Note for Max-flow

Recap

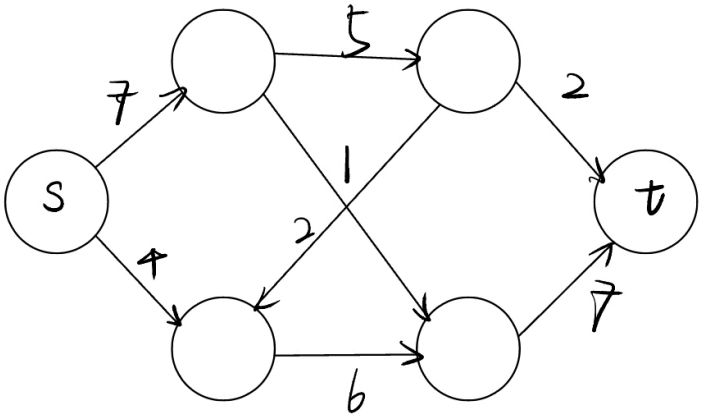
Part 1

- Max-flow is an algorithm that we can solve in polynomial time => It is fast and good.
- ***What is a max-flow problem?***

Before that, we are about to know what is the flow network.

- ***What is the flow network?***

Draw a picture



Basically, we have a starting point or a ***source*** called S, and an end point T called ***sink***. There are some cities each vertex including s and t. => ***s and t***

- ***transport goods***

(1) We can imagine that we are transporting goods from s here to t,

(2) but it is ***not allowed*** to accumulate goods in these intermediate cities. => ***So, whatever goes into a city or in vertex has to go out again except for S & T***.

- ***Flow Conservation***

We can also think of it as a system of water pipes. So, you want to send as much water from S to T as possible. But there is a ***limit***. => In addition, to have what is called ***flow conservation***, ***whatever goes in has to go out*** in the above four nodes.

- ***Capacities***

We also have these numbers on the edges of the graph. And these numbers are called capacities.

For instance, 7 means that we can send in most seven units of whatever we want to send per unit of time. => Litter of water or Number of trucks per day.

- ***Application***

The max-flow problem ***can be used as a black box inside algorithms for other problems***. So if we can solve the max-flow problem, we can use an algorithm for that to help solve other problems.

- ***Formally Definition of Flow Network*** => (G, s, t, c), no self-loops and no antiparallel edges.

A flow network consists of a directed graph $G = (V, E)$, a source $s \in V$, a sink $t \in V \setminus \{s\}$, and a capacity $fn : c. V \times V \to \mathbb{R}$ such that $c(u, v) \geq 0, \forall u, v \in V$, and if $(u, v) \notin E$ then $c(u, v) = 0$.

Note: C. It maps vertex pairs to real numbers. Capacities can never be negative. If we have a vertex pair that is not an edge of the graph, then the capacity has to be 0.

Now we have a flow network but what we want to find in a flow network is a flow.

- **Special requirement for G**

Before we continue, we should declare one thing.

We require that G has no self-loops and no anti-parallel edges.

Self-loop means an edge with the endpoints equal to each other. So it connects a vertex to itself.

Antiparallel edges mean that you have a pair of distinct vertices that have edges in opposite directions. => Solution is to add an extra vertex.

- ***Flow***

Definition of Flow => A flow is a function takes pairs of vertices maps them to real number the flow values. And it should satisfy the following constraints.

A flow in G is a $fn : f. V \times V \to \mathbb{R}$ such that,

1. $0 \leq f(u, v) \leq c(u, v), \forall u, v \in V$ and this is ***Capacity Constraints***.
2. $\sum_{v \in V} f(u, v) = \sum_{v \in V} f(v, u), \forall u \in V \setminus \{s, t\} \Leftrightarrow \sum_{(u,v) \in E} f(u, v) = \sum_{(v,u) \in E} f(v, u)$ and this is ***Flow Conservation.***

   Why it is equivalent?

   We have zero capacity on all pairs that are not connect all ordered pairs that don't correspond to an edge. So whether we sum over all the edges makes no difference because there is no contribution from this u, v is not in edge and the same on the right side.

If we have these two types of requirements satisfied, then we call this function of flow.

- ***Quality of our flow***

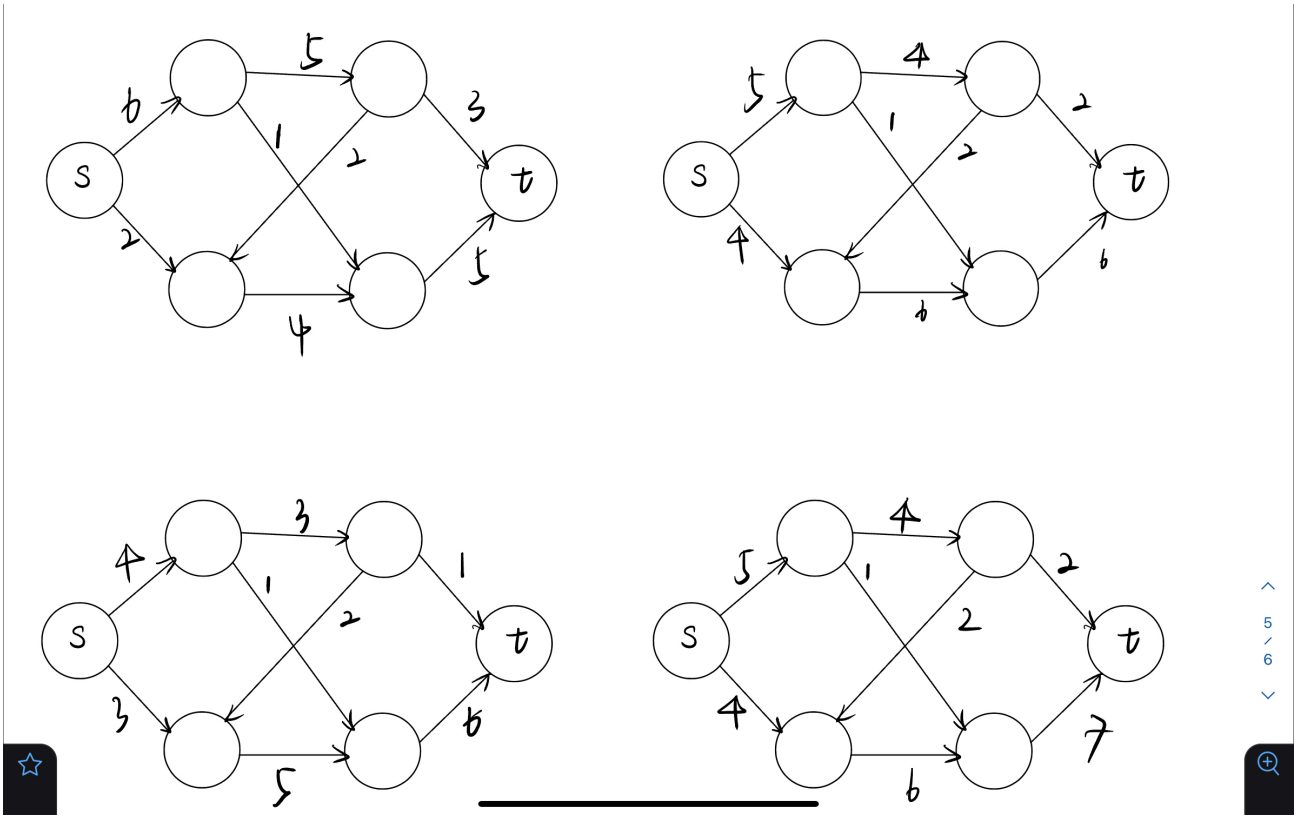
However, in the example, we want to transport as much as possible from $s \to t$. So we need to define something that expresses how good our flow is.

The value $|f|$ of f is defined as $|f| = \sum_{v \in V} (f(s, v) - f(v, s))$ => (total flow leaving s - total flow entering s).

- ***What is Max-flow?***

A max-flow is a flow of maximum value. We want to maximize the equation above.

- ***Candidate flow and the*** $|f|$



The first one is not a flow as it violates the capacity constraint. The second one is not a flow as it violates the flow conservation where the right bottom node, 7 in and 6 out.

The third one is a flow with $|f| = 7$, as the source sends out 7 units, while there is nothing in the source. The fourth one is a flow with $|f| = 9$.

- ***How do we find such a maximum flow?***

To solve this question, we need to be familiar with the following concepts first.

- ***Residual Capacity*** => Why do we need the special requirement above?

Given flow f in G, the residual capacity $c_f : V \times V \to \mathbb{R}$ is defined as,

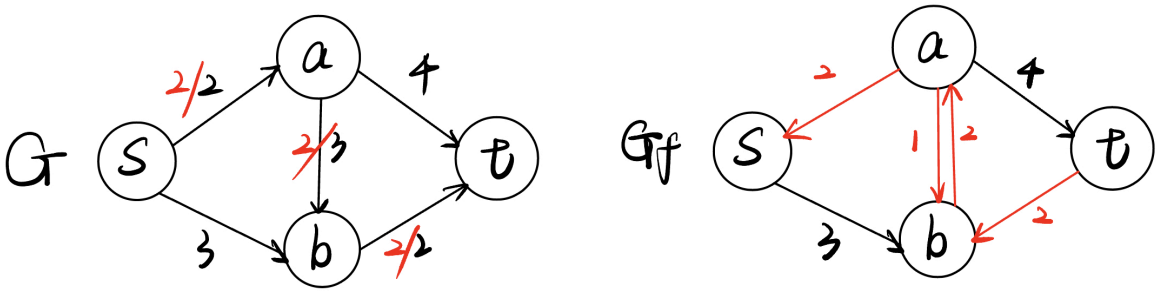
$$c_f(u,v) = \begin{cases} c(u,v) - f(u,v), & if\ (u,v) \in E \\ f(v,u), & if\ (v,u) \in E => cancellation \\ 0, & otherwise \end{cases}$$

- ***Residual Network***

The residual network $G_f = (V, E_f)$ where $E_f = \{(u,v) \in V \times V | c_f(u,v) > 0\}$.

G_f is a flow network with a capacity function c_f. G_f is allowed to have antiparallel edges.

- ***Practice***

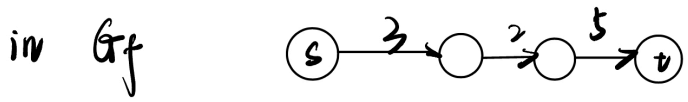


Then, let's go back how to find a maximum flow.

Ford-Fulkerson(G) => Informal

1. $f \leftarrow 0$ => It initializes the flow to be 0 everywhere.
2. while there is an augmenting path $p = s \rightsquigarrow t$ in G_f

   find a max flow f_p along p. (We need to find the bottleneck, the below example is $f_p = 2$)

   $f \leftarrow f \uparrow f_p$ (f is augmented with f_P)
3. return f

Here we only call it a method instead of an algorithm is that we don't specify how we pick this augmenting path.

Right now, let's define this augmentation symbol.

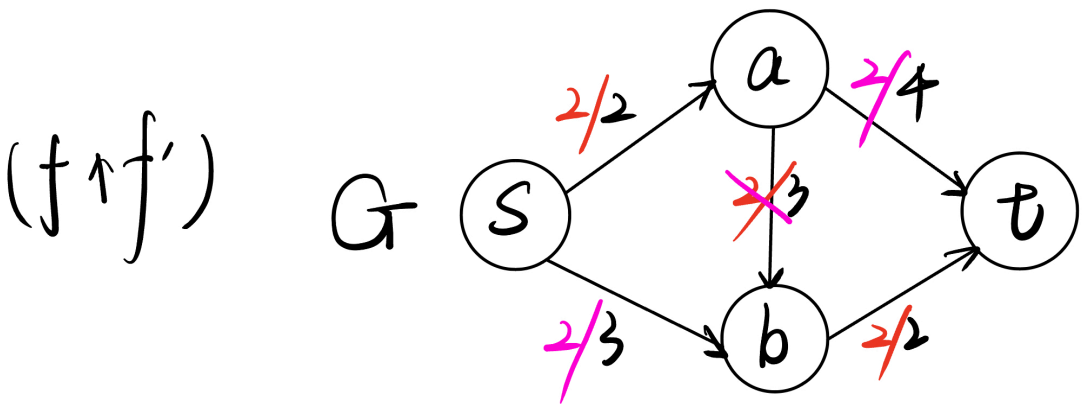
- ***Augmented Flow***

Given flow f in G and given flow f' in G_f. => f' is what we called f_p before. => But why do we need to minus the flow that we send in the opposite direction?

Then the augmented flow $f \uparrow f' : V \times V \rightarrow \mathbb{R}$ is,

$$(f \uparrow f')(u,v) = \begin{cases} f(u,v) + f'(u,v) - f'(v,u), & if\ (u,v) \in E \\ 0, & otherwise \end{cases}$$

- ***Augmented Practice***



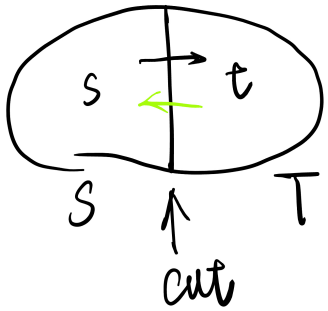
- ***Max-flow-min-cut Theorem***

Why the Ford-Fulkerson can find the max-flow? There is a theorem which is called max-flow-min-cut theorem.

Before we state the theorem, there are still some definitions.

- ***Cut***

A cut is a partition of V into subsets S and T such that $s \in S$ and $t \in T$.



- ***Net Flow*** => The total flow going from $S \to T$ minus the total flow going from $T \to S$.

Given a cut (S, T), we define the net flow across (S, T) as $f(S, T) = \sum_{u \in S} \sum_{v \in T} (f(u, v) - f(v, u))$.

- ***Capacity of the Cut*** => We only look at the edges going from $S \to T$ and add the capacities.

The capacity $c(S, T)$ is $c(S, T) = \sum_{u \in S} \sum_{v \in T} c(u, v)$.

- ***Lemma 1***

$f \uparrow f'$ is a flow in G of value $|f \uparrow f'| = |f| + |f'|$.

Proof

How do we show this is a flow first? => ***Capacity Constraints***

Let $(u, v) \in E$. Then,

$$\begin{aligned}(f \uparrow f')(u, v) &= f(u, v) + f'(u, v) - f'(v, u) \\ &\leq f(u, v) + c_f(u, v) - 0 \\ &= f(u, v) + c(u, v) - f(u, v) = c(u, v)\end{aligned}$$

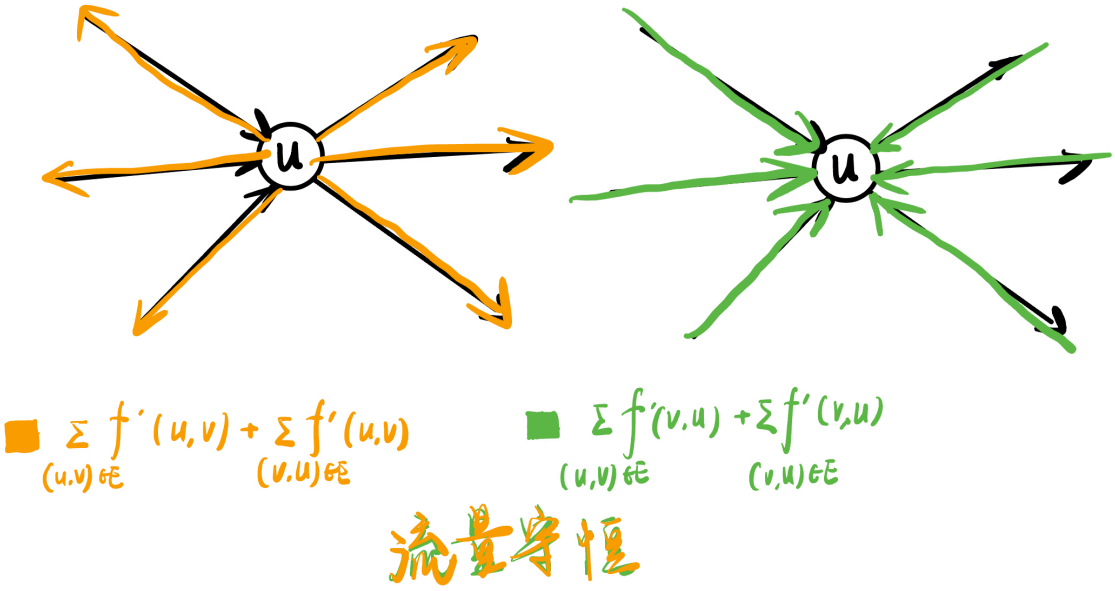
$$\begin{aligned}(f \uparrow f')(u, v) &= f(u, v) + f'(u, v) - f'(v, u) \\ &\geq f(u, v) - c_f(v, u) => \textit{ignore the } 2^{nd} \textit{ term} \\ &= f(u, v) - f(u, v) = 0\end{aligned}$$

Then, => ***Flow Conservation***

Let $u \in V \setminus \{s, t\}$. Then,

$$\sum_{(u,v) \in E} (f \uparrow f')(u, v) - \sum_{(v,u) \in E} (f \uparrow f')(v, u) = \sum_{(u,v) \in E} (f(u, v) + f'(u, v) - f'(v, u)) - \sum_{(v,u) \in E} (f(v, u) + f'(v, u) - f'(u, v))$$

The first term can be canceled as they represent the flow going out from u and going in to u. => ***Flow Conservation***



Similarly, the other two are exactly reversed. The total amount of f' flow going into u.

Therefore, everything got canceled.

$$\sum_{(u,v)\in E}(f\uparrow f')(u,v)-\sum_{(v,u)\in E}(f\uparrow f')(v,u)=0$$

$$|f\uparrow f'|=\sum_{(s,u)\in E}(f\uparrow f')(s,u)-\sum_{(u,s)\in E}(f\uparrow f')(u,s)$$

$$=\sum_{(s,u)\in E}(f(s,u)+f'(s,u)-f'(u,s))-\sum_{(u,s)\in E}(f(u,s)+f'(u,s)-f'(s,u))$$

$$=|f|+|f'|=>\ by\ definition\ of\ s,\ 0\ to\ enter$$

- ***Lemma 2***

Given flow f in G and given any cut (S,T), $f(S,T)=|f|$. => The net flow going out of the source is equal to the net flow going into the sink. (consider the $T=t$).

Proof

$$f(S,T)=\sum_{u\in S}\sum_{v\in T}(f(u,v)-f(v,u))$$

$$=\sum_{u\in S}\sum_{v\in S}(f(u,v)-f(v,u))+\sum_{u\in S}\sum_{v\in T}(f(u,v)-f(v,u))$$

The first term is 0 as when (u,v) switched their roles in the sum, we are subtracting that term again. The first term can be understood that the sum of the edge that both endpoints are in S.

Simplify

$$=\sum_{u\in S}\sum_{v\in V}(f(u,v)-f(v,u))=>Net\ Flow\ Going\ Out\ Of\ u$$

$$=|f|$$

We have flow conservation in u. The net flow going out of u is 0 except possibly when u is equal to s. So, the sum is the amount of flow going out of the source minus the amount of flow going into the source. And that is equal to the value of the flow.

- ***Corollary 1***

For any flow f and any cut (S, T), $|f| \leq c(S, T)$. => The amount of flow that can go from s (the value of the flow) is upper bounded by the capacity of the edges that go from S to T, it is intuitively correct as the capacity is a bottleneck for how much flow you can send from $S \rightarrow T$.

Proof

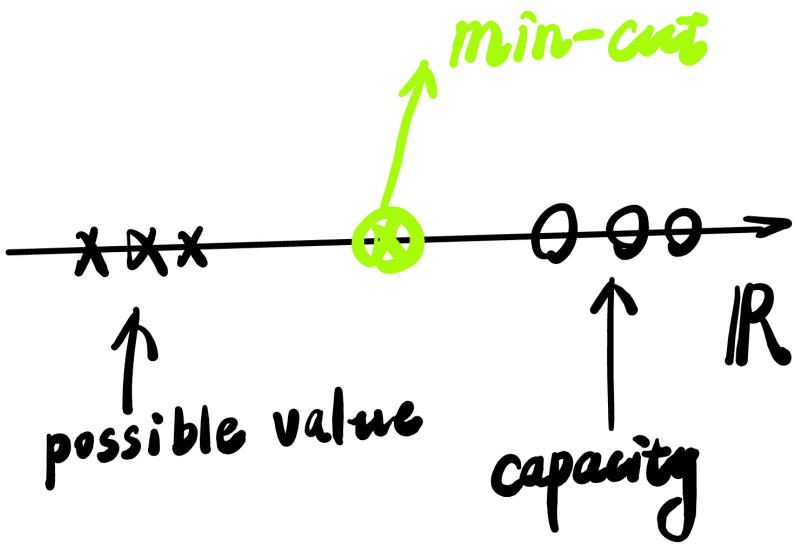
$$
\begin{aligned}
|f| = f(S, T) &=> Lemma\ 2 \\
&= \sum_{u \in S} \sum_{v \in T} ((f(u, v) - f(v, u))) \\
&\leq \sum_{u \in S} \sum_{v \in T} (c(u, v) - 0) \\
&= c(S, T)
\end{aligned}
$$

Now, let us state the max-flow-min-cut theorem.

- ***Max-flow-min-cut Theorem***

First, let us illustrate the corollary that we have proved above.

Illustrate the Corollary



1. Put in all possible flow values that you can have in your graph.
2. Put in all the possible capacities of cuts that you can have.

The corollary is saying that all the XS are to the left of all the circles. No flow value can be larger than any capacity of a cut.

While the max-flow-min-cut theorem says that if you take the largest flow value here and you take the smallest capacity of the cut, they actually meet.

Statement

Let f be a flow in G. Then the following 3 statements are equivalent.

1. f is a max flow
2. there is no augmenting path (in G_f)
3. $\exists$ cut (S, T) such that the value of the flow $|f| = c(S, T)$

Why do we show interest in this theorem? What are the relationships between the FF method?

- If Ford-Fulkerson terminates that means there's no augmenting path, then the max-flow-min-cut theorem tells f is a max flow.

The big question is, does FF always terminate?

Part 2

- ***Proof of Theorem***

$1 \implies 2$ ***Contradiction + Lemma 1***

Assume for contradiction that, f is a max flow and that there is an augmenting path p (in G_f).

Then by ***Lemma 1***, $f \uparrow f_p$ is a flow in G of value $|f \uparrow f_p| = |f| + |f_p| > |f|$. This means, $f \uparrow f_p$ has a larger value.

However, f is a max flow and that is contradicted by the assumption.

Therefore, $1 \implies 2$.

$2 \implies 3$ ***Construct cut + Single Edge***

Let $S = \{v \in V | v \ is \ reachable \ from \ s \ in \ G_f\}$. So S is simply **the vertices that you can reach from the source** in the residual network by some path. 在残存网络中，集合S中的任意一个点到原点都有路径。 Let $T = V \setminus S$.

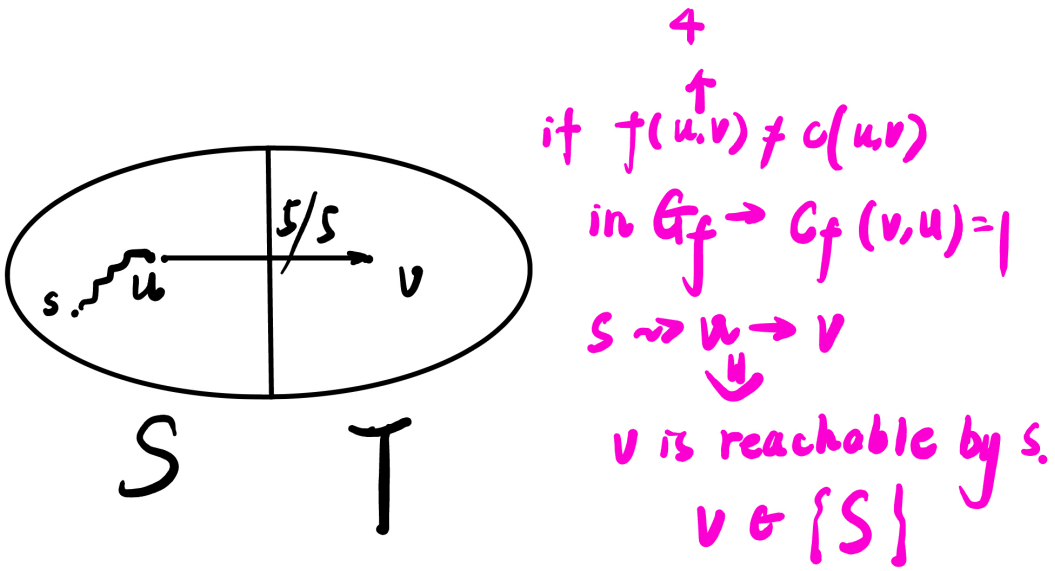
Now we need to argue that this is a cut.

It is clearly a partition of the vertex set, but then we needed the source to be in S and the sink to be in T.

Is there a path is as reachable from S in G_f that clearly? => It just takes the empty path, so it clearly s belongs to S.

What about the sink being in the T? => $t \in T$ because otherwise t would be reachable from s in G_f, contradicting (2) => Augmenting path.

Therefore, the cut has been constructed. Then, we need to argue why $|f| = c(S, T)$.



Assume $(u, v) \in E$. Then $f(u, c) = c(u, v)$. Above.

Now assume $(v, u) \in E$. We take the same case in the opposite direction. Therefore, $f(v, u) = 0$.

Otherwise, *v is reachable from s*, which means $v \in S$.

By ***Lemma 2***,

$$|f| = f(S, T) = \sum_{u \in S} \sum_{v \in T} (f(u, v) - f(v, u))$$
$$= \sum_{u \in S} \sum_{v \in T} (c(u, v) - 0) = c(S, T)$$

$3 \implies 1$ ***Corollary 1***

Let f' be any flow in G. By ***Corollary 1***,

$$|f'| \leq c(S, T) => cut\ from\ Theorem(3)$$
$$= |f| \implies f\ is\ a\ max\ flow$$

Therefore, $3 \implies 1$.

Running Time

- ***How fast of FF method?***

In general, F.F. does not terminate. => Capacities need to be irrational numbers for this to occur, otherwise, it always terminates.

Assume integer capacities. Then F.F. has a running time $O(E \cdot |f^*|)$ where f^* is a max flow.

Why it is true?

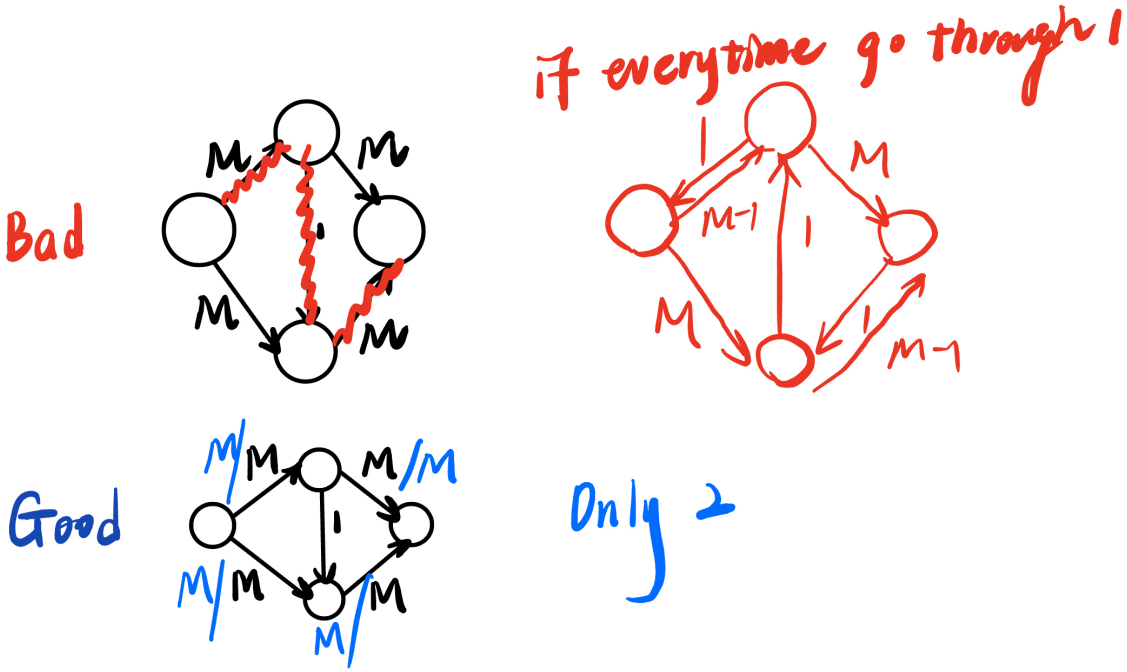
If all the capacities are integers, we know that F.F increases the value of the flow by at least 1 in each step. Therefore, the number of iterations is bounded by the value of the max flow. And in each step, we need to find an augmenting path and you can do any linear time search procedure to find a path from $s \to t$.

找增广路径的时间是线性的，边集合的大小。每次找到一个路径至少增加1，最多找到最大流的值个。

Is this fast or slow? => It depends on the value of the $|f^*|$, and also the $|E|$.

Example

Bad choice is every time, go through the middle path, then increment by 1 every time.



Good choice is not selecting the middle path. Only takes 2 steps.

- **Edmonds-Karp(G)** => E.K. No matter about the capacities look like, including irrational numbers.

Then, we introduce an algorithm which is called **Edmonds-Karp**. It is an implementation that when selecting a path always chooses the shortest path, for example, the algorithm will select the length of the path equal to 2 instead of the bad one with 3.

1. $f \leftarrow 0$ => Initialize flow to be 0.
2. while $\exists$ augmenting path

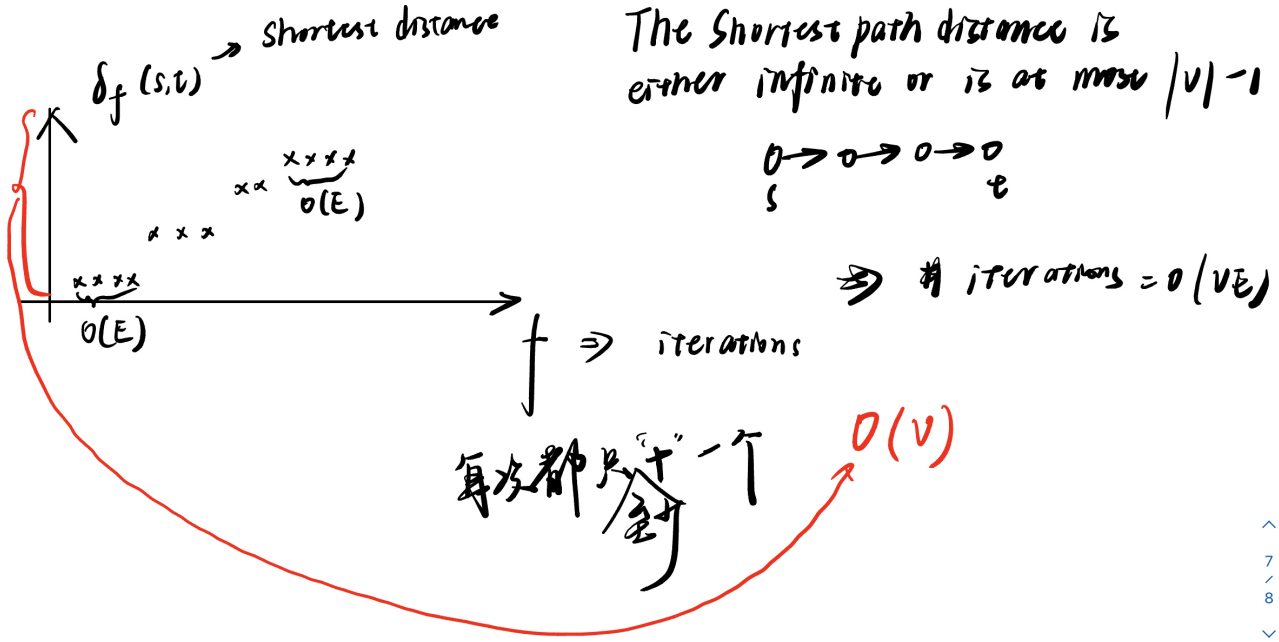
   let p be a shortest such path

   $f \leftarrow f \uparrow f_p$
3. return f

- ***Theorem 1***

The number of iterations of E.K. is $O(VE)$.

Proof



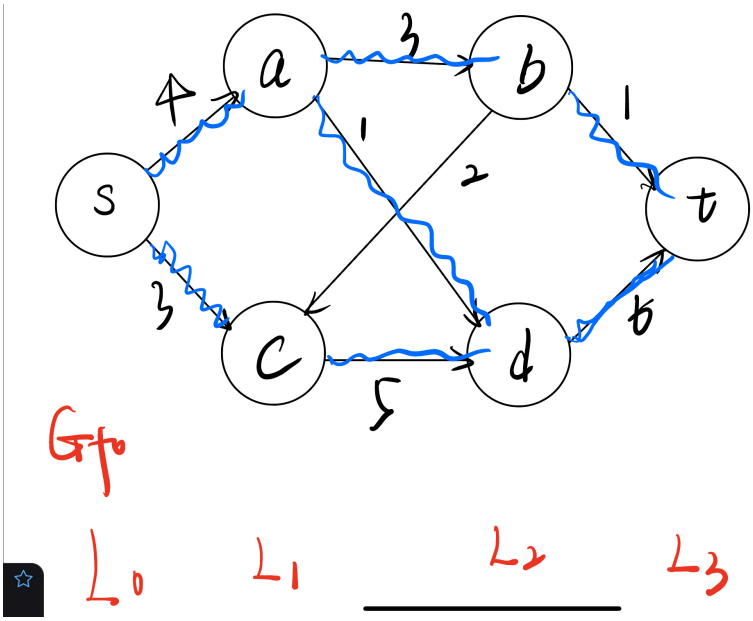
Then, let's prove in each step, the #iteration is bounded by $O(E)$.

But right now, we need to show that we cannot keep the distance the same for more than $O(E)$.

Consider consecutive flows $f_0, f_1, \dots f_k$ found by E.K. such that the $\delta_{f_0}(s,t) = \delta_{f_1}(s,t) = \dots = \delta_{f_k}(s,t)$.

For $d = 0, 1, \dots,$ let $L_d = \{v \in V | \delta_{f_0}(s,v) = d\}$.

A forward edge of G_{f_0} is an edge (u,v) in G_{f_0} such that $u \in L_d$, $v \in L_{d+1}$ for some d. A back edge of G_{f_0} is the reverse of a forward edge.



Forward edges are in blue, while $b \to c$ is not the forward edge as the distance from $s \to b = 2$, $s \to c = 1$.

- ***Claim 1***

For $i = 0, 1, \dots, k$, E.K. finds an augmenting path in G_{f_i} consisting only of forward edges of G_{f_0}.

Claim 1 implies that $k = O(E)$ since at least one forward edge disappears in each iteration. (as in each step, we are removing at least 1 blue edge from the G_{f_0}. While the number of blue edges is smaller than the number of edges. Remove at least 1 forward edge because we can always remove the bottleneck forward edge by replacing it to be the back edge in each step.)

Proof

It is clear for $i = 0$ as we can only use forward edges because it's the shortest path, we have to go one step forward each time.

G_{f_1} is obtained from G_{f_0} by removing forward edges and adding back edges since $\delta_{f_0}(s,t) = \delta_{f_1}(s,t)$. Thus ***Claim 1*** true for $i = 1$.

- ***Claim 2*** $\delta_{f_{k+1}}(s,t) \geq \delta_{f_k}(s,t)$

Proof

$G_{f_{k+1}}$ is obtained from G_{f_0} by removing forward edges and inserting back edges. Thus ***Claim 2*** holds.

Then, claim 1 and 2 can imply the proof above. => ***Theorem 1***.

- ***Corollary 2***

E.K. can be implemented to run in $O(VE^2)$. => BFS.

- ***Integrality Theorem***

Given integer capacities, F.F. will find an integer-value flow f, with $|f|$ an integer.