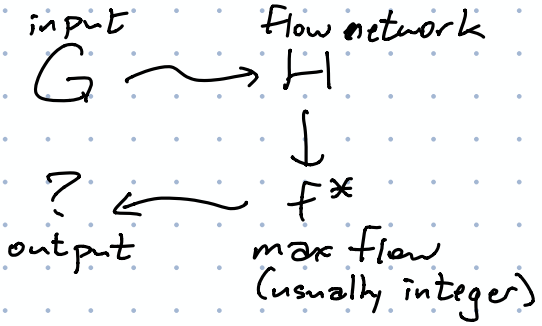
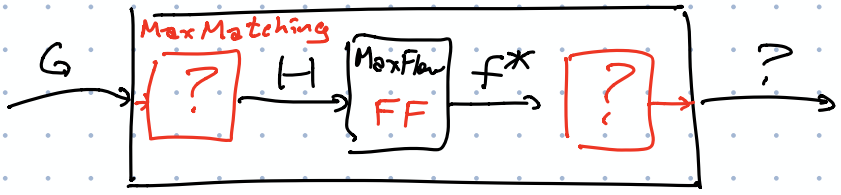
Applications of max flow/min cut

- edge-disjoint paths
- vertex-disjoint paths
- vertex capacities
- bipartite maximum matching





- Transform input
- Transform output

} Features in solution ⇌ paths in max flow

- Time (in terms of original input)
- Prove correct

Exam Scheduling

$N = n + r + pt$ = total input size

- n classes
- r rooms
- t time slots
- p proctors

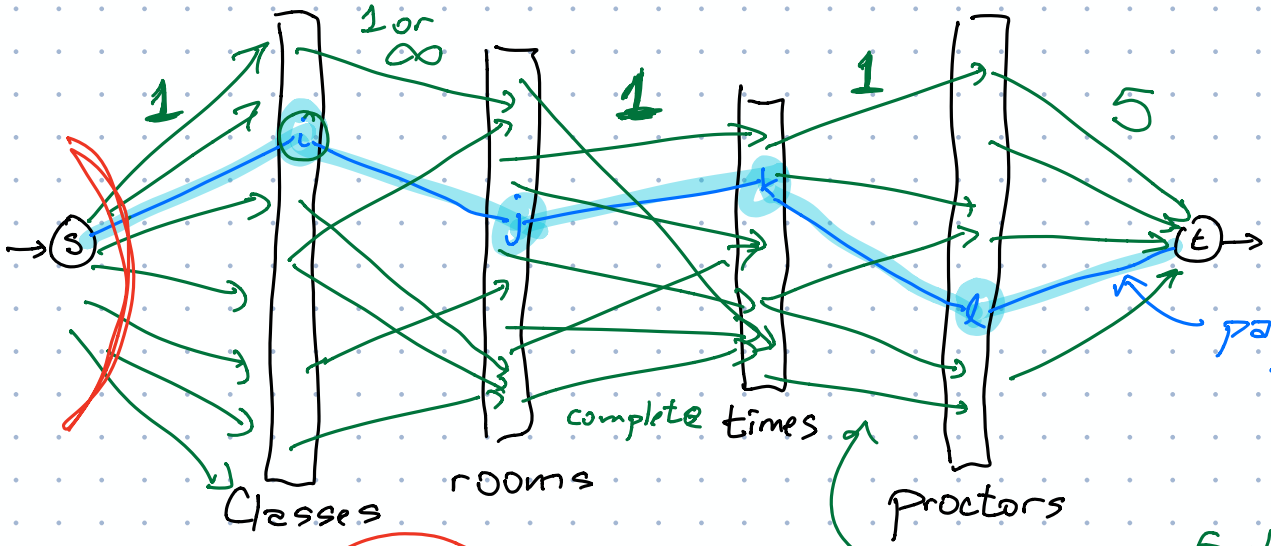
$E[1..n]$ — enrollment
$S[1..r]$ — # seats
$A[1..t, 1..p]$ — availability

$A(k,l) = T \iff$ proctor l is available at time k

- Every class needs scheduling
- $E \le S$ in any exam
- ≤ 1 exam per room per time slot
- Each proctor oversees ≤ 5 exams

Schedule = set of 4-tuples (i, j, k, l) — one per exam

class i, room j, time k } capacity 1
proctor l } available



Each class has ≤ 1 exam

$f[i] \leq S[j]$

At most one exam in any room at any time

~~Availability~~ $A[k,l] = \text{True}$

Each proctor can only watch one exam at any time

Each proctor writes ≤ 5 exams

Feasible Integer flow = Σ paths = set of successfully scheduled exams

Is $|\text{Max Flow in } H| = \#\text{classes}$?

$V = n + r + t + p + 2 = O(N)$

$E \leq n + nr + rt + tp + p = O(N^2)$

$\text{Time} = O(VE) = \boxed{O(N^3)}$

Tuple Selection

Input: • Finite sets $X_1, X_2, \ldots, X_d$ (represent discrete resources)

- $c(x)$ For all $x \in X_i$ For all i
- $c(x,y)$ For all $x \in X_i, y \in X_{i+1}$ For all i

Output: largest set of tuples $(x_1, x_2 \ldots x_d) \in X_1 \times X_2 \times \cdots \times X_d$

subject to constraints

- For each index i, each $x \in X_i$ appears in $\leq c(x)$ tuples
- For each index i each $x \in X_i$ and $y \in X_{i+1}$ appear in $\leq c(x,y)$ tuples

Max Matching:

Input $G = (L \cup R, E)$

Output max matching

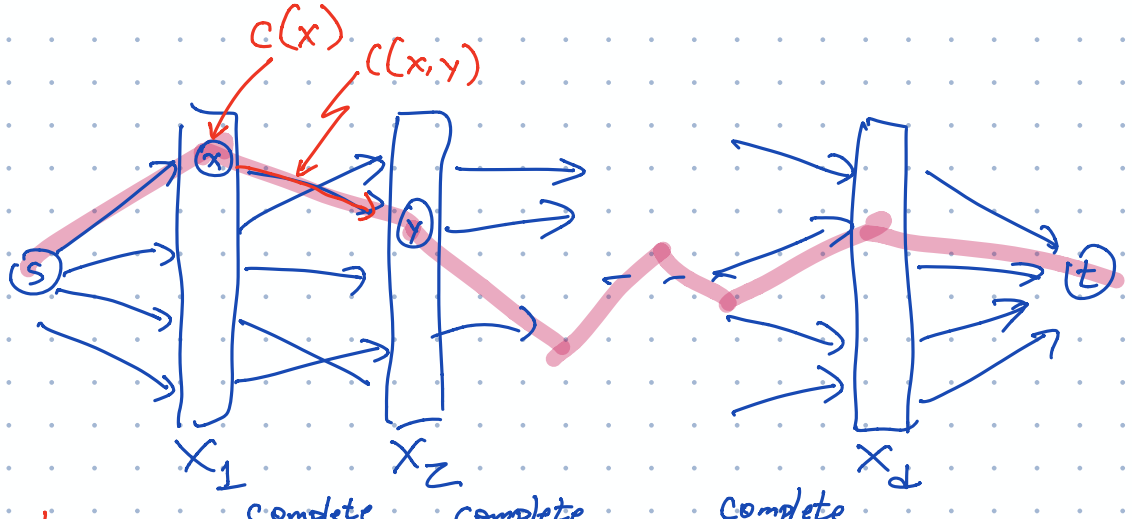
set of pairs $(x,y) \in L \times R$

$X_1 = L$

$X_2 = R$

$c(x,y) = \begin{cases} 1 & \text{if } xy \in E \\ 0 & \text{if } xy \notin E \end{cases}$

$c(x), c(y) = 1$

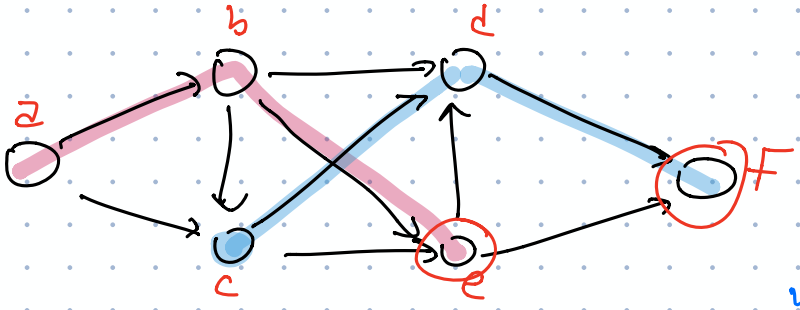


largest valid set of tuples ⟺ paths ⟺ maximum feasible flow

Disjoint-Path Cover

Input: dag $G=(V,E)$ — general graphs → NP-hard

Output: min # disjoint paths that cover every vertex



Intuition: We want to assign (match) a successor to (with) as many vertices as possible

a → b
b → e
c → d
d → f
e ✗
f ✗

paths = # verts with no successor

Reduce to max matching

Build $H=(L \cup R, E)$

$L = V$

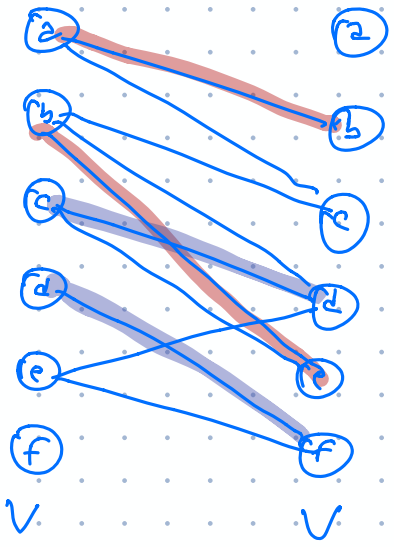
$R = V$ (copy)

Find max matching M

↓

paths in G

paths $= \#V - \#M$



$O(VE)$ time

Project Selection ("Open-pit mining")

Input: dag $G=(V,E)$

Profit $\$(v)$ for every vertex v

V = projects

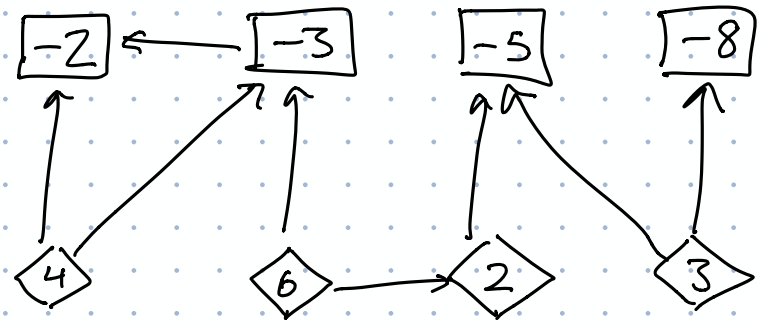
E = dependencies

$u \to v$ means u can only be done after v.

Output: Subset $S \subseteq V$

s.t. for all $u \to v$, $u \in S \Rightarrow v \in S$

$\max \$(S) = \sum_{v \in S} \$(v)$

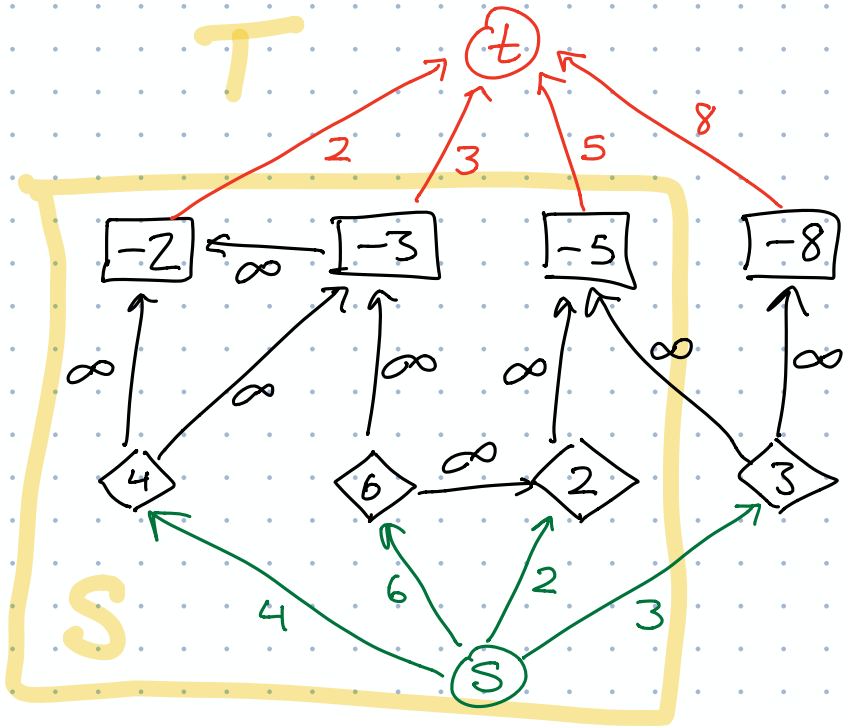


Partition $V \longrightarrow S \cup T$

reduce to min-cut problem

S — select

T — throw out



Build H

Compute max flow f^*

return $P - |f^*|$

$O(VE)$ time

$$P = \sum_{v} \max\{\$(v), 0\} = \sum_{\$(v)>0} \$(v)$$

— max possible profit (w/o depend.)

$profit(S) = P - \|S,T\|$ ← Claim

For any $X \subseteq V$ $\quad cost(X) = \sum_{\substack{\$(v)<0 \\ v \in X}} -\$(v) = \sum_{v \in X} c(v \to t)$

$$yield(X) = \sum_{\substack{\$(v)>0 \\ v \in X}} \$(v) = \sum_{v \in X} c(s \to v)$$

$$profit(X) = yield(X) - cost(X) = \sum_{v \in X} \$(v)$$

$$P = yield(V) = yield(S) + yield(T)$$

$$\|S,T\| = cost(S) + yield(T)$$

$$P - \|S,T\| = yield(S) - cost(S) = profit(S)$$ ✓