Network Flow

November 23, 2016

Types of Networks

- Internet
- Telephone
- Cell
- Highways
- Rail
- Electrical Power
- Water
- Sewer
- Gas
- ...

Maximum Flow Problem

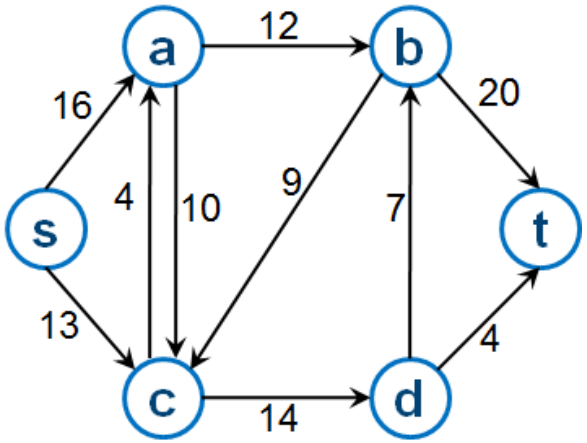
- How can we maximize the flow in a network from a source or set of sources to a destination or set of destinations?
- The first efficient algorithm for finding the maximum flow was conceived by two Computer Scientists, named Ford and Fulkerson.

Network Flow

- A Network is a directed graph G
- Edges represent pipes that carry flow
- Each edge $< u, v >$ has a maximum capacity $c_{<u,v>}$
- There is a source node s in which flow arrives
- There is a sink node t out which flow leaves

Network Flow

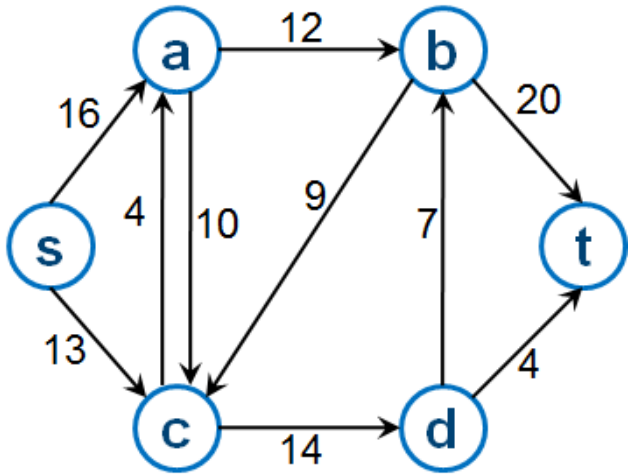
- The network flow problem is as follows:
  - Given a connected directed graph G
    - with non-negative integer weights,
    - (where each edge stands for the capacity of that edge),
- Two different vertices, s and t, called the source and the sink,
  - such that the source only has out-edges and the sink only has in-edges,
- Find the maximum amount of some commodity that can flow through the network from source to sink.



Each edge stands for the capacity of that edge.

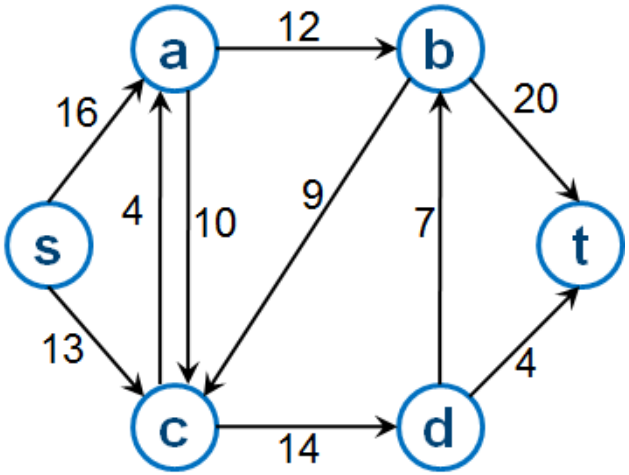
Network Flow

- One way to imagine the situation is imagining each edge as a pipe that allows a certain flow of a liquid per second.
  - The source is where the liquid is pouring from, and the sink is where it ends up.
  - Each edge weight specifies the maximal amount of liquid that can flow through that pipe per second.
  - Given that information, what is the most liquid that can flow from source to sink per second, in the steady state?

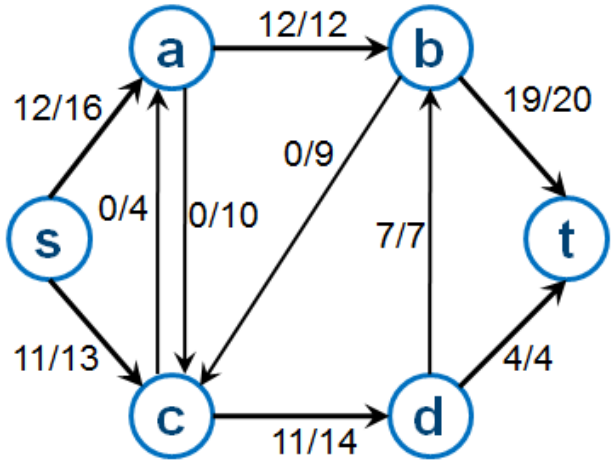


Each edge stands for the capacity of that edge.

Network Flow



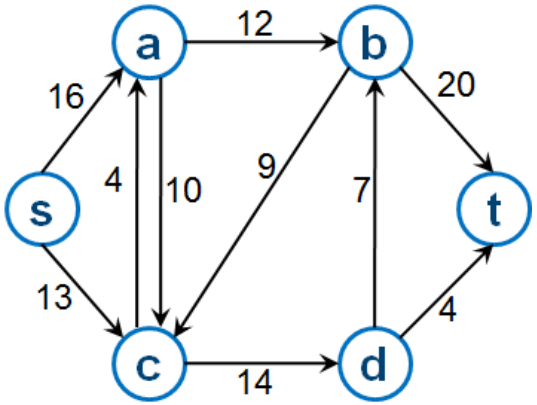
This graph contains the capacities of each edge in the graph.



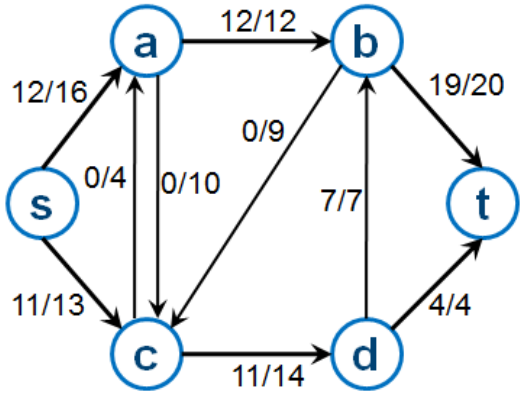
An example of a flow in the graph.

- The flow of the network is defined as the flow from the source, or into the sink.
- For the situation above, the network flow is 23.

Network Flow Rules



Capacities

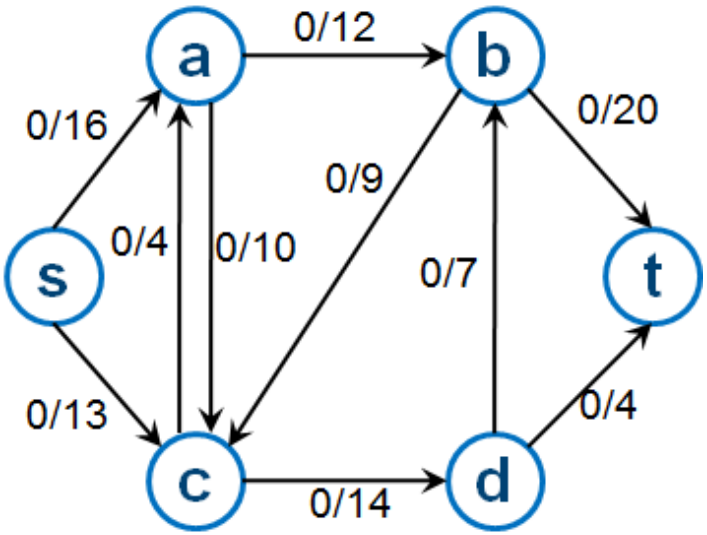


Flow

- The Conservation Rule:
  - In order for the assignment of flows to be valid, we must have the sum of flow coming into a vertex equal to the flow coming out of a vertex, for each vertex in the graph except the source and the sink.
- The Capacity Rule:
  - Also, each flow must be less than or equal to the capacity of the edge.
- The flow of the network is defined as the flow from the source, or into the sink.
  - For the situation above, the network flow is 23.

Network Flow

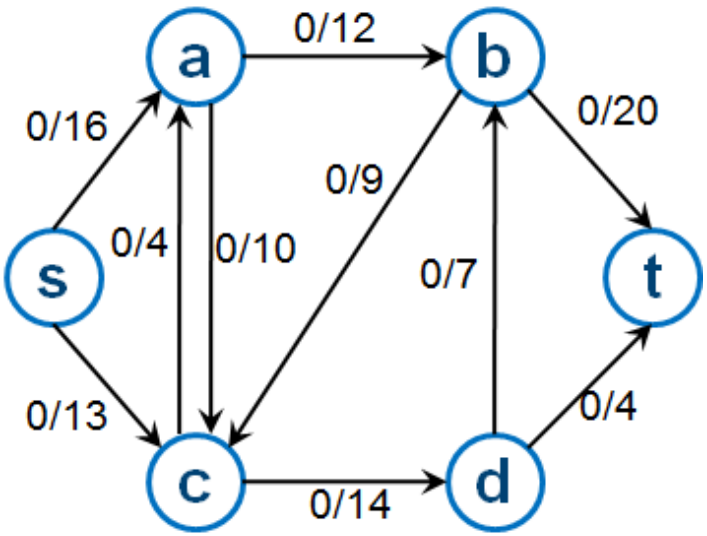
- In order to determine the maximum flow of a network, we will use the following terms:
  - Residual capacity – is simply an edge's unused capacity.
    - Initially none of the capacities will have been used, so all of the residual capacities will be just the original capacity.



Using the notation:
used/*capacity*.
Residual Capacity:
capacity − *used*.

Network Flow

- Residual capacity of a path – the minimum of the residual capacities of the edges on that path, which will end up being the max excess flow we can push down that path.
- Augmenting path – defined as one where you have a path from the source to the sink where every edge has a non-zero residual capacity.



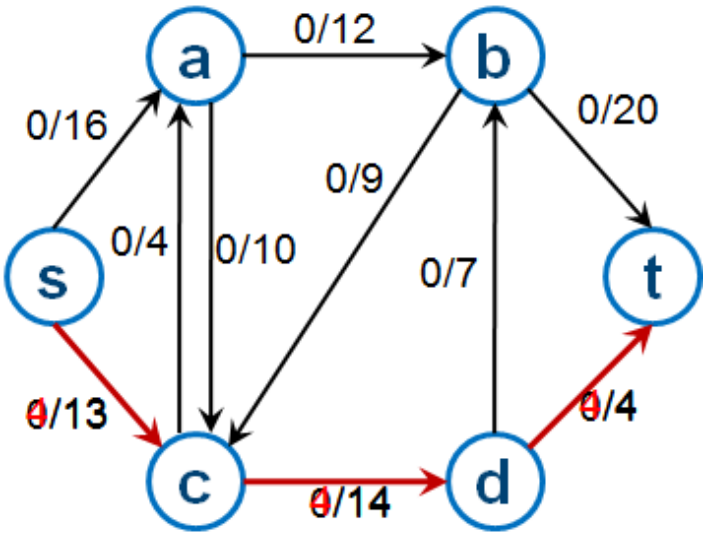
Using the notation:
used/*unused*.
Residual Capacity:
unused − *used*.

Ford-Fulkerson Algorithm

While there exists an augmenting path

Add the appropriate flow to that augmenting path

- So we're going to arbitrarily choose an the augmenting path s, c, d, t in the graph below:
  - And add the flow to that path.



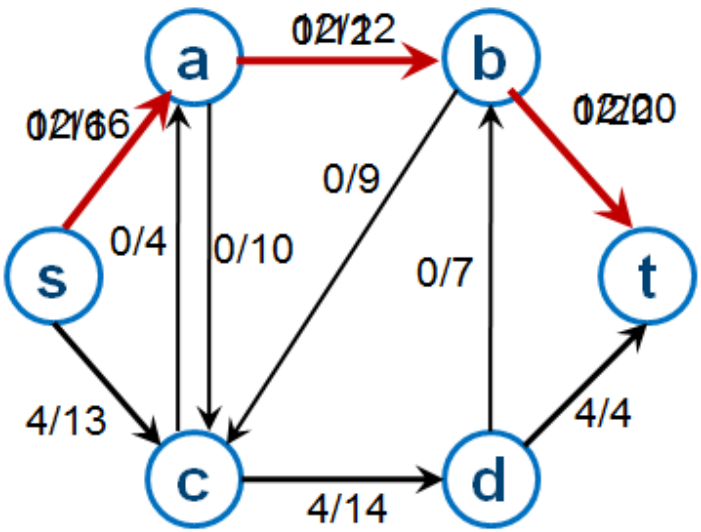
- Residual capacity of a path – the minimum of the residual capacities of the edges on that path.
- 4 in this case, which is the limiting factor for this path's flow.

Ford-Fulkerson Algorithm

While there exists an augmenting path

Add the appropriate flow to that augmenting path

- Choose another augmenting path (one where you have a path from the source to the sink where every edge has a non-zero residual capacity.)
  - s, a, b, t



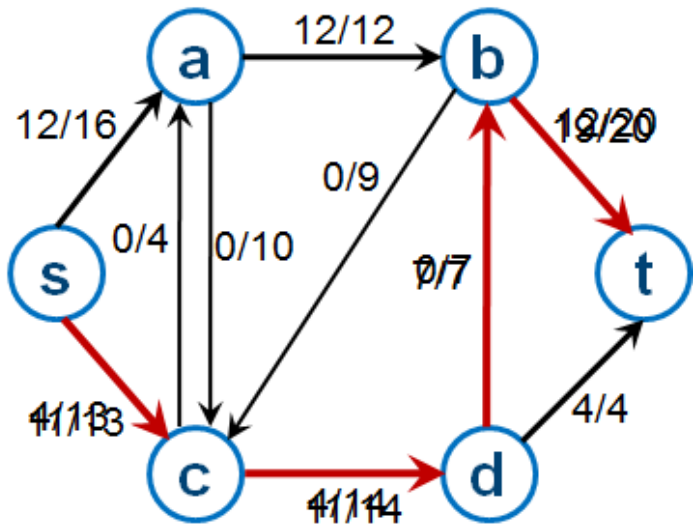
- Residual capacity of a path – the minimum of the residual capacities of the edges on that path.
- 12 in this case, which is the limiting factor for this path's flow.

Ford-Fulkerson Algorithm

While there exists an augmenting path

Add the appropriate flow to that augmenting path

- Choose another augmenting path (one where you have a path from the source to the sink where every edge has a non-zero residual capacity.)
  - s, c, d, b, t



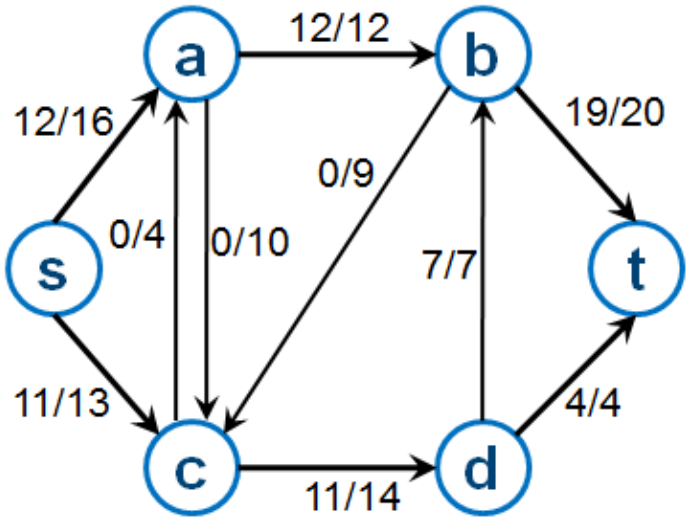
- Residual capacity of a path – the minimum of the residual capacities of the edges on that path.
- 7 in this case, which is the limiting factor for this path's flow.

Ford-Fulkerson Algorithm

While there exists an augmenting path

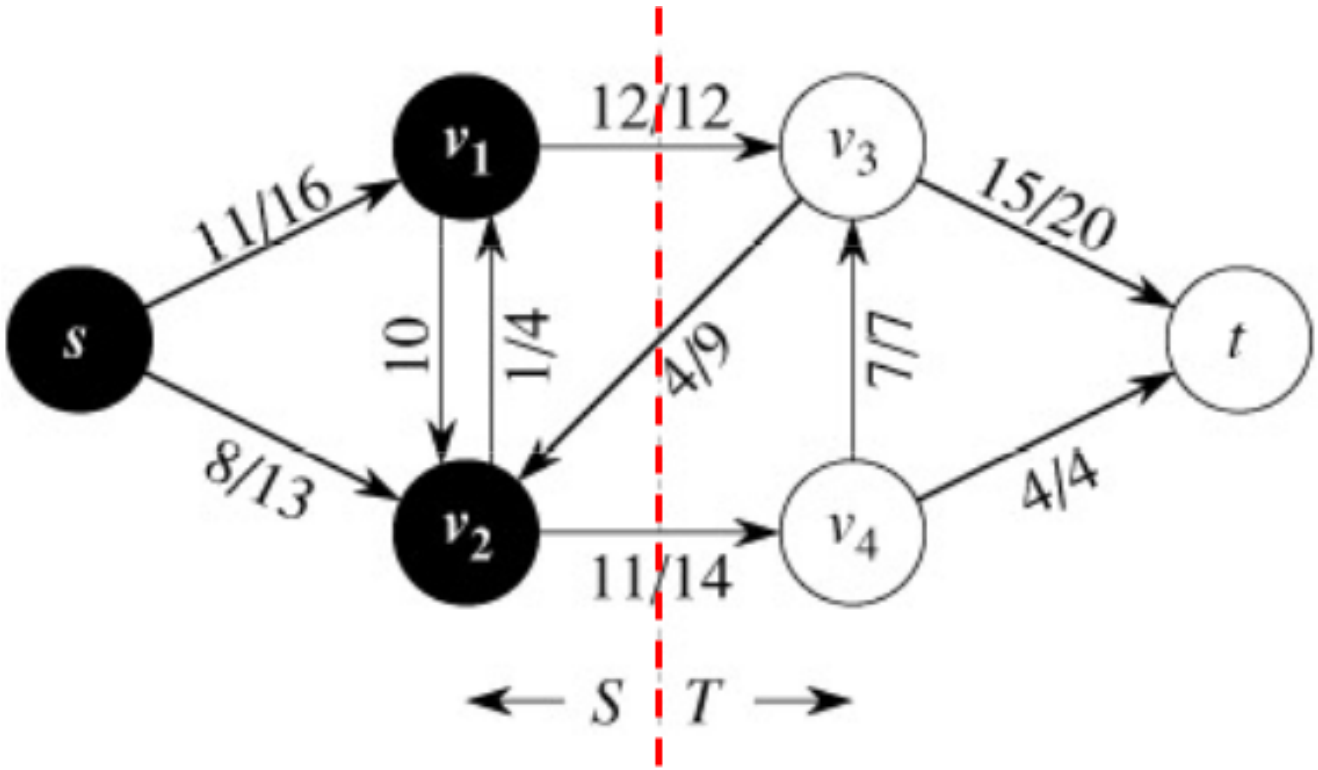
Add the appropriate flow to that augmenting path

- Are there any more augmenting paths?
  - No! We're done
  - The maximum flow = 19 + 4 = 23



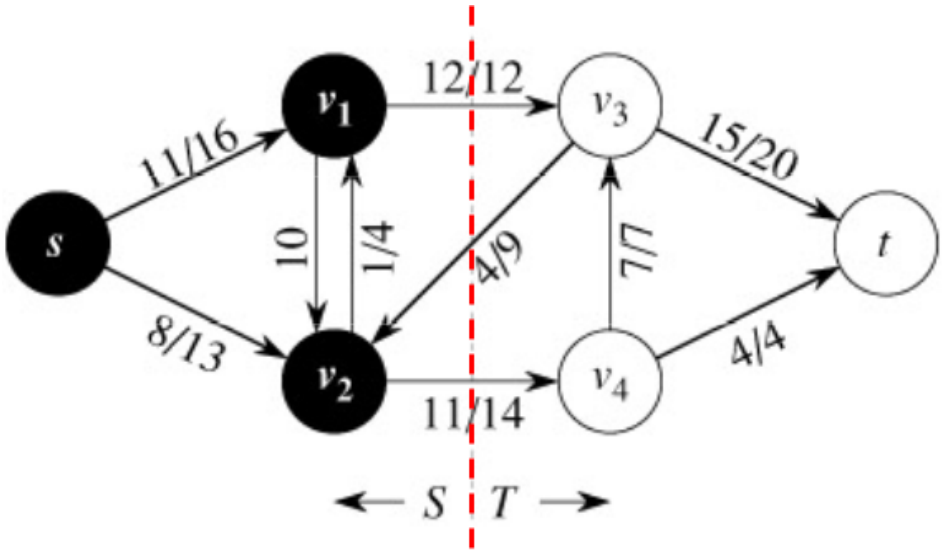
Cuts of Flow Networks

- A *cut*(S, T) of a flow network is a partition of V into S and $T = V - S$ such that $s \in S$ and $t \in T$.



The Net Flow through a Cut (S,T)

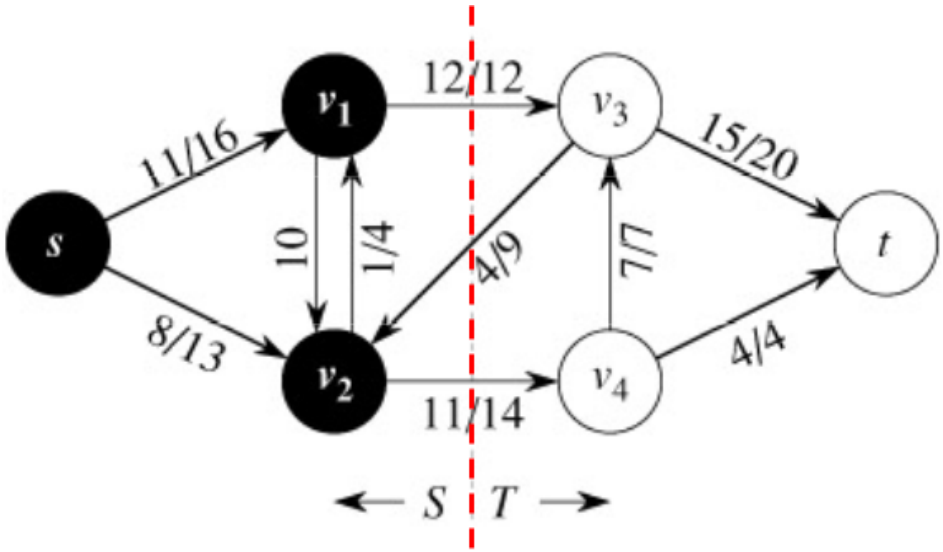
- $f(S,T) = \sum_{u \in S, v \in T} f(u,v)$



- $f(S,T) = 12 - 4 + 11 = 19$

The Capacity of a Cut (S,T)

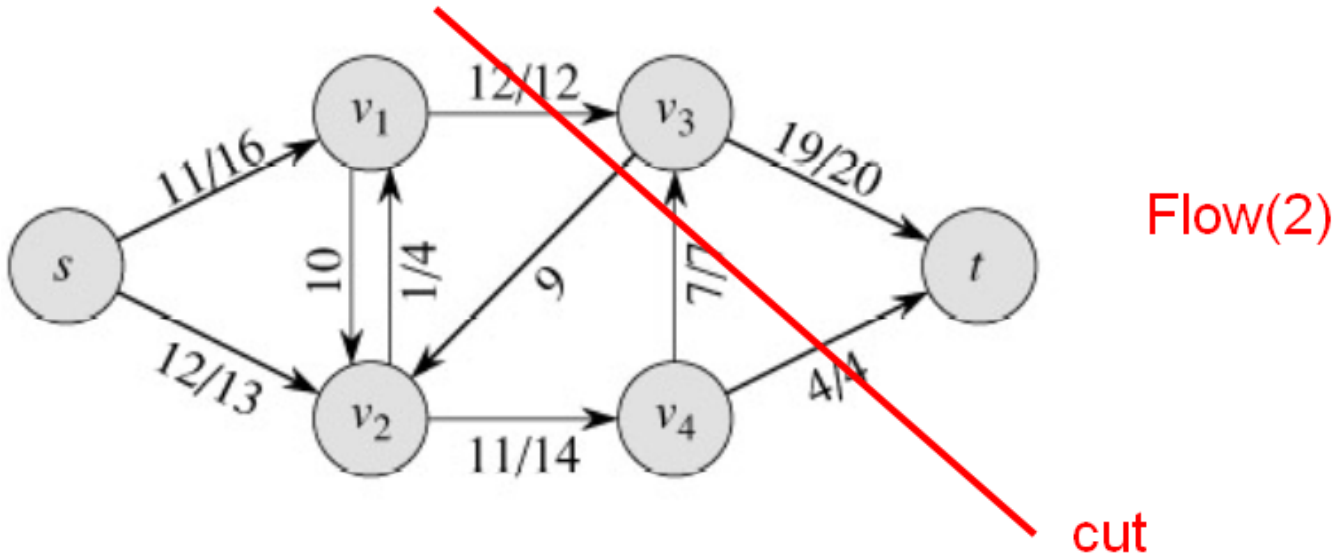
- $c(S,T) = \sum_{u \in S, v \in T} c(u,v)$



- $c(S,T) = 12 + 0 + 14 = 26$

Augmenting Paths – example

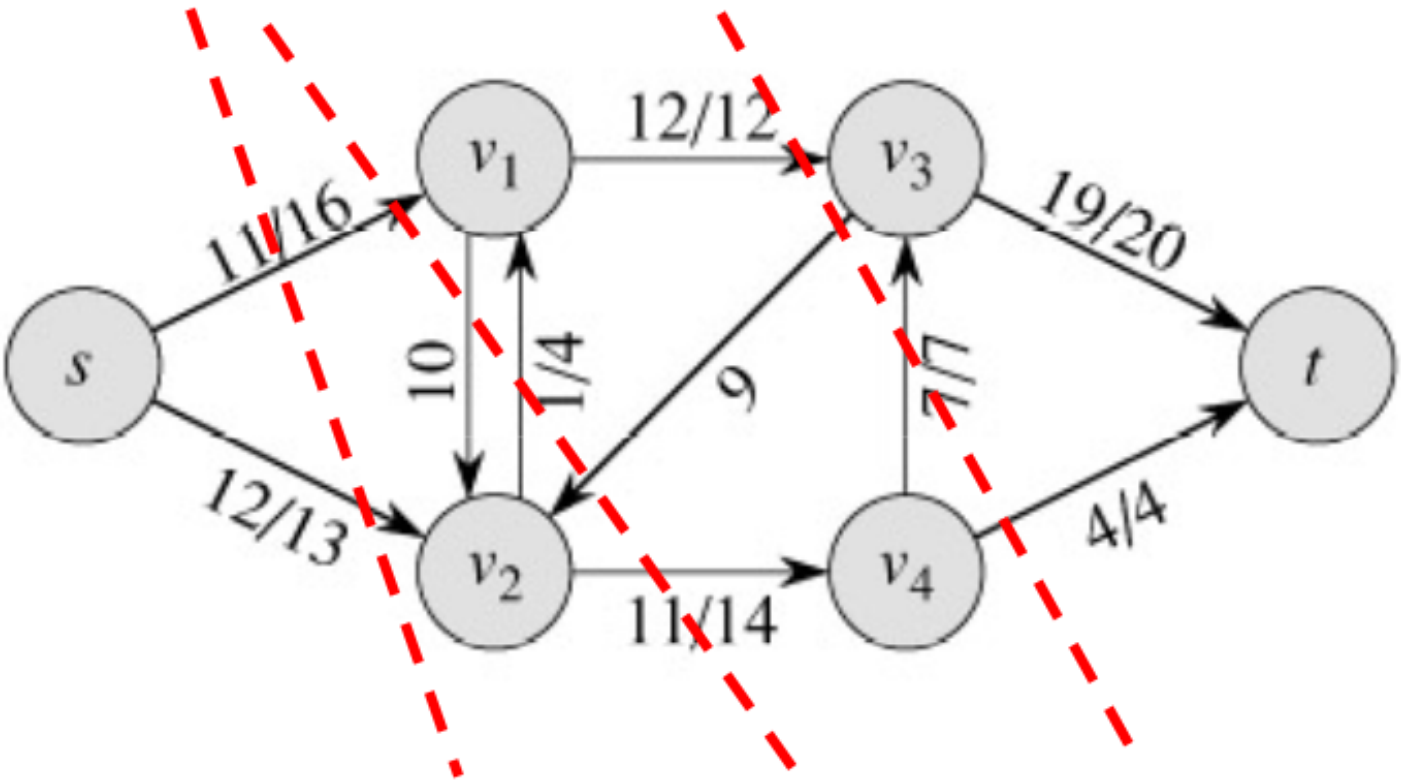
- The maximum possible flow through the cut = 12 + 7 + 4 = 23



- The network has a capacity of at most 23.
- This is called a minimum cut.

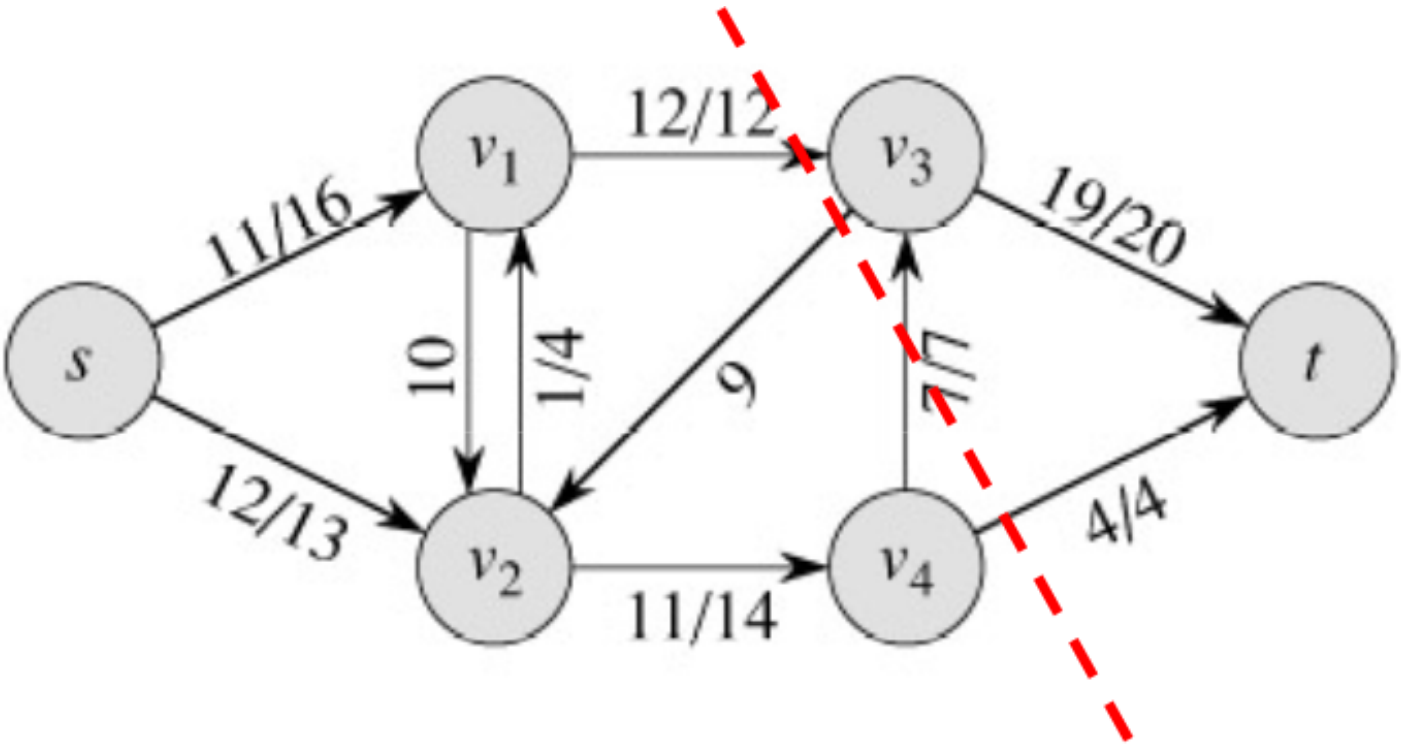
Net Flow of a Network

- The net flow across any cut is the same and equal to the flow of the network $|f|$.



Bounding the Network Flow

- The value of any flow f in a flow network G is bounded from above by the capacity of any cut of G.

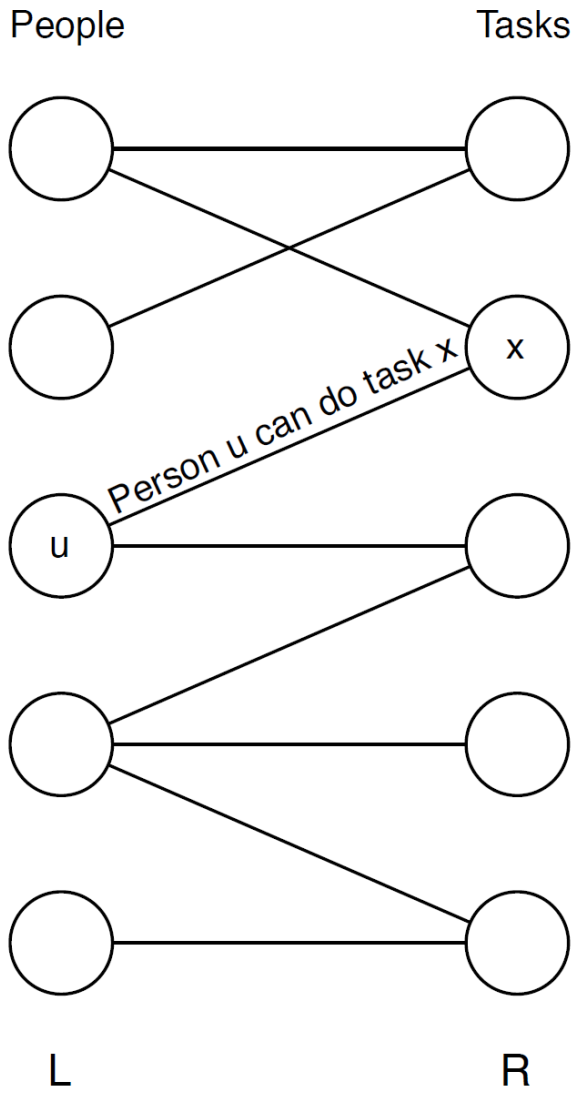


Max-Flow Min-Cut Theorem

- If f is a flow in a flow network $G = (V, E)$, with source s and sink t, then the following conditions are equivalent:
  1. f is a maximum flow in G.
  2. The residual network G_f contains no augmented paths.
  3. $|f| = f(S, T)$ for some cut (S, T) (a min-cut).

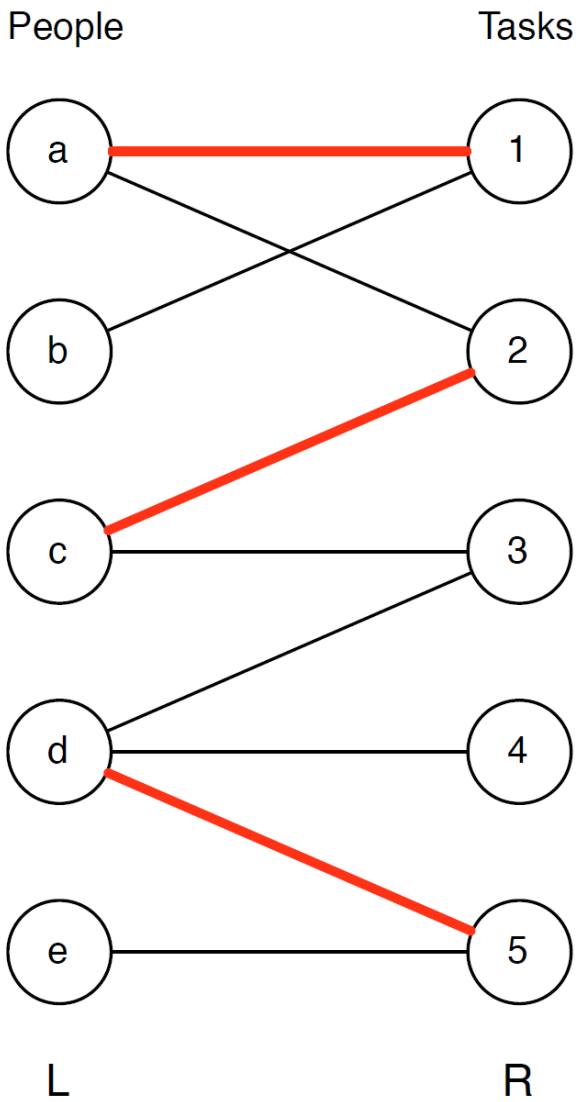
Bipartite Matching

- Assume that we have a set of people L and set of jobs R.
- Each person can do only some of the jobs.
- We can model this as a bipartite graph.



Bipartite Matching

- A matching provides an assignment of people to tasks.
- We want to have as many tasks done as possible.
- So, we want a maximum matching: one that contains as many edges as possible.
- The matching on the right is not maximum.



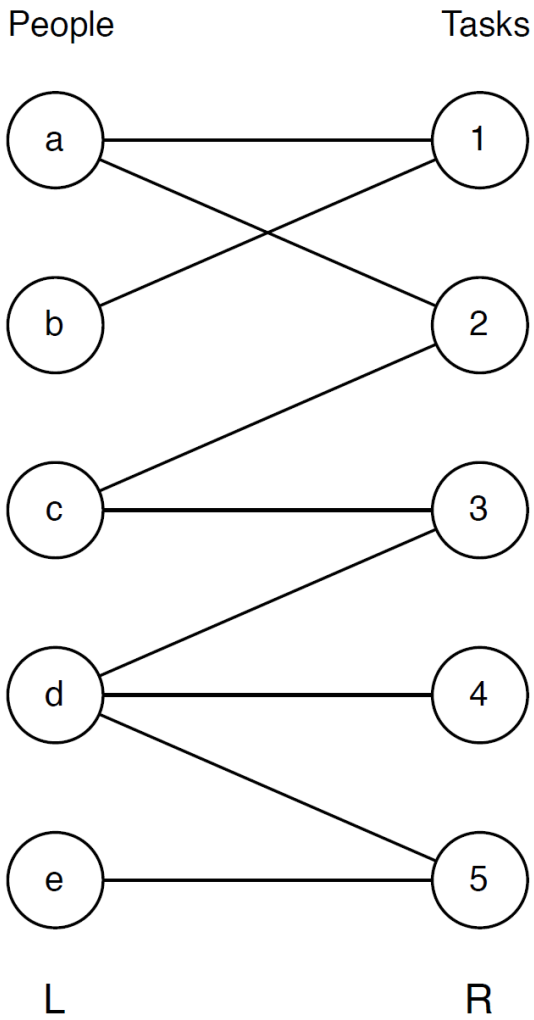
Maximum Bipartite Matching

- Maximum Bipartite Matching: Given a bipartite graph $G = (L \cup R, E)$, find an $S \subseteq L \times R$ that is a matching and is as large as possible.
- S is a perfect matching if every vertex is matched.

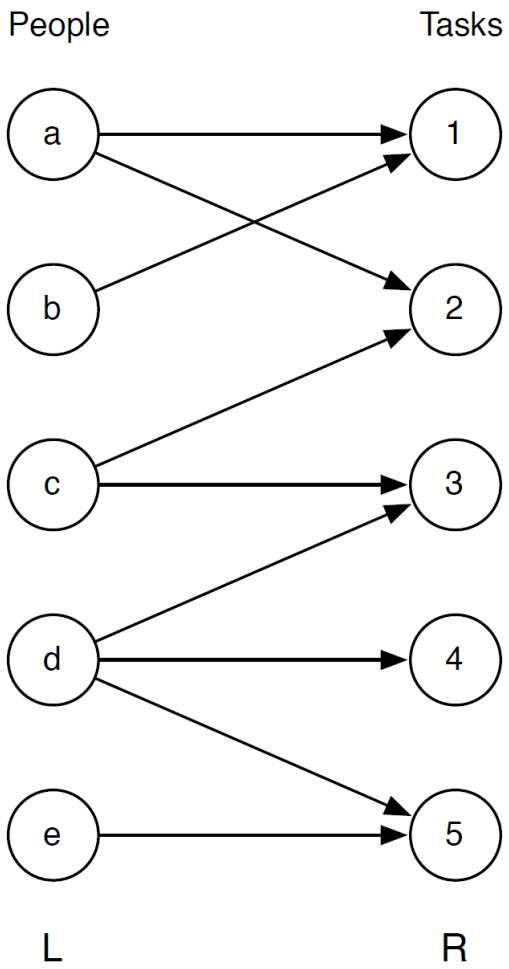
Solution by Transformation

- Given an instance of bipartite matching,
- Create an instance of network flow.
- The solution to the network flow problem can be used to find the solution to the bipartite matching.

Transforming Bipartite Matching to Network Flow

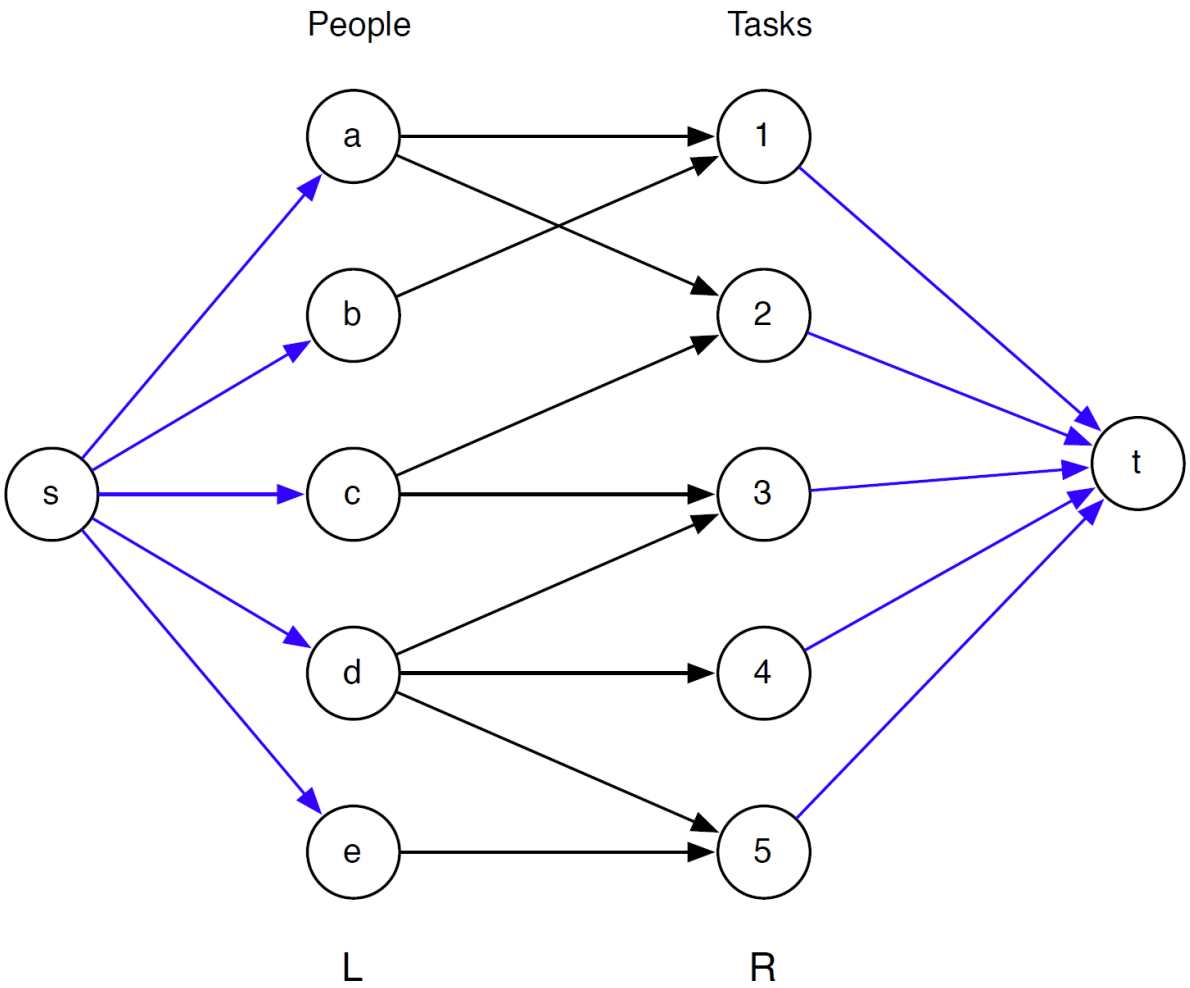


Transforming Bipartite Matching to Network Flow



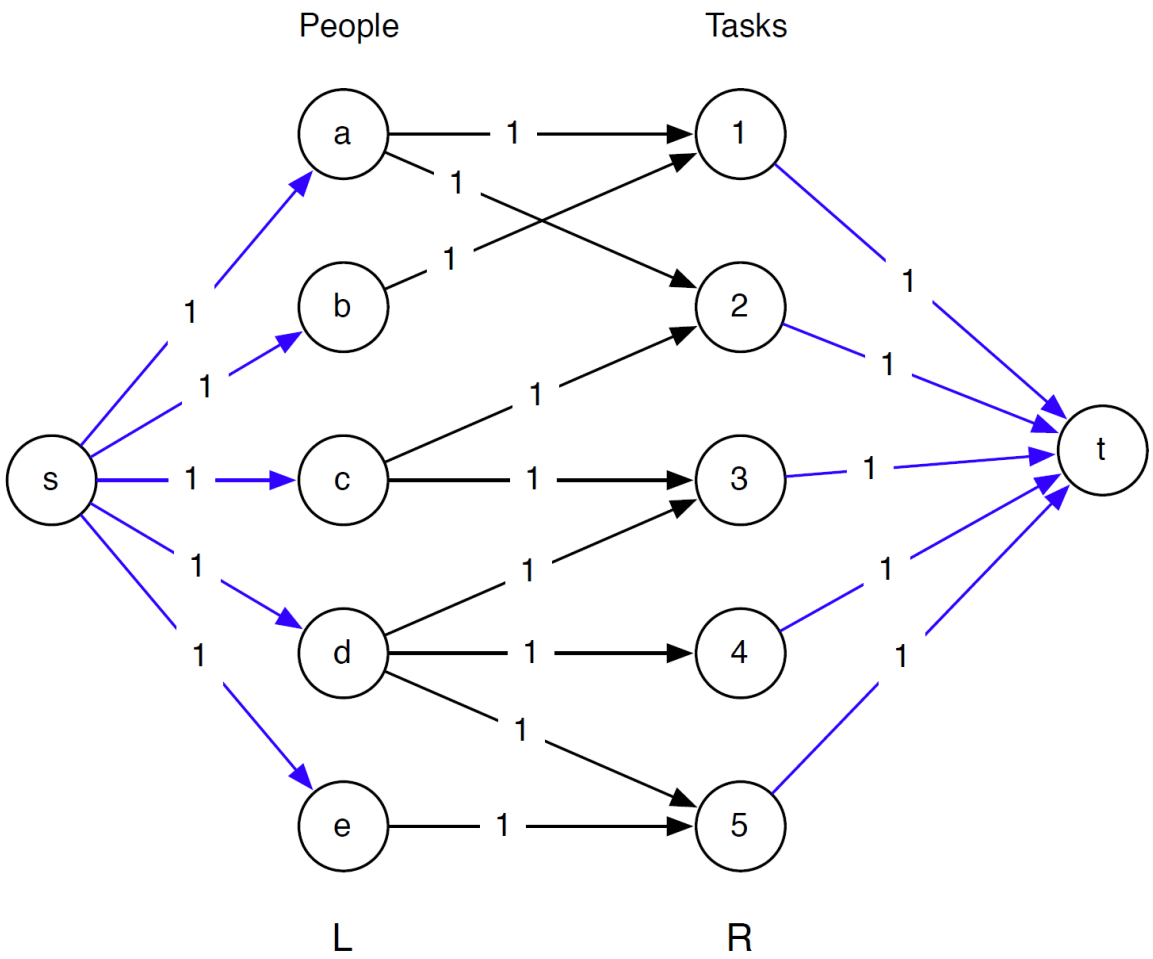
Direct the edges from L to R.

Transforming Bipartite Matching to Network Flow



- Add new vertices s and t.
- Add an edge from s to every vertex in L.
- Add an edge from every vertex in R to t.

Transforming Bipartite Matching to Network Flow



- Make all the capacities 1.
- Solve maximum network flow problem on this new graph G'.
- The edges used in the maximum network flow will correspond to the largest possible matching!

The stable marriage problem

- The stable marriage problem is not a network flow problem, but it is a matching problem. The scenario is as follows:
  - We are given two sets, V_M and V_F (male and female) of the same size. Also, every man $v \in V_m$ ranks every woman $u \in V_F$ and every woman $u \in V_F$ ranks every man in V_M.
  - We will write $u <_v u'$ if v would rather marry u than u'.
  - The stable marriage problem is to find a matching $E \subset V_M \times V_F$ such that if (v, u) and (w, z) are in E , then either $u <_v z$ or $w <_z v$.

Solution

- The Gale-Shapley algorithm is a solution to the stable marriage problem that involves a number of rounds.
- Heuristically, at each round every “unengaged” man proposes to his most preferred woman that he has not already proposed to. If this woman is unengaged or engaged to someone she prefers less than her new suitor, she breaks off her current engagement and accepts the new proposal.
- These iterations continue until everybody is engaged. It can be shown that at this point we have a solution to the stable marriage problem.