

CS 331, Fall 2026
lecture 13 (10/12)

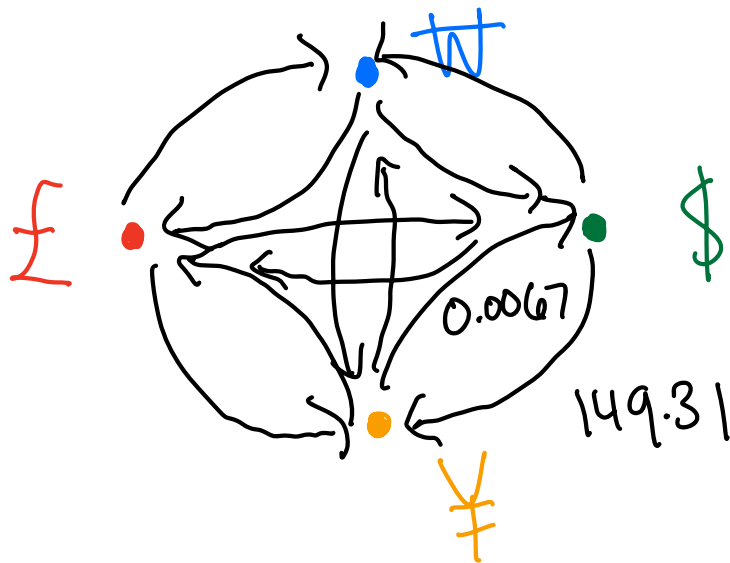
Today: - Arbitrage
- A^* search
- Maxflow
- Flow reductions

Arbitrage (Part V, Section 5.1)

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

Vertices = currencies

$w(u, v)$ = exchange | 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

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

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

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

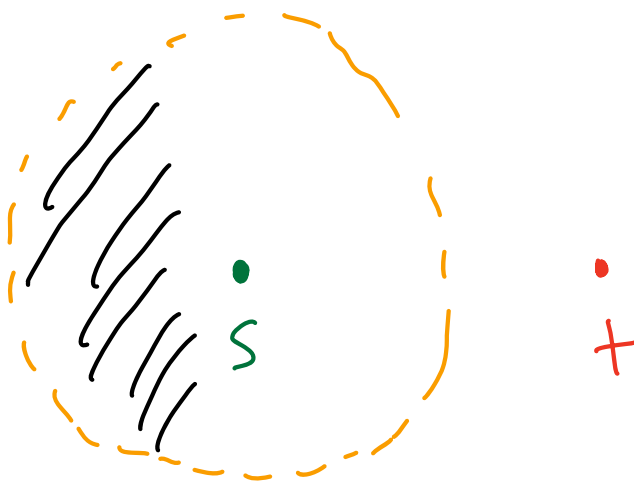
A* Search (Part V, Section 5.2)

Motivation: Dijkstra solves SSSP

What if we only care about $s \rightarrow t$ 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 $h: V \rightarrow \mathbb{R}$
(heuristic/
price function)

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

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

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

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

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

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

Total edge weight:

$$\begin{aligned} \sum_{e \in P} w_e^h &= \sum_{e \in P} w_e - \cancel{h(s)} + \cancel{h(v_2)} \\ &\quad - \cancel{h(v_2)} + \cancel{h(v_3)} \\ &\quad \dots \\ &\quad - \cancel{h(v_{k-1})} + h(v) \end{aligned} \quad \begin{array}{l} \text{telescoping} \\ \text{sum} \end{array}$$

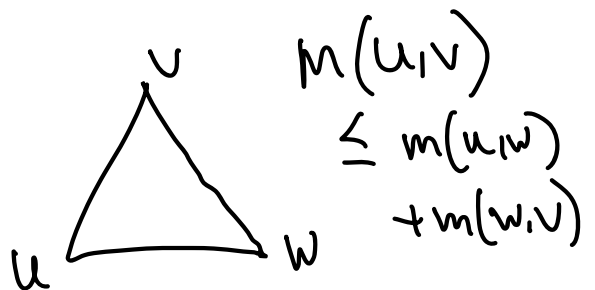
- Takeaways:
- Shortest paths unchanged
 - Shortest path distances change by $h(v) - h(s)$

What is a good h ?

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

Example: (quasi-) metric $h(v) = m(v, +)$

distance function,
triangle ineq.



e.g. Euclidean distance

$$h(v) = \underbrace{\|v - +\|}$$

"straight-line distance"

Why are metrics consistent?

$$\underbrace{W(u,v)}_{\text{assume}} - m(u, \textcolor{red}{+}) + m(v, \textcolor{red}{+}) \geq 0$$

$h(u) \qquad h(v)$

assume
= $m(u,v)$

triangle ineq.

Interestingly, can use shortest path metric δ

to make negative edges all nonnegative

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

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

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

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

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

Flows (Part V, Section 4.1)

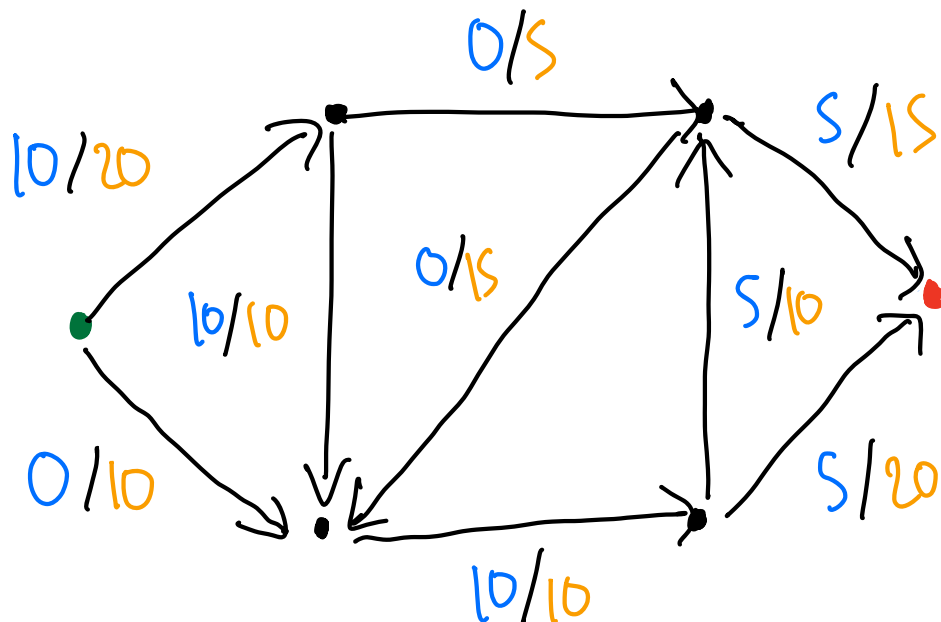
One of most powerful reduction tools

"how much stuff can be sent in G ?"

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

Flow $f \in \mathbb{R}^E$ is feasible if

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



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:

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

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...

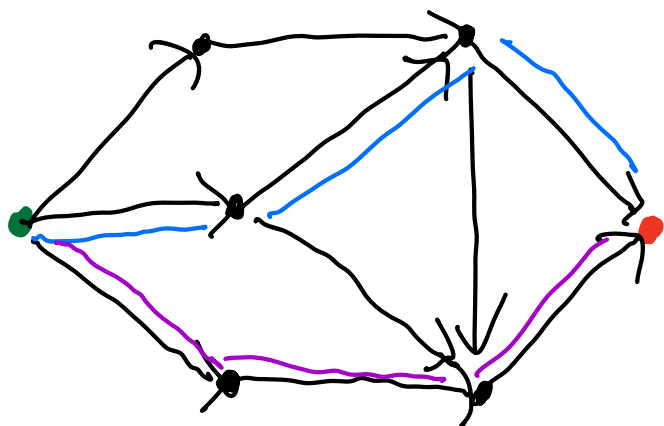
$(\Rightarrow)$ feasible sol'n G to feasible flow in G'

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

Disjoint paths

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

Output: Max # disjoint $s-t$ paths
Share no edge



⊛ 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 $\mathcal{F}(s) = k$

$(\Leftarrow)$ Given flow of value k , repeatedly "peel off" $s-t$ paths until no more $\mathcal{F}(s)$

Bipartite matching

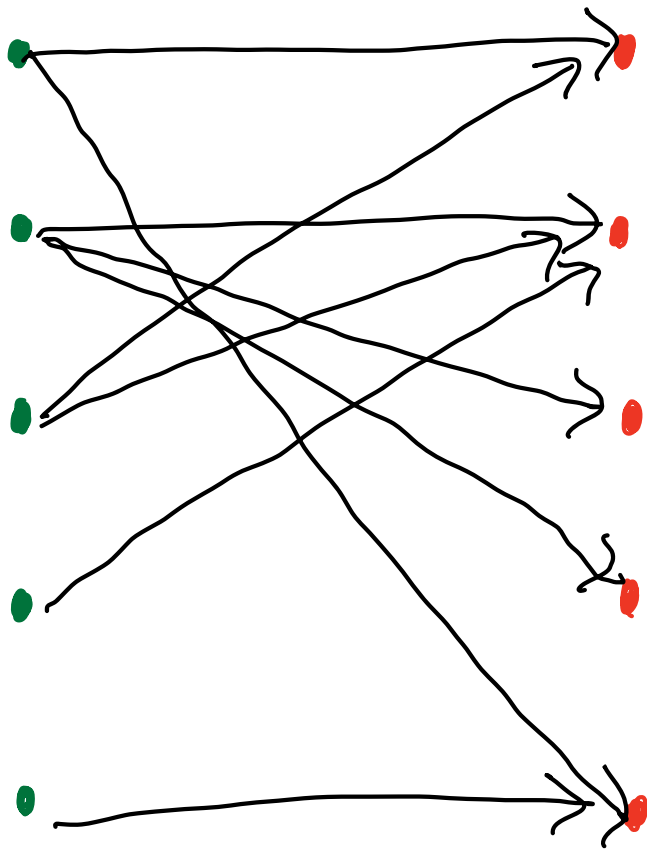
Input: bipartite graph $(L \cup R, E)$

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

Output: $\max |M|$ over matchings M

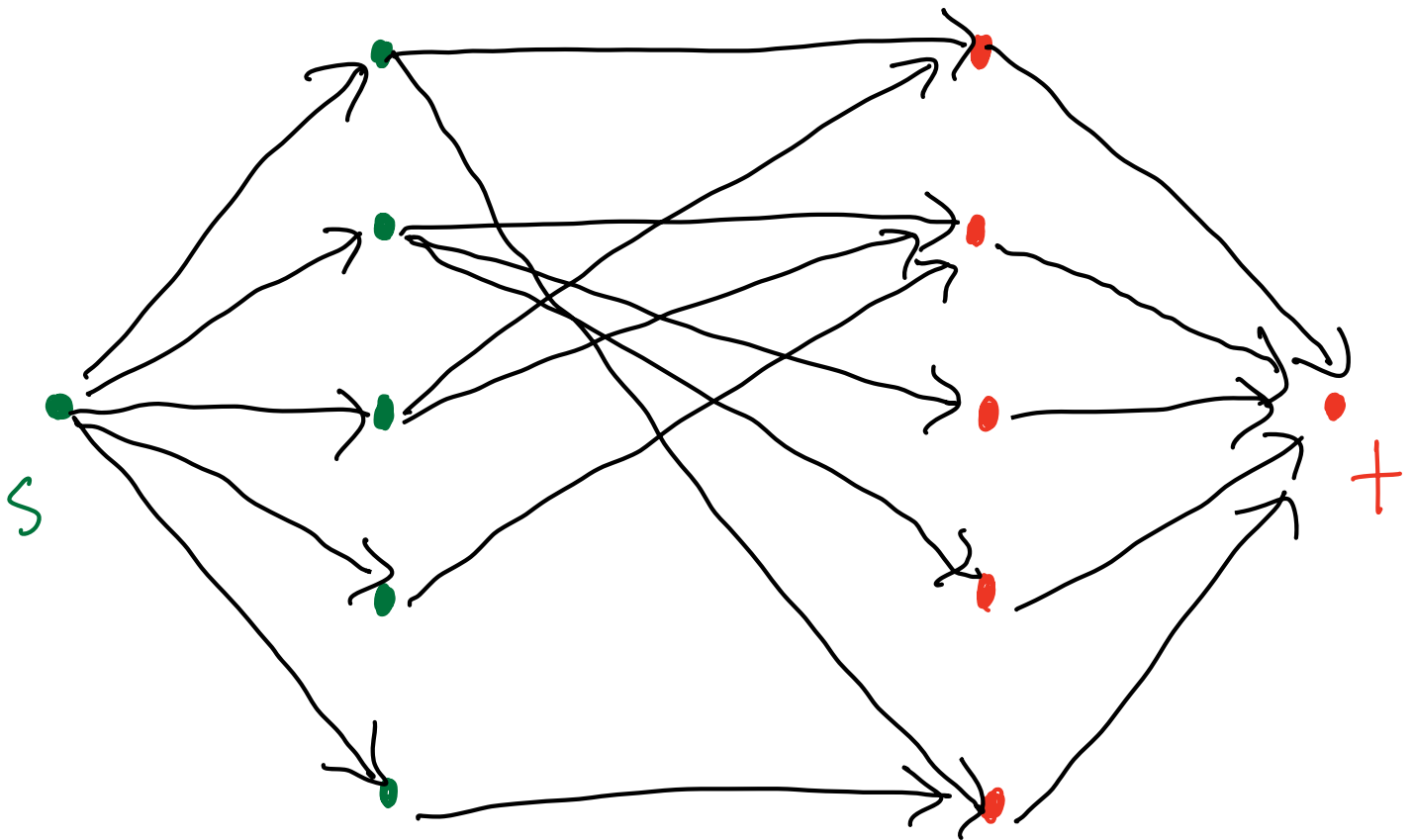
(every vertex belongs to ≤ 1 edge)

Example



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

Reduce to flow!



All edges capacity = 1

Claim: $s \rightarrow +$ 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 $+$, creates one matched pair

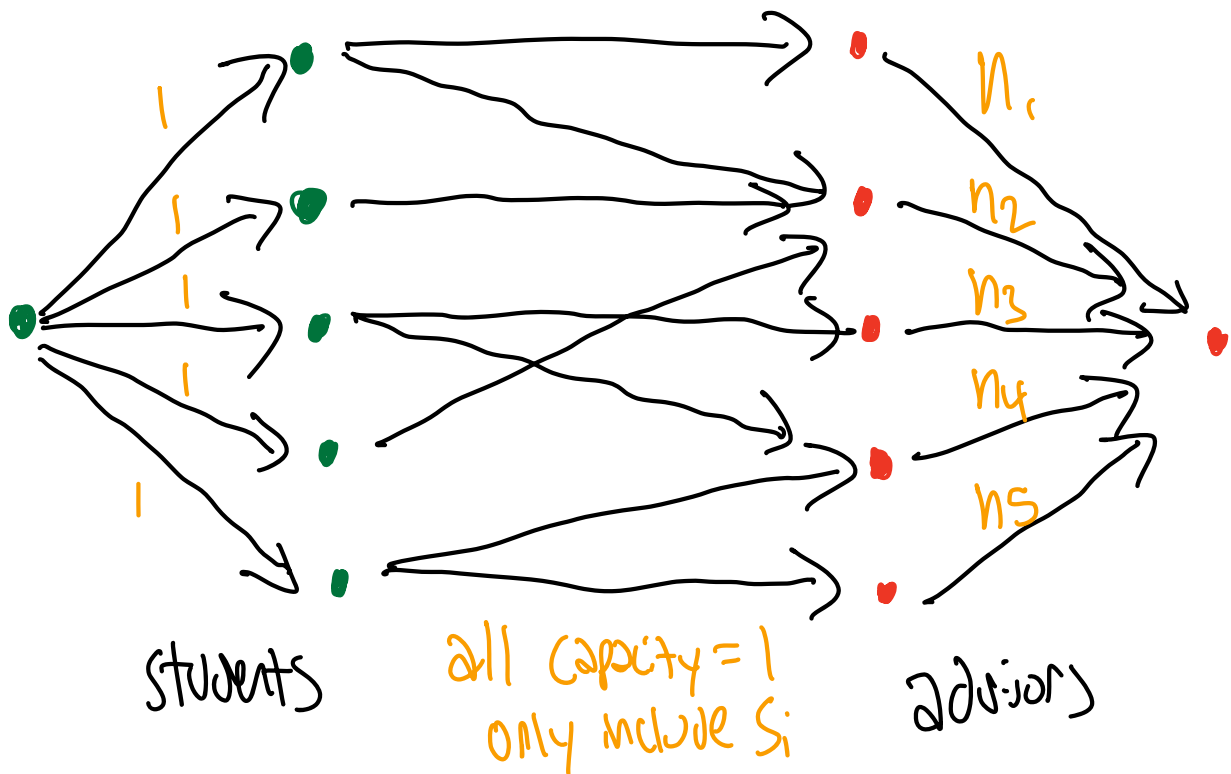
Generalized matching

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