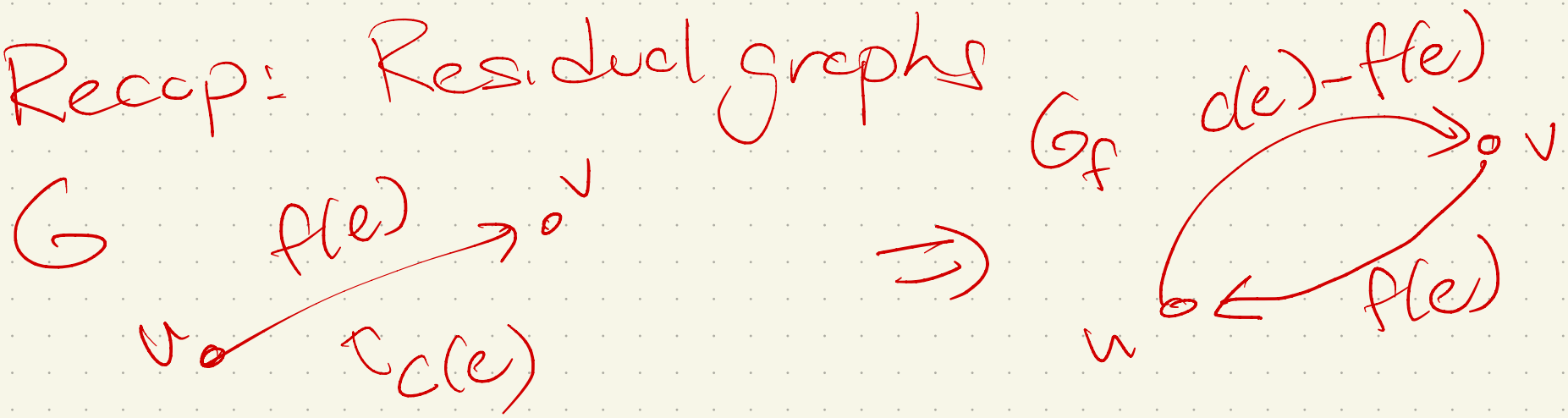
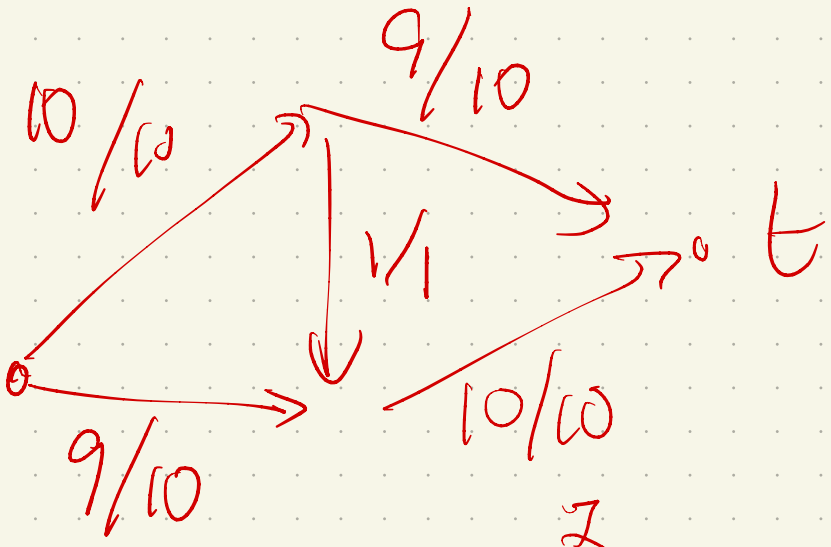
Complexity & Algorithms, Spring 2026

Flows:
algorithms &
Applications

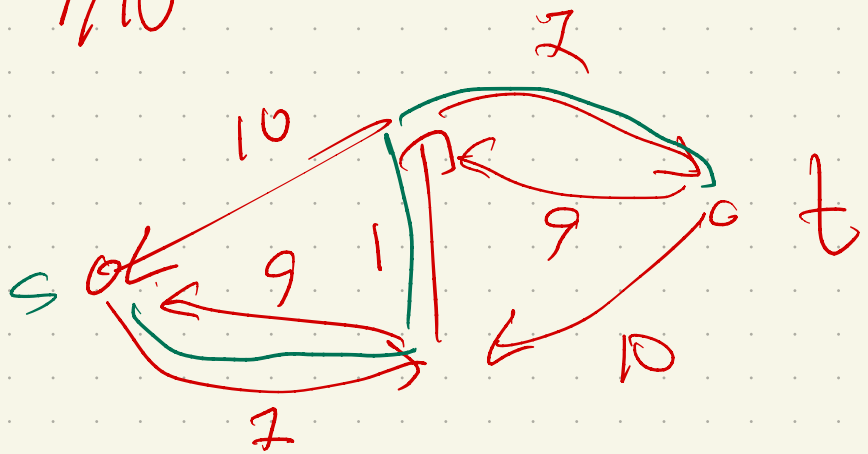
Recap: Residual graphs



"no valid flow path left"



$\hookrightarrow G_f$



Computing Flows

Ford-Fulkerson:

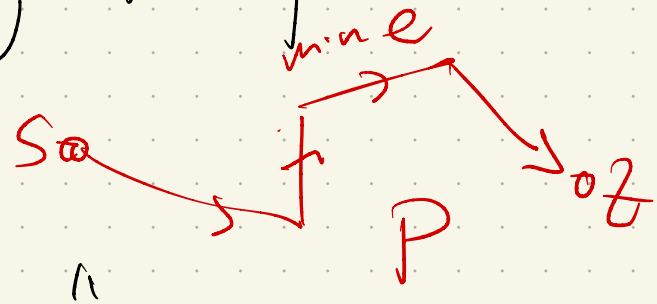
$O((V+E) \cdot f)$ — f: max flow value

Compute any augmenting path in G_f
+ push flow
When no paths in G_f from s to t,
↳ must have a cut!
(use G_f to find it)

[Since any flow ≤ any cut, we've hit the max]

Edmunds-Karp: (take 1)

In G_f, find the largest possible augmenting path



↪ largest "bottleneck":

$$\max_{\text{paths } P: s \rightsquigarrow t} \left\{ \min_{e \in P} \{ w(e) \} \right\}$$

How? Variant of MST → $E \log V$

And each round, flow increases by $\frac{f}{E}$

time to find P ⇒ $O((E \log V) \cdot (E \ln f))$

↑ # of repetitions

EK algorithm #2

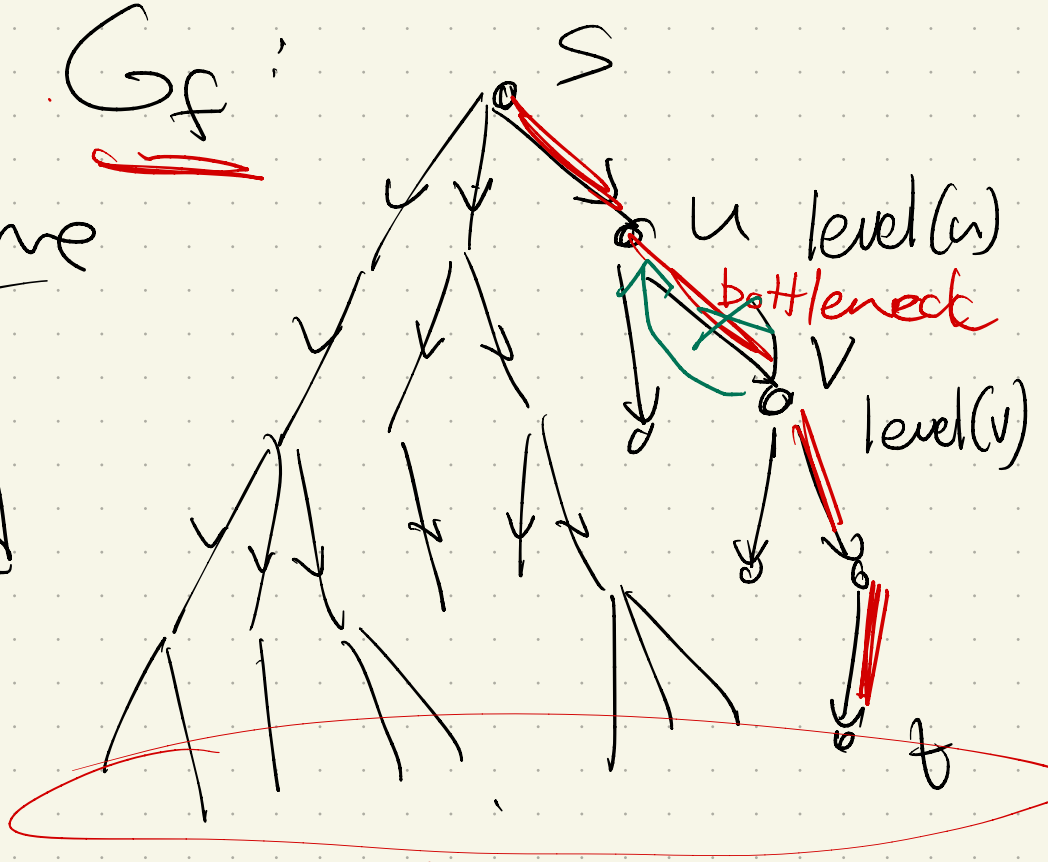
(avoid any f in runtime)

- Choose path with fewest edges
  - ↳ BFS! So: $O(V+E)$

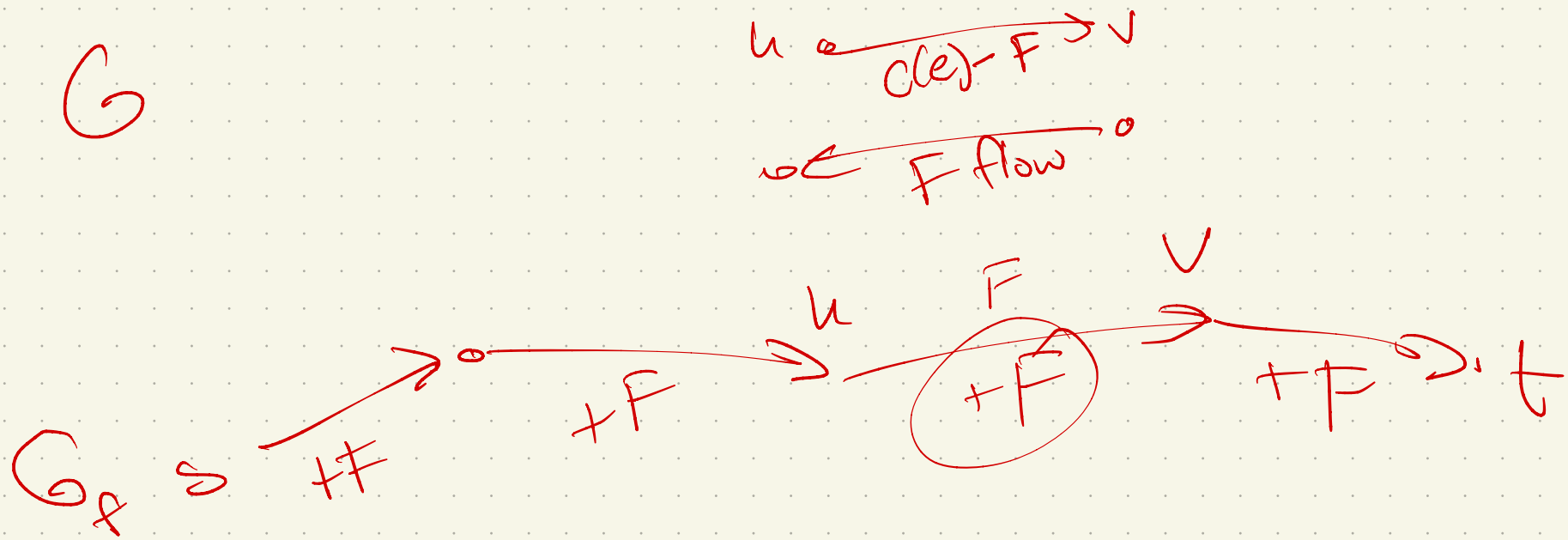
Each time: G_f:

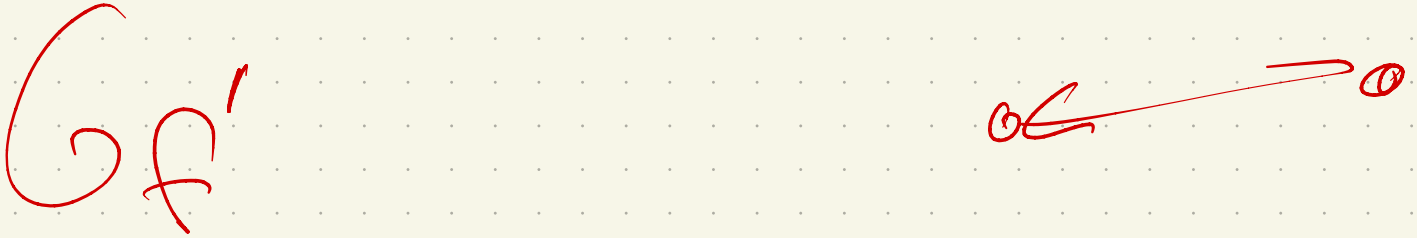
flow saturates some edge on path

↳ edge is reversed in next residual graph.



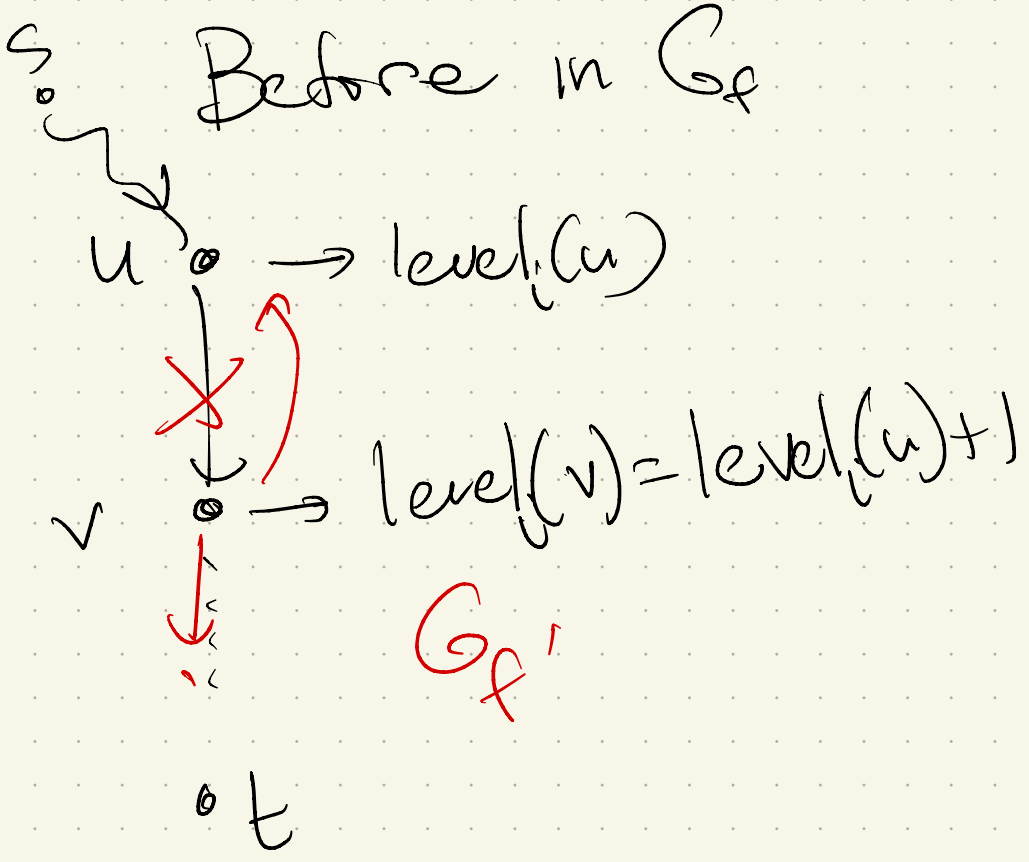
↳ f' $G_{f'}$ ↳ BFS tree


G
u
c(e) - F
v
F flow
u
F
v
G_f s
+F
+F
+F
t


G_f'

So, round i : $u \to v$

round $i+1$: $v \to u$ (4 levels go up)



if it ever reappears:

we must have re-flipped $v \to u$,

so

$$\text{level}_j(u) = \text{level}_j(v)+1 \quad (v \to u)$$

$$\geq \text{level}_i(v)+1$$

$$\geq \text{level}_i(u)+2$$

In other words:

Every time an edge is resaturated, level goes up +2.

Only V levels $\Rightarrow$

total # of augmenting paths in all iterations is $\leq E \cdot \frac{V}{2}$

Runtime: #rep: $O(E \cdot V)$

time per loop iteration: $O(V+E)$

$\Rightarrow O(E^2 V)$

And... not done!

Technique	Direct	With dynamic trees	Source(s)
Blocking flow	$O(V^2E)$	$O(VE \log V)$	[Dinitz; Karzanov; Even and Itai; Sleator and Tarjan]
Network simplex ← LPs	$O(V^2E)$	$O(VE \log V)$	[Dantzig; Goldfarb and Hao; Goldberg, Grigoriadis, and Tarjan]
Push-relabel (generic)	$O(V^2E)$	–	[Goldberg and Tarjan]
Push-relabel (FIFO)	$O(V^3)$	$O(VE \log(V^2/E))$	[Goldberg and Tarjan]
Push-relabel (highest label)	$O(V^2\sqrt{E})$	–	[Cheriyan and Maheshwari; Tunçel]
Push-relabel-add games	–	$O(VE \log_{E/(V \log V)} V)$	[Cheriyan and Hagerup; King, Rao, and Tarjan]
Pseudoflow	$O(V^2E)$	$O(VE \log V)$	[Hochbaum]
Pseudoflow (highest label)	$O(V^3)$	$O(VE \log(V^2/E))$	[Hochbaum and Orlin]
Incremental BFS	$O(V^2E)$	$O(VE \log(V^2/E))$	[Goldberg, Held, Kaplan, Tarjan, and Werneck]
Compact networks	–	$O(VE)$	[Orlin]

Figure 10.10. Several purely combinatorial maximum-flow algorithms and their running times.

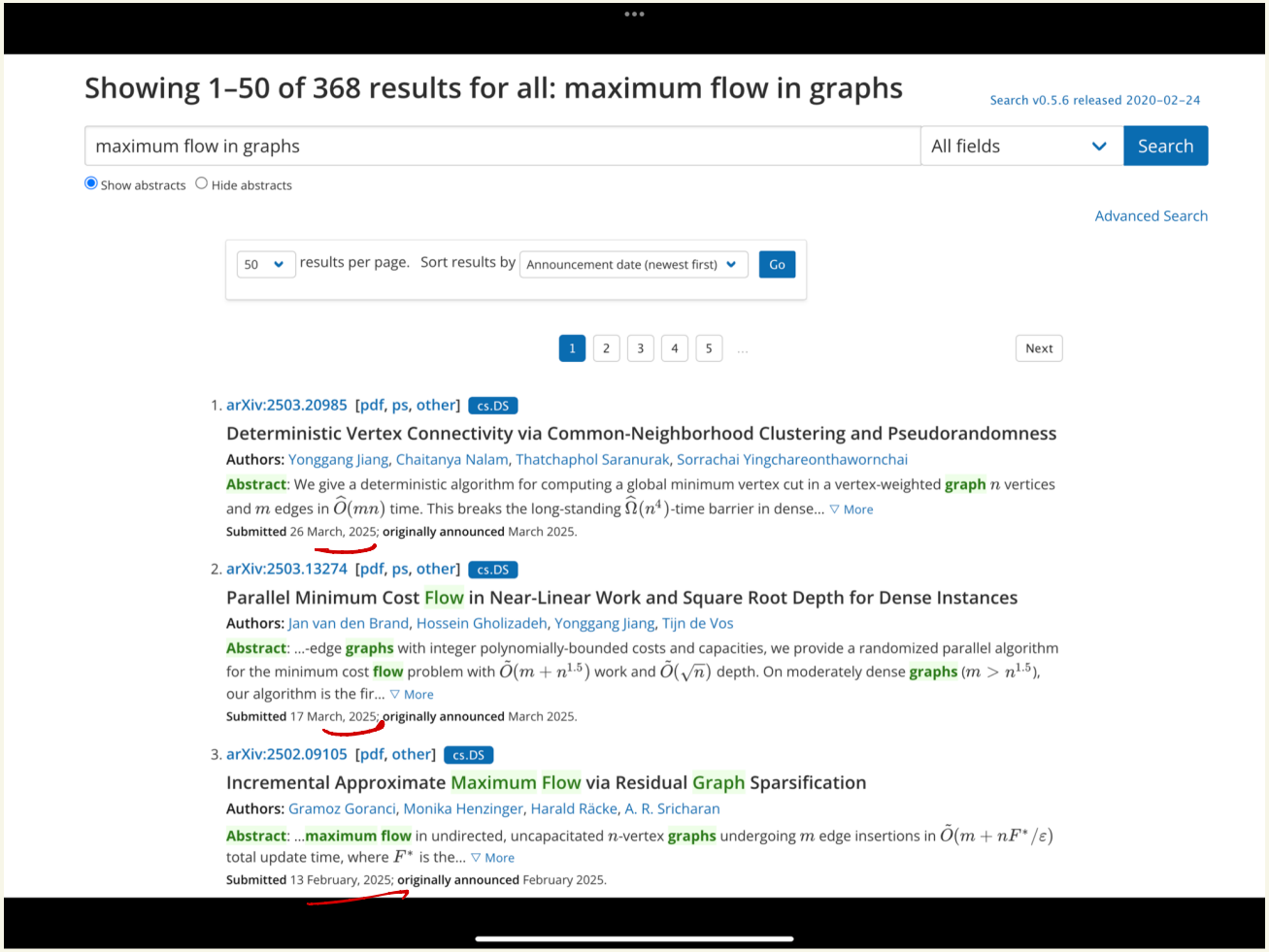
Many use very different techniques

→ Best known!

FF: $O((V+E)F)$

- linear programming
- complex data structures
- not residual graphs

Still active:



Showing 1–50 of 368 results for all: maximum flow in graphs

Search v0.5.6 released 2020-02-24

maximum flow in graphs | All fields | Search

Show abstracts | Hide abstracts

Advanced Search

50 results per page. Sort results by Announcement date (newest first) Go

1 2 3 4 5 … Next

1. arXiv:2503.20985 [pdf, ps, other] cs.DS

Deterministic Vertex Connectivity via Common-Neighborhood Clustering and Pseudorandomness

Authors: Yonggang Jiang, Chaitanya Nalam, Thatchaphol Saranurak, Sorrachai Yingchareonthawornchai

Abstract: We give a deterministic algorithm for computing a global minimum vertex cut in a vertex-weighted graph n vertices and m edges in $\widehat{O}(mn)$ time. This breaks the long-standing $\widehat{\Omega}(n^4)$-time barrier in dense... ▽ More

Submitted 26 March, 2025; originally announced March 2025.

2. arXiv:2503.13274 [pdf, ps, other] cs.DS

Parallel Minimum Cost Flow in Near-Linear Work and Square Root Depth for Dense Instances

Authors: Jan van den Brand, Hossein Gholizadeh, Yonggang Jiang, Tijn de Vos

Abstract: ...-edge graphs with integer polynomially-bounded costs and capacities, we provide a randomized parallel algorithm for the minimum cost flow problem with $\tilde{O}(m + n^{1.5})$ work and $\tilde{O}(\sqrt{n})$ depth. On moderately dense graphs ($m > n^{1.5}$), our algorithm is the fir... ▽ More

Submitted 17 March, 2025; originally announced March 2025.

3. arXiv:2502.09105 [pdf, other] cs.DS

Incremental Approximate Maximum Flow via Residual Graph Sparsification

Authors: Gramoz Goranci, Monika Henzinger, Harald Räcke, A. R. Sricharan

Abstract: ...maximum flow in undirected, uncapacitated n-vertex graphs undergoing m edge insertions in $\tilde{O}(m + nF^*/\varepsilon)$ total update time, where F^* is the... ▽ More

Submitted 13 February, 2025; originally announced February 2025.

Plus work for special classes of graphs: planar, sparse, etc.

Topics in Ch. 11

A mess of different ideas! Reductions

① Matchings: identify a way to pair up items

Build G': More pairs $\Leftrightarrow$ larger flow

② Disjoint paths:

Modify G:

Find paths that avoid each other via a flow in modified G

③ "Tuple" Selection

Build a graph: flow paths give selection

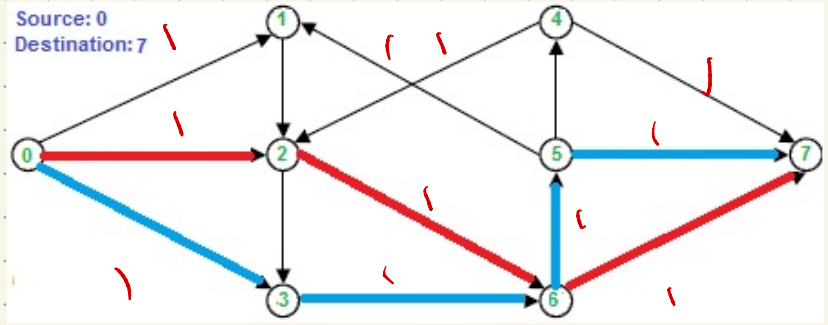
Magic: Max flows always give solution

First problem

What if we want non-intersecting paths from s to t?

Two variants:

- Edge disjoint: No 2 paths visit the same edge



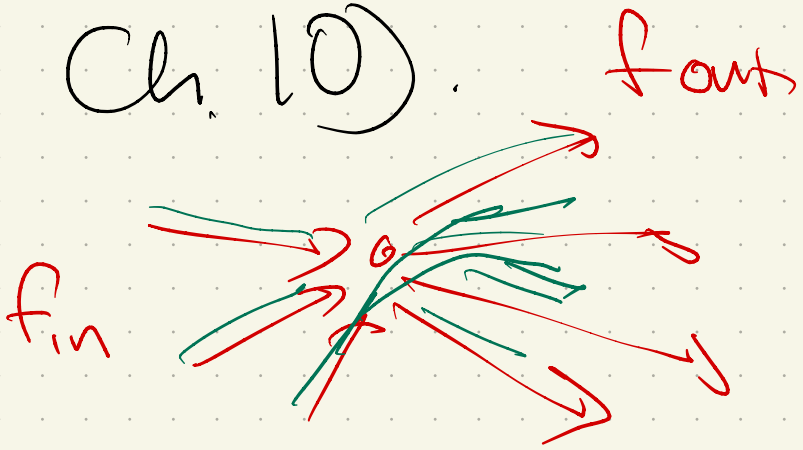
- Vertex disjoint: no 2 paths visit the same vertex

Note: different! And both model useful cases

Key idea for both:

Transform into flow graphs.

Then, decompose flow into paths (see Ch. 10).



Note: Reductions!

Input graph G ⟶ Flow network G'

Edge disjoint:

Input: unweighted directed graph $G = (V, E)$
plus $s, t \in V$.

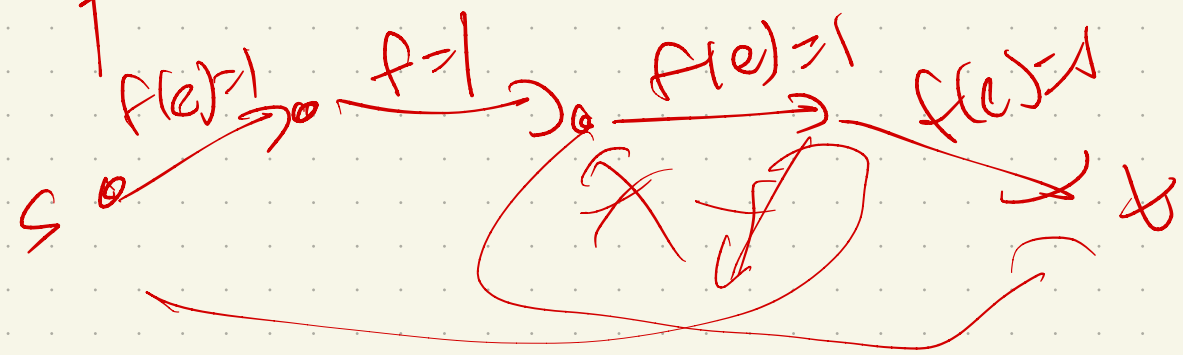
For each edge $e = \overrightarrow{uv}$:

- give it $c(e) = 1$ ← $O(E)$

↳ flow paths "use up" edge e

run Max flow on G'

paths = value(f) $O(VE)$



Correctness: Since all flows are integral, & capacity of every edge is = 1

$\Rightarrow$ no edge will be in 2 paths.

Therefore, we have a set of k disjoint paths

$\Longleftrightarrow$ there is a flow of value k.

Since we find largest flow

$\Rightarrow$ get max # of paths

Runtime:

Build G': $O(E+V)$

flow in G': Orlin $O(VE)$

Convert to soln in G:

↳ $O(E)$

Vertex disjoint:

Build a new graph $\tilde{G}$:

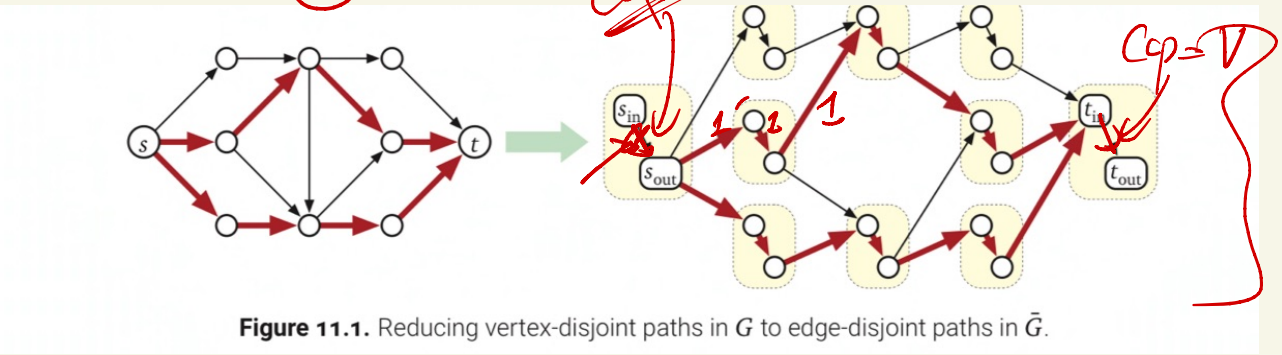


Figure 11.1. Reducing vertex-disjoint paths in G to edge-disjoint paths in $\bar{G}$.

$V(\tilde{G}) =$ 2 copies of each $v \in V(G)$: v_{in}, v_{out}

$E(\tilde{G}) =$ $\forall e = \overrightarrow{u,v} \in E(G)$: build $u_{out} \to v_{in}$

$\forall v \in V$, build edge $v_{in} \to v_{out}$

Add capacity = 1 to each edge

$f \leftarrow$ Maxflow$(\tilde{G}, s_{in}, t_{out})$

s_{out} t_{in}

For each edge $s \to v$ with flow = 1, start tracing path.

$\tilde{V} = 2V$ $\tilde{E} = V + E$

Runtime: $O(E+V)$ to build $\tilde{G}$ Orlin: $O(\tilde{V}\tilde{E}) = O(VE)$

Correctness:

Consider max flow f:

Any flow path that enters v_{in} will exit v_{out},
so $f(v_{in} \to v_{out}) = 1 = c(v_{in} \to v_{out})$

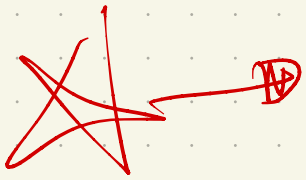
↳ Flow decomposes into paths which visit each vertex ≤ 1 time.

So: Flow of value k in $\tilde{G}$

$\iff$ k vertex disjoint paths in G.

Step back: reductions again!

In these examples algorithm is usually:



Build $\tilde{G}$ from input

Run max flow

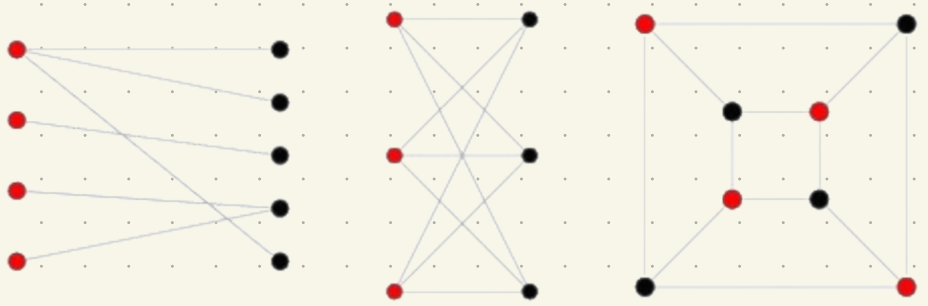
Correctness:

solution to input problem $\Longleftrightarrow$ flow of value k in $\tilde{G}$

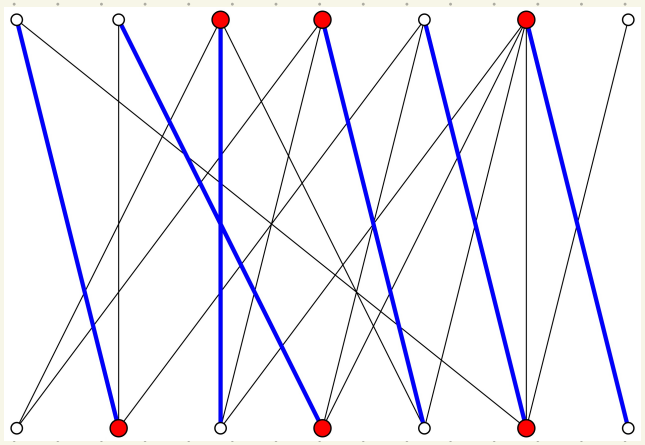
Bipartite Graphs

Any graph where vertices can be divided into 2 sets (usually L & R) s.t. no edges exist inside L or R

Ex:



Maximum matching:
find edges (no 2 edges per vertex)



Instead, use flows:

Convert G to $\tilde{G}$:

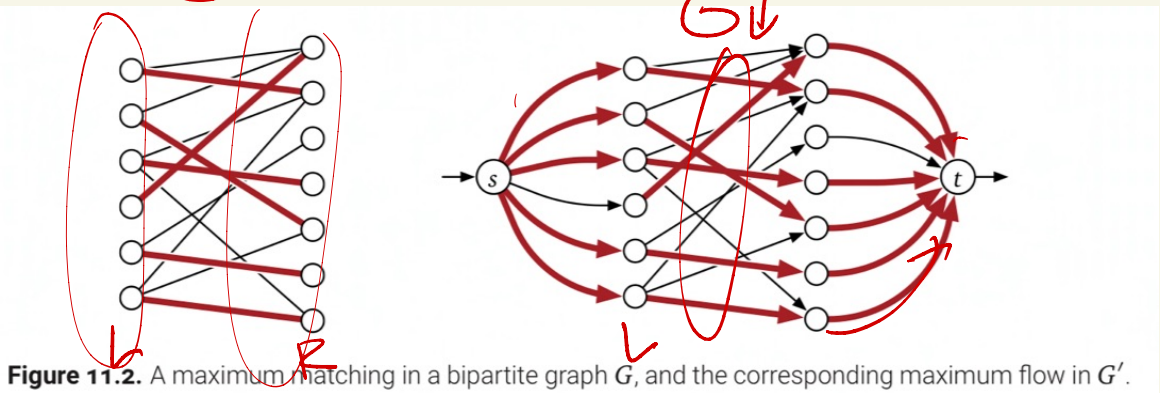


Figure 11.2. A maximum matching in a bipartite graph G, and the corresponding maximum flow in G'.

FF: $(\tilde{G})$

$O((V+E)\cdot f)$

$O((V+E)\cdot V)$

Build $\tilde{G}$: $\tilde{V} = s + t + L + R$

$\tilde{E} = s \to v, \ \forall v \in L$

$u \to t, \ \forall u \in R$

$\forall e = uv \in E, u \in L, v \in R, u \to v$

& capacities: all $= 1$

Algorithm: Given $G = (V, E)$ with $V = L \cup R$ (bipartite)

// build $\tilde{G}$ (see last slide)

$\tilde{V} = V + 2$ vertices

$\tilde{E} = E + V$ edges

$O(V+E)$ to build

// run flow

$f \leftarrow$ MaxFlow$(\tilde{G})$ $O(\tilde{V}\tilde{E}) = O(VE)$

// get matching

$\forall uv$, $u \in L$ & $v \in R$, if $f(e) = 1$, add to matching $O(E)$

Runtime: $O(VE)$

Correctness:

① Any matching in G of size k

$\Rightarrow$ flow in $\tilde{G}$ of $|f| = k$

Given matching M, build the flow f':

② Any flow in $\tilde{G}$ $\Rightarrow$ matching in G

Given flow f, build a matching M:

$L \to R$ w/ flow $= 1$, in matching

$\Rightarrow$ max flow $\Leftrightarrow$ max matching

Aside: How??

(FF is somehow improving matching...)

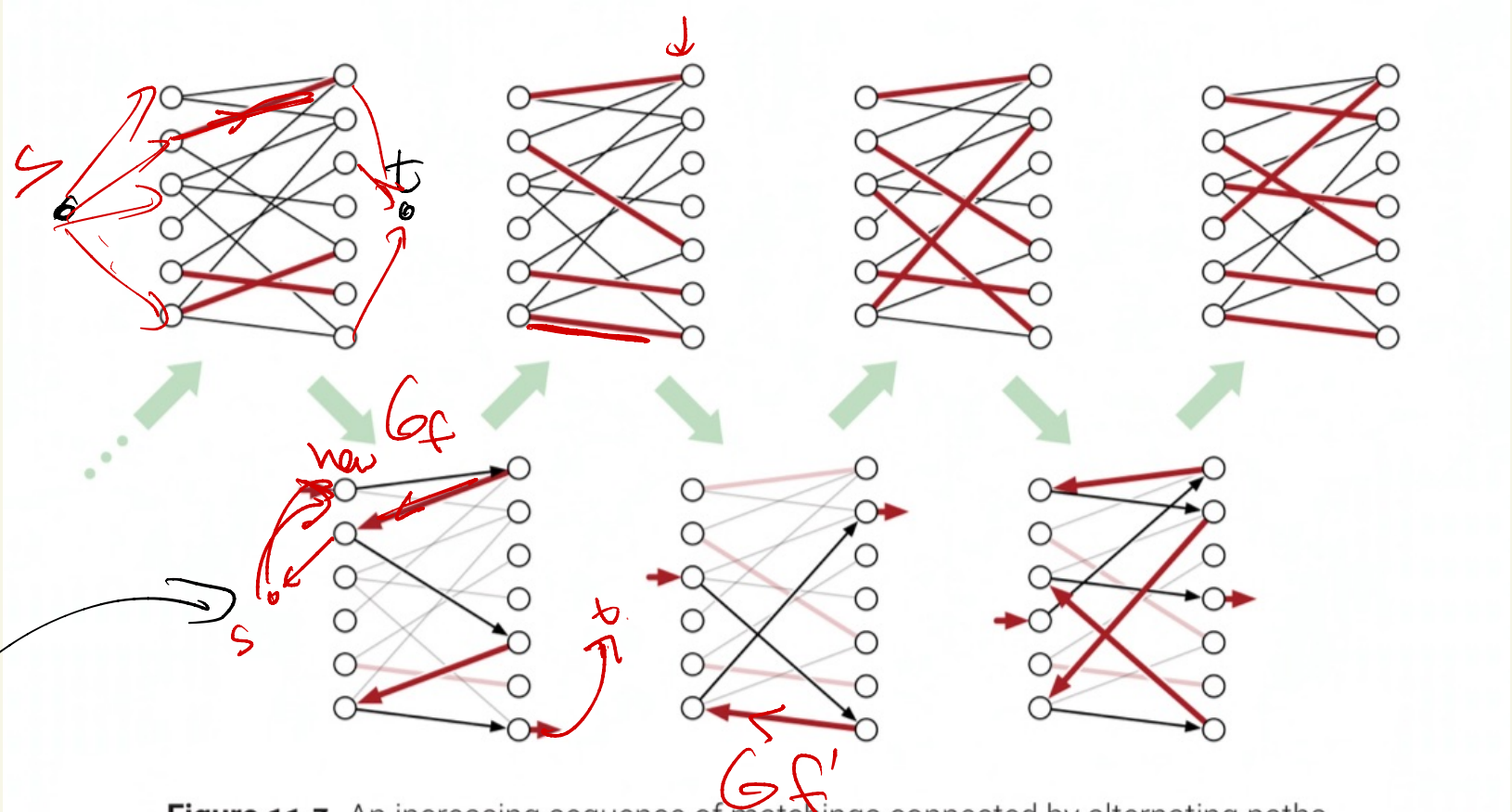


Figure 11.3. An increasing sequence of matchings connected by alternating paths.

Augmenting paths

Aside: can do better: $O(\sqrt{V}E)$

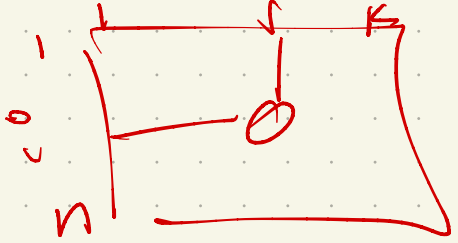
Crazier "word problem" examples

A company sells k products, & keeps records on customers.

Goal: Design a survey to send to n customers, to get feedback.

- Each customer's survey shouldn't be too long, & should ask only about products they purchased
- Each product needs some # of reviews from different customers

Input: - k products
- n customers
- records of who bought what:
a_{ij} for $i \le k$, $j \le n$



- For each customer, C_i is max # of products to ask them about
C[1..n]
- for each product, P_i is minimum # of reviews needed P[1..k]

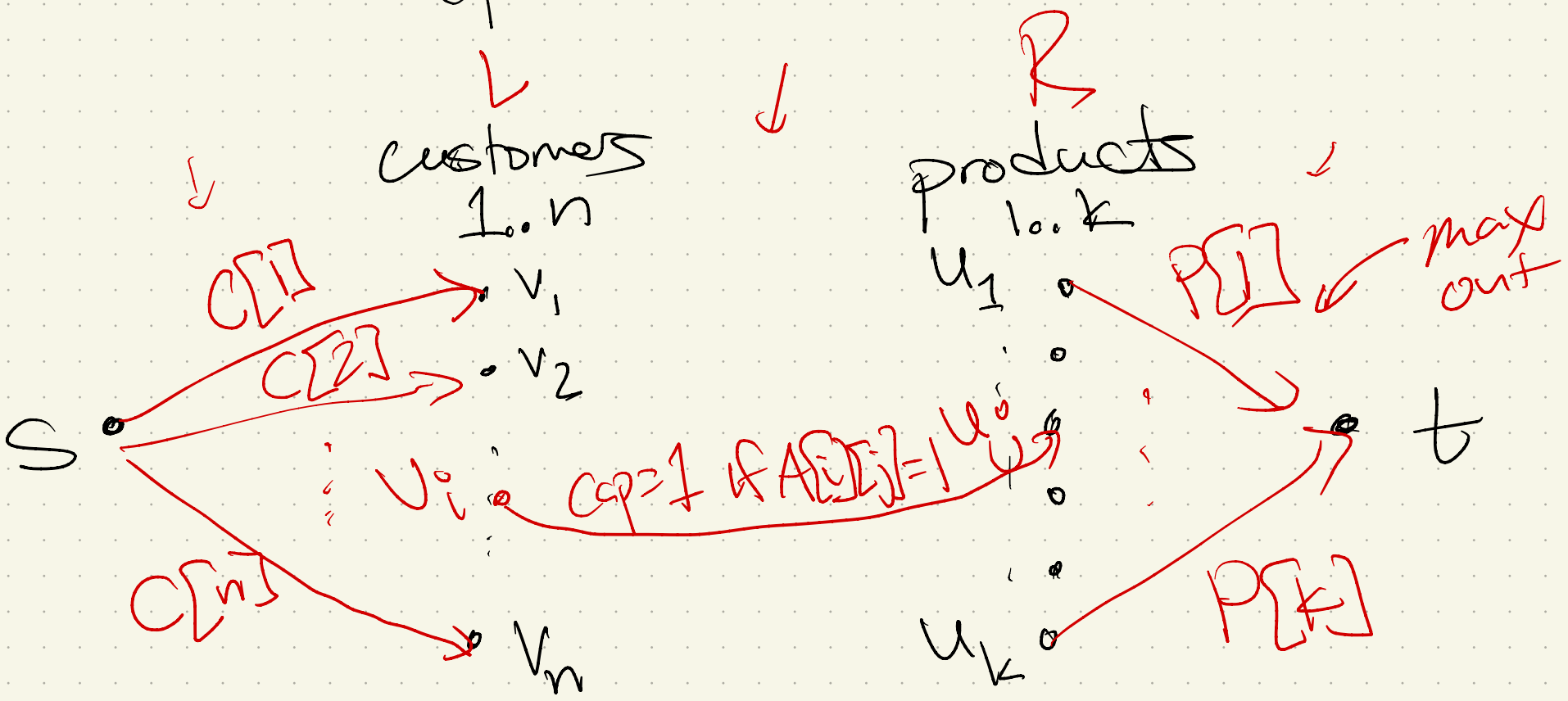
How to design? flows!

Algorithm: Input: $C[1..n]$ ~ # of asks (max)

$P[1..k]$ ~ # of reviews needed

$A[1..n][1..k]$ ~ { customer i purchased product j

Build $\tilde{G}$: (picture first)



f ← Run Maxflow(G)

if val(f) = $\sum_{i=1}^{k} p[i]$

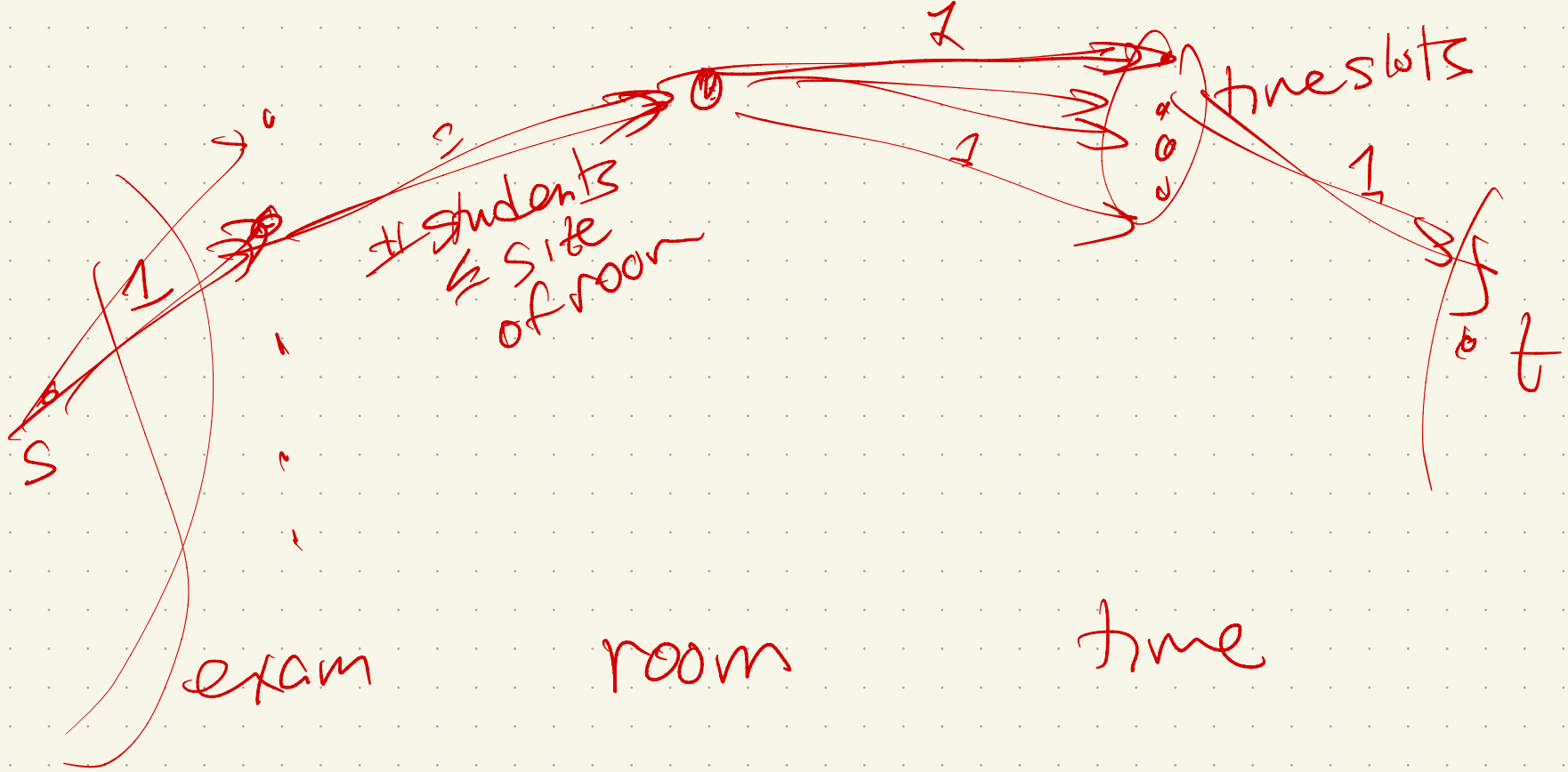
↳ yes: design survey

↓ Orlin

Runtime: $O(VE) = O((k+n)(nk))$

$V = 2 + k + n$

$E \leq n + k + nk$ ← upper bound for L → 12


1
timeslots
1
1
students ≤ size of room
1
s
t
exam
room
time