Algorithms - Spring '25

Flow applications

Recap

- HW due ~~Friday~~ Monday
- Office hours: today: 1-3pm

  tomorrow: might move on zoom

  ↳ will post on slack by 1pm with details
- Readings: start Friday
- Poll: last topic → LP

Topics in Ch. 11

A mess of different ideas!

① Matchings: identify a way to pair up items

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

② Disjoint paths:
Modify G: $\longrightarrow$ turn into flow network
Find paths that avoid each other.

③ "Tuple" Selection
Build a graph: flow paths give selection

Magic: All use flows to solve.

Step back: reductions again!

In these examples, algorithm is usually:

Build $\tilde{G}$ from input

Run max flow

↳ encode problem solution

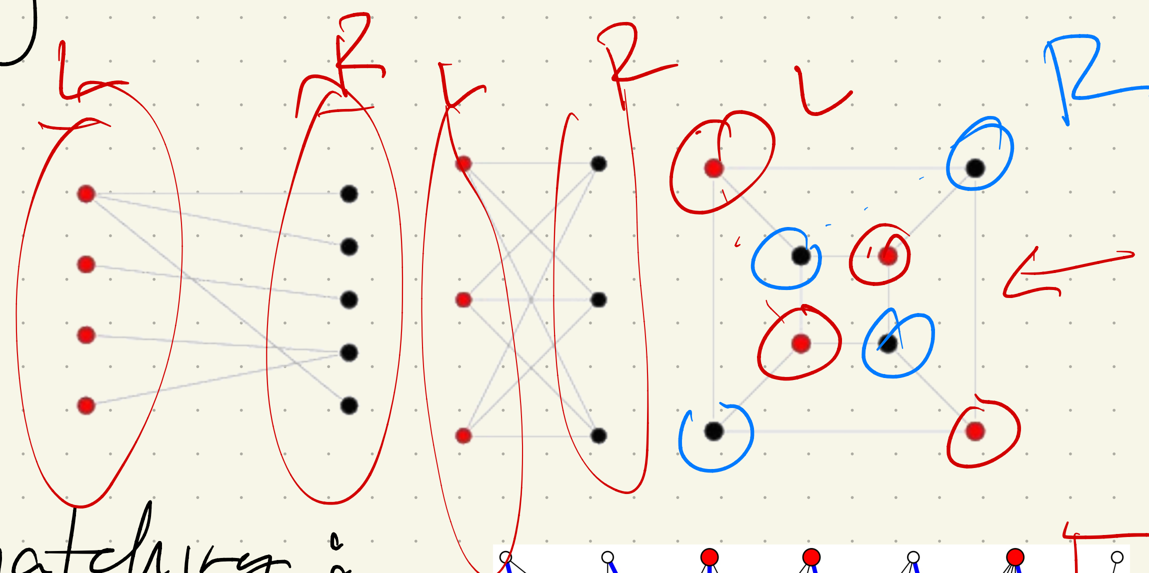
Correctness:

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

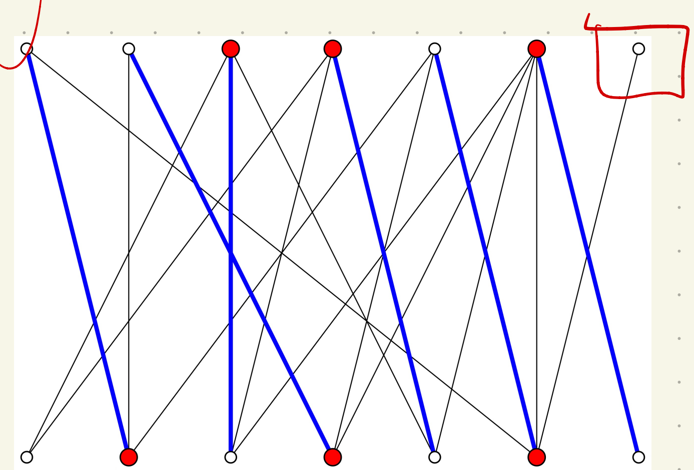
Bipartite Graphs

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 $G=(V,E)$ to $\tilde{G}$:

Suppose $V = L \cup R$

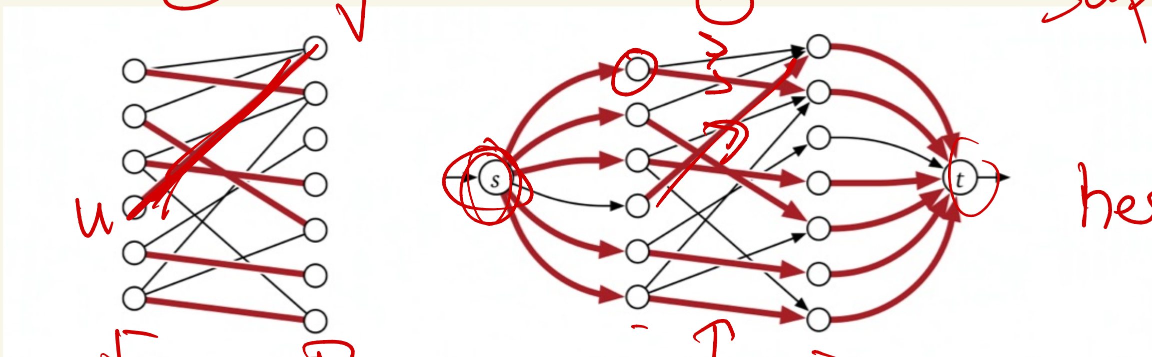


Figure 11.2: A maximum matching in a bipartite graph G, and the corresponding maximum flow in G'.

here: $f \le \tilde{V}$

Build $\tilde{G}$: $\tilde{V} = V \cup \{s\} \cup \{t\}$

$\tilde{E} = \{e \in E, e = \{u,v\}, u \in L, v \in R \mid u \to v\} \cup \{\forall u \in L, s \to u\} \cup \{\forall v \in R, v \to t\}$

& capacities: set all capacities $= 1$ (all edges in $\tilde{E}$)

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

// build $\tilde{G}$

$\tilde{V} \leftarrow L \cup R \cup \{s,t\}$

$\tilde{E}$: Add edges: $\forall u \in L$, $s \to u$ and $\forall v \in R$, $v \to t$
$\forall e=\{u,v\} \in E$, $u \in L$ & $v \in R$, add $u \to v$ to $\tilde{E}$
$\forall e \in \tilde{E}$, set $c(e) \leftarrow 1$

// run flow

$FF(\tilde{G})$

// get matching

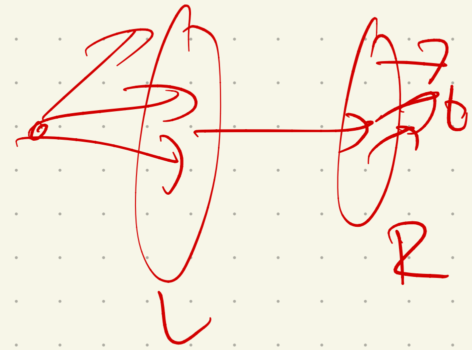
return $val(f)$

if $f(e)=1$ for e going from $L \to R$,
add e to matching

Runtime: Orlin: $O(\tilde{V}\tilde{E})$

FF: $O((\tilde{V}+\tilde{E})\cdot f)$

Input: $G=(V,E)$



Here: $\tilde{V} = V+2$

$\tilde{E} = V+E$

and $f \le V$

Overall: Orlin $\tilde{V}\tilde{E} = (V+2)(V+E)$

FF: $(\tilde{V}+\tilde{E}) f \le \underline{(V+2+V+E)\cdot V}$

$\hookrightarrow O(V(V+E)) = O(VE)$

Correctness: Reduction:

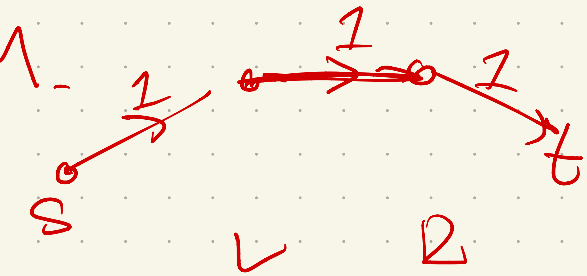
① Any matching in G $\Rightarrow$ flow in $\tilde{G}$

valid: edge + vertex constraints are OK.

Given matching M, build the flow f':

Consider edge $e \in$ matching M.

Build flow path of value 1:



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

Given flow f, build a matching M:

All edges go $s \to L$, $L \to R$, $R \to t$

Consider $s \to L$ edges: one per vertex in L.

Then flow path goes $L \to R$, $R \to t$

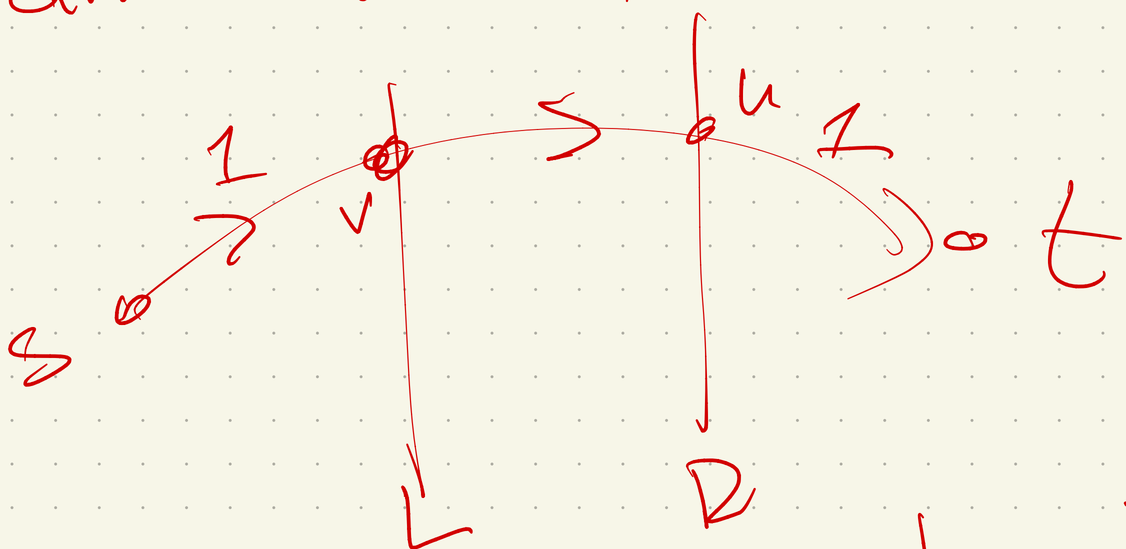
$\Rightarrow$ max flow $\Longleftrightarrow$ max matching

Why matching?

Consider $L \to R$ edges with flow 1.

Claim: at most one per $v \in L$

and $u \in R$.



Why? Only one edge to v,

with capacity 1.

Use flow decomp $\Rightarrow$ get matching.

Aside: How??

(FF is somehow improving matching...)

in residual G_f

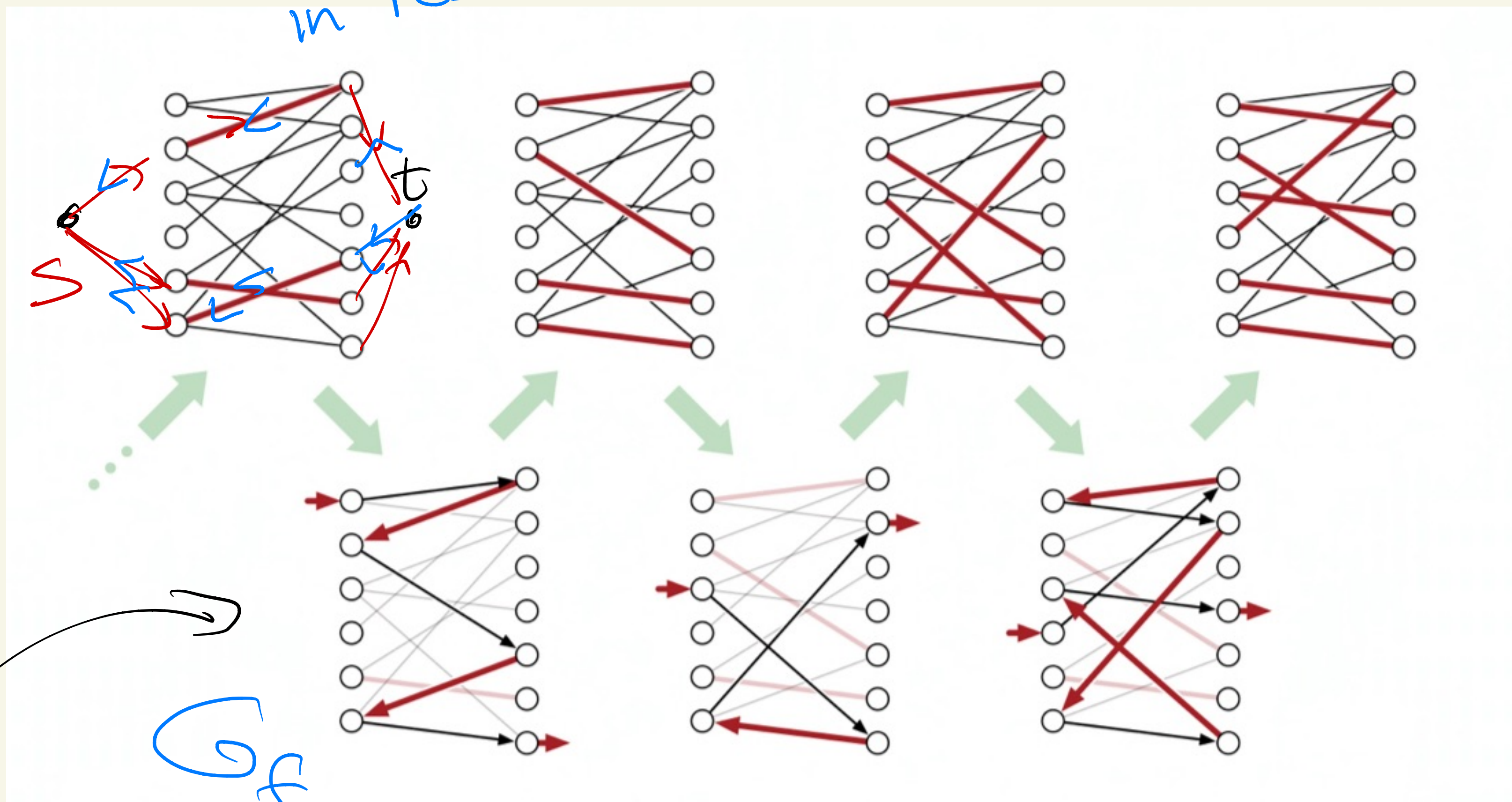


Figure 11.3. An increasing sequence of matchings connected by alternating paths.

Augmenting paths

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: $A[i,j]$
a_{ij} for $i \le k$, $j \le n$

- For each customer, c_i is max # of products to ask them about $C[i]$

- for each product, p_i is minimum # of reviews needed $P[i]$

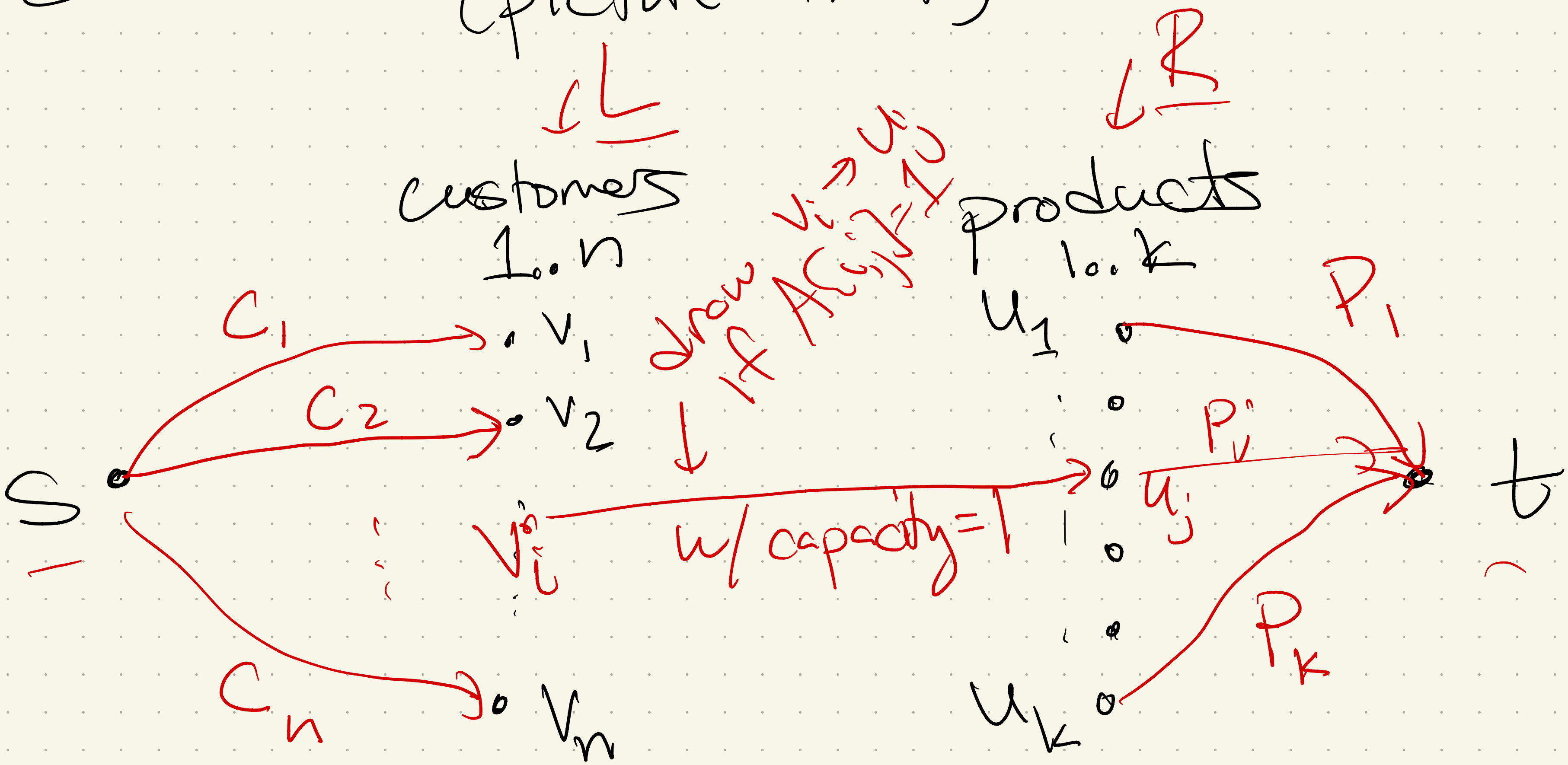
How to design?

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

$P[1..k]$ ← # of reviews needed

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

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



More formally:

// Build G:

$L \leftarrow$ one vertex per customer: $\{v_1, \dots, v_n\}$

$R \leftarrow$ one vertex per product: $\{u_1, \dots, u_k\}$

$V \leftarrow L \cup R \cup \{s, t\}$

set capacities: $s \rightarrow v_i$ gets cap. c_i

$u_j \rightarrow t$ gets cap p_j

and $v_i \rightarrow u_j$ gets cap $= 1$, $\forall i+j$

// Run max flow

FF(G)

// Can we find assignment?: $f = \sum_{j=1}^{k} p_j$: Yes!

If edge $v_i \rightarrow u_j$ has flow $= 1$,

have customer i review product j

Correctness

Suppose can find customer assignments.

Build flow:

Send 1 flow path $s \to$ customer $\to$ product $\to t$ ✓

Total: $\sum p_j$

Suppose G has a flow of value $\sum_{i=1}^{k} p_i$.

Use flow paths:

know no customer will get $> c_i$ (b/c capacities)

know each product must get p_i

know each product only gets reviewed by purchasers

Runtime:

Orlin: $V \cdot E$

$$V = 2 + \overset{\text{products}}{k} + \overset{\text{customers}}{n} = O(k+n)$$

$$E \le n + k + \underbrace{nk}_{L \to R \text{ edges}} = O(nk)$$

$$\Rightarrow O((n+k)(nk))$$

Another: Exam scheduling

Input: n classes, r classrooms
t time slots, p proctors

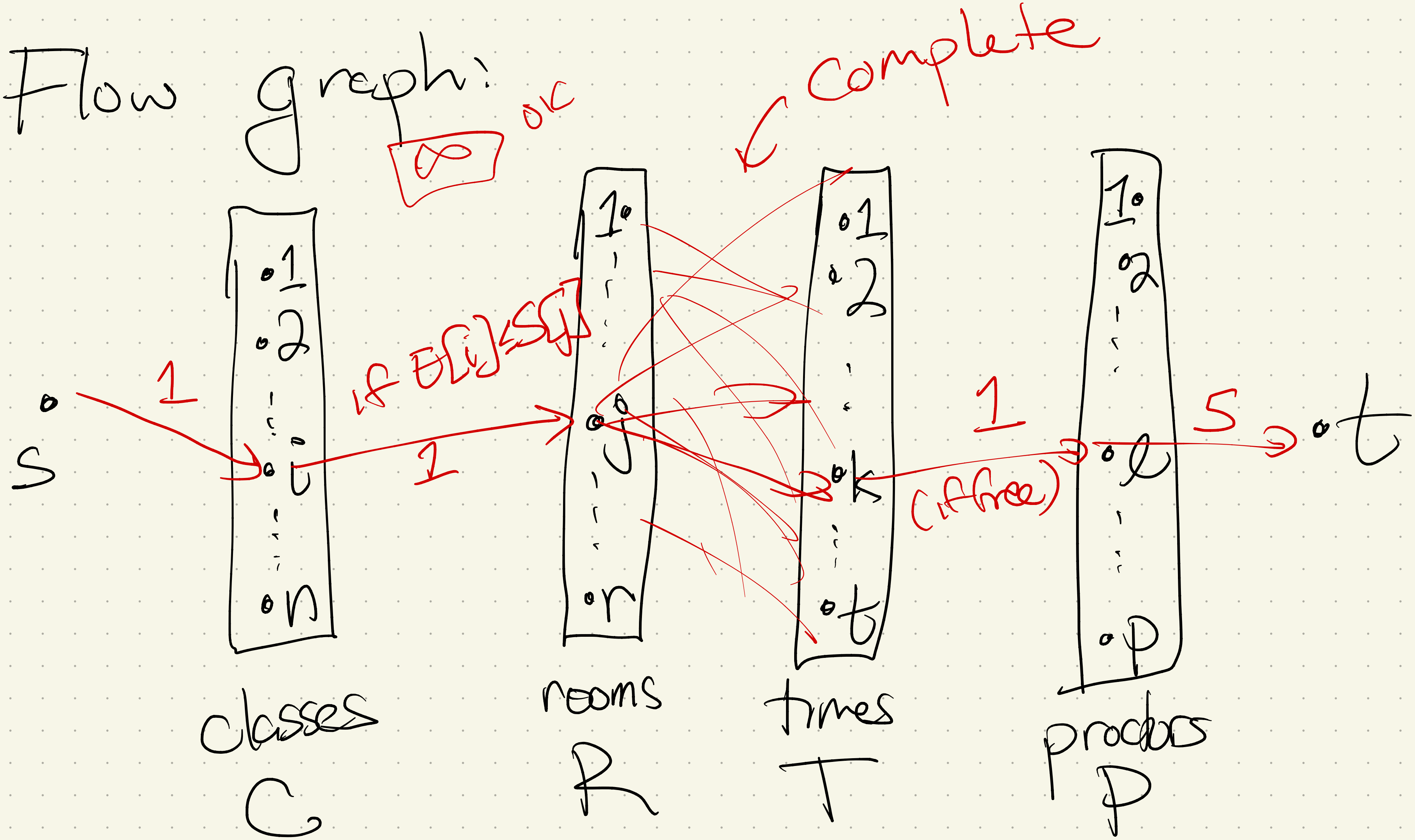
$E[1..n]$: # of students in each class

$S[1..r]$: capacity of each classroom

$A[1..t, 1..p]$: $A[k, \ell]$ is true if ℓ^{th} proctor is free at time slot k

& each proctor gets ≤ 5 classes.

Flow graph:



Edges: $\forall v \in C$, $s \to v$ with cap $= 1$,
so flow paths "assign" 1 class to
valid room, time & proctor

Then $C \to R$ edges:

If $E[i] \leq S[j]$: class will fit in room.

So add edge $i \to j$ [for $i \in C$ + $j \in R$].

capacity = $\cancel{2}$ 1

Then $R \to T$ edges:

add all edges $j \to k$ with capacity = 1, since each room is open to start at every time

Next: $T \to P$ edges

If $A[k, \ell]$ is true, then proctor ℓ is open at time k

↳ add edge of capacity $= 1$

(so can't be assigned 2)

Finally: $P \to t$

Add all $\ell \to t$ edges, for $\ell \in P$

Capacity $= \cancel{?}$ 5

Then: find max flow.

If $= n$, done! If $< n$: problem.

r.t nodes

room1

Find flow paths:

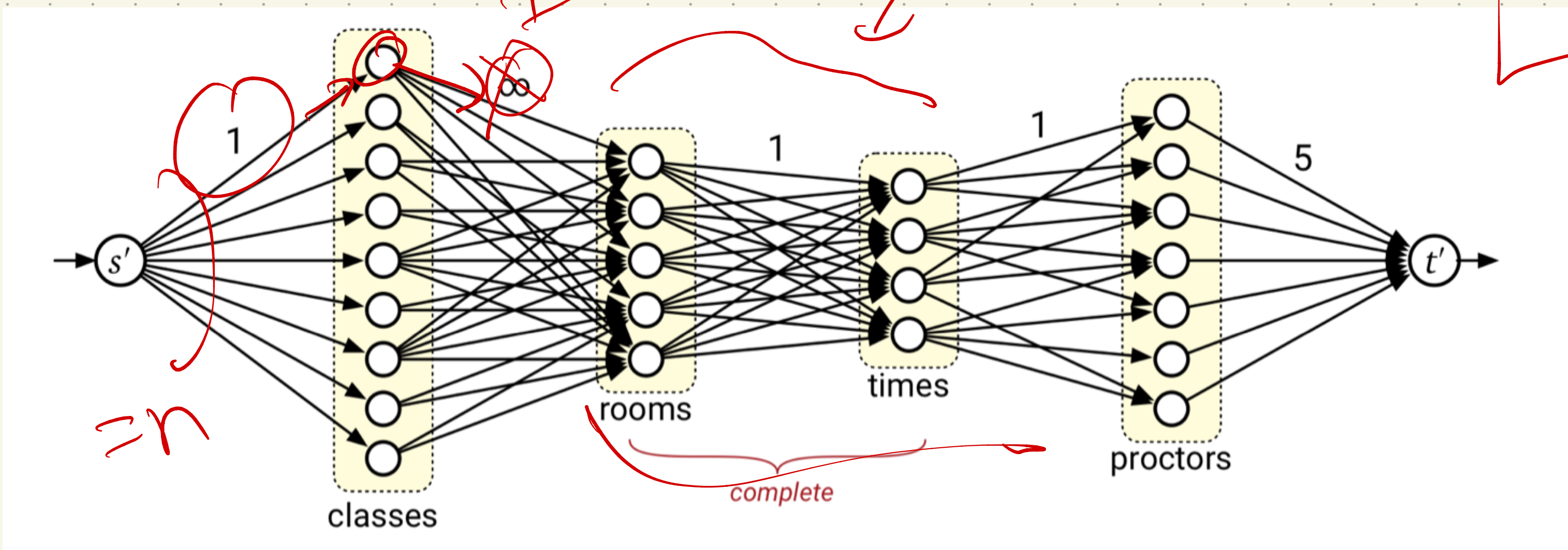


Figure 11.5. A flow network for the exam scheduling problem.

Must go $s \to i \to j \to k \to l \to t$.

So: if $\exists$ flow of value n, can find assignment of exams.

Other way:

If can assign rooms, classes, times, & proctors, can also use each assignment to build a flow path of value 1 in a flow

G. So, assignment $\Rightarrow$ flow.

Runtime:

$V =$

$E =$

+ Orlin $= O(VE) =$