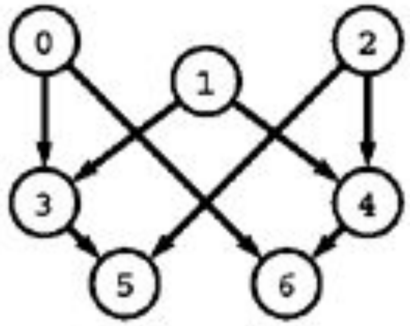
Maximum Flow (Part 3)

Reduction to Maximum Flow

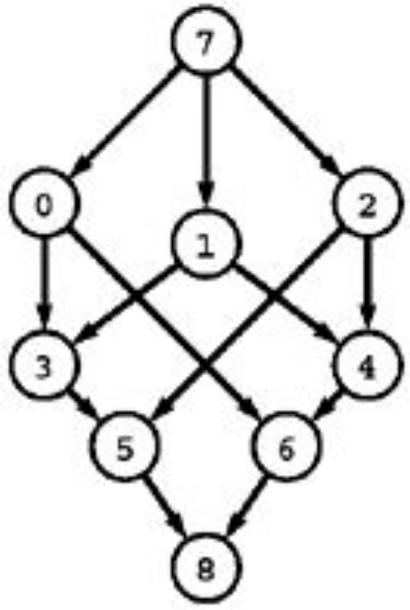
R1: Multiple sources and sinks

- Problem:
  - What if you have a problem with more than one source and more than one sink?
  - The network at the top has three sources (0, 1, and 2) and two sinks (5 and 6).
- Reduction: Create a network which
  - is a copy of the original network
  - with the addition of a new source 7 and
  - a new sink 8.
  - There is an edge from 7 to each original-network source with capacity equal to the sum of the capacities of that source's outgoing edges,
  - an edge from each original-network sink to 8 with capacity equal to the sum of the capacities of that sink's incoming edges.

	cap
0-3	2
0-6	3
1-3	3
1-4	1
2-4	1
2-5	1
3-5	2
4-6	3



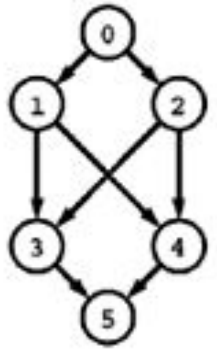
	cap
0-3	2
0-6	3
1-3	3
1-4	1
2-4	1
2-5	1
3-5	2
4-6	3
7-0	5
7-1	4
7-2	2
5-8	3
6-8	6



R2: Removing vertex capacities

- Problem:
  - Given a flow network, find a maxflow satisfying additional constraints specifying that the flow through each vertex must not exceed some fixed capacity.

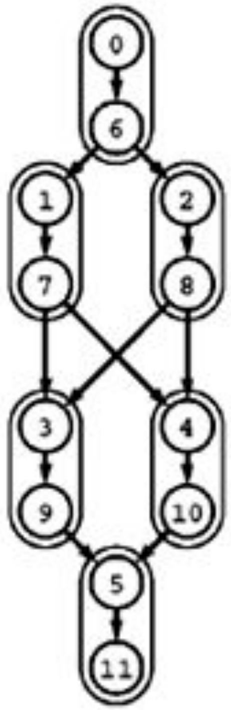
	cap
0-1	2
0-2	3
1-3	3
1-4	1
2-3	1
2-4	1
3-5	2
4-5	3



	0	1	2	3	4	5
capV	4	4	3	1	3	4

- Reduction: Create a new network
  - Associate a new vertex u* (where u* denotes u+V) with each vertex u,
  - add an edge u->u* whose capacity is the capacity of u,
  - include an edge u*->v for each edge u->v.

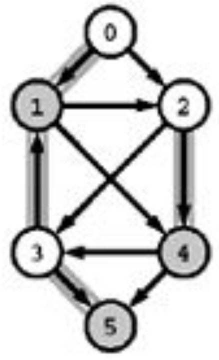
	cap
6-1	2
6-2	3
7-3	3
7-4	1
8-3	1
8-4	1
9-5	2
10-5	3
0-6	4
1-7	4
2-8	3
3-9	1
4-10	3
5-11	4



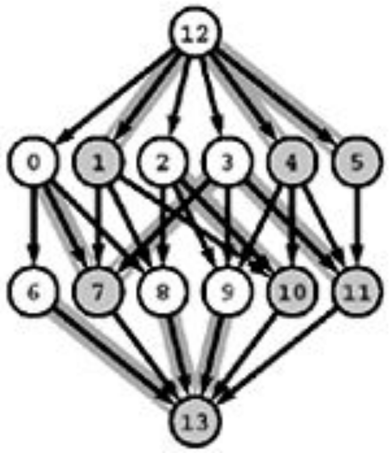
R3: Reduction to acyclic network

- Problem:
  - Find a maxflow in an acyclic network.
- Reduction: Create a new network where
  - Each vertex u in the top network corresponds to two vertices u and u* (where u* denotes u+V) in the bottom network,
  - each edge u-v in the top network corresponds to an edge u-v* in the bottom network.
  - Additionally, the bottom network has uncapacitated edges u-u*
  - a source s with an edge to each unstarred vertex,
  - a sink t with an edge from each starred vertex.

0-1	2
0-2	3
1-2	3
1-4	2
2-3	2
2-4	1
3-1	3
3-5	2
4-3	3
4-5	3



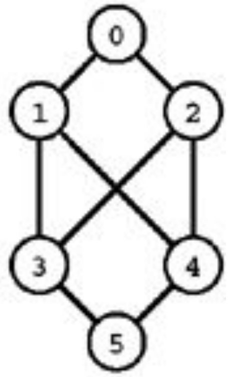
0-6	25	12-0	5	6-13	5
1-7	25	12-1	5	7-13	5
2-8	25	12-2	3	8-13	3
3-9	25	12-3	5	9-13	5
4-10	25	12-4	6	10-13	6
5-11	25	12-5	0	11-13	0
0-7	2				
0-8	3				
1-8	3				
1-10	2				
2-9	2				
2-10	1				
3-7	3				
3-11	2				
4-9	3				
4-11	3				



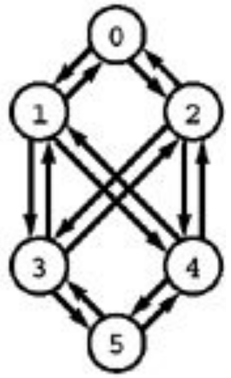
R4: Reduction from undirected networks

- Problem:
  - Given an undirected weighted graph find the maximum flow
- Reduction: Create a network s.t.
  - we can consider it to be a directed network with edges in each direction.

	cap
0-1	2
0-2	3
1-3	3
1-4	1
2-3	1
2-4	1
3-5	2
4-5	3

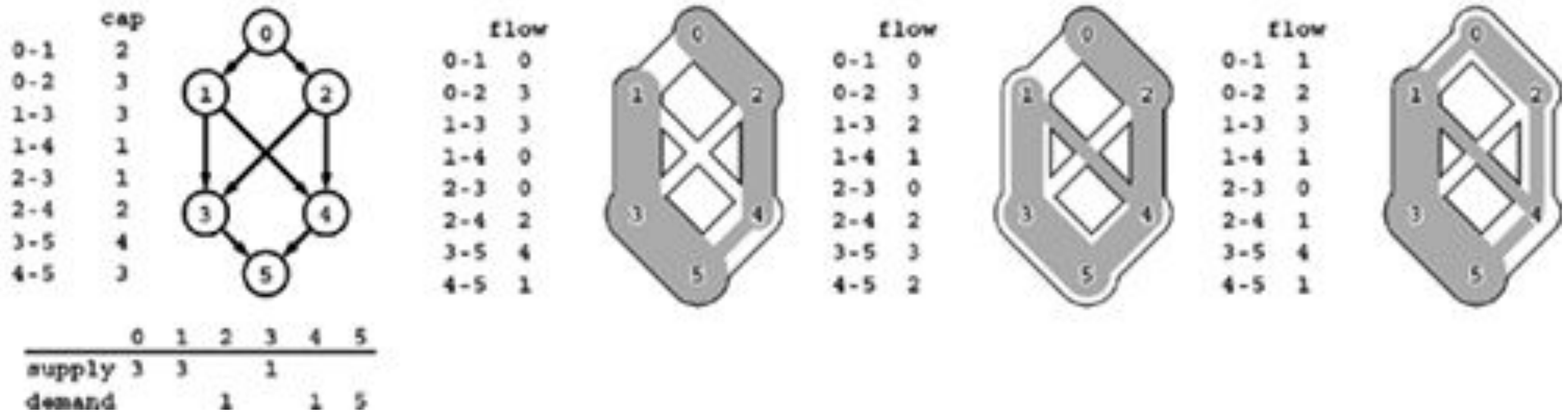


	cap
0-1	2
1-0	2
0-2	3
2-0	3
1-3	3
3-1	3
1-4	1
4-1	1
2-3	1
3-2	1
2-4	1
4-2	1
3-5	2
5-3	2
4-5	3
5-4	3



R5: Feasible flow

- Problem:
  - Suppose that a weight is assigned to each vertex in a flow network and is to be interpreted as supply (if positive) or demand (if negative), with the sum of the vertex weights equal to zero. Define a flow to be feasible if the difference between each vertex's outflow and inflow is equal to that vertex's weight (supply if positive and demand if negative). Given such a network, determine whether or not a feasible flow exists.



	cap
0-1	2
0-2	3
1-3	3
1-4	1
2-3	1
2-4	2
3-5	4
4-5	3

	0	1	2	3	4	5
supply	3	3		1		
demand			1		1	5

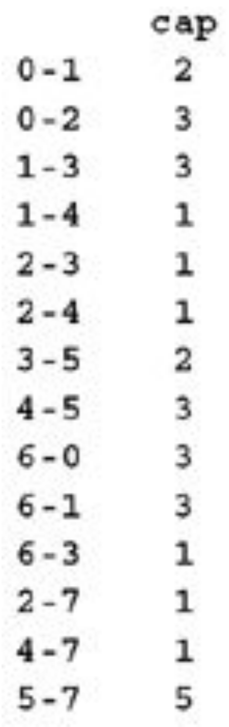
	flow
0-1	0
0-2	3
1-3	3
1-4	0
2-3	0
2-4	2
3-5	4
4-5	1

	flow
0-1	0
0-2	3
1-3	2
1-4	1
2-3	0
2-4	2
3-5	3
4-5	2

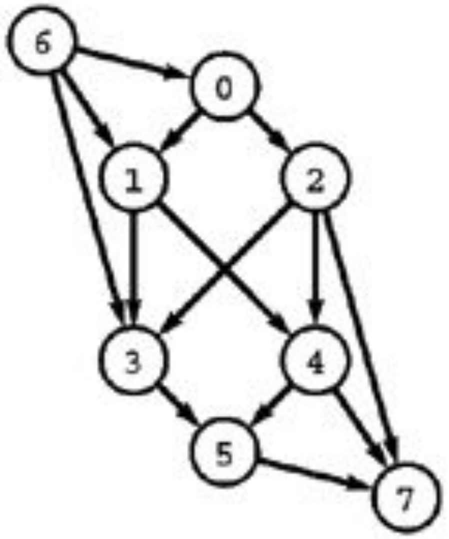
	flow
0-1	1
0-2	2
1-3	3
1-4	1
2-3	0
2-4	1
3-5	4
4-5	1

R5: Reduction from feasible solution

- Reduction: Create a network
  - by adding edges from a new source vertex to the supply vertices (each with capacity equal to the amount of the supply) and
  - edges to a new sink vertex from the demand vertices (each with capacity equal to the amount of the demand).
  - The network has a feasible flow if and only if this network has a flow (a maxflow) that fills all the edges from the sink and all the edges to the source.

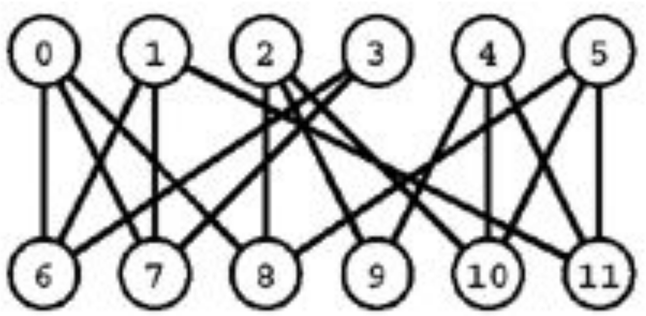


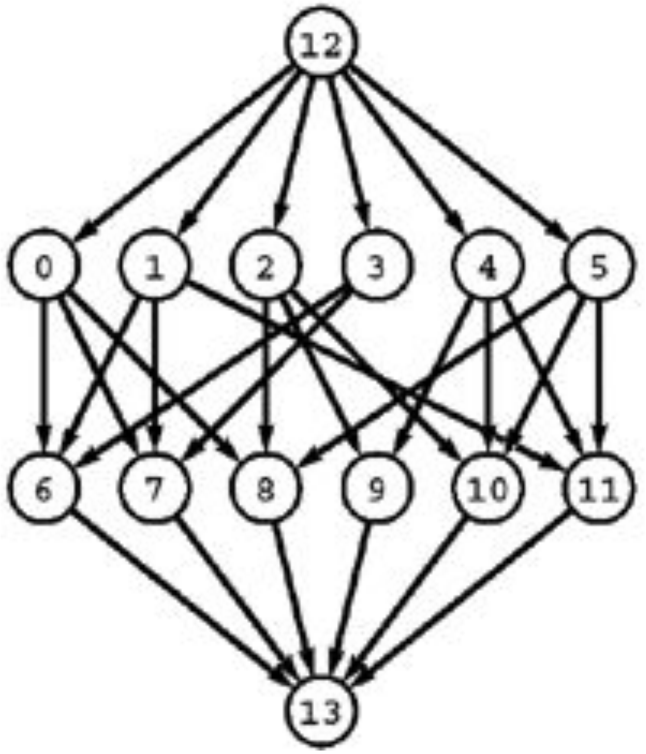
	cap
0-1	2
0-2	3
1-3	3
1-4	1
2-3	1
2-4	1
3-5	2
4-5	3
6-0	3
6-1	3
6-3	1
2-7	1
4-7	1
5-7	5



R6: Bipartite matching

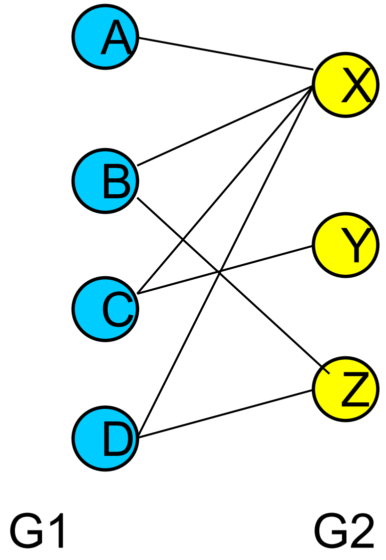
- Problem:
  - Given a bipartite graph, find a set of edges of maximum cardinality such that each vertex is connected to at most one other vertex.
- Reduction: Construct an st-network
  - by directing all the edges from the top row to the bottom row,
  - adding a new source with edges to each vertex on the top row,
  - adding a new sink with edges to each vertex on the bottom row, and
  - assigning capacity 1 to all edges.
  - In any flow, at most one outgoing edge from each vertex on the top row can be filled and at most one incoming edge to each vertex on the bottom row can be filled, so a solution to the maxflow problem on this network gives a maximum matching for the bipartite graph.

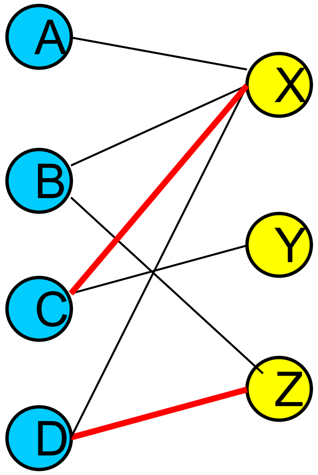




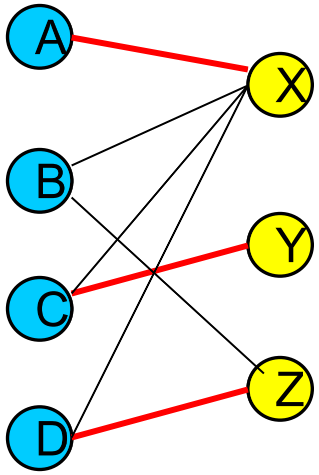
Model for Matching Problem

- Group1 on leftmost set, Group2 on rightmost set, edges if they are compatible





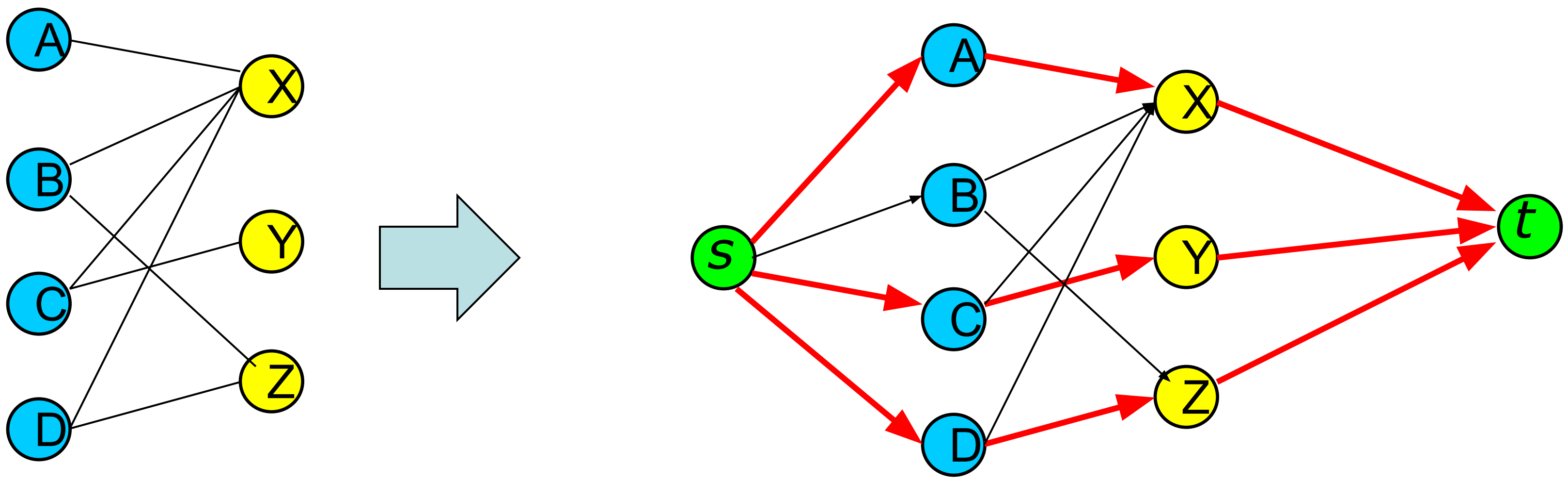
A matching



Optimal matching

Solution Using Max Flow

- Add a supersouce, supersink, make each undirected edge directed with a flow of 1



Since the input is 1, flow conservation prevents multiple matchings

Edge Connectivity

- What is the minimum number of edges that need to be removed to separate a given graph into two pieces?
- Find a set of edges of minimal cardinality that does this separation.

Vertex Connectivity

- What is the minimum number of vertices that need to be removed to separate a given graph into two pieces?
- Try yourself