminus \{e_1\}} f(e) + f(e_1) \pm \delta = \sum_{e \in out(v) \setminus \{e_2\}} f(e) + f(e_2) \pm \delta = \sum_{e \in out(v)} f'(e)$$

If one of  $e_1, e_2$  forward edge and other one is backward edge then either  $e_1, e_2 \in in(v)$  (when  $e_1$  is forward and  $e_2$  is backward) or  $e_1, e_2 \in out(v)$  (when  $e_1$  is backward and  $e_2$  is forward). Now if  $e_1, e_2 \in in(v)$ ,  $f'(e_1) + f'(e_2) = f(e_1) + \delta + f(e_2) - \delta = f(e_1) + f(e_2)$  and if  $e_1, e_2 \in out(v)$  then  $f'(e_1) + f'(e_2) = f(e_1) - \delta + f(e_2) + \delta = f(e_1) + f(e_2)$ . Hence

$$\sum_{e \in in(v)} f'(e) = \sum_{e \in in(v)} f(e) = \sum_{e \in out(v)} f(e) = \sum_{e \in out(v)} f'(e)$$

Hence  $f'$  is a valid flow. ■

**Lemma 1.2.2**

At any iteration Given  $G_f$  if the flow,  $f'$  obtained after flow augmentation of  $f$  by  $\delta$  then

$$|f'| = |f| + \delta$$

**Proof:** Since we augment flow along an  $s \rightsquigarrow t$  path, the first edge of the path is always in  $out(s)$ . Let the first edge is  $e = (s, u)$ . Now  $e$  has to be a forward edge because otherwise  $(u, s) \in E$  and then there is an incoming edge in  $G$  which is not possible. Hence

$$|f'| = \sum_{e \in out(s)} f'(e) = \sum_{e \in out(s) \setminus \{e\}} f(e) + f'(e) = \sum_{e \in out(s) \setminus \{e\}} f(e) + f(e) + \delta = \sum_{e \in out(s)} f(e) + \delta = |f| + \delta$$

Hence we have the lemma. ■

**Lemma 1.2.3**

At every iteration of the Ford-Fulkerson Algorithm the flow values and the residual capacities of the residual graph are non-negative integers.

**Proof:** Initial flow and the residual capacities are non-negative integers. Let till  $i^{th}$  iteration the flow values and the residual capacities were non-negative integers. Let the flow after  $i^{th}$  iteration was  $f$ . Hence  $\forall e \in E$ ,  $f(e) \in \mathbb{Z}_0$ . Therefore in the  $G_f$  for all  $e \in E_f$ ,  $c_f(e) \in \mathbb{Z}_0$ . Hence  $\delta \in \mathbb{Z}_0$ . Therefore  $\forall e \in E$ ,  $f'(e) \in \mathbb{Z}_0$ . And therefore for all  $e \in E_{f'}$  where  $G_{f'}$  is the residual graph of the flow  $f'$ ,  $c_{f'}(e) \in \mathbb{Z}_0$ . Hence by mathematical induction the lemma follows. ■

At any iteration let  $P$  be the path from  $s \rightsquigarrow t$ . Then for all  $e \in P$ ,  $c_f(e) > 0$ . Therefore  $\delta = \min_{e \in P} c_f(e) \geq 1$ . Therefore the algorithm must stop in at most  $\sum_{e \in out(s)} c_e$  since we can have the value of a flow to be at max the value of the sum of capacities of edges in  $out(s)$  and therefore we can increase the flow at max that many times.

#### Lemma 1.2.4

If  $f$  is a max flow then there is no  $s \rightsquigarrow t$  path in  $G_f$ .

**Proof:** Suppose there is an  $s \rightsquigarrow t$  path  $P$  in  $G_f$ . We will show that then  $f$  is not a max flow following the algorithm. Then  $\forall e \in P$ ,  $c_f(e) > 0$ . Hence  $\delta = \min_{e \in P} c_f(e) \geq 1$ . Now after the flow augmentation process of  $f$  by  $\delta$  we get a new valid flow  $f'$  by Lemma 1.2.1 and by Lemma 1.2.2 we have  $|f'| = |f| + \delta > |f|$ . Hence  $f$  is not a maximum flow. Hence contradiction. Therefore there is no  $s \rightsquigarrow t$  path in  $G_f$ . ■

### 1.2.1 Max Flow Min Cut

#### Definition 1.2.2: Cut Set

For a graph  $G = (V, E)$  and a subset  $A \subseteq V$ , the cut  $(A, V \setminus A)$  is a bipartition of  $V$  where the edges  $E_A$  of the graph  $G_A = (A, V \setminus A, E_A)$  is the set  $E_A = E \cap (A \times (V \setminus A))$ .

Now if  $s, t$  are two vertices of  $G$  then an  $s - t$  Cut  $(A, V \setminus A)$  is a cut such that  $s \in A$  and  $t \in V \setminus A$ .

Now we define for a cut  $(A, V \setminus A)$  the *Capacity of the Cut*  $(A, V \setminus A) = \sum_{e \in E_A} c_e$ . For an  $s - t$  cut  $(A, V \setminus A)$  we denote the capacity of the cut by  $cap(A)$ . A *Min  $s - t$  Cut* is a  $s - t$  cut of minimum capacity. Then we have the following relation between cut and flow.

#### Lemma 1.2.5

Given a graph  $G = (V, E)$ ,  $s, t, c_e \in \mathbb{Z}_0$  for all  $e \in E$  for any flow  $f$  and a  $s - t$  cut  $(A, V \setminus A)$

$$|f| \leq cap(A)$$

**Proof:** Given  $f$  and the  $s - t$  cut  $(A, V \setminus A)$  we have

$$\begin{aligned} |f| &= \sum_{e \in out(s)} f(e) \\ &= \sum_{v \in A} \left[ \sum_{e \in out(v)} f(e) - \sum_{e \in in(v)} f(e) \right] \\ &= \sum_{\substack{e=(u,v), \\ u \in A, v \notin A}} f(e) - \sum_{\substack{e=(u,v), \\ u \notin A, v \in A}} f(e) && \text{[Edges for both endpoints in } A \text{ are canceled out]} \\ &= \sum_{e \in out(A)} f(e) - \sum_{e \in in(A)} f(e) \\ &\leq \sum_{e \in out(A)} f(e) \leq \sum_{e \in out(A)} c_e = cap(A) \end{aligned}$$

Hence we have the lemma. ■

Having this lemma we have for any flow  $f$  and  $s - t$  cut  $(A, V \setminus A)$  we have

$$|f| \leq cap(A) \implies \max_f |f| \leq \min_{s-t \text{ cut } (A, V \setminus A)} cap(A)$$

So we have the following theorem that the value of maximum flow is equal to the capacity of minimum cut.

**Theorem 1.2.6 Max Flow Min Cut**

Given a graph  $G = (V, E)$ ,  $s, t, c_e \in \mathbb{Z}_0$  for all  $e \in E$ . Then the following are equivalent:

- (1)  $f$  is a maximum flow.
- (2) There is no  $s \rightsquigarrow t$  path in  $G_f$
- (3) There exists an  $s - t$  cut of capacity  $|f|$

**Proof:**

(1)  $\implies$  (2): This is by [Lemma 1.2.4](#).

(2)  $\implies$  (3): We are given a flow  $f$  such that there is no  $s \rightsquigarrow t$  path in  $G_f$ . We will construct a  $s - t$  cut which has the capacity  $|f|$ . Now take  $A$  to be all the vertices reachable from  $s$  in  $G_f$ . This is a valid  $s - t$  cut since  $s \in A$  and as there is no  $s \rightsquigarrow t$  path in  $G_f$ ,  $t \notin A$ . Now

$$|f| = \sum_{e \in \text{out}(A)} f(e) - \sum_{e \in \text{in}(A)} f(e)$$

Now  $\forall e = (u, v) \in E$  where  $u \in A$  and  $v \notin A$  we have  $c_{u,v} = f(u, v) \implies c_{u,v} - f(u, v) = 0$  since otherwise  $c_{u,v} - f(u, v) \neq 0 \implies c_{u,v} > f(u, v) \implies (u, v) \in E_f$  and therefore  $v$  is reachable from  $s$  but  $v \notin A$ , contradiction. Therefore  $(u, v)$  is a backward edge and hence  $f(u, v) = 0$ . Now  $\forall e = (u, v) \in E$  where  $u \notin A$  and  $v \in A$  we have  $f(u, v) = 0$  since otherwise  $f(u, v) > 0 \implies (v, u) \in E_f$  and therefore  $u$  is reachable from  $s$  but  $u \notin A$ , contradiction. Hence we have

$$|f| = \sum_{e \in \text{out}(A)} f(e) - \sum_{e \in \text{in}(A)} f(e) = \sum_{e \in \text{out}(A)} c_e = \text{cap}(A)$$

(3)  $\implies$  (1): Now by [Lemma 1.2.5](#) we have for any flow  $f$  and  $s - t$  cut

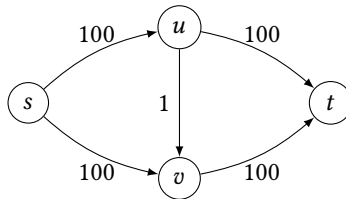
$$|f| \leq \text{cap}(A) \implies \max_f |f| \leq \min_{s-t \text{ cut } (A, V \setminus A)} \text{cap}(A)$$

Now given  $f$  there exists an  $s - t$  cut of capacity  $|f|$ . Hence  $f$  is a max flow. ■

Hence at the end of the [Ford-Fulkerson Algorithm](#) let the flow returned by the algorithm is  $f$ . The algorithm terminates when there is no  $s \rightsquigarrow t$  path in  $G_f$ . Hence by Max Flow Min Cut Theorem we have  $f$  is a maximum flow. This completes the analysis of the Ford-Fulkerson Algorithm.

Since the capacities of the edges can be very large we want an algorithm return the maximum flow with running time  $\text{poly}(n, m, \log c_e)$  where  $n$  is the number of vertices and  $m$  is number of edges and  $\log c_e$  basically means number of bits at most needed to represent the capacities.

But Ford-Fulkerson algorithm takes does not run in  $\text{poly}(n, m, \log c_e)$  instead  $\text{poly}(n, m, c_e)$  as the while loop in the algorithm takes  $\text{poly}(c_e)$  many iterations. For example in the following graph: it takes around 100 steps



and in general Ford-Fulkerson takes  $O(|f_{\max}|)$  time. For this reason we will now discuss a modification of the Ford-Fulkerson Algorithm which takes  $\text{poly}(n, m, \log c_e)$  time, Edmonds-Karp Algorithm.

### 1.2.2 Edmonds-Karp Algorithm

To get a  $\text{poly}(n, m, \log c_e)$  time algorithm we will always pick the shortest  $s \rightsquigarrow t$  path in the residual graph. This algorithm is known as the Edmonds-Karp Algorithm

Suppose  $f_i$  be the total flow after  $i^{\text{th}}$  iteration. And  $G_{f_i}$  be the residual graph with respect  $f_i$ . Then  $f_0(e) = 0$  for all  $e \in E$  and  $G_{f_0} = G$ . Also suppose  $\text{dist}_i(v) = \text{Shortest } s \rightsquigarrow v \text{ path distance in the residual graph } G_{f_i}$ . Hence  $\text{dist}_i(s) = 0$  for all  $i$  and  $\text{dist}_i(t) = \infty$  at the end of the algorithm.

**Note:-**

In  $i^{\text{th}}$  iteration of the Ford-Fulkerson Algorithm or Edmonds-Karp Algorithm if  $P$  is the  $s \rightsquigarrow t$  in the residual graph  $G_{f_i}$  where  $e = (u, v) \in P$  and  $c_{f_i}(u, v) = \delta = \min_{e \in P} c_{f_i}(e)$  then the edge  $(u, v)$  is not present in the next residual graph  $G_{f_{i+1}}$ . Thus at least one edge disappears in each iteration of Ford-Fulkerson or Edmonds-Karp Algorithm.

Now we will prove following two lemmas which will help us to prove that the Edmond-Karp algorithm takes  $O(mn)$  iterations.

**Lemma 1.2.7**

At any iteration  $i$ ,  $\forall v \in V$ ,  $\text{dist}_i(v) \leq \text{dist}_{i+1}(v)$

**Lemma 1.2.8**

For any edge  $e = (u, v) \in E$  the number of iterations where either  $(u, v)$  appears or  $(v, u)$  appears is at most  $O(n)$  i.e.

$$|\{i: (u, v) \notin G_{f_i}, (u, v) \in G_{f_{i+1}}\}| + |\{i: (v, u) \notin G_{f_i}, (v, u) \in G_{f_{i+1}}\}| = O(n)$$

### 1.3 Preflow-Push/Relabel Algorithm