CS 331, Fall 2025

Lecture 13 (10/13)

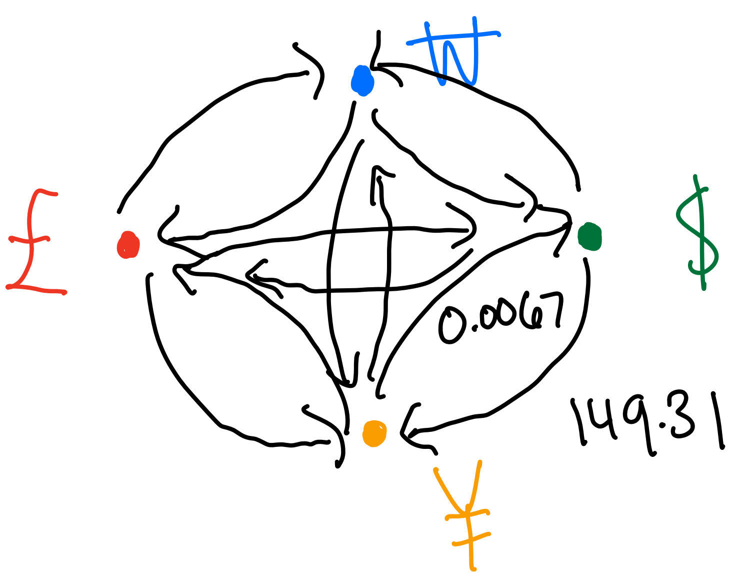
Today:
- Arbitrage
- A^* search
- Maxflow
- Flow reductions

Arbitrage (Part V, Section 5.1)

$G = (V, E, w)$ exchange network

Vertices = currencies

$w_{(u,v)}$ = exchange 1 unit of u for v



Question: Can we make trades and end up w/ more of currency than started?
(arbitrage)

Equivalently, $\exists$ cycle C s.t.

$$\prod_{e \in C} w_e > 1 ?$$

Idea: reduction to negative-weight cycle

$$\prod_{e \in C} w_e > 1 \iff \sum_{e \in C} \underbrace{\left(-\log(w_e)\right)}_{:= w'_e} < 0$$

Make new graph $G' = (V, E, w')$

Detect NWC in $O(mn)$ time $\Rightarrow$ arbitrage in G

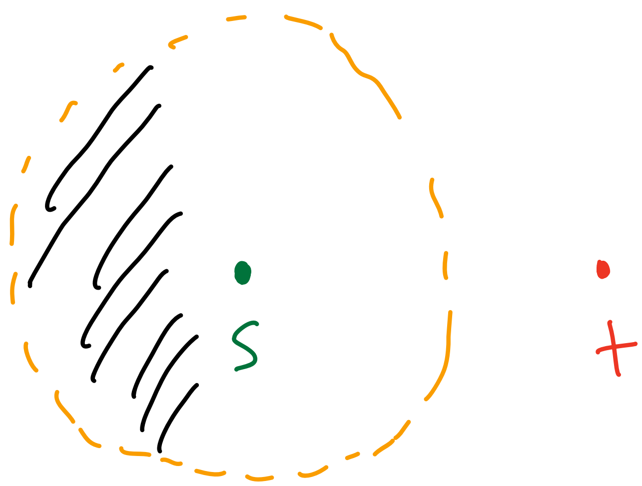
A* Search (Part V, Section 5.2)

Motivation: Dijkstra solves SSSP

What if we only care about s-t shortest path?

Goal: Pull t out of queue as fast as possible

Would like to deemphasize searching using info about t



Idea: Modify edge weights using $h : V \to \mathbb{R}$

(heuristic/price function)

Assign shorter paths to vertices close to t w/o changing the shortest paths!

Define $G^h = (V, E, w^h)$

$$w^h_{(u,v)} = w_{(u,v)} - \underbrace{h(u)}_{\text{savings when leaving } u} + \underbrace{h(v)}_{\text{cost of entering } v}$$

Claim: fix $v \in V$. All s-v paths P have

$$\underbrace{\sum_{e \in P} w^h_e}_{\text{new weight}} = \underbrace{\sum_{e \in P} w_e}_{\text{old weight}} \underbrace{- h(s) + h(v)}_{\text{Change: no dependence on } P}$$

Proof: let P be $(v_1 = s, v_2), (v_2, v_3), \ldots (v_{k-1}, v_k = v)$

Total edge weight:

$$\sum_{e \in P} w^h_e = \sum_{e \in P} w_e - h(s) + \cancel{h(v_2)} - \cancel{h(v_2)} + \cancel{h(v_3)} \cdots - \cancel{h(v_{k-1})} + h(v)$$

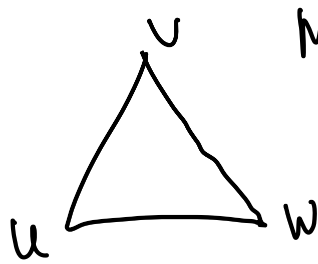
telescoping sum

Takeaways:
- Shortest paths unchanged
- Shortest path <u>distances</u> change by $h(v) - h(s)$

What is a good h?

1) Smaller for vertices near t
2) all edge weights in G^h nonneg: "consistent"

Example: $\underbrace{\text{(quasi-) metric}}_{\text{distance function, triangle ineq.}}$ $h(v) = m(v, t)$



$m(u,v) \le m(u,w) + m(w,v)$

e.g. Euclidean distance

$$h(v) = \underbrace{\|v - t\|}_{\text{"straight-line distance"}}$$

Why are metrics consistent?

$$\underbrace{w_{(u,v)}}_{\substack{\text{assume}\\ = m(u,v)}} - \underbrace{m(u,t)}_{h(u)} + \underbrace{m(v,t)}_{h(v)} \geq 0$$

triangle ineq.

Interestingly, can use <u>shortest path metric</u> δ to make negative edges all nonnegative

Floyd-Warshall: $O(n^3)$ APSP

Johnson: $O(mn\log(n))$ APSP

1) Compute all $\delta(s,v) = h(v)$ $\quad O(mn)$

2) Form G^h $\quad O(m)$

3) Run Dijkstra from all v $\quad O(mn\log(n))$

Flows (Part V, Section 4.1)

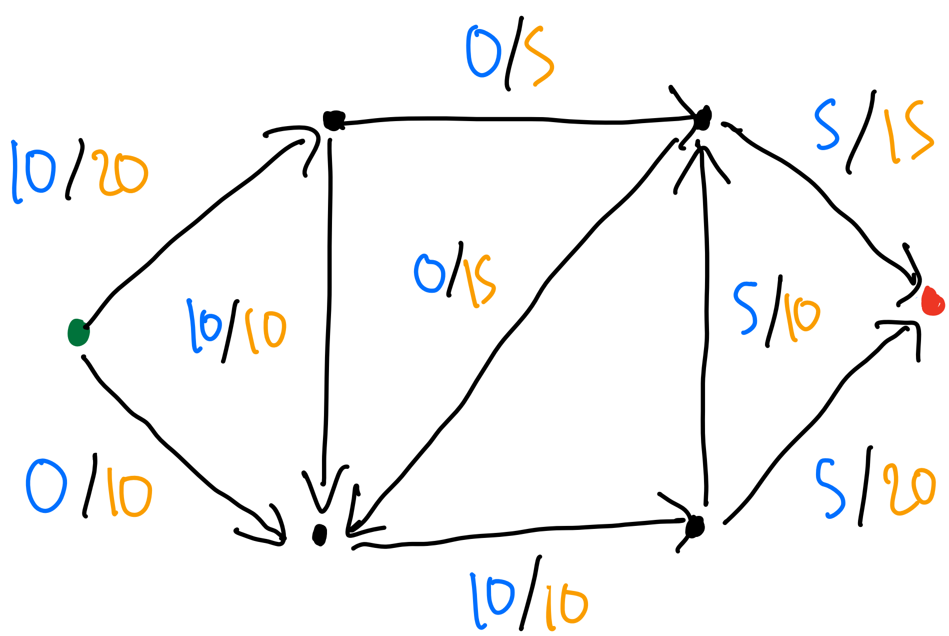
One of most powerful reduction tools

"how much stuff can be sent in G?"

$$G = (V, E, \underbrace{c}_{\text{capacities}})$$

Flow $f \in \mathbb{R}^E$ is <u>feasible</u> if

$$0 \le f_e \le c_e \quad \forall e \in E$$



Net flow @ $v \in V$:

$$\partial f(v) = \underbrace{\sum_{(v,u)\in E} f_{(v,u)}}_{\text{produced by } v} - \underbrace{\sum_{(u,v)\in E} f_{(u,v)}}_{\text{consumed by } v}$$

Simple observation:

$$\sum_{v\in V} \partial f(v) = \sum_{v\in V} \sum_{(v,u)\in E} f_{(v,u)} - \sum_{v\in V} \sum_{(u,v)\in E} f_{(u,v)} = 0$$

We call f an s-t flow if:

- $\partial f(s) = -\partial f(t)$
- $\partial f(v) = 0 \quad \forall v \notin \{s, t\}$

s-t maxflow problem:

$$\max \underbrace{\partial f(s)}_{\text{total "stuff" prod. by source } s} \text{ s.t. } f \text{ is feasible } s\text{-}t \text{ flow}$$

Next time: how to solve maxflow

Today: applications!

Flow reductions (Part V, Section 5.3)

Idea: you want to solve graph problem on G

Instead, show you can recover solution from maxflow on some other G'

Proof of correctness: Convert any...

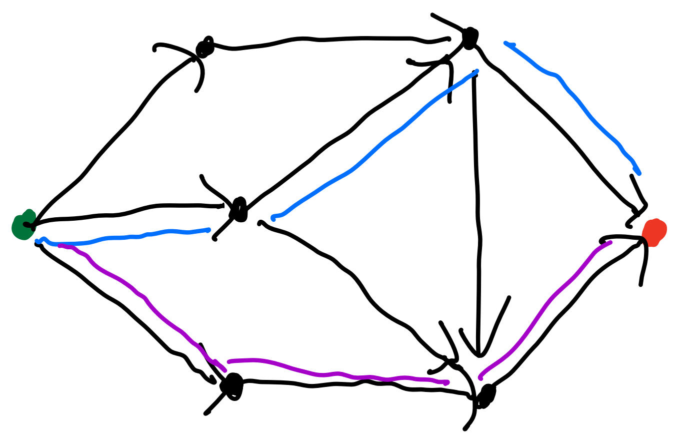
($\Rightarrow$) feasible sol in G to feasible flow in G'

($\Leftarrow$) feasible flow in G' to feasible sol in G

Disjoint paths

Input: $G = (V, E)$, $s, t \in V$

Output: Max # $\underbrace{\text{disjoint}}_{\text{share no edge}}$ s-t paths



(☆) We'll show next class if all capacities integer, so is maxflow

Claim: it's just the s-t maxflow value (☆) if every edge has capacity $c_e = 1$

($\Rightarrow$) Given k disjoint paths, can create feasible flow f with $\partial f(s) = k$

($\Leftarrow$) Given flow of value k, repeatedly "peel off" s-t paths until no more $\partial f(s)$

Bipartite matching

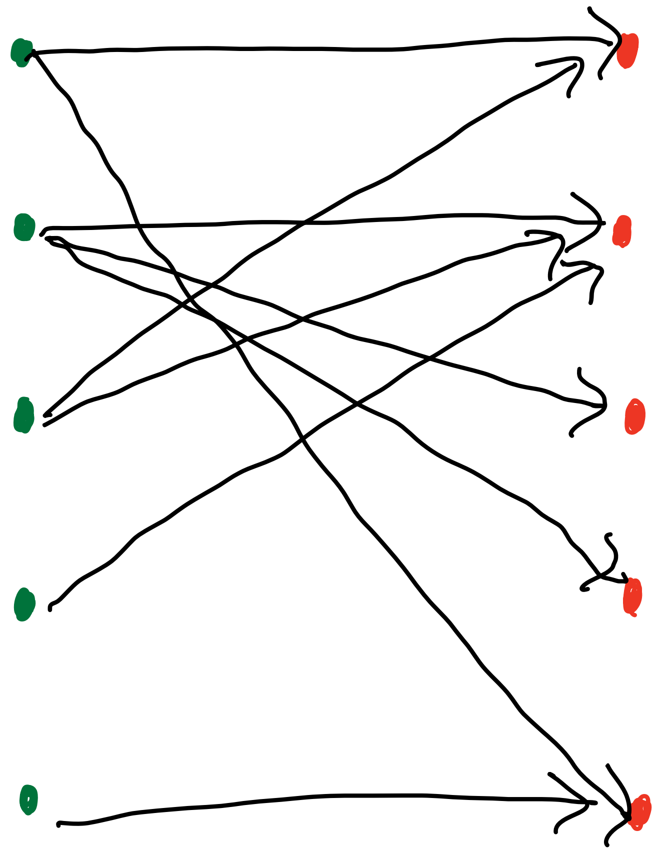
Input: bipartite graph $(L \cup R, E)$

(all edges $(u,v) \in E$ have $u \in L$, $v \in R$)

Output: max $|M|$ over matchings M

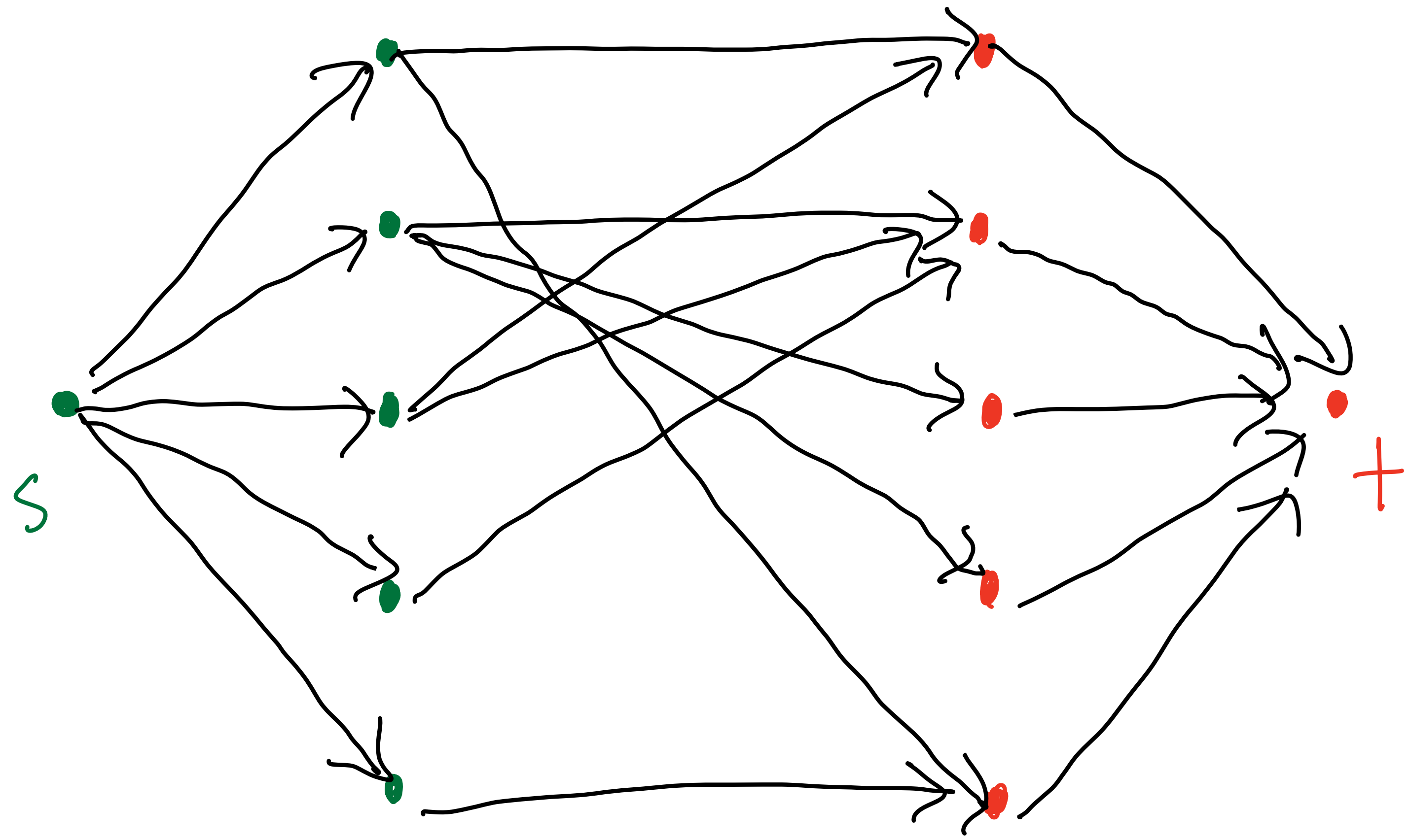
(every vertex belongs to ≤ 1 edge)

Example



(in stable matching, Part IV, Section 5, all pairs $L \times R$ are available $\Rightarrow$ max matching $= \frac{n}{2}$)

Reduce to flow!



All edges capacity = 1

Claim: s-t maxflow value = max matching size

($\Rightarrow$) Given matching, can extend to flow of same value

($\Leftarrow$) Given flow, where is each (s,u) going?

Extend path to t, creates one matched pair

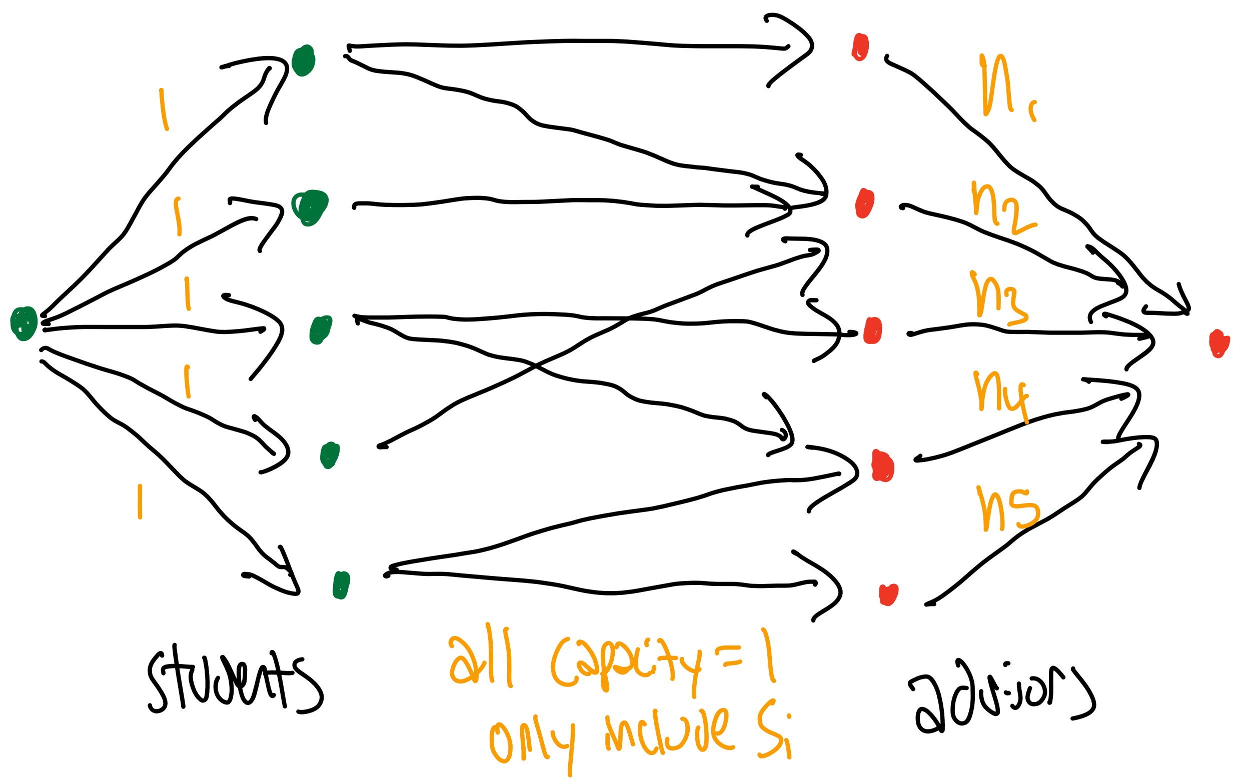
Generalized matching

Above example has many extensions.

Say we have... a advisors
b students

Each advisor $i \in [a]$ can...

- only advise students $S_i \subseteq [b]$ (specific expertise)
- only has n_i slots available



Maxflow value = max # advisable students