



CSE 421

Algorithms

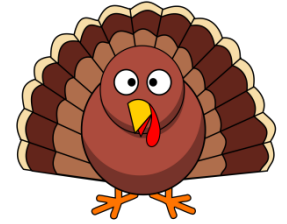
Richard Anderson

Lecture 23

Network Flow



Today



- Maxflow-MinCut Theorem
- Network Flow Applications

Network Flow Definitions

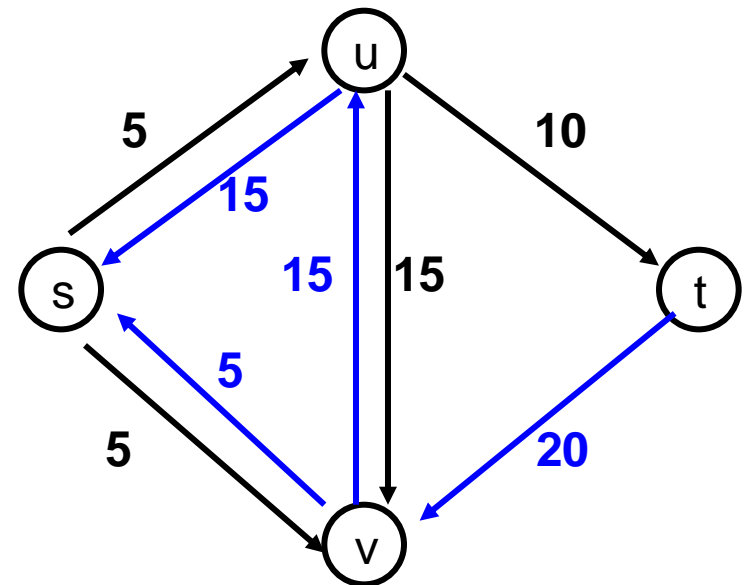
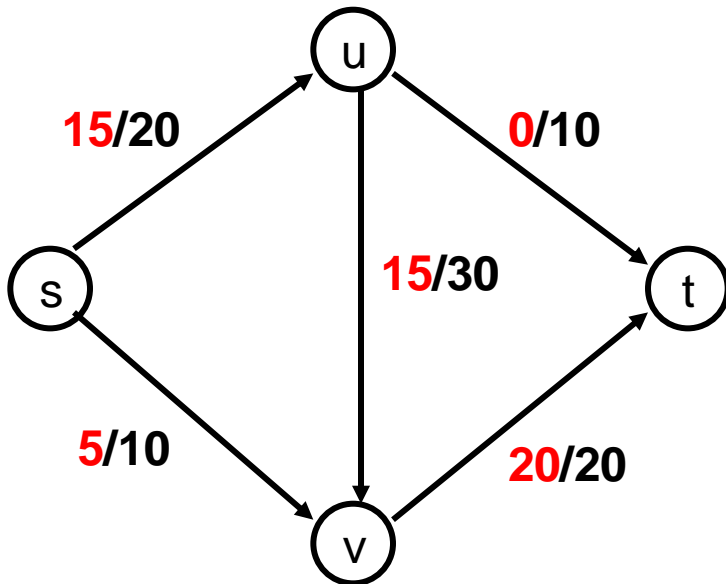
- Flowgraph: Directed graph with distinguished vertices s (source) and t (sink)
- Capacities on the edges, $c(e) \geq 0$
- Problem, assign flows $f(e)$ to the edges such that:
 - $0 \leq f(e) \leq c(e)$
 - Flow is conserved at vertices other than s and t
 - Flow conservation: flow going into a vertex equals the flow going out
 - The flow leaving the source is as large as possible

Lecture Review

- Residual Graph – how a flow can be augmented
- Ford Fulkerson Algorithm
- Correctness Proof for FF Algorithm
 - FF Terminates with a valid flow
 - When FF completes residual graph is disconnected
 - If the Residual Graph is disconnected, then the flow is maximum
 - Max Flow, Min-Cut Theorem

Residual Graph

- Flow graph showing the remaining capacity
- Flow graph G , Residual Graph G_R
 - G : edge e from u to v with capacity c and flow f
 - G_R : edge e' from u to v with capacity $c - f$
 - G_R : edge e'' from v to u with capacity f



Ford-Fulkerson Algorithm (1956)

while not done

 Construct residual graph G_R

 Find an s-t path P in G_R with capacity $b > 0$

 Add b units along in G

If the sum of the capacities of edges leaving S is at most C , then the algorithm takes at most C iterations

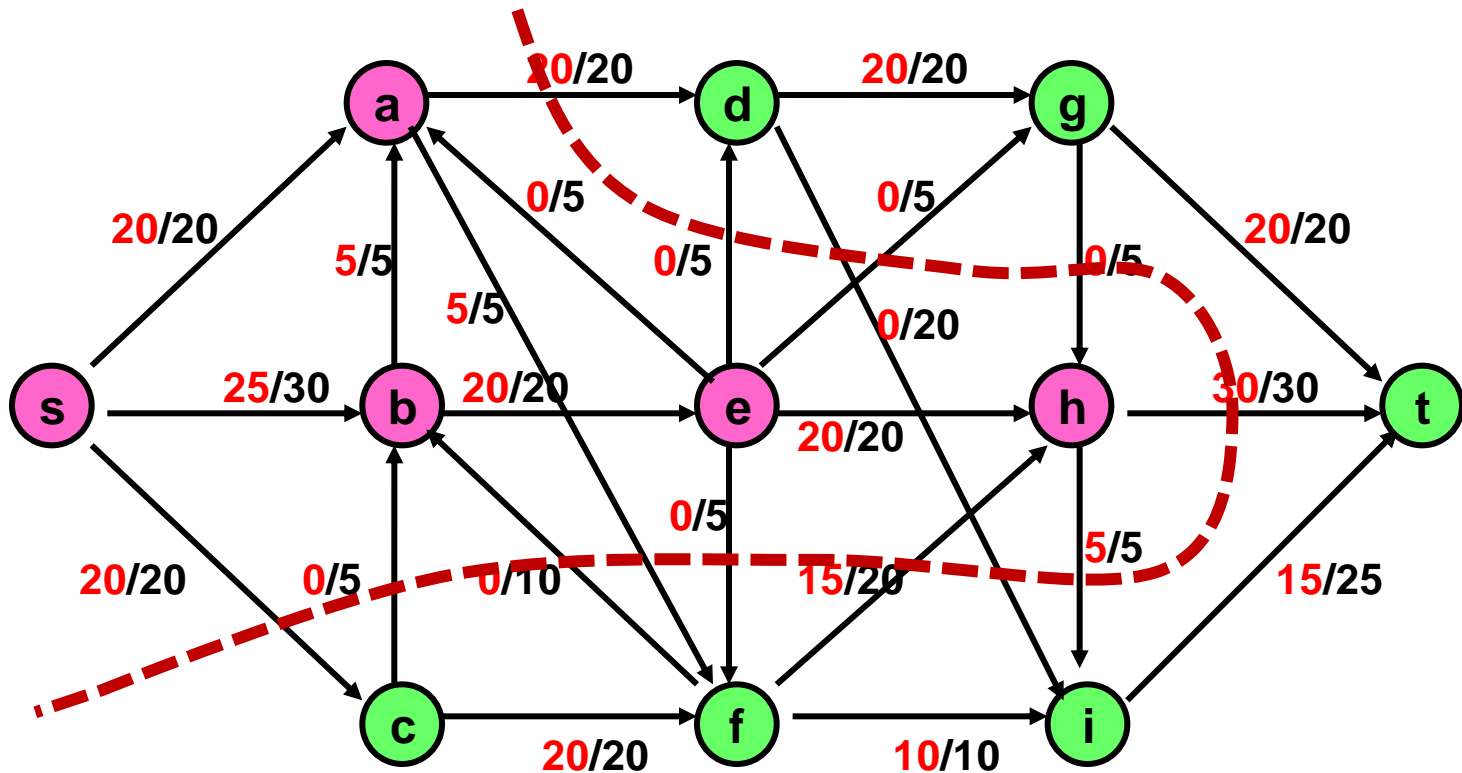
Cuts in a graph

- Cut: Partition of V into disjoint sets S , T with s in S and t in T .
- $\text{Cap}(S,T)$: sum of the capacities of edges from S to T
- $\text{Flow}(S,T)$: net flow out of S
 - Sum of flows out of S minus sum of flows into S

- $\text{Flow}(S,T) \leq \text{Cap}(S,T)$

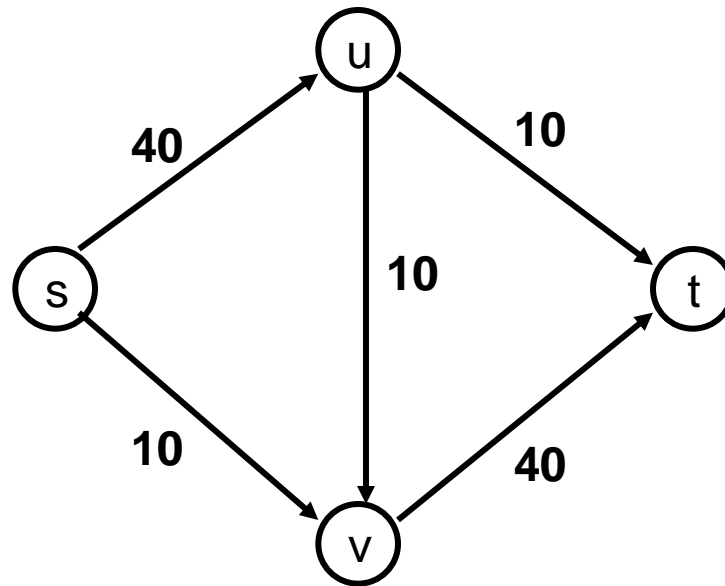
What is $\text{Cap}(S,T)$ and $\text{Flow}(S,T)$

$S = \{s, a, b, e, h\}$, $T = \{c, f, i, d, g, t\}$

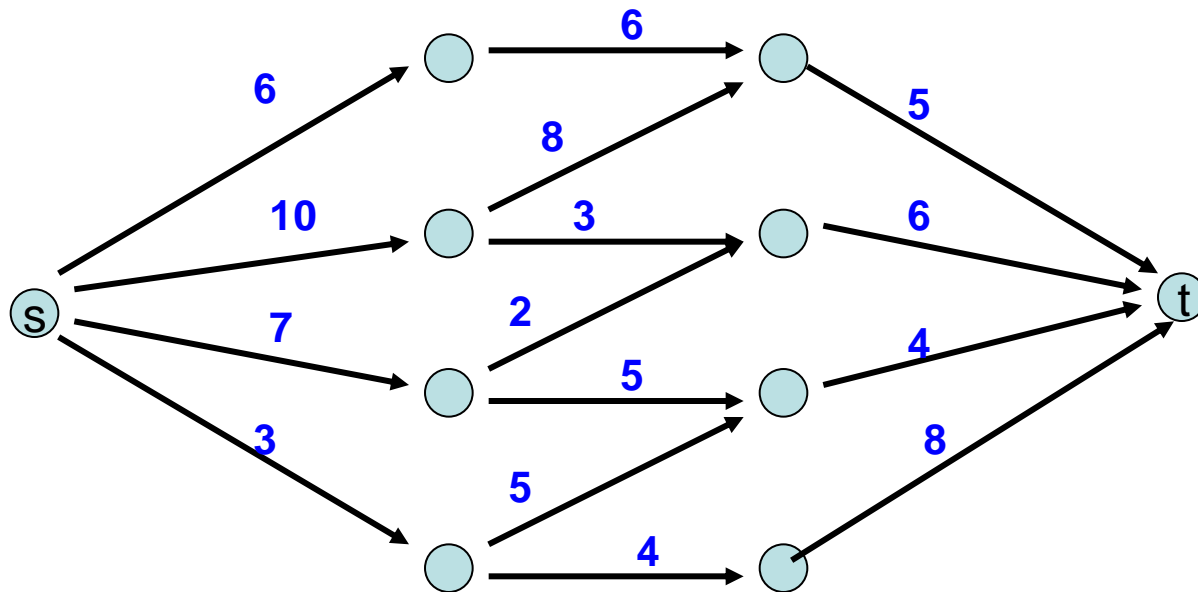


$\text{Cap}(S,T) = 95$, $\text{Flow}(S,T) = 80 - 15 = 65$

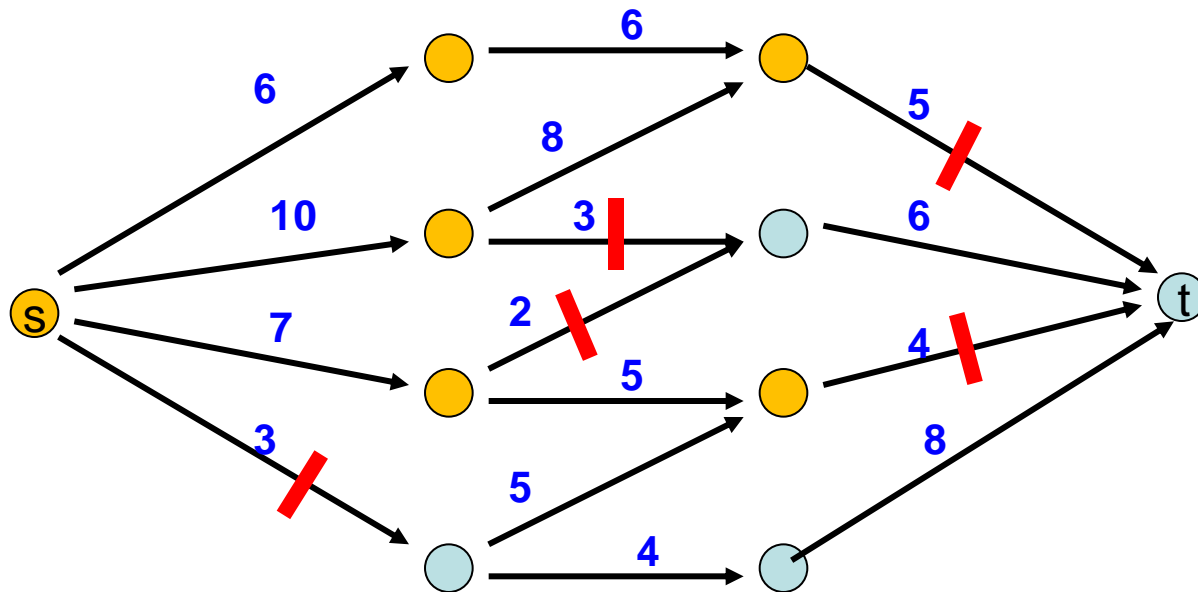
Minimum value cut



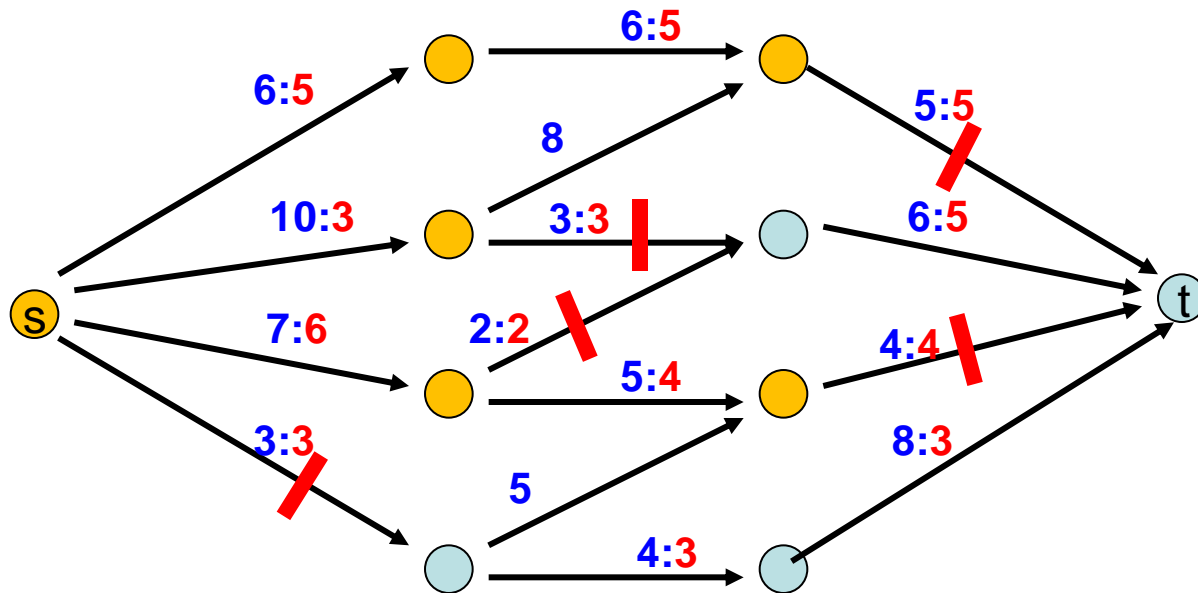
Find a minimum value cut



Find a minimum value cut

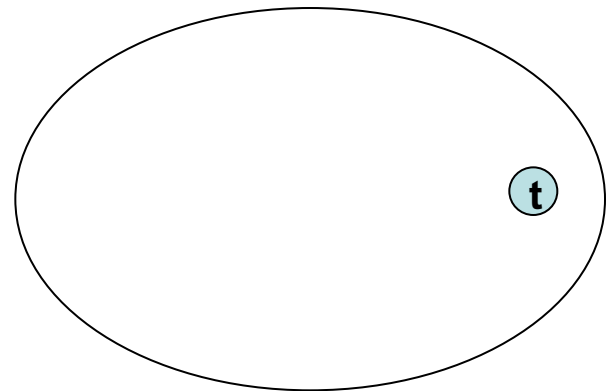
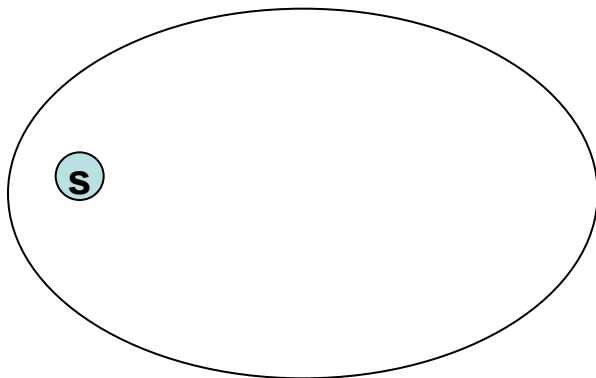


Find a minimum value cut

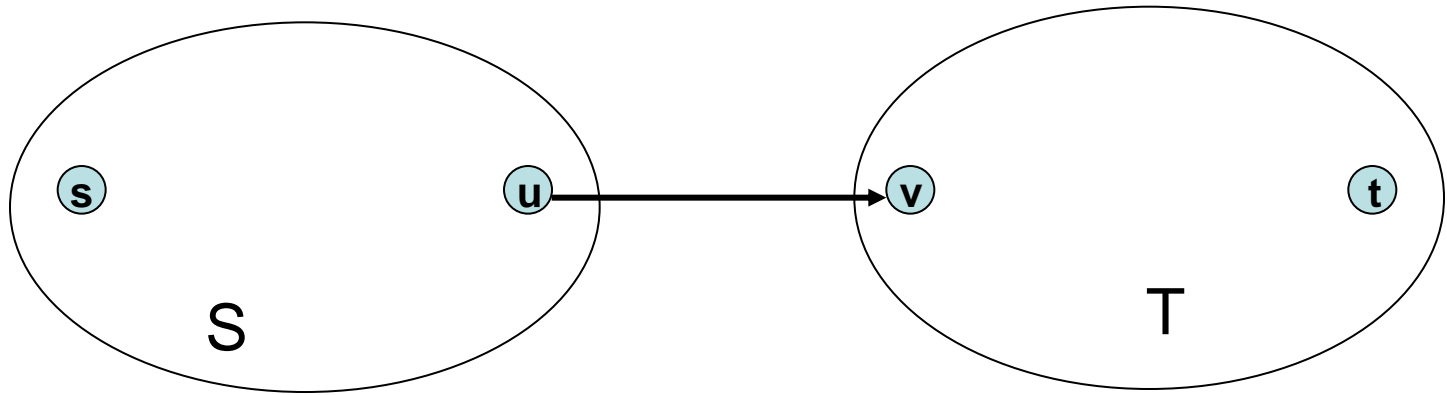


MaxFlow – MinCut Theorem

- There exists a flow which has the same value of the minimum cut
- Proof: Consider a flow where the residual graph has no s-t path with positive capacity
- Let S be the set of vertices in G_R reachable from s with paths of positive capacity



Let S be the set of vertices in G_R reachable from s with paths of positive capacity



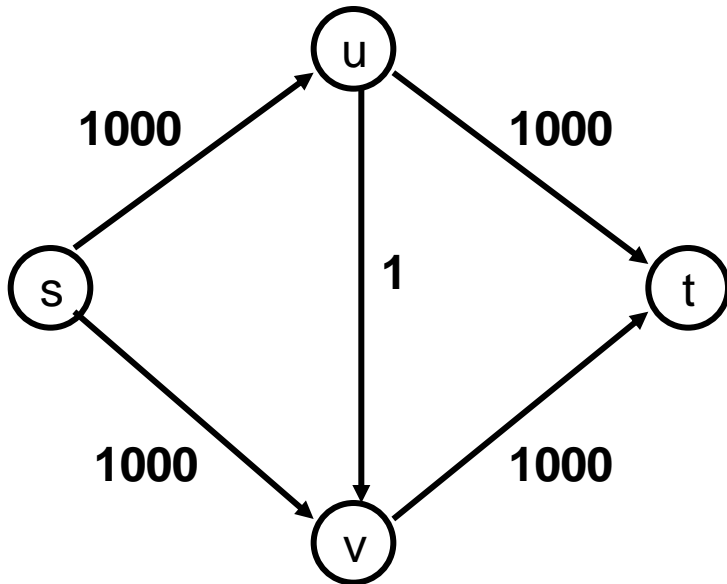
What can we say about the flows and capacity between u and v ?

Max Flow - Min Cut Theorem

- Ford-Fulkerson algorithm finds a flow where the residual graph is disconnected, hence FF finds a maximum flow.
- If we want to find a minimum cut, we begin by looking for a maximum flow.

Performance

- The worst case performance of the Ford-Fulkerson algorithm is horrible



Better methods of finding augmenting paths

- Find the maximum capacity augmenting path
 - $O(m^2 \log(C))$ time algorithm for network flow
- Find the shortest augmenting path
 - $O(m^2 n)$ time algorithm for network flow
- Find a blocking flow in the residual graph
 - $O(mn \log n)$ time algorithm for network flow

Problem Reduction

- Reduce Problem A to Problem B
 - Convert an instance of Problem A to an instance of Problem B
 - Use a solution of Problem B to get a solution to Problem A
- Practical
 - Use a program for Problem B to solve Problem A
- Theoretical
 - Show that Problem B is at least as hard as Problem A

Problem Reduction Examples

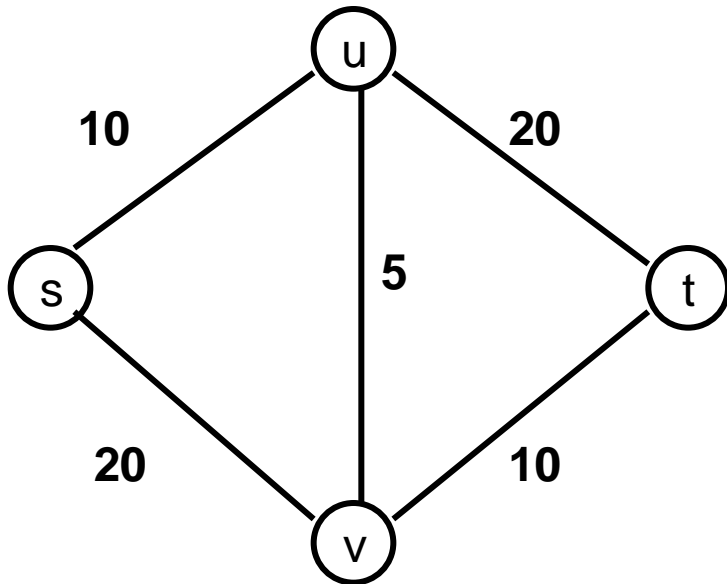
- Reduce the problem of finding the Maximum of a set of integers to finding the Minimum of a set of integers

Find the maximum of: 8, -3, 2, 12, 1, -6

Construct an equivalent minimization problem

Undirected Network Flow

- Undirected graph with edge capacities
- Flow may go either direction along the edges (subject to the capacity constraints)













Construct an equivalent flow problem

Bipartite Matching

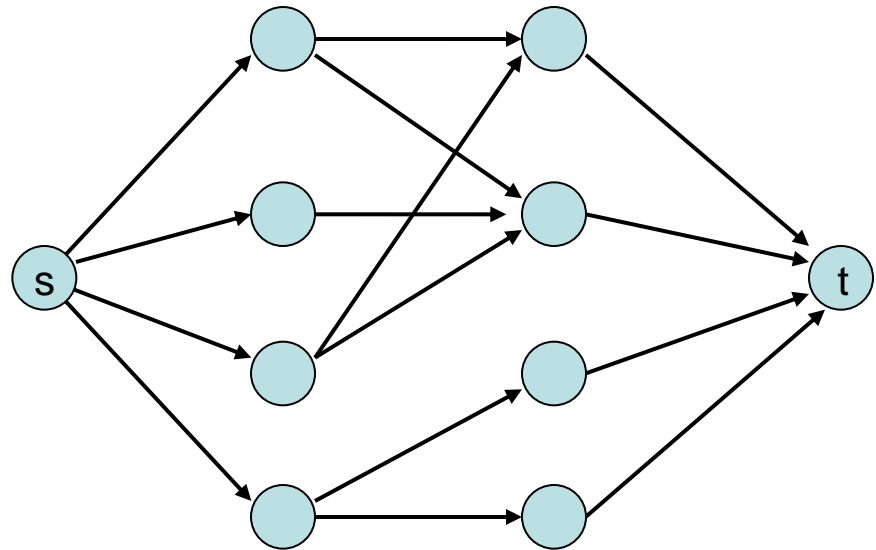
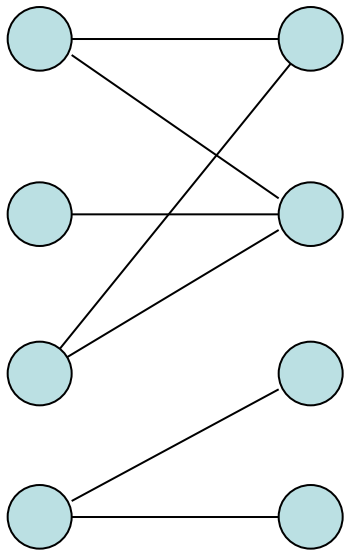
- A graph $G=(V,E)$ is bipartite if the vertices can be partitioned into disjoint sets X,Y
- A matching M is a subset of the edges that does not share any vertices
- Find a matching as large as possible

Application

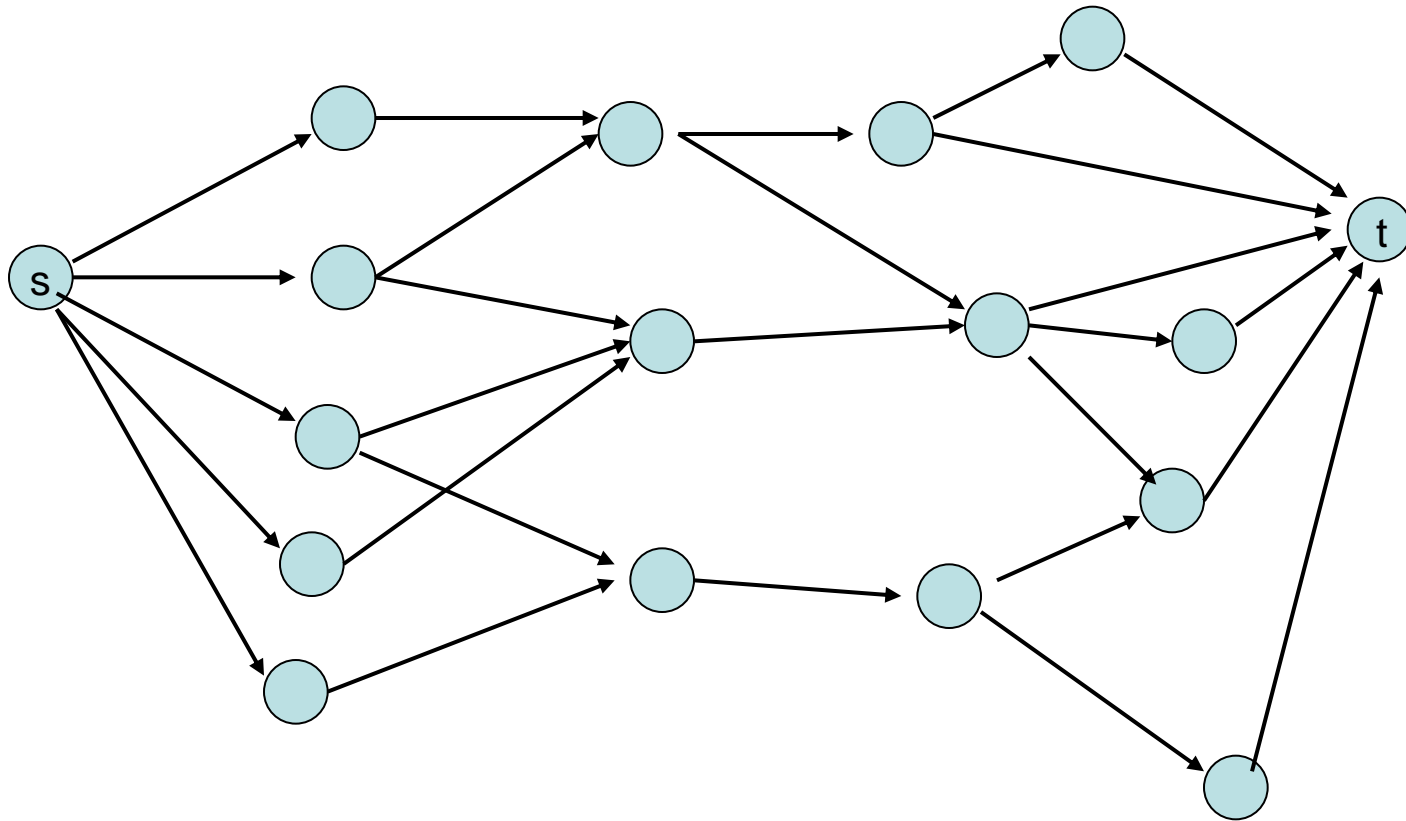
- A collection of teachers
- A collection of courses
- And a graph showing which teachers can teach which courses

RA			311
PB			331
ME			332
DG			401
AK			421

Converting Matching to Network Flow



Finding edge disjoint paths



Construct a maximum cardinality set of edge disjoint paths