Advanced Algorithms

Lecture 13: Flows, cuts, review

Announcements

- **Mid-term on Thursday (in class)**
- HW 2 grades out — direct-message TAs + instructor on Piazza (or write to Vivek).
- HW 3 due Wednesday after the fall break

Last week

- Shortest path in graphs (see notes)
  - Dijkstra's algorithm ($O(m+n) \log n$) time — imitation of BFS
  - DP based, "Bellman-Ford" algorithm — $O(n\,(m+n))$ time
  - If there are short paths, can be quite good

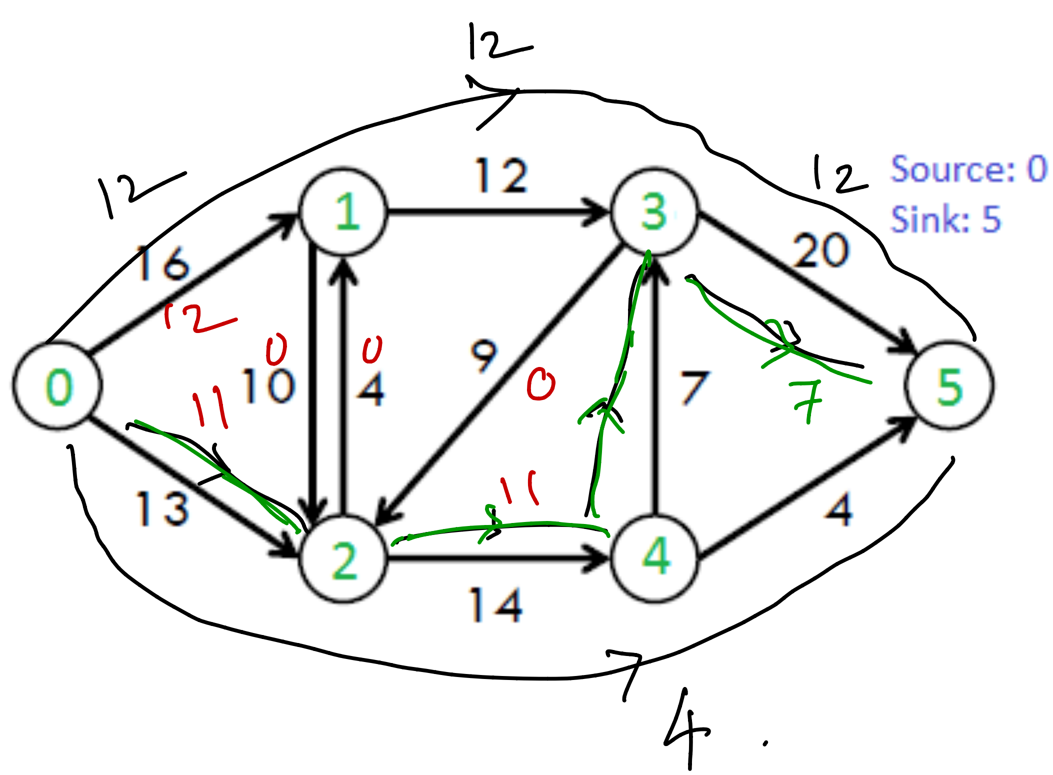
Plan for today

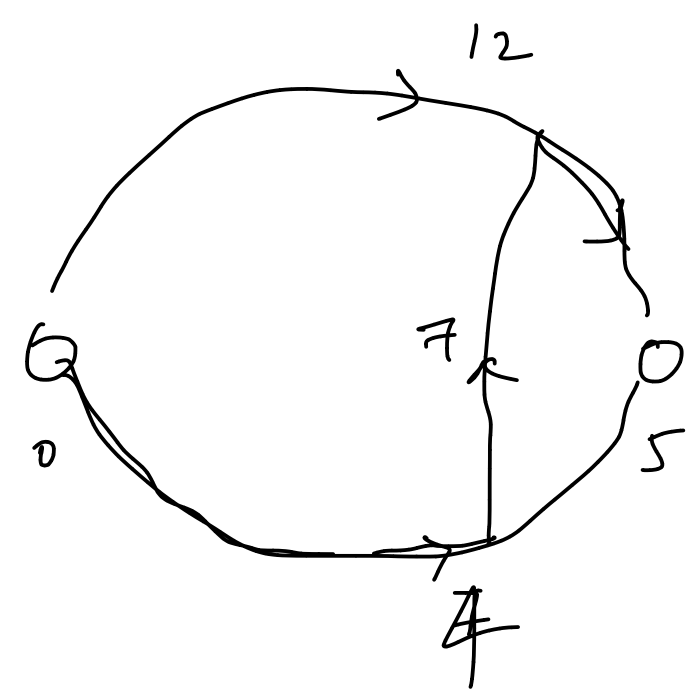
- Two problems: "flows" and cuts
- Connections
- Mid-term review

Maximum flow

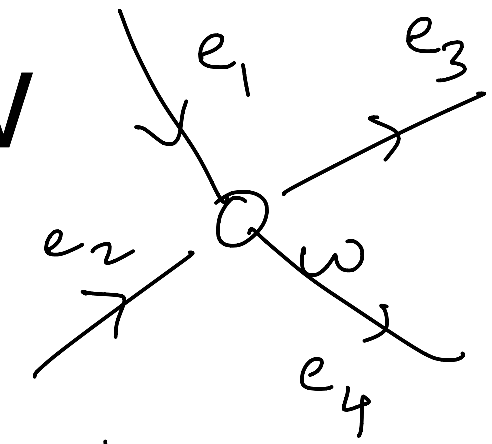
communication networks, shipping goods, …

Problem: given a (directed) graph G = (V, E) with edge **capacities** (> 0), source u, sink v, find the max possible "rate" at which one can send "information" from u to v.





Two definitions of flow



equivalent formulations of flow

- Set of paths, each carrying some flow; constraint: on every edge e, total flow on that edge (summed over paths using that edge) is $\le$ capacity(e).
- Flow values on edges

views a flow as a fn from $E \to \mathbb{R}_{\geq 0}$.

for every $e \in E$, we have a value f_e.

(capacity)

$\star$ $f_e \le$ capacity(e)

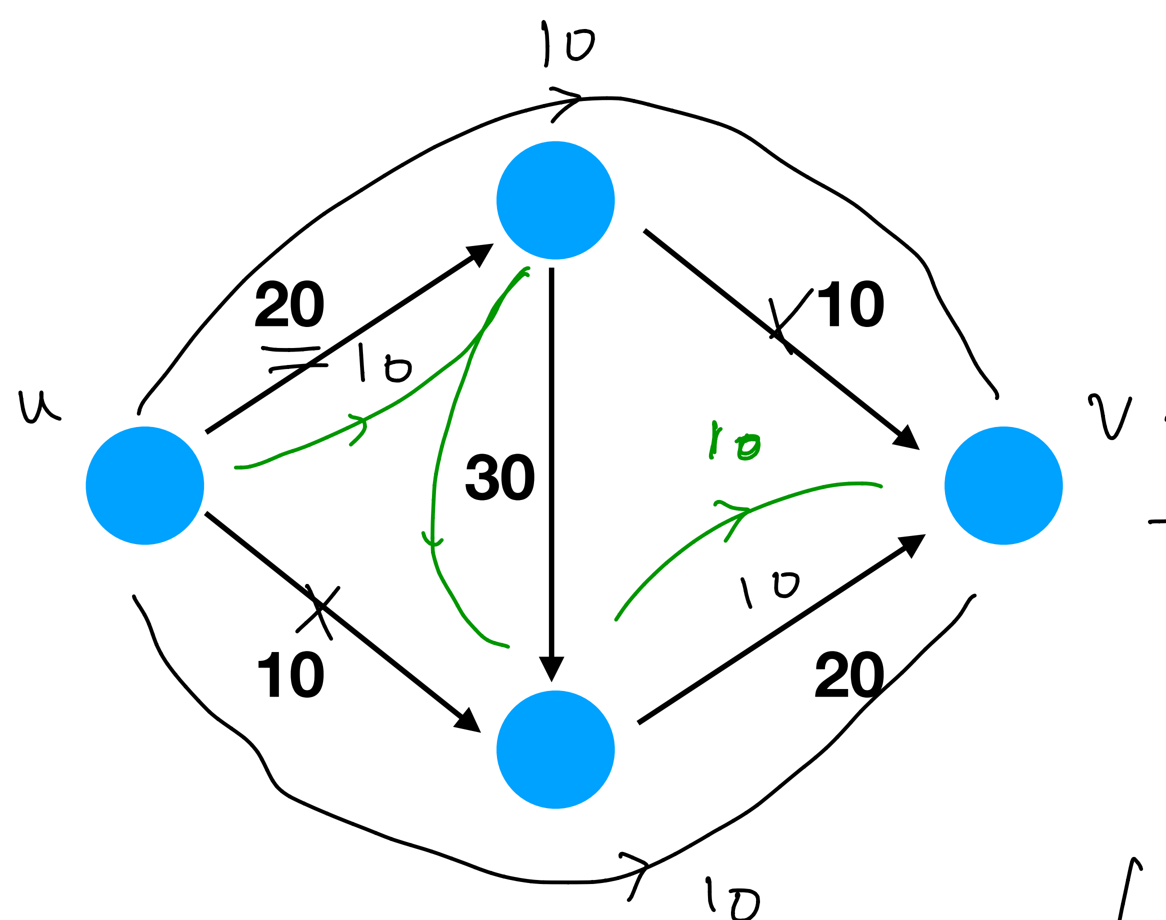
(conservation).

$\star$ for all vertices $w \neq u, v$, total flow into w = total flow out of w.

$$f_{e_1} + f_{e_2} = f_{e_3} + f_{e_4}.$$

Greedy procedure

10 + 10 + 10 = 30 units of flow.



→ Find ~~shortest~~ any path from u → v, send as much flow as possible

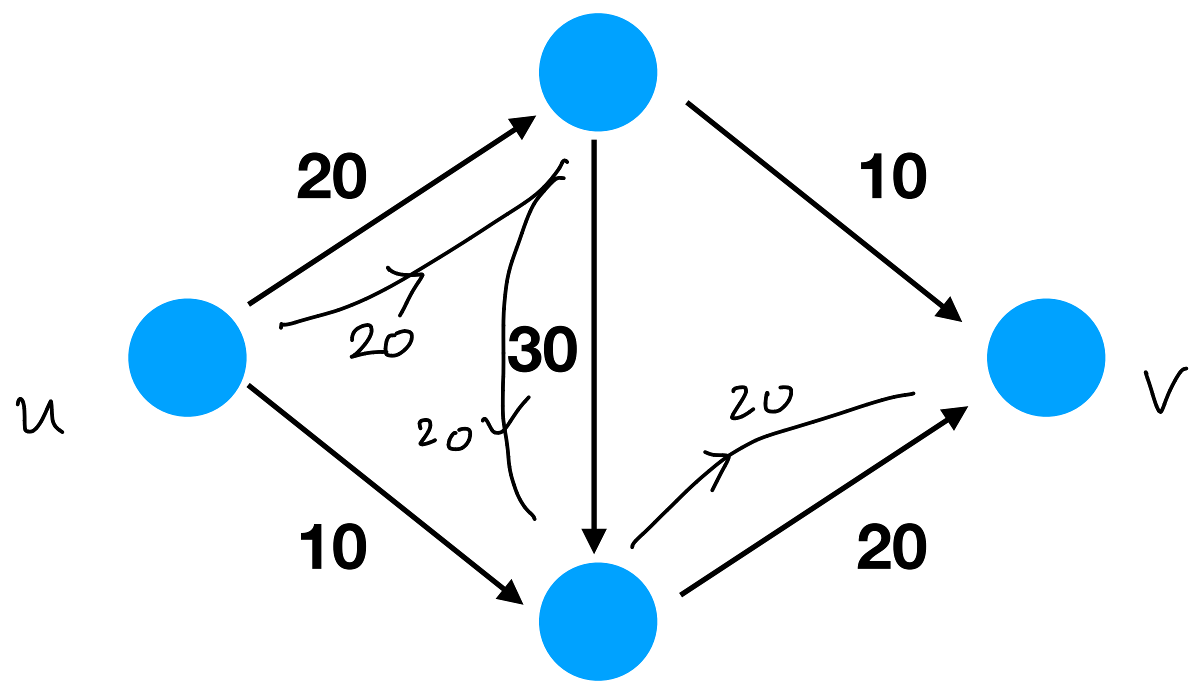
→ Update capacities on edges (i.e., keep only "remaining" capacity)

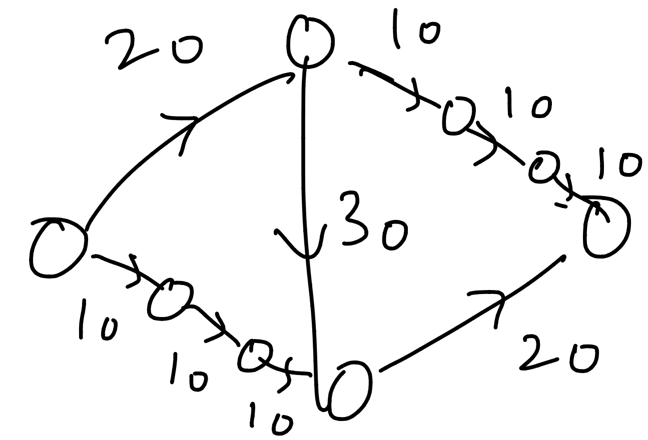
— Repeat.

* Algorithm always finds a feasible flow.

Issues

- Depends on path chosen!





Need to “push back” flow, correct “mistakes” — Ford Fulkerson algorithm…

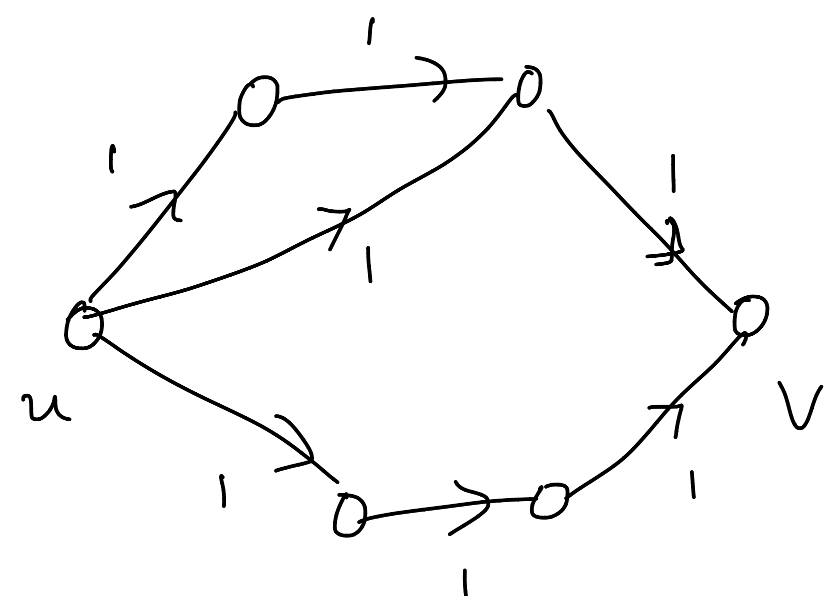
Min cut problem

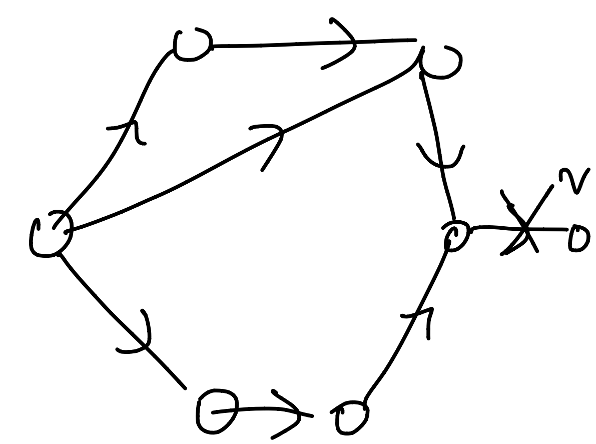
A cut is a set of edges .

Problem: given a (directed) graph *G* = (V, E) with edge **costs** (> 0), source *u*, sink *v*, find the min possible set of edges to "cut" so that there's no path from *u* —> *v*

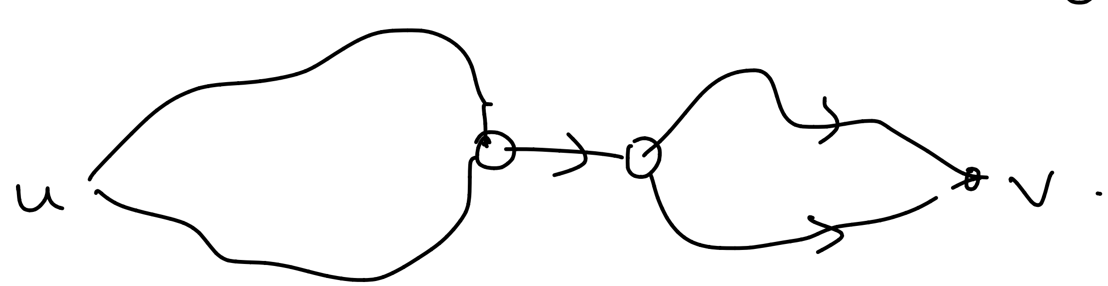
min total cost .

Blowing bridges…

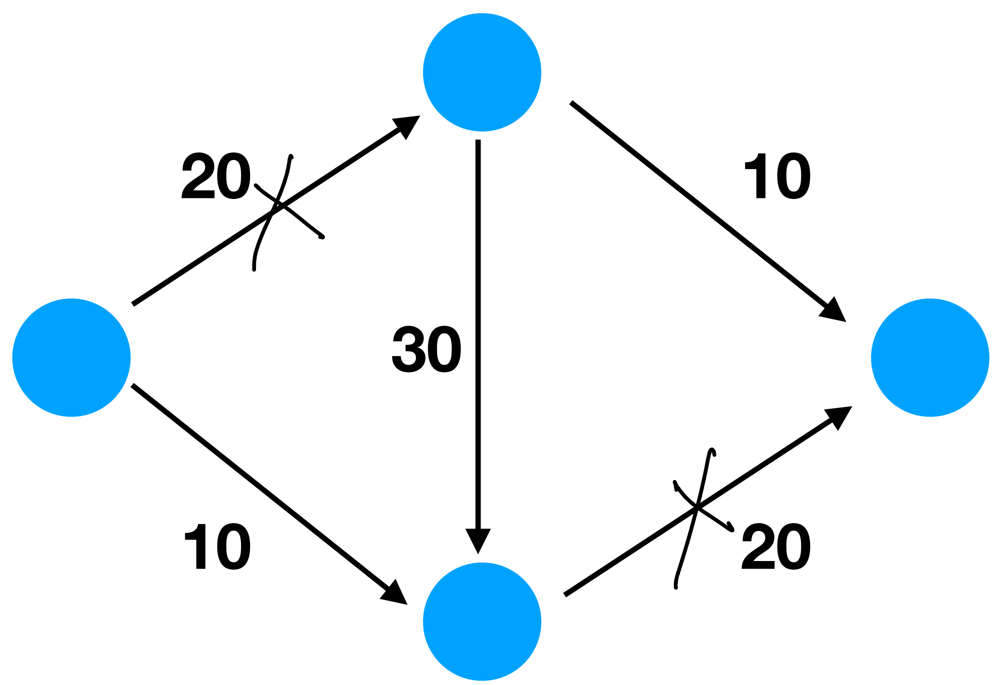




- Undirected graphs
- Image segmentation



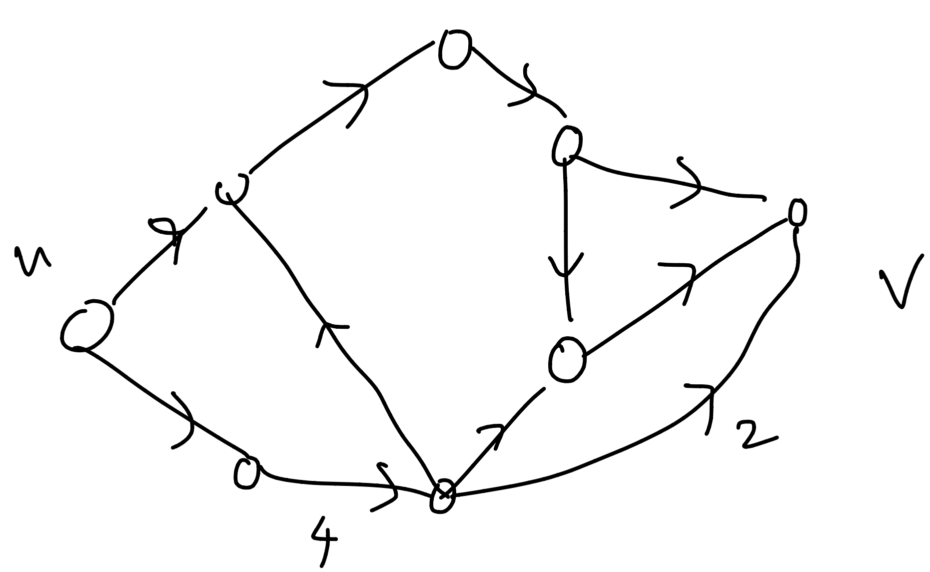
Minimum cut



Min-cut value = 30.

Flows and cuts

Theorem (easy): G = (V, E) be a weighted directed graph, and u, v be vertices. Let “F” be any flow, interpreting wts as capacities. Let “C” be any cut, interpreting wts as costs. Then F <= C.



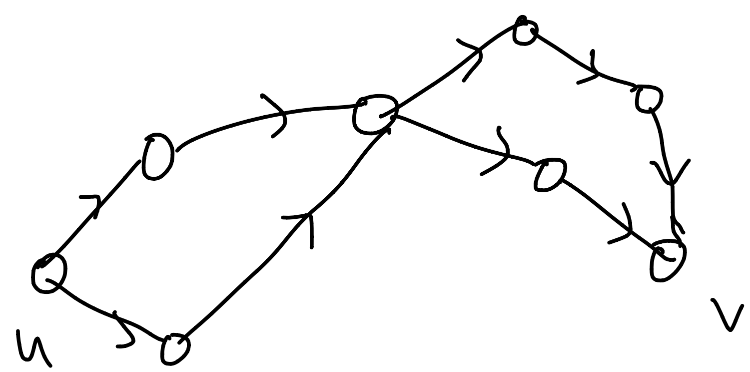
F: value of some flow from $u \to v$.

C: value of some cut (i.e., sum of costs of a set of edges that disconnects v from u.).

Claim: $F \leq C$.

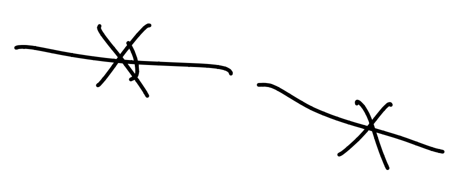
Flow <= cut

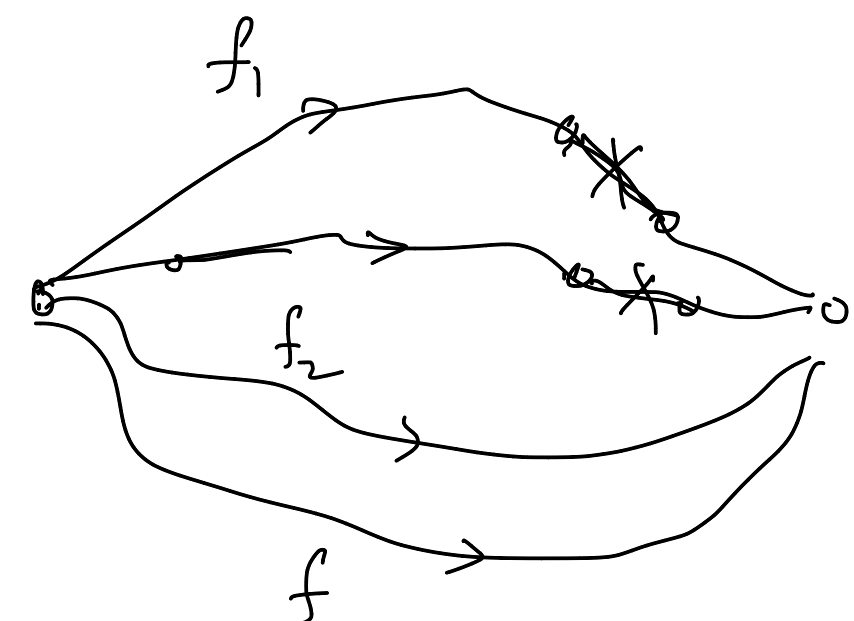
Example: think of all edge wts as 1.



Any flow ≡ collection of edge-disjoint paths.

Cut ≡ some set of edges whose removal disconnects v from u.



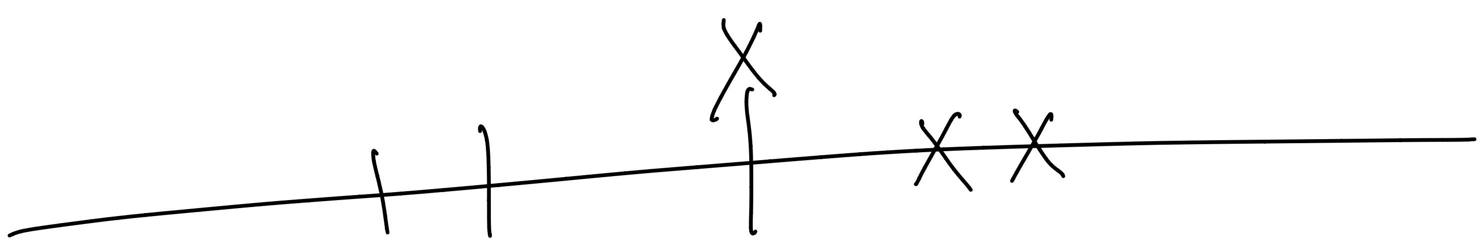


Any path must have at least one of the cut edges. ⟹ # paths ≤ # of cut edges.

Proof of Ford-Fulkerson

→ Value of ANY flow $\leq$ Value of ANY cut.

Value of max flow $\leq$ Value of min cut.



Proof of FF: (Correctness) exhibits a cut whose value is equal to the value of the flow produced.

Comments

- Max-flow min-cut theorem (in any graph, max flow = min cut.) (FF-Theorem).

- Many applications — e.g., no bottleneck => many edge disjoint paths

- Algorithms for cut == algorithms for flow

Review

HW1, 2

- Divide and conquer / recursion — inductive analysis, recurrences
- Dynamic programming (store answers to sub-problems)

HW3

- Greedy algorithms (easy to design + implement, often hard to reason), local search
- Basics of graphs

- Using randomness in algorithm design
- Optimization and linear programming