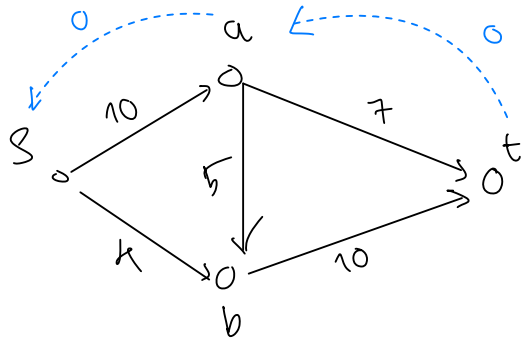
Flow and Cuts

Given a digraph $G = (V, E)$



s : source

t : sink

a, b can be source or sink

Properties of Flows and Cuts in Flow Network

- Flow network G has no self-loops and no parrallel edges.
- Each edge has a capacity $c(u, v)$
- Each edge has a "flow" $0 \leq f(v, v) \leq c(u, v)$
- Convient to assume that:

  edge $(u, v) \in E \iff$ edge $(v, u) \in E$

  ($c(v, u) = 0$ if not specified)
- Total flow from set of sources to set of sinks, denoted as:

$$f(X, Y) = \sum_{(u,v) \in E} f(u, v)$$

$$\text{s.t.} \begin{cases} u \in X \\ v \in Y \end{cases}$$

Flow Conservation

- Total flow going in equals Total flow going out for any vertex that is not source or sink:

$$\sum_{(u,v) \in E} f(u, v) = \sum_{(v,u) \in E} f(v, u)$$

- "Cuts" is a partion of V into pair of sets (source, sink)

  Example, some valid cuts from the given example are:

  $(\{s\}, \{a, b, t\})$

  $(\{s, a\}, \{b, t\})$

  $(\{s, a, b\}, \{t\})$

  Total cuts a graph can have: $2^{|V|-2}$

- The zero-flow network: basically a digraph G where capacity $c(u, v)$ of any edge is zero

Net Flow:

Difference between total flow forward and total flow backward between 2 cuts S and T:

$$\text{net flow} = f(S,T) - f(T,S)$$

Lemma 1 Equality of Net Flow accross 2 similar Cuts

Let (S,T) and (S',T') be 2 cuts s.t. $|S \oplus S'| = 1$

Then: $f(S,T) - f(T,S) = f(S',T') - f(T',S')$

Proof:

Since $|S \oplus S'| = 1$, $\exists$ a lone vertex v.

W2OG, assume $S' = S + v \iff T' = T - v$

Net flow in (S,T) cut:

$$\begin{aligned}
& f(S,T) - f(T,S) \\
= \; & f(S+v,T) - f(\{v\},T) - f(T,S+v) - f(T,\{v\}) \\
= \; & f(S',T) - f(T,S') \\
= \; & f(S',T-v) + f(S',\{v\}) - f(T-v,S') - f(\{v\},S') \\
= \; & f(S',T') + f(T',S')
\end{aligned}$$

Lemma 2 Equality of Net Flow accross Distinct Cuts

From Lemma 1, we can easily prove that any two cuts in a given flow network $G = (V,E)$ have the same net flow.

The Value of a Flow

Lemma 2 suggests that we dont need to care ab which cut associate with the flow's value, be we know that we would get the same result.

Value of Flow

The value of flow f is net flow out of s

. Equals to net flow of f accross all possible cuts $(\{s\}, V-s)$

Maximum Flow Problem - Minimum Cut Problem

Maximum Flow Problem:

Assigning a flow value $f(u,v)$ to every edge (v,v) s.t.:

- Capacity Constraints:

$$0 \leq f(v,v) \leq c(u,v)$$

- Flow Conservation:

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

Are respected, the goal is to maximize total flow:

$$\text{Total flow} = \sum_{(s,v)} f(s,v)$$

or

$$\text{Tota flow} = \sum_{(v,t)} f(v,t)$$

Minimum Cut Problem

Find a cut (S,T) s.t. the total capacity is minimized

$$\text{Total capacity of cut } (S,T) = \sum_{\substack{u \in S \\ v \in T}} c(u,v)$$

Maximum Flow Problem = Minimum Cut Problem

By definition, Minimum Cut Problem is essentially "identifying the bottleneck in the network" - the set of edges that are also the upper bound of "the maximal flow we can push through the network"

Max-flow min-cut theorem

Maximum Flow = Minimum Cut

Ford - Fulkerson Algorithm

High Level

Create a residual graph G', which is the same as the original graph but with residual edges (reverse edges)

Repeatedly do the followings:

- Find augmenting path - any path from s to t, including the residual edges

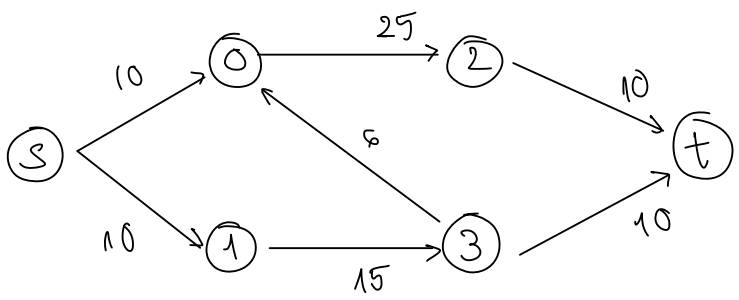
  Note: only path with no edge reached its capacity is valid
- Update all the edges in the path by the remaining capacity of the "smallest edge" - the bottleneck

Remaining capacity: $c_f(u,v) = c(u,v) - f(u,v)$

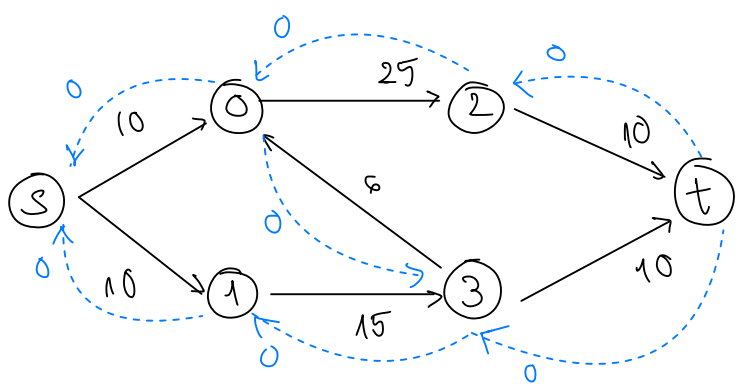
Terminate when there is no more augmenting path, the total final flow value on each edge is the Max-flow, and also Min-cut

Example:

Given graph G =



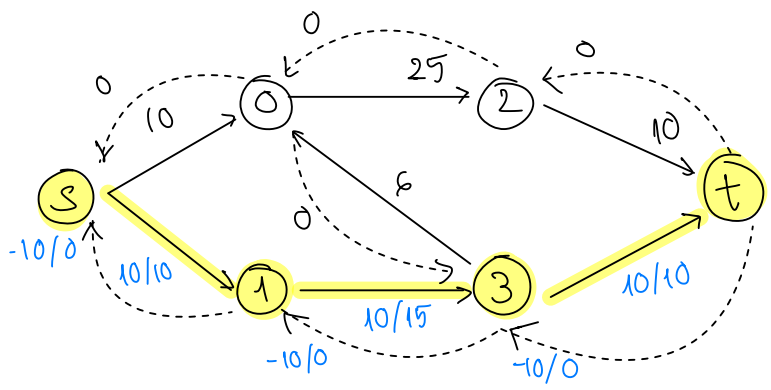
Then residual graph G' =



First iteration:

Augmenting path:

s → 1 → 3 → t



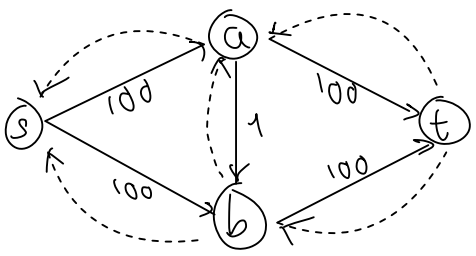
Increase flow by minimum remaining capacity:

$f(u,v) := f(u,v) + \min\{c_f(u,v)\}$

Continue until there is no more augmenting path.

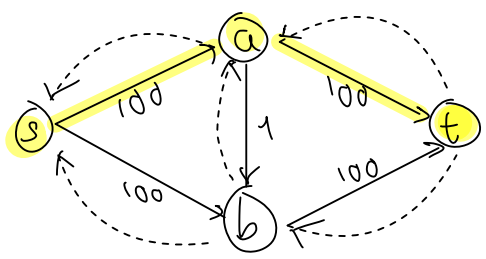
Problem with Ford - Fulkerson

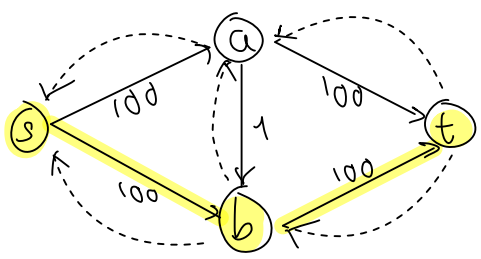
Given this example



Can easily see that the maximum flow of this network is

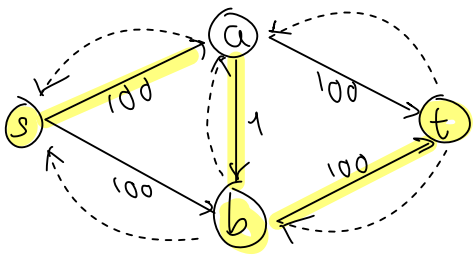
$100 + 100 = 200$ with these 2 paths $\Rightarrow$ takes 2 iterations



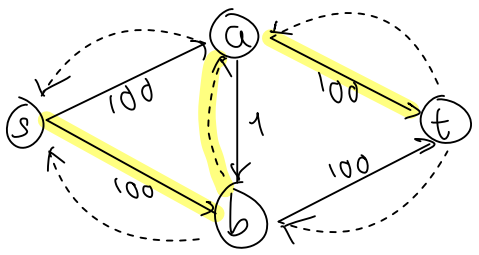


$\Rightarrow$ takes 2 iterations

But it is also possible for DFS to include this path



or



$\Rightarrow$ Takes at most 200 iterations

Time Complexity of Ford - Fulkerson

- Assume all capacities are integer, then each iteration of Ford - Fulkerson increase the total flow by at least 1 unit. Let f^* denote max-flow then: $O(f^*)$
- Ford - Fulkerson use DFS to find augmenting path: $O(E)$

  $\Rightarrow O(E \cdot f^*)$

Note: If capacities $\in \mathbb{R}$, then the "remaining capacity" might be so small, that updating the flow with it never reach the capacity, leading to the algorithm runs indefinitely

Edmonds - Karp Maximum Flow algorithm

Very much the same as Ford - Fulkerson but use BFS to find the shortest augmenting path, instead of randomly chose any path

⇒ Edmonds - Karp solves the problems with Ford - Fulkerson:

- Better actual runtime
- Converge with real values

Time Complexity of Edmonds - Karp

- Number of iterations: $O(E \cdot V)$
- Each iteration can be implemented in $O(E)$ --> updating the edges

⇒ Total complexity $O(E^2 \cdot V)$ → strong polynimial

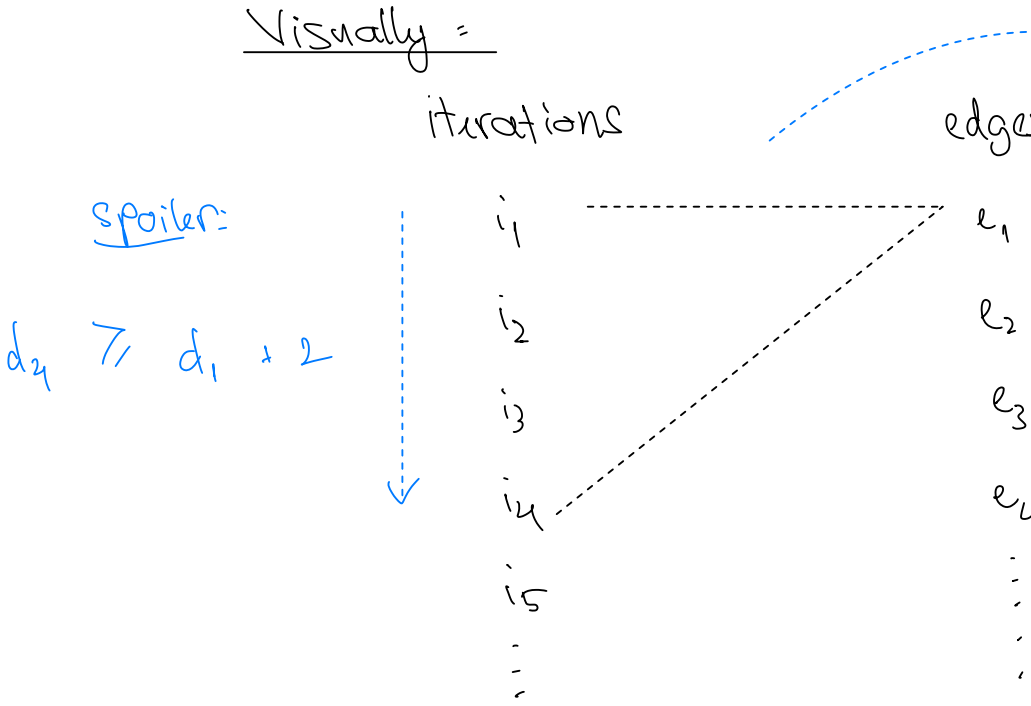
Proof for number of iterations:

- Bottleneck count of an edge: number of iterations i s.t. edge is a bottleneck edge

We will prove the bottleneck count of an edge is $O(V)$

- So that, with the knowledge of "each iteration increments the bottleneck count of at least one edge", we conclude that the number of iterations is $O(E \cdot V)$

Visually:



e_1 is bottleneck edge in iterations i_1, i_2, i_3

($\Rightarrow$ e_1' bottleneck count = 3

⇒ Strategy is to bound e_1' bottleneck count, then we can bound the iterations

Prove bottleneck count is bounded by $O(V)$

- Define $d(v, i)$ as the minimum length path from s to v after i^{th} iteration.

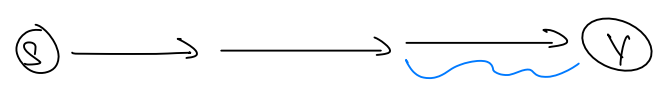
Lemma: For all v in V, $i \geq 1$, we have $d(v, i-1) \leq d(v, i)$

Proof by contradiction

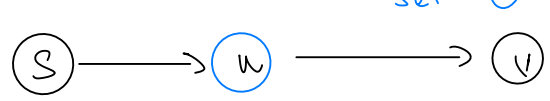
- Assume after i iterations, $d(v, i-1) > d(v, i)$

Visually. Augmenting path at $i-1$ and i^{th} iterations

$i-1^{th}$ iteration:



i^{th} iteration:



- Let set $U = \{ v : d(v, i-1) > d(v, i) \}$ (violator set)

$\Rightarrow$ $v \in U$ to be vertex with minimum distance in U

- Let vertex u be the vertex right before vertex v in i^{th} iteration

Let k denote $d(v, i)$

$$\Rightarrow \begin{cases} d(v, i) = k \\ d(u, i) = k-1 \end{cases}$$ (prefix of shortest path is itself shortest pat

Definition of v: minimum $d(v, i)$

And distance $s \to v$ is larger than distance $s \to u$

$\Rightarrow$ $d(u, i-1) \leq d(u, i) = k-1$

(up until vertex u, the shortest distance is non-decreasing)

- Now we show contradiction in the definition of v, which implies

$d(v, i-1) > d(v, i) = k$

The goal is to show $d(v, i-1) \leq k$.

We know: $c_f(u, v) \geq 0$ (constraint of c_f)

- Case 1: $c_f(u, v) > 0$

$\Rightarrow$ (u, v) appears in some augmenting path

$\Rightarrow$ $d(v, i-1) \leq d(u, i-1) + 1$

$= k$

Contradicts $d(v, i-1) > k$

. Case 2: $c_f(u, v) = 0$

We have $c_{f'}(u, v) > 0$ (after i iterations, definition of u)

⇒ Residual capacity of edge (u, v) increase from 0 to >0 in 1 iteration from $(i-1)$ to i.

⇒ More flow is added to residual edge (v, u) in 1 iteration from $(i-1)$ to i

⇒ Residual edge (v, u) appears in the augmenting path at iteration i, or after $i-1$ iterations

Recall the process: 1. select augmented path
2. update the flow

⇒ $d(v, i-1) = d(u, i-1) - 1$

$= k - 2$

$\leqslant k$

Contradicts $d(v, i-1) > k$

Conclusion: Shortest distance from source to any vertex in flow network is non-decreasing through the iterations

Now we have the tool to bound the number of bottleneck edges, which in turns bound the number of iterations in Edmonds-Karp

Lemma Bottleneck

If an edge (u, v) is a bottleneck at iteration i in some path And shortest distance from source to u and v at iteration $i-1$ are: $d(u, i-1) = k$

$d(v, i-1) = k+1$

Then only after some unknown iterations $i+1, i+2, \ldots, i'$ where

$d(u, i'-1) \geqslant d(u, i-1) + 2$

or $d(v, i'-1) \geqslant d(v, i-1) + 2$

can edge (u, v) becomes a bottleneck again in some path

Proof:

- We know: (u, v) appears in augmenting path at iteration i as bottleneck ⇒ Residual capacity $c_f(u, v) = 0$ after iteration i

- Easy to see that: edge (u,v) can only be available to be included in any path after some iteration i' $(i' > i)$ if its residual capacity is "restored", $c_{f'}(u,v) > 0$ after i' iteration
- What happen during iterations $i \to i'$ so that edge (u,v) can "restore" its residual capacity ?

  $\Rightarrow$ Only when we add flow to residual edge (v,u) at some iterations between $i \to i'$

- Let $\begin{cases} d(u, i-1) = k \\ d(v, i-1) = k+1 \end{cases}$

  Technical lemma: $d(v, i'-1) \geq d(v, i-1) = k+1$

  Edge (v,u) is on some p between $i+1 \to i'$

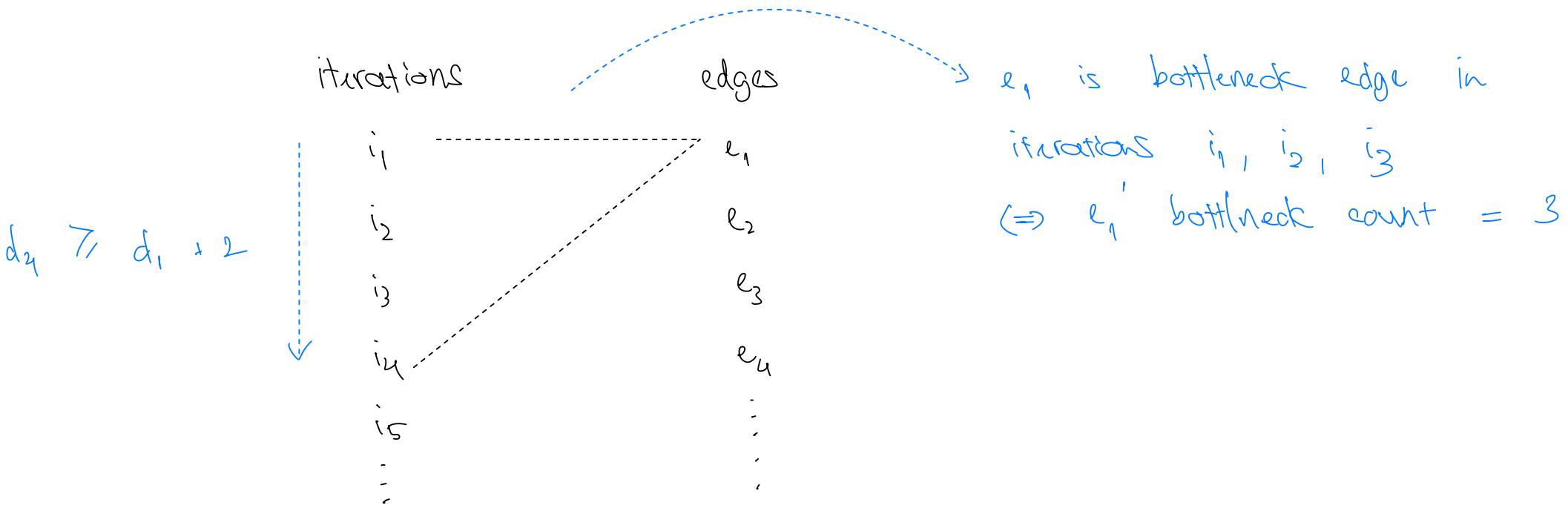
  $\Rightarrow d(u, i'-1) = d(v, i'-1) + 1$

  $\Rightarrow$ $d(u, i'-1) \geq k+2$

  End of proof.

Conclusion: With Technical lemma and Bottleneck lemma, we can now bound the number of iterations Edmonds-Karp.

Recall this visualization:



Bottleneck Lemma bound each bottleneck by shortest distances.

i.e every time a bottleneck count increase, the shortest distance from source to that bottleneck edge increase at least 2.

In other words, when:

bottleneck count of (u,v) increase $0 \to 100$

$\Leftrightarrow$ shortest distance from source to vertex $u \geq 2.99$

We know that: distance either $\begin{cases} \leq |V|-1 \\ \infty \quad \text{(unreachable)} \end{cases}$

So, any edge's bottleneck count is bounded by $|V| - 1$

Hence number of iterations is also bounded by $O(|V|)$