Lecture 12: Network Flow II

Polynomial-time Algorithms for Max Flow

Network Flow Recap

Directed graph

Capacity: $0 \le f(u,v) \le c(u,v)$

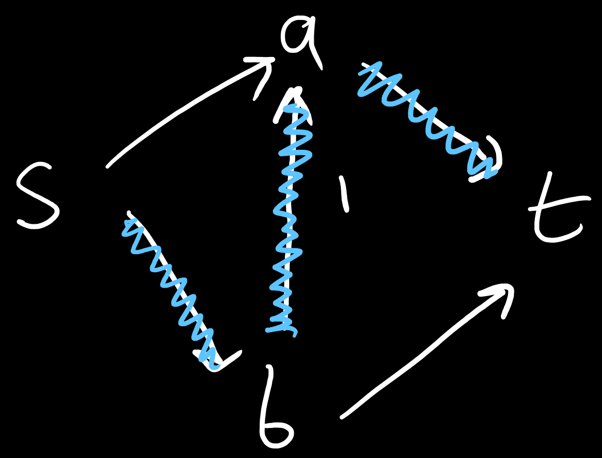
Conservation: "flow in" = "flow out" for all $v \notin \{s,t\}$

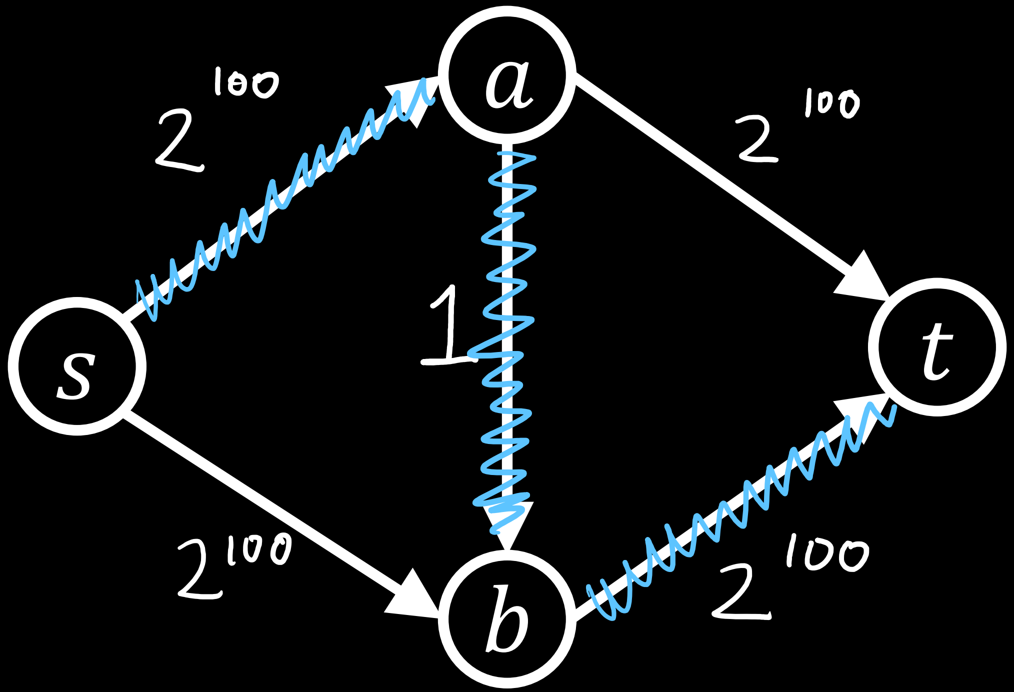
Goal: Find <u>max</u> s-t flow

Ford-Fulkerson: while $\exists$ augmenting path, push flow

Motivation: Worst case for Ford-Fulkerson

Runtime: $O(mF)$





Runtime is tight!

Goal: Polynomial time (regardless of capacities)

Edmonds-Karp Algorithm

Shortest Augmenting Paths (S.A.P.)

Ford-Fulkerson

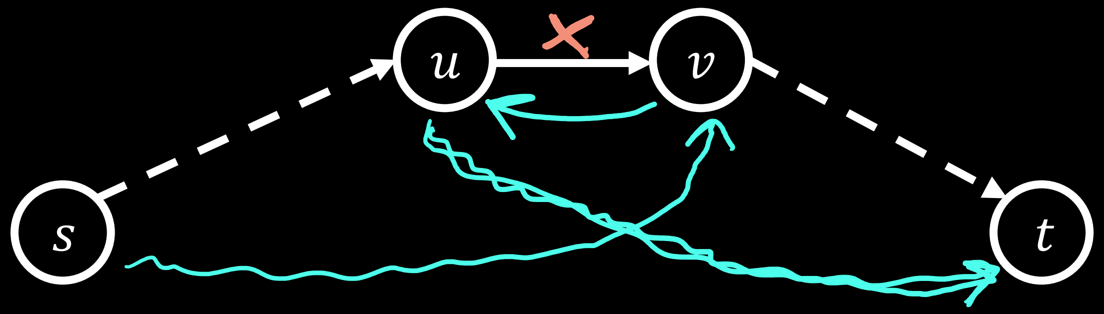
Use S.A.P.S instead of arbitrary augmenting paths
(e.g. Use BFS to find paths)

Edmonds-Karp Algorithm (Analysis)

Theorem: Takes $O(nm)$ iterations

Let d be the distance from s to t in G_f

Claim: d never decreases



By contradiction, d does not decrease.

$\square$

Edmonds-Karp Algorithm (Analysis)

Claim: After m iterations, d must increase

Each augmenting path deletes an edge

After m augmentations, all shortest paths have been used up

Either done, or d increased. $\square$

$d \leq n$ BFS takes $O(m)$ - up to nm of them $O(nm^2)$

Redundancy in Edmonds-Karp

- might do m augmenting paths for same d
- BFS tells you <u>all</u> the shortest paths

Can we find many S.A.Ps from one BFS?

Blocking Flows

<u>Def</u> A blocking flow is a set of augmenting paths of length d such that after augmenting, no more paths of length d.

Dinic's Algorithm

while flow is not maximum
find a blocking flow and augment them

Goal: Find blocking flows

The "Layered Graph"

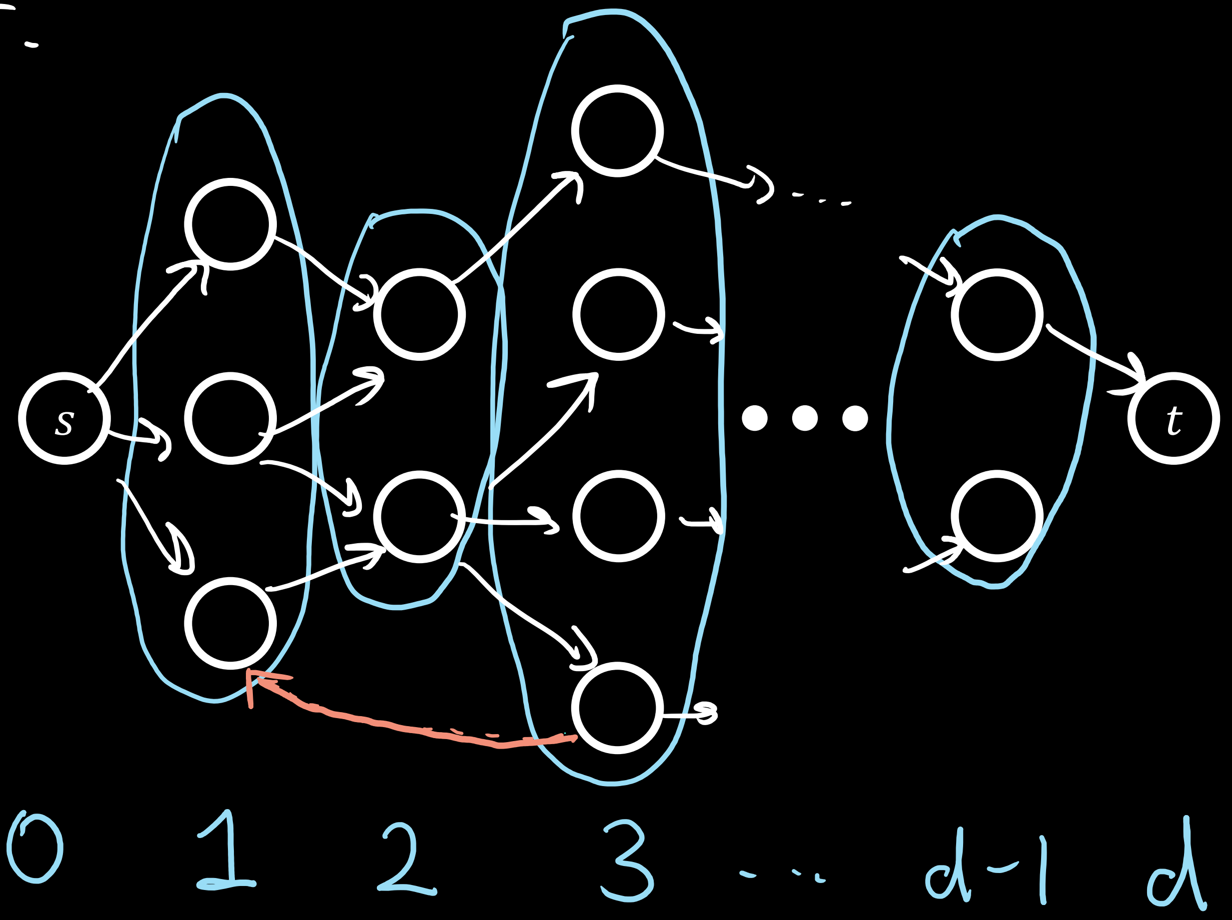
Care about (u, v) s.t.

$$d[v] = d[u] + 1$$

Naively: $O(m^2)$

m augmenting paths

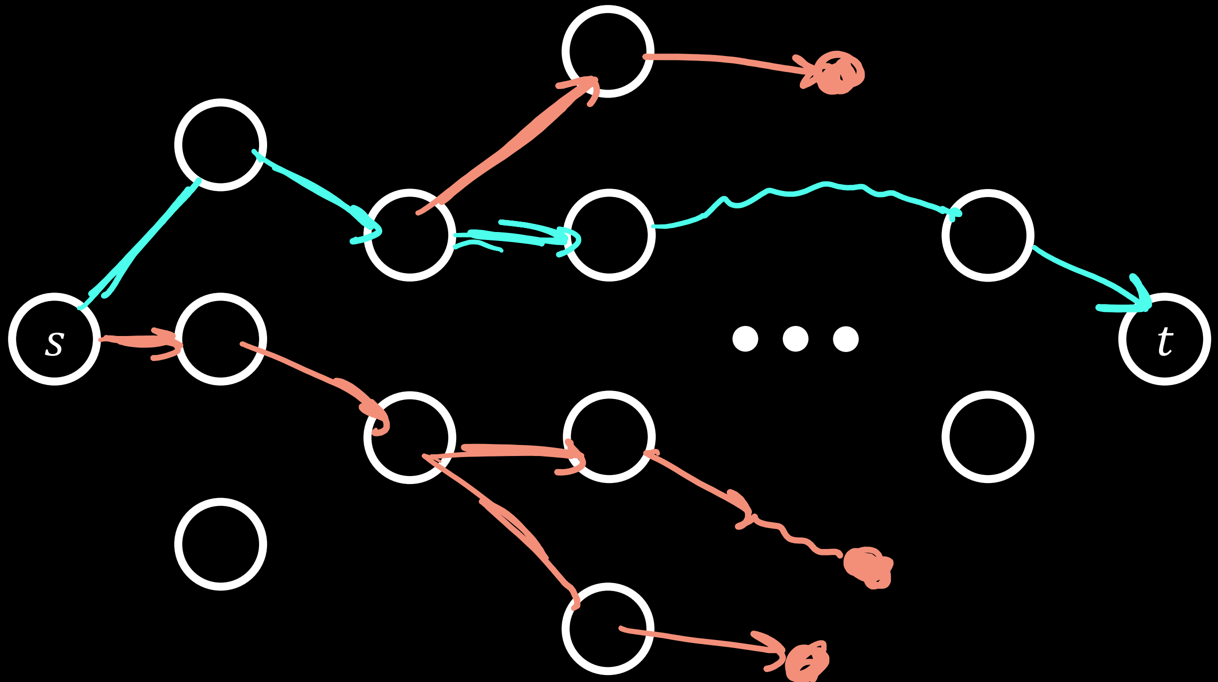
m time per DFS



Finding blocking flows

Green path takes $O(d) = O(n)$

Every time we find an edge that doesn't work mark it as "dead"



Runtime analysis

Each DFS takes $O(d)$ + # dead edges marked

Total cost $= \sum_{i=0}^{m} O(d)$ + # dead in iteration i

$= O(nm)$ + # total dead edges ever

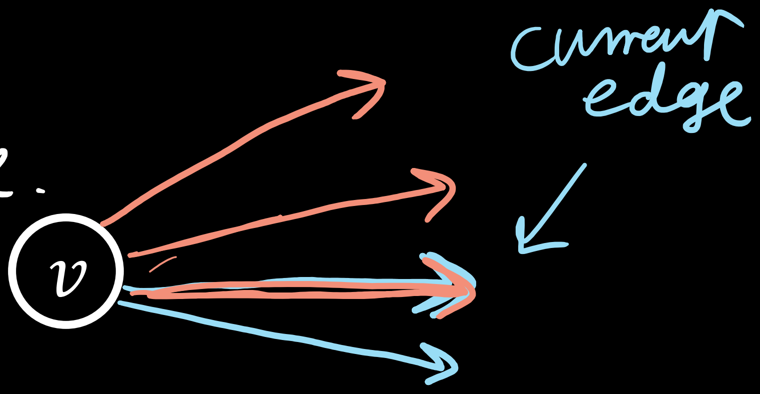
$= O(nm + m)$

$= O(nm)$ to find one blocking flow

n blocking flows $\Rightarrow$ Dinic's takes $O(n^2 m)$ time

Implementation

- Store index of "current edge"
- Increment counter when dead edge.



current_edge[v] = $\cancel{3}$ 4

Dinic's on unit-capacity graphs

Capacities = 1

<u>Lemma 1</u> Blocking flow can be found in $O(m)$

All edges are used in at most one path

$\Rightarrow$ Total length of all paths = m

Search cost $\leq m + O(m) = O(m)$ $\square$

Successful

marking dead

Dinic's on unit-capacity graphs

<u>Lemma 2</u> Need $\leq 2\sqrt{m}$ blocking flows

Iteration k $(d \geq k)$

- Paths are length $\geq k$ $(= d)$
- Path are edge disjoint
- At most m/k paths (at most m/k more flow)

blocking flows $\leq k + m/k$

$\leq 2\sqrt{m}$.

$k = m/k$

$k = \sqrt{m}$

$\Rightarrow$ Takes $\boxed{O(m\sqrt{m}) \text{ time}}$