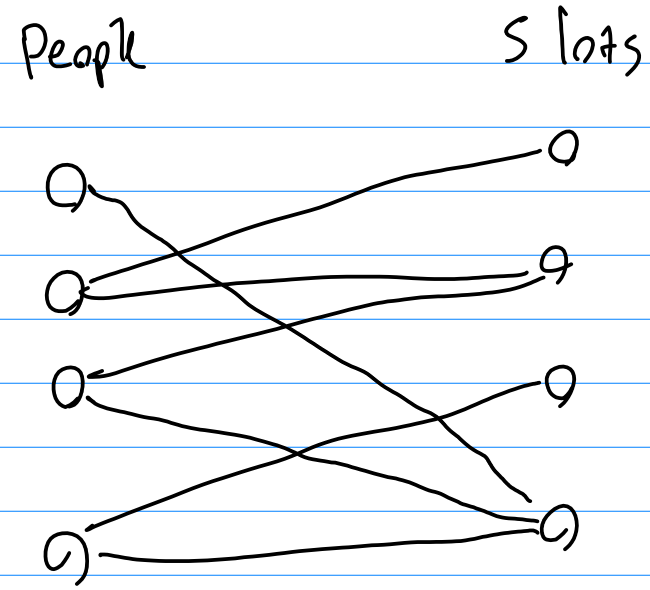
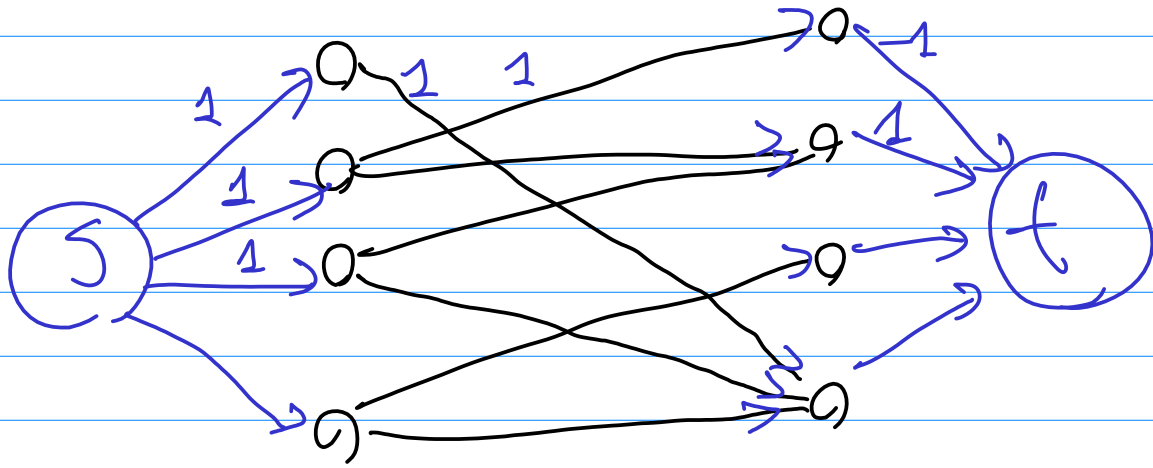
Bipartite Matching (& other flow applications)



draw edge if person can make slot

Each person needs one slot
each slot can take 1 person

Q: how many people can be accommodated?



Answer: max flow, draw S, t,
edges S→left, right→t,

orient all edges to right.

why?

(1) any matching of size m gives size-m flow

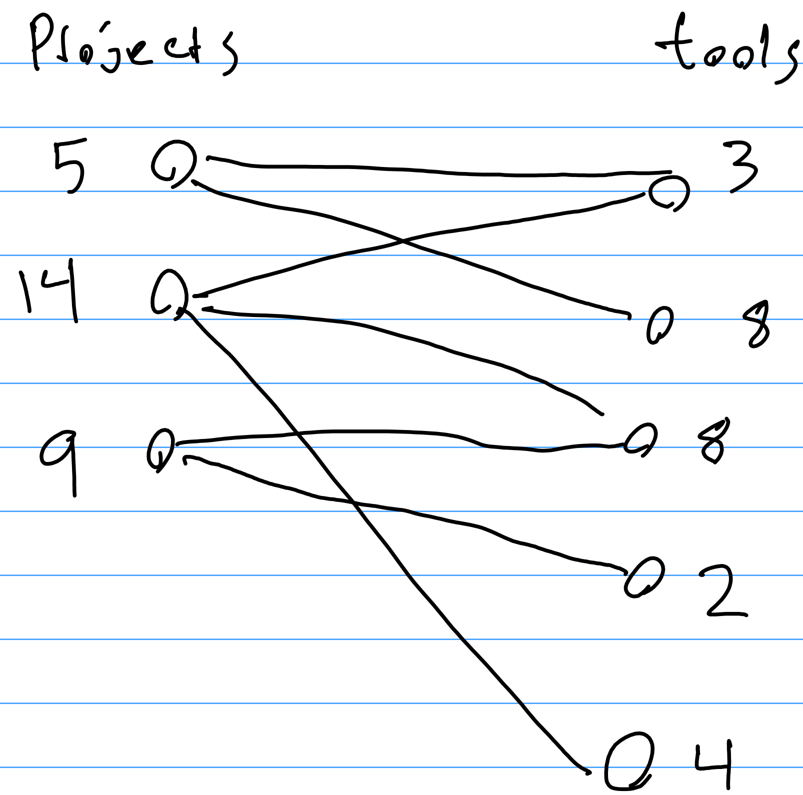
(2) any size-m (integer) flow gives size-m matching [path decomposition]

$\Rightarrow$ Max matching = max flow

Variants:

- 3 people per slot?
  [right $\to$ t edges capacity 3]
- 4 slots per person?
  [s $\to$ left capacity 4]

Project Selection

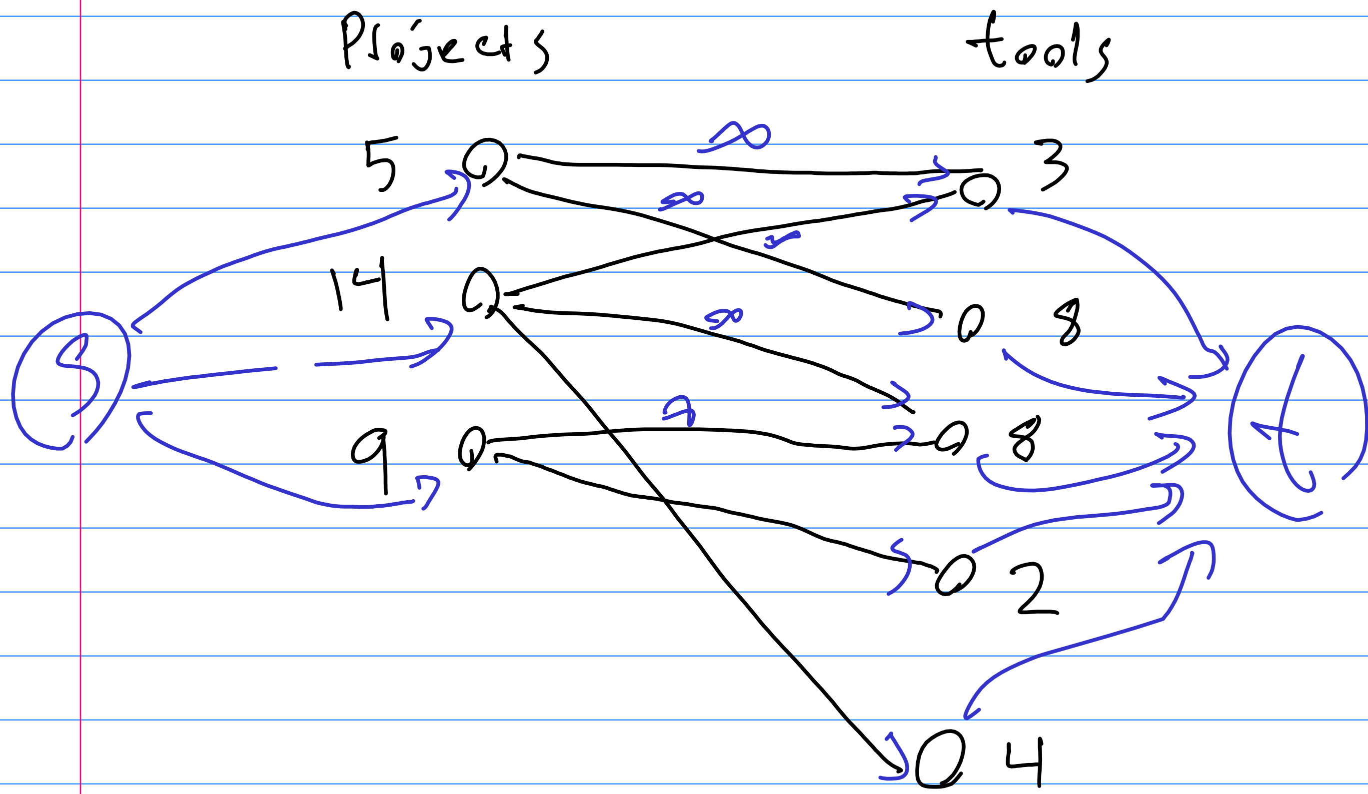


Given: <u>projects</u> to do, get paid p_i

<u>tools</u> you need, cost c_i

each project needs a subset of tools
tools can be reused.

Q: Choose subset of projects to do
& tools to buy
s.t. each project you do, you buy the tools
& Maximize
(total value of projects done)
− (total cost of tools bought)



Answer: max flow (= min cut)

Draw edges s → projects, tools → t
capacity = given value

∞ capacity on project → tool edges

⇒ min cut only cuts s → project or tool → t
edges.

any s-t cut cuts
- projects you don't complete
- tools you do buy

Valid s-t cut
⇒ no s → project → tool → t path
over uncut edges

⇒ for every (project, tool) pair (P, l)
where P needs l
either s → P cut or l → t cut
⇒ either P canceled or l bought

⇒ valid choice to build / buy.

Cost of cut =
cost(S) = (total value of canceled projects)
+ (total price of bought tools)

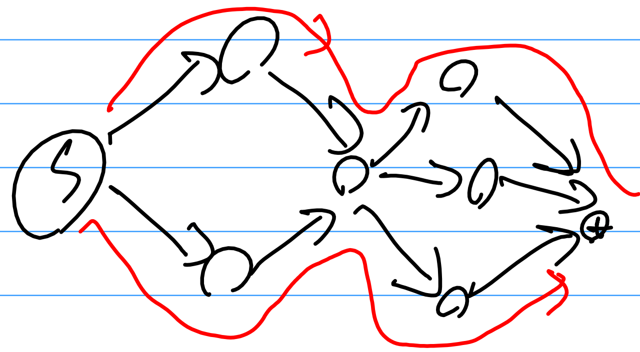
independent of choice

= (total value of all projects)
− (profit made by choice)

⇒ min cut = max profit

Edge Disjoint Paths:

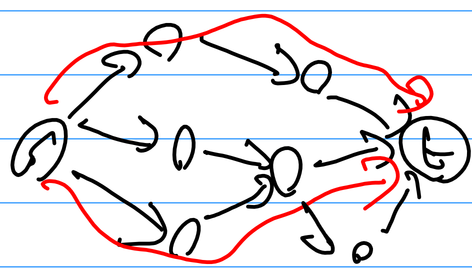
Find max # edge-disjoint s-t paths in graph



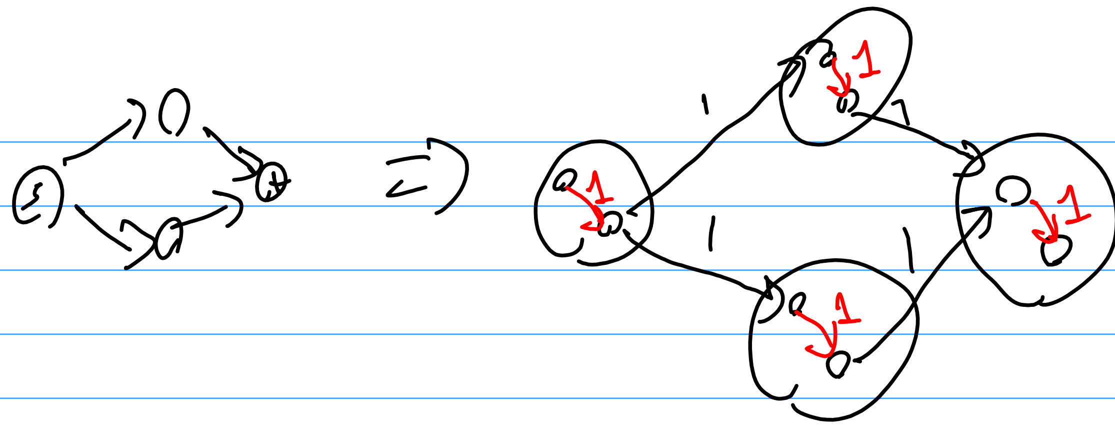
A: max flow w/ capacity 1

Vertex-disjoint paths:

Max # vertex-disjoint s-t paths
[reusing s & t is OK]



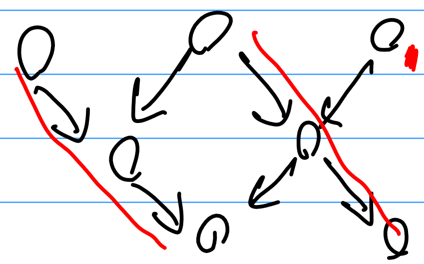
A: split vertices into "in"- and "out" node



regular edge $(u,v) \Rightarrow (u_{out}, v_{in})$
add (u_{in}, u_{out}) edges $\forall u$
all capacity 1
Ans = flow(s_{out}, t_{in})

Min disjoint Path Cover

Given a DAG find min # vertex-disjoint paths that cover all vertices



ans = 3

Soln: bipartite matching between in-nodes & out-nodes
$\Rightarrow$ each vertex gets ≤ 1 selected edge in & out
$\Rightarrow$ can start at unmatched in-nodes and follow edges to get paths covering graph

(# paths) = (# unmatched in vertices) = V − (# matched pairs)
$\Rightarrow$ min # paths $\Leftrightarrow$ max matching