

# Algorithms - Spring '25

Flow applications



## Recap

- HW due ~~Friday~~ Monday
- Office hours: Today: 1-3pm
  - tomorrow: might move on zooms  
↳ will post on slack by 1pm with details
- Readings: Start next Monday

# 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  $G$ :  $\rightarrow$  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

Correctness:

solution to  
input problem

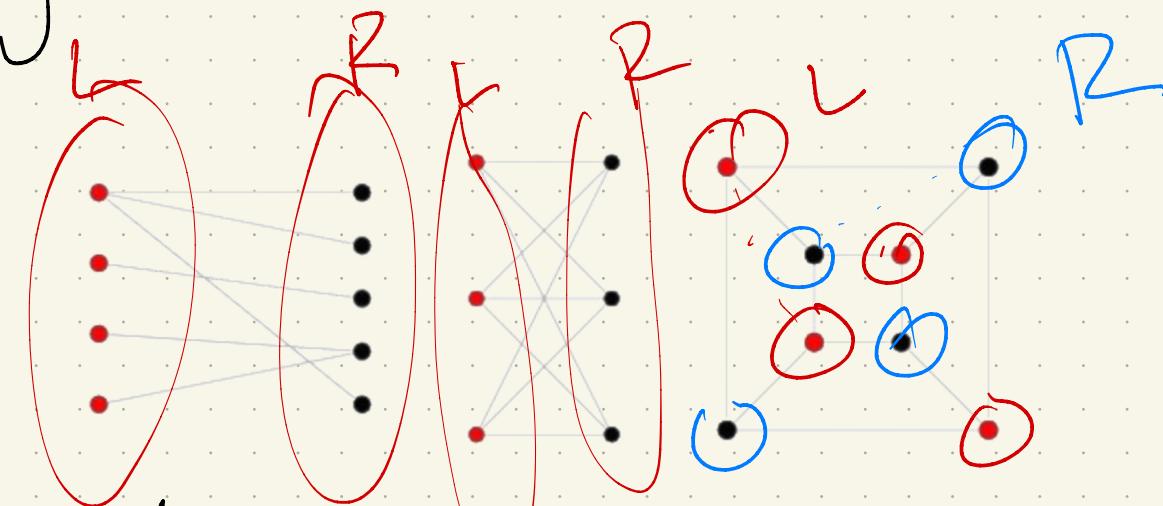


flow of  
value  $k$   
in  $\tilde{G}$

# Bipartite Graphs

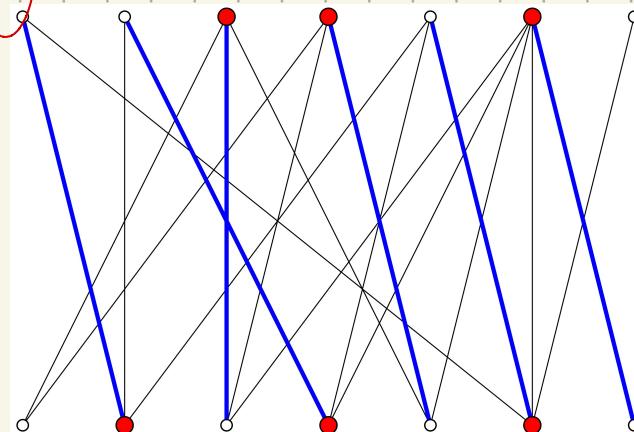
Any graph where vertices can be divided into 2 sets (usually L & R)  
s.t. no edges exist inside L or R

Ex:



Maximum matching :

Find edges  
(no 2 edges  
per vertex)



Instead, use flows:

$G = (V, E)$

Convert  $G$  to  $\tilde{G}$ :

Suppose  $V = L \cup R$

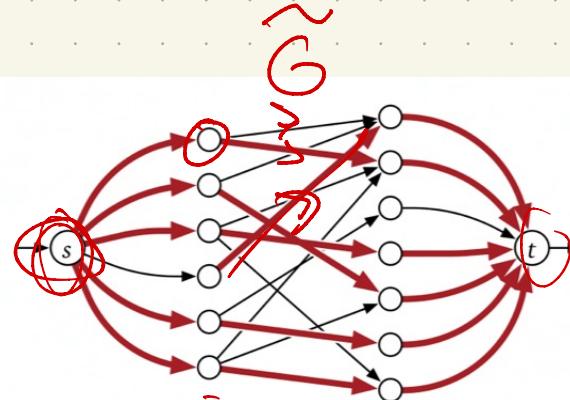
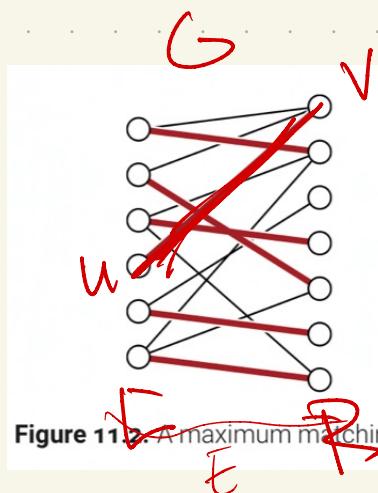


Figure 11.2: A maximum matching in a bipartite graph  $G$ , and the corresponding maximum flow in  $G'$ .

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

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

$\tilde{E} = \{e \in E, e = \{u, v\}, u \in L, v \in R\}$   
 $\quad \quad \quad \cup \{u \rightarrow v\} \cup \{t \leftarrow u, s \rightarrow u\}$   
 $\quad \quad \quad \cup \{\forall v \in R, v \rightarrow t\}$

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

Algorithm: Given  $G = (V, E)$

with  $V = L \cup R$  (bipartite)

// build  $\tilde{G}$   
 $\tilde{V} \leftarrow L \cup R \cup \{s, t\}$   
 $E$ : Add edges:  $\forall u \in L, s \rightarrow u$  and  $\forall v \in R, v \rightarrow t$   
 $\forall e = \{u, v\} \in E, u \in L \wedge v \in R$ , add  $u \rightarrow v$  to  $E$   
 $\forall e \in E$ , set  $c(e) \leftarrow 1$   
// run flow

$FF(\tilde{G})$

// get matching

return  $val(F)$

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

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

Here:  $\tilde{V} =$

$\tilde{E} =$

and  $f \leq$

Overall

Correctness:

① Any matching in  $G$

$\Rightarrow$  flow in  $\tilde{G}$

Given matching  $M$ , build the flow  $f'$ :

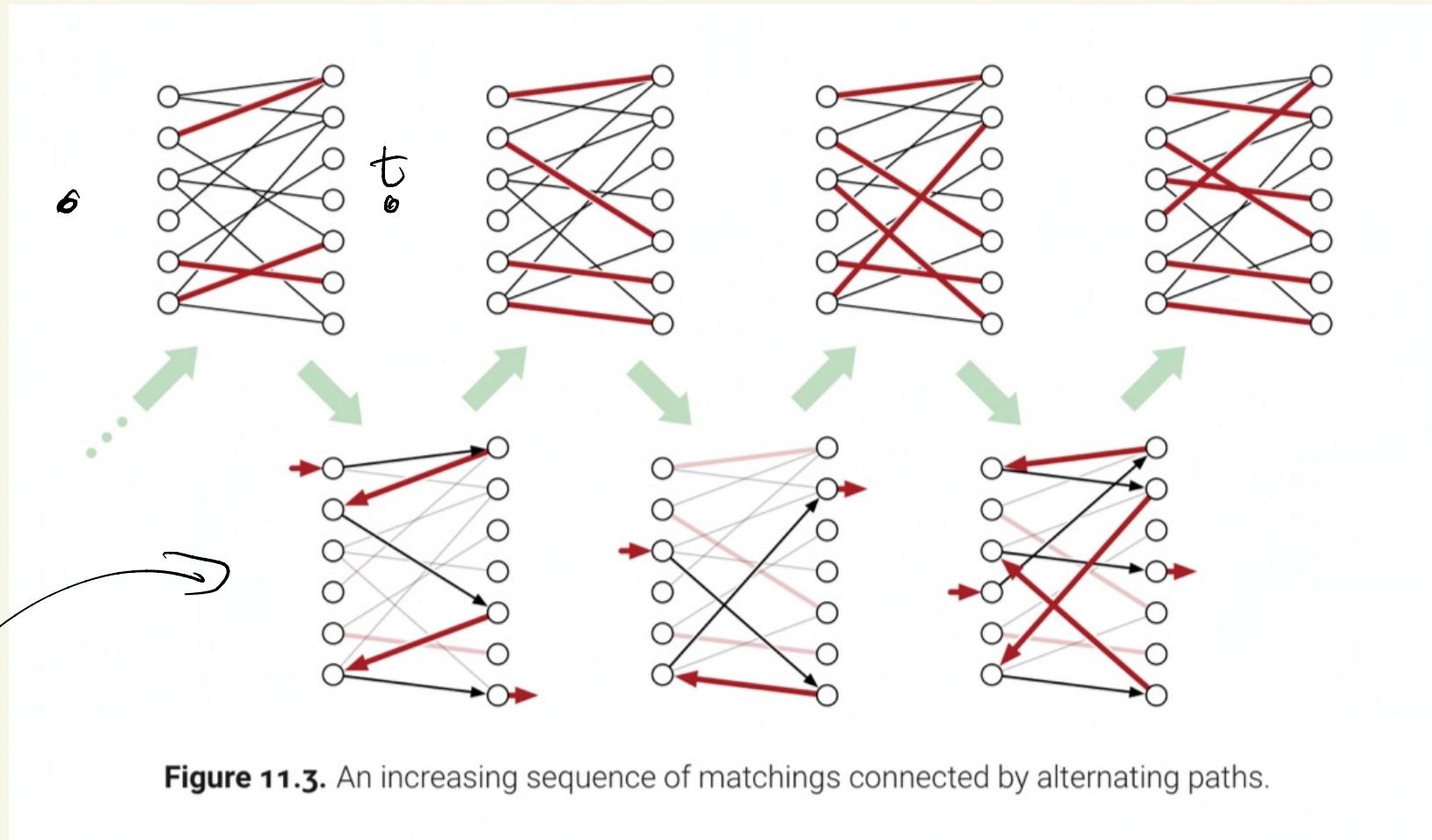
② Any flow in  $\tilde{G} \Rightarrow$  matching in  $G$

Given flow  $f$ , build a matching  $M$ .

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

Aside: How??

(FF is somehow improving matching...)



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 feed back.

- 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 :-

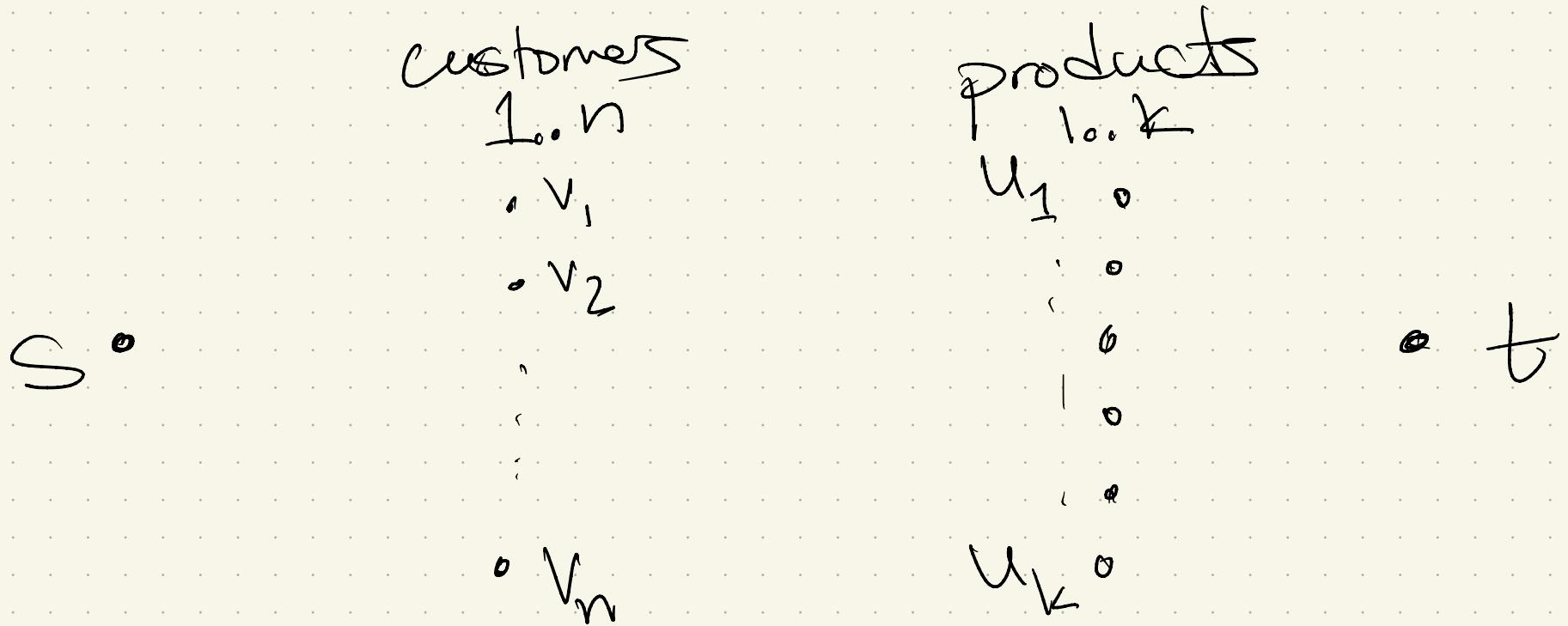
- k products
- n customers
- records of who bought what :  
 $a_{ij}$  for  $i \leq k, j \leq n$

- For each customer,  $C_i$  is max # of products to ask them about
- for each product,  $P_i$  is minimum # of reviews needed.

How to design?

Algorithm: Input:  $C[1..n]$   $\leftarrow$  # of tasks (max)  
 $P[1..k] \leftarrow$  # of reviews needed  
 $A[1..n][1..k] \leftarrow$  {customer j purchased product i}

Build  $\tilde{G}^o$  (picture first)



More formally:

//Build G!

P ←  
R ←  
V ←

//Run max flow

//Can we find assignment?

## Correctness

Suppose can find customer assignments.  
Build flow:

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

Runtmei

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

$A[1..t, 1..P]$ :  $A[k, l]$  is true  
if  $l$ th proctor is free at time slot  $k$

& each proctor gets  $\leq 5$  classes.

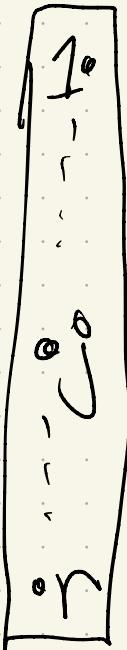
# Flow graph:

$s^0$



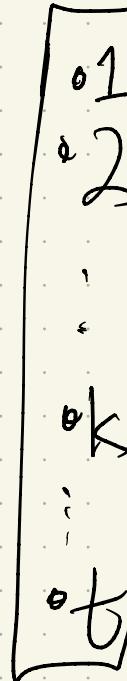
classes

C



rooms

R



times

T



proctors

P

$t^0$

Edges:  $\forall v \in C, s \rightarrow v$  with  $cap = 1$ ,

so flow paths "assign" 1 class to  
valid room, time & proctor

Then  $C \rightarrow R$  edges:

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

So add edge  $i \rightarrow j$  [for  $i \in C, j \in R$ ].  
capacity = ?

Then  $R \rightarrow T$  edges:

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

Next:  $T \rightarrow P$  edges

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

→ add edge of capacity = 1  
(so can't be assigned 2)

Finally:  $P \rightarrow t$

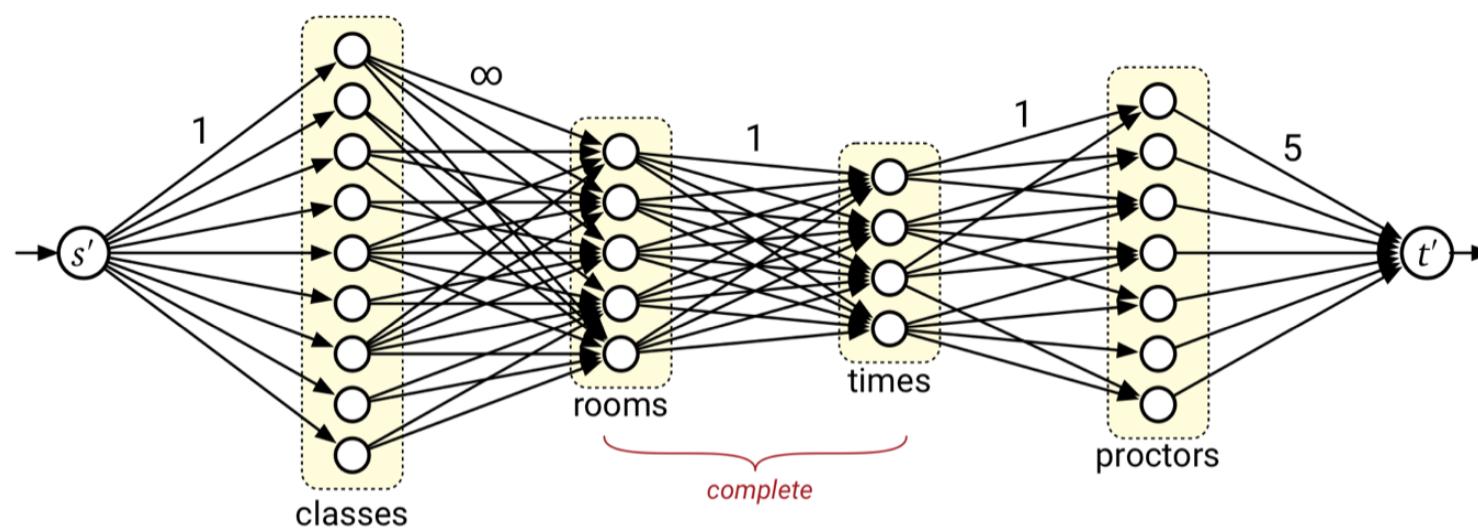
Add all  $l \rightarrow t$  edges, for  $l \in P$

Capacity = ?

Then: find max flow.

If  $= n$ , done!

Find flow paths:



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

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

So: If I 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  
G. So, assignment  $\Rightarrow$  flow.

Runtime:

$$V =$$

$$E =$$

$$+ O(nk) = O(VE) =$$