SC1007
Data Structures and Algorithms

Bipartite Graph – Matching Problem

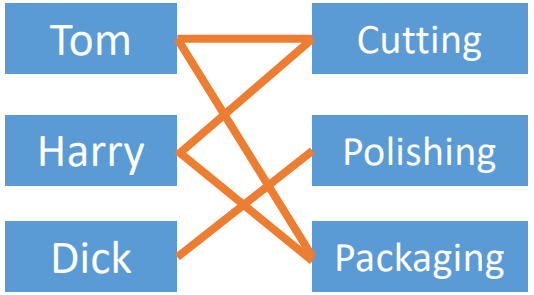
Maximum Flow Algorithm

Dr Liu Siyuan (syliu@ntu.edu.sg)

N4-02C-117a

Office Hour: Mon & Wed 4-5pm

Matching Problem

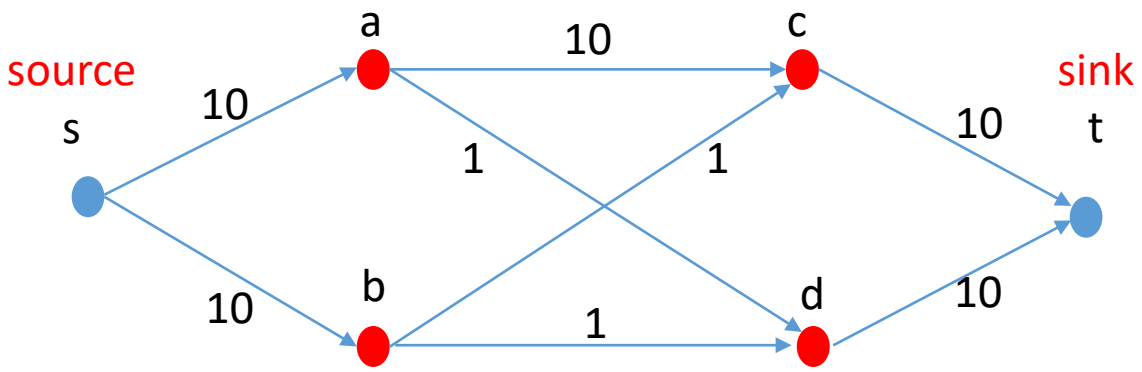


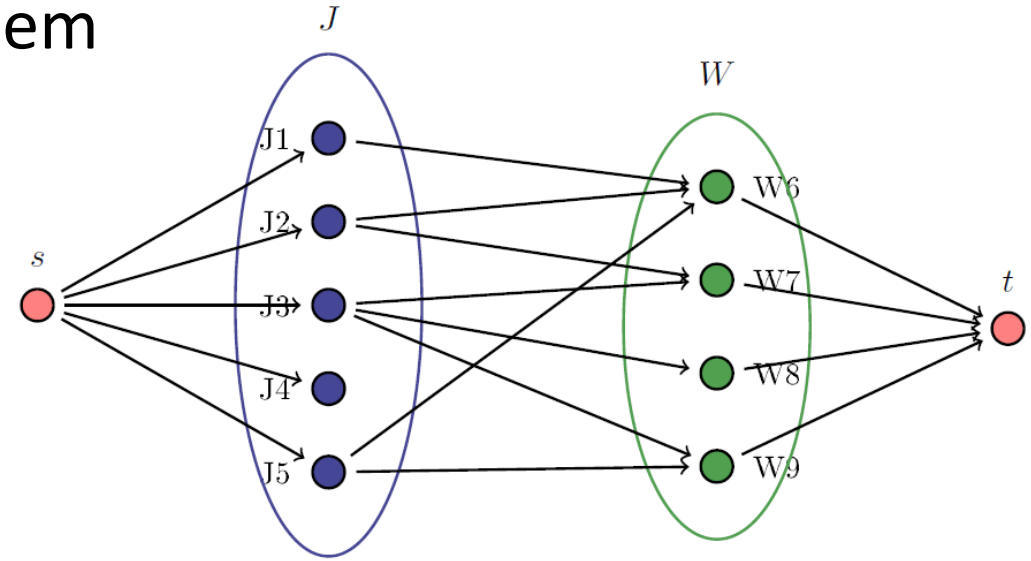
- A graph that vertex V can be partitioned into two subsets, e.g., J and W.
- J and W are two disjoint sets.
- Every edge connects a vertex in J to one in W.
- This graph is known as Bipartite Graph.

- Matching:
  - A subset of edges that are mutually non-adjacent
  - No two edges have an endpoint in common
- Problem: Matching with the maximum number of edges

Maximum Matching == Maximum Network Flow

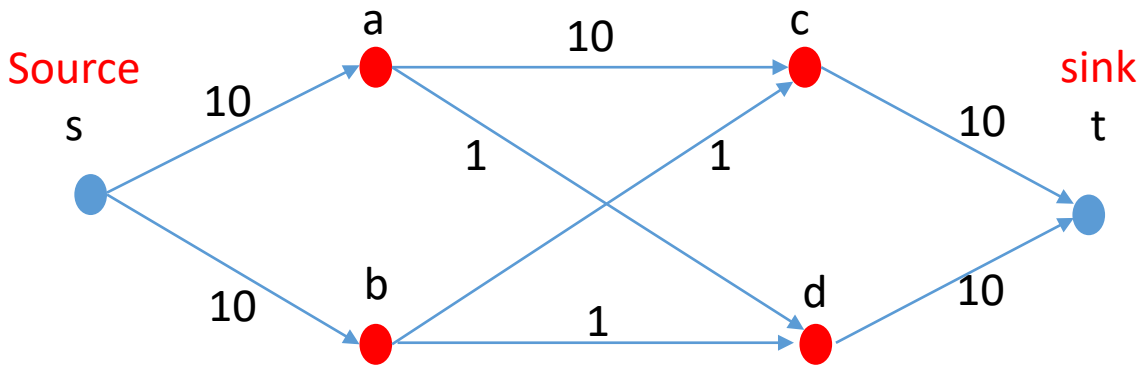
- Network flow
- Introduce two vertices: a source *s* and a sink *t*, flow water from s to t as much as possible withing exceeding the capacities.
- A flow network G=(V,E) is a directed graph in which each edge (j, w) has a non-negative weight.
- The capacity is {0, 1} for matching problem





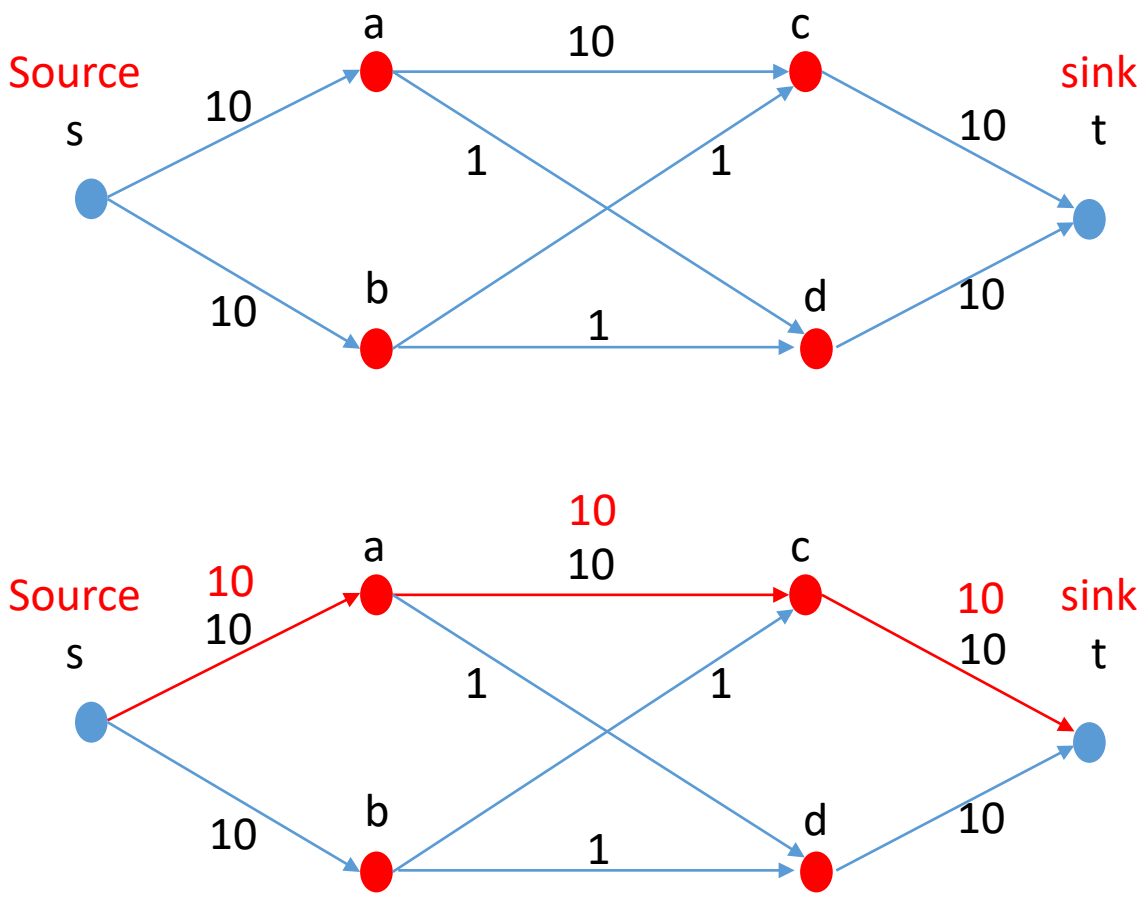
Maximum Network Flow

- Let G={V, E} be a directed graph with weight (capacity) c(j,w) on edge (j,w)
- A flow is an (integer) function, f, that is chosen for each edge so that $f(j,w)$ <= $c(j,w)$
- Maximize the flow allocation $max \sum_{(j,t)\in E} f(j,t)$



Maximum Network Flow: Example

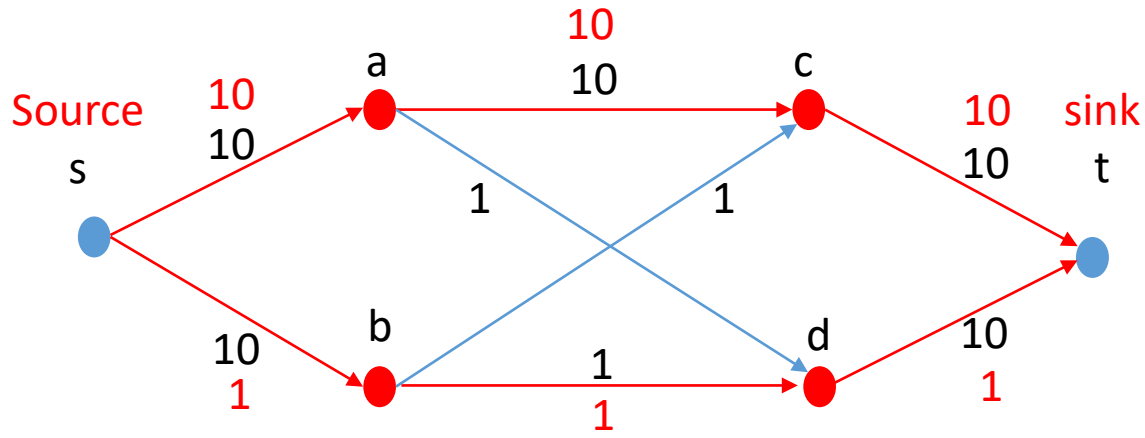
- Flow 1:



Flow is of size 10

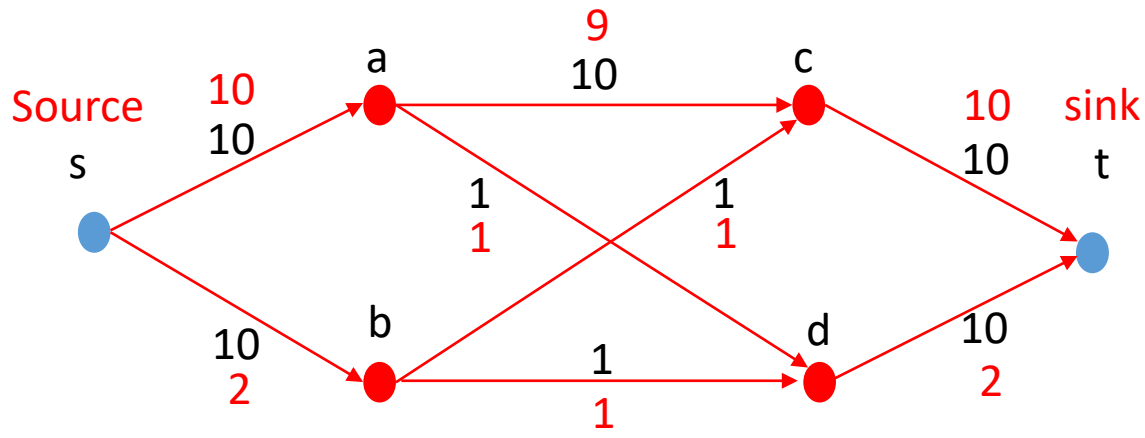
Maximum Network Flow: Example

- Flow 2:



Flow is of size 10+1 = 11

- Flow 3:



Flow is of size 10+2 = 12

Not obvious

The Ford-Fulkerson Method

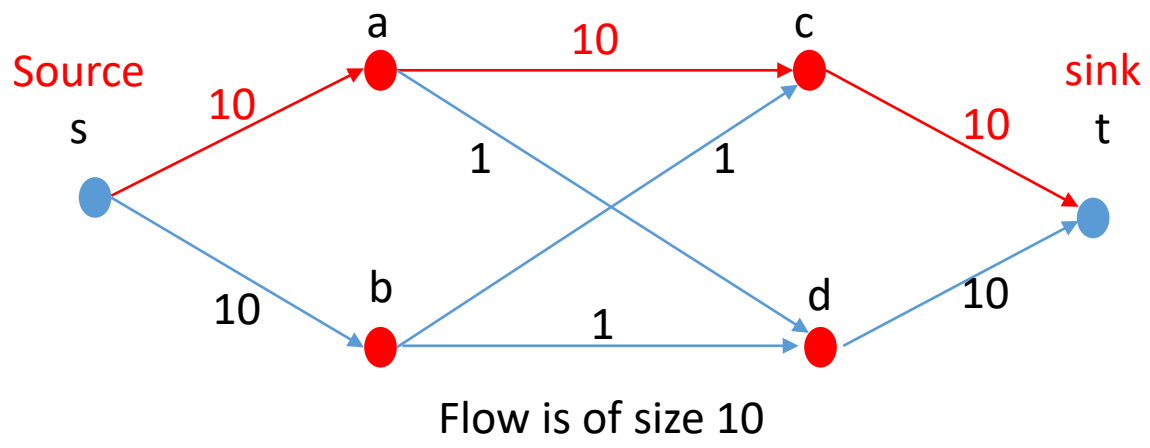
- An iterative improvement strategy
- Proposed by L. R. Ford Jr. and D. R. Fulkerson in 1956
- Iteratively find an additional flow (match) in the network
  - A residual network is used to find the available flow
  - $c_f(j,w) = c(j,w) - f(j,w)$

- Initially, there is no flow
- Residual network == original network
- $c(w,j) = 0, if\ (j,w) \in E$

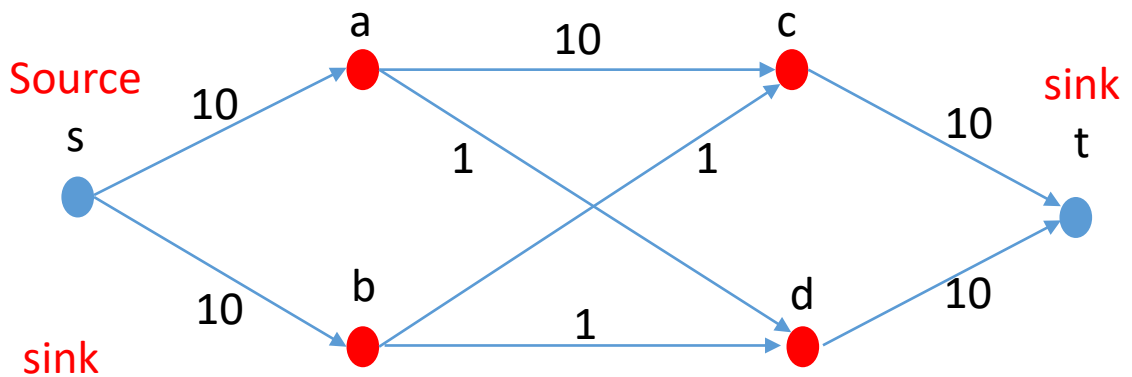
Ford-Fulkerson Method

1. Find a path from s to t
2. Increase the flow along the previous paths to augment them by minimum residue c_{min} along the new path
3. For each (j,w) on the path, set $f(j,w) = c_{min}, f(w,j) = -c_{min}$
4. Update the residual network, in which $c_f(j,w) = c_f(j,w) - f(j,w)$
5. Keep augmenting paths until there are no more to augment

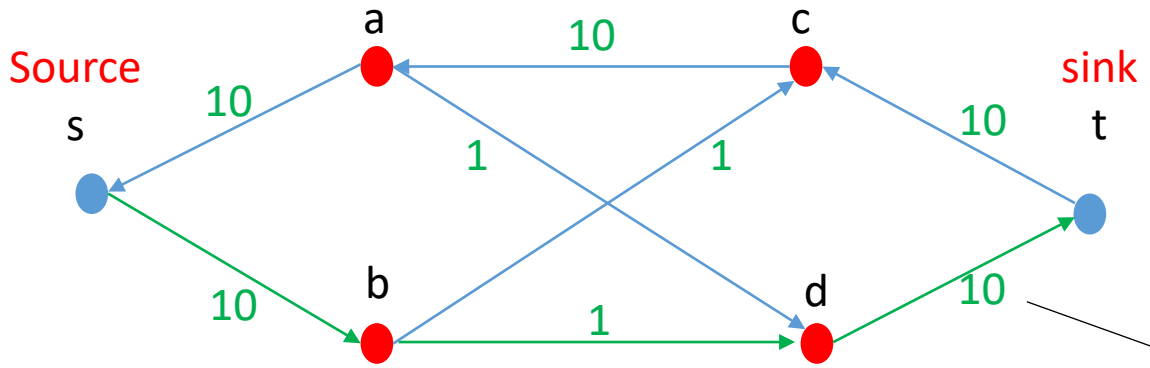
Example



Flow is of size 10

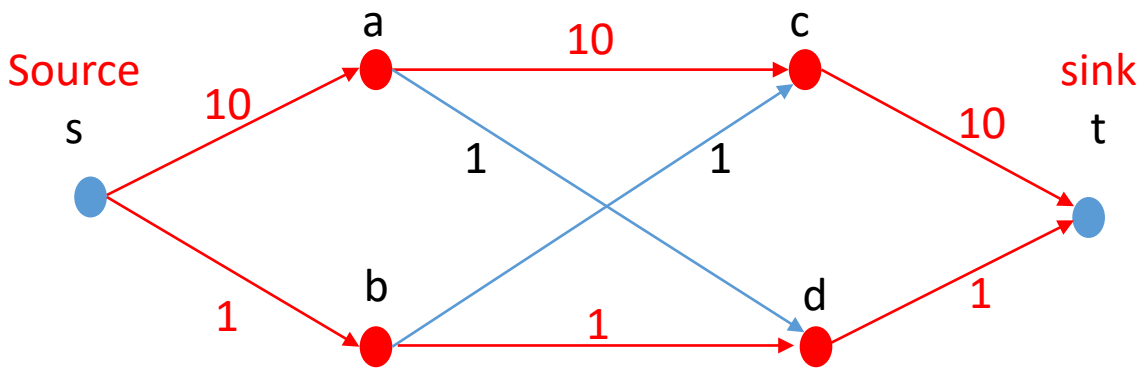


- Residual Graph, R $c_f(j,w) = c(j,w) - f(j,w)$

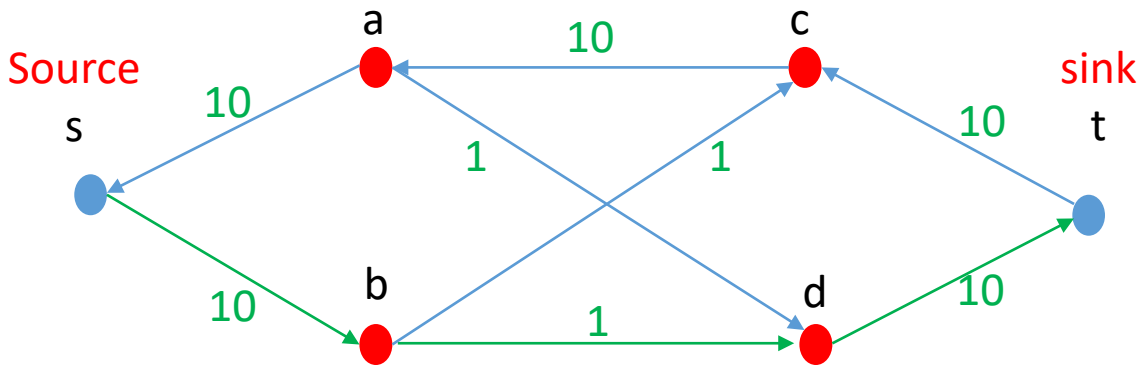


Augmenting path

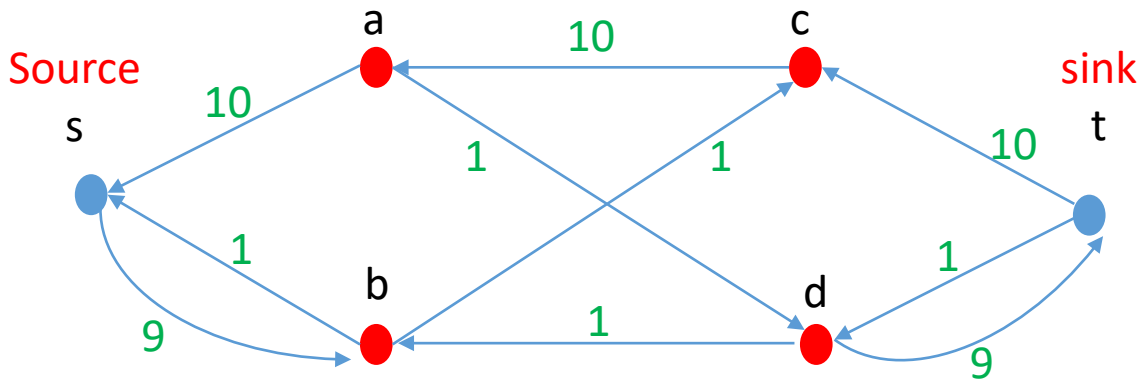
Example



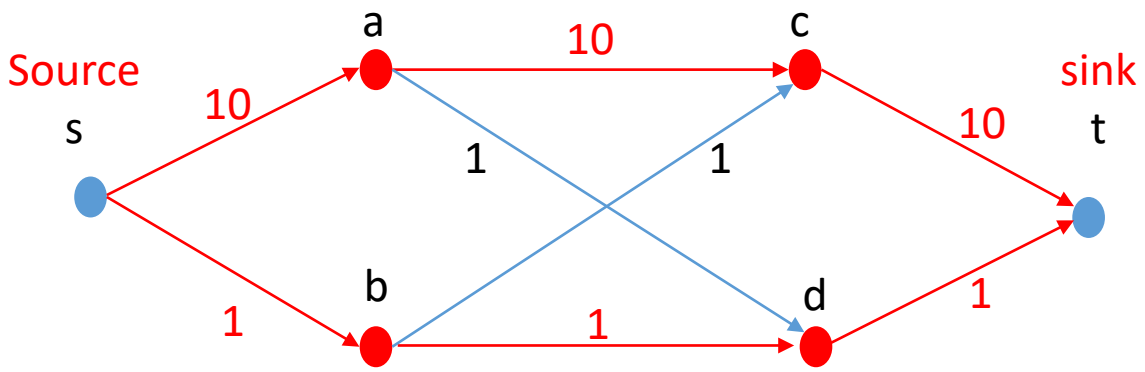
Flow is of size 10+1 = 11



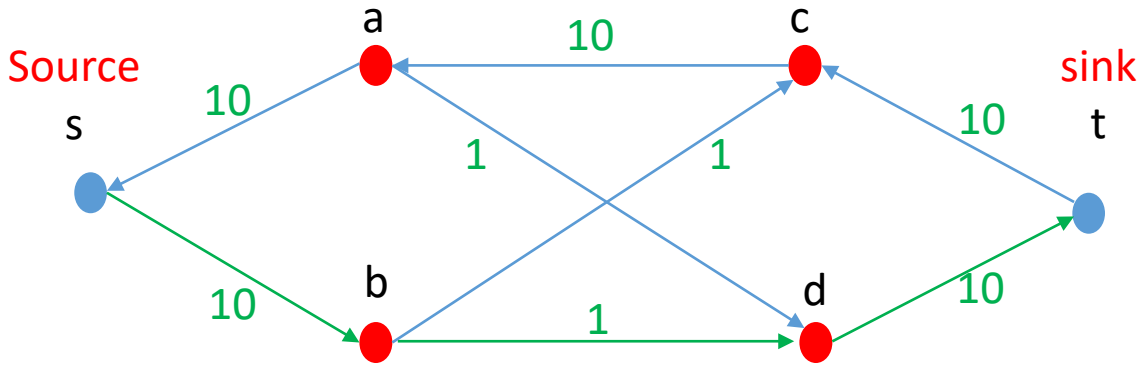
- Residual Graph, R $c_f(j,w) = c(j,w) - f(j,w)$



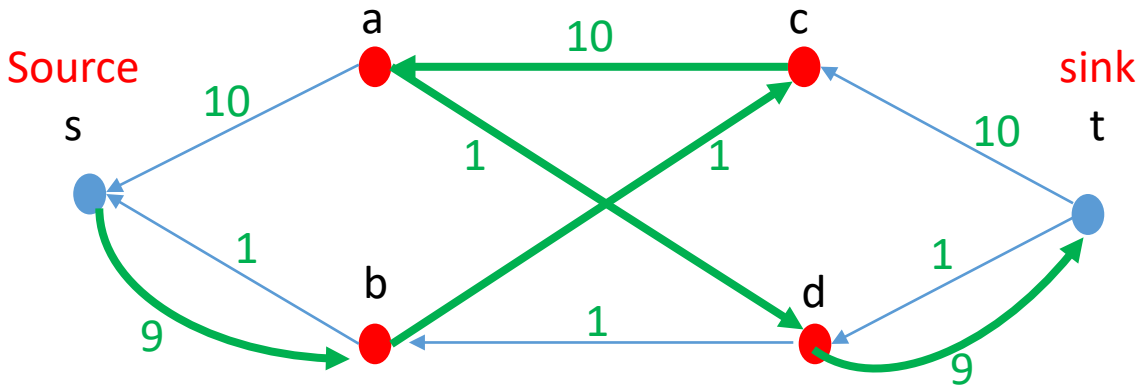
Example



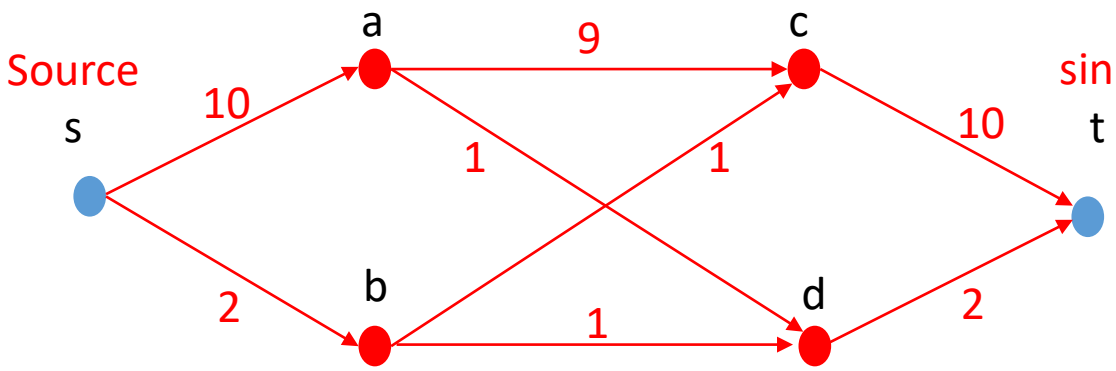
Flow is of size 10+1 = 11

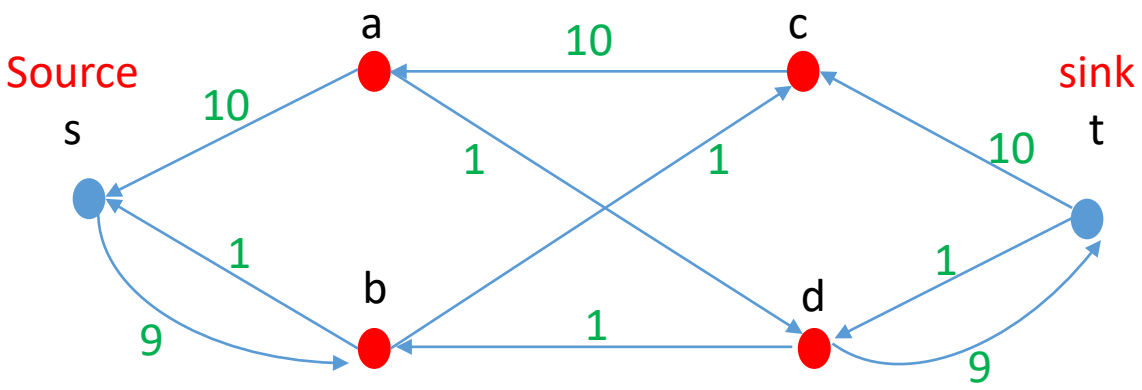


- Residual Graph, R $c_f(j,w) = c(j,w) - f(j,w)$



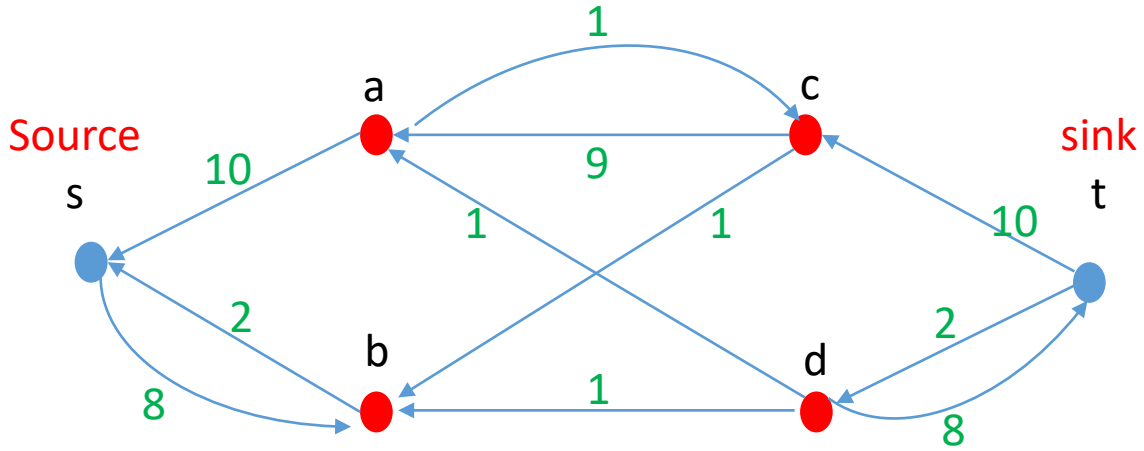
Example





Flow is of size 11+1 = 12

- Residual Graph, R $c_f(j,w) = c(j,w) - f(j,w)$

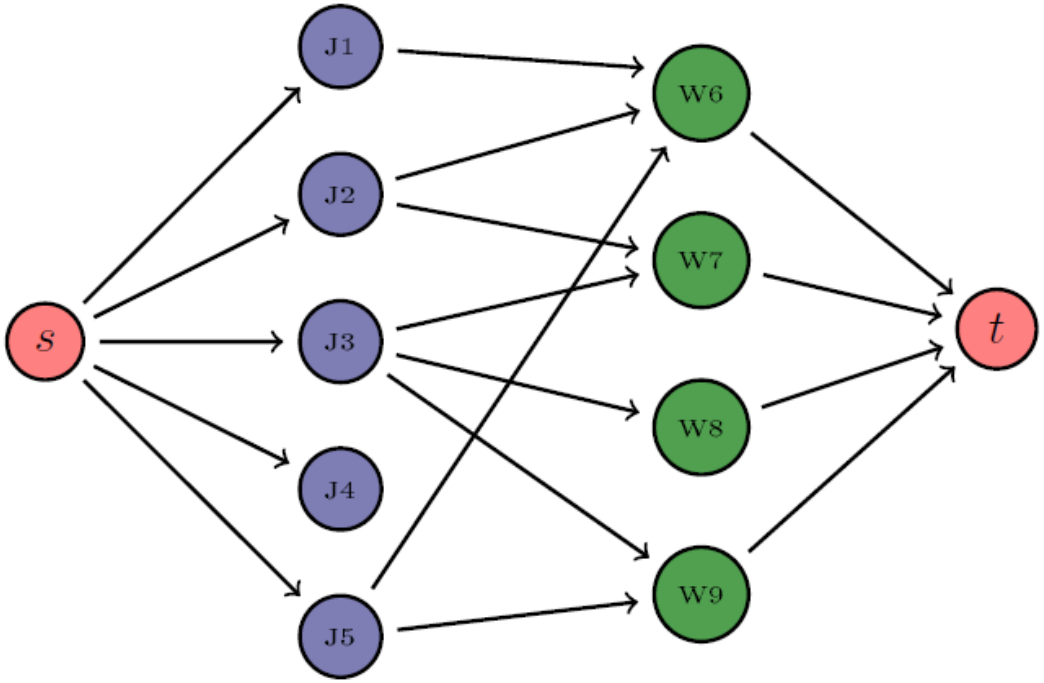


Algorithm 2 Ford-Fulkerson

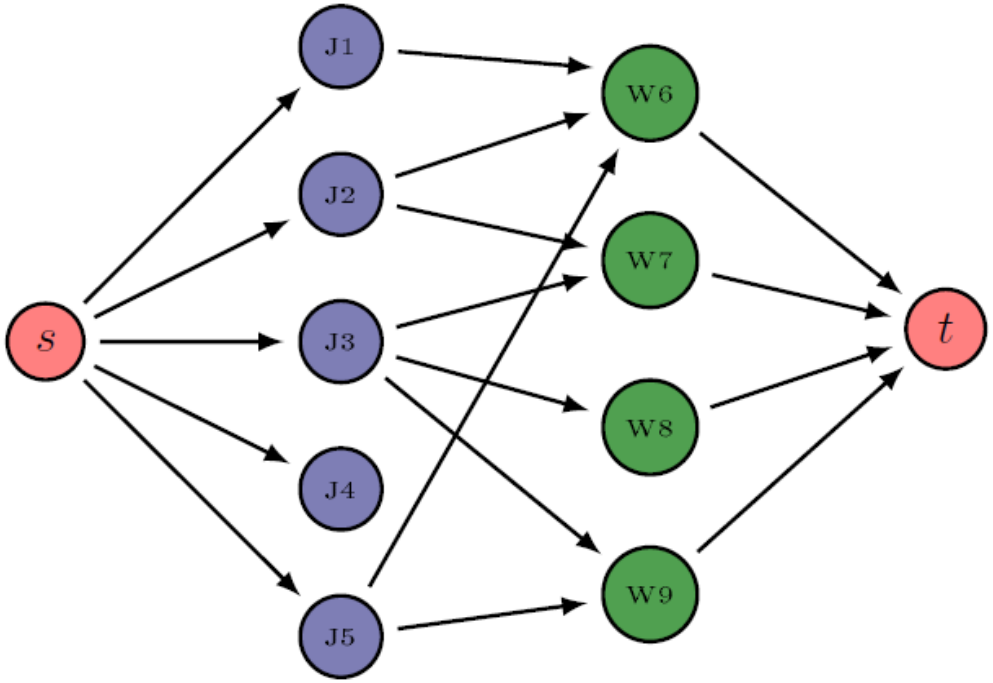
function FORD-FULKERSON(Graph G, Vertex s, Vertex t)
  for each edge $(u, v) \in E[G]$ **do** ▷ Initialization of Flows
    $f[u, v] \leftarrow 0$
    $f[v, u] \leftarrow 0$
  end for
  while Finding a path from s to t in G_f **do**
    $c_{min}(p) \leftarrow \min\{c_f(u, v) : (u, v) \in p\}$
    for each edge $(u, v) \in p$ **do**
      if $(u, v) \in E$ **then**
        $f[u, v] \leftarrow f[u, v] + c_{min}(p)$ ▷ adding the new flow
      else
        $f[v, u] \leftarrow f[v, u] - c_{min}(p)$
      end if
      $c_f(u, v) \leftarrow c_f(u, v) - c_{min}(p)$ ▷ Update Residual Graph, G_f
      $c_f(v, u) \leftarrow c_f(v, u) + c_{min}(p)$
    end for
  end while
end function

- $c_f(u, v) = c(u, v)$
- $c_f(v, u) = 0$: G_f is the same as the original graph G
- G_f *is the residual graph*
- $c_{min}(p)$ is always 1 for matching problem here

First …

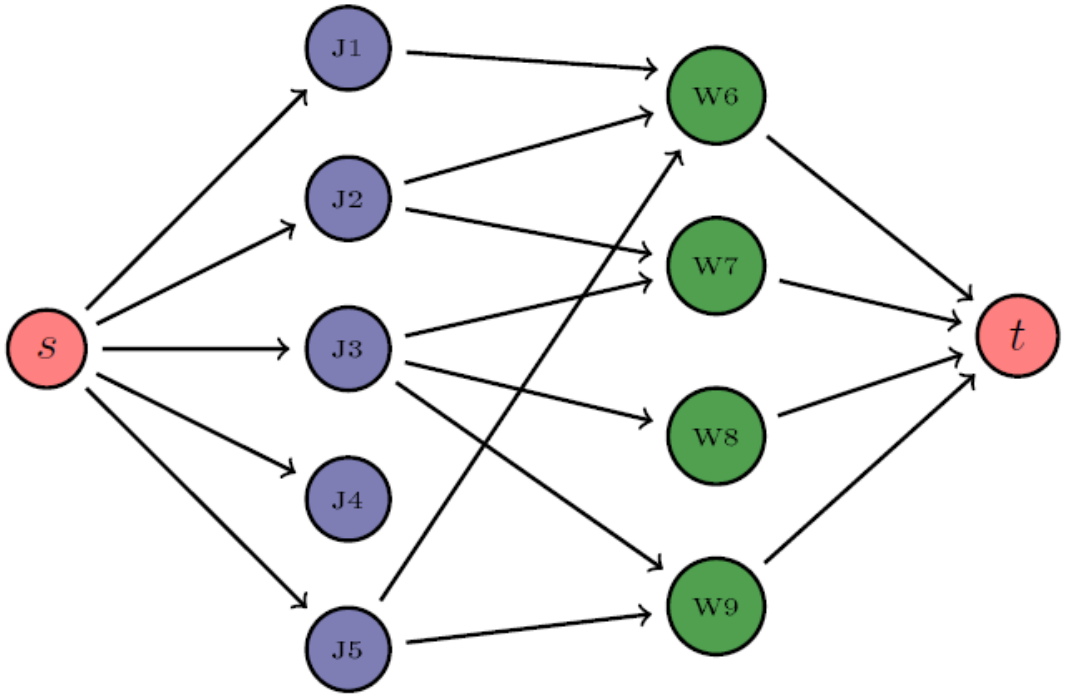


Network Graph

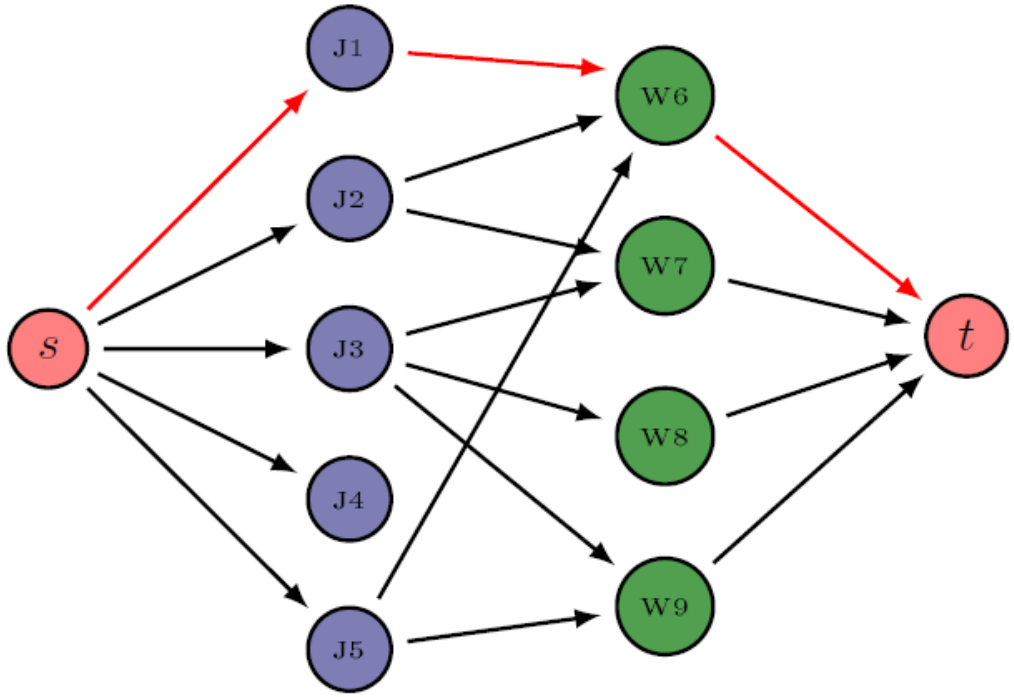


Residual Graph

Next ...

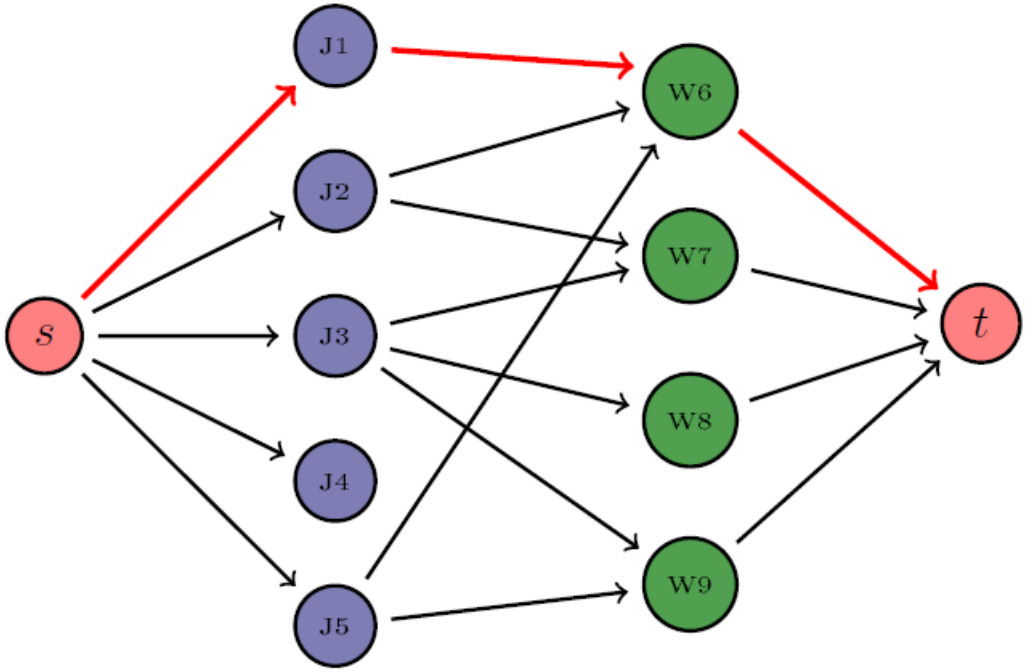


Residual Graph

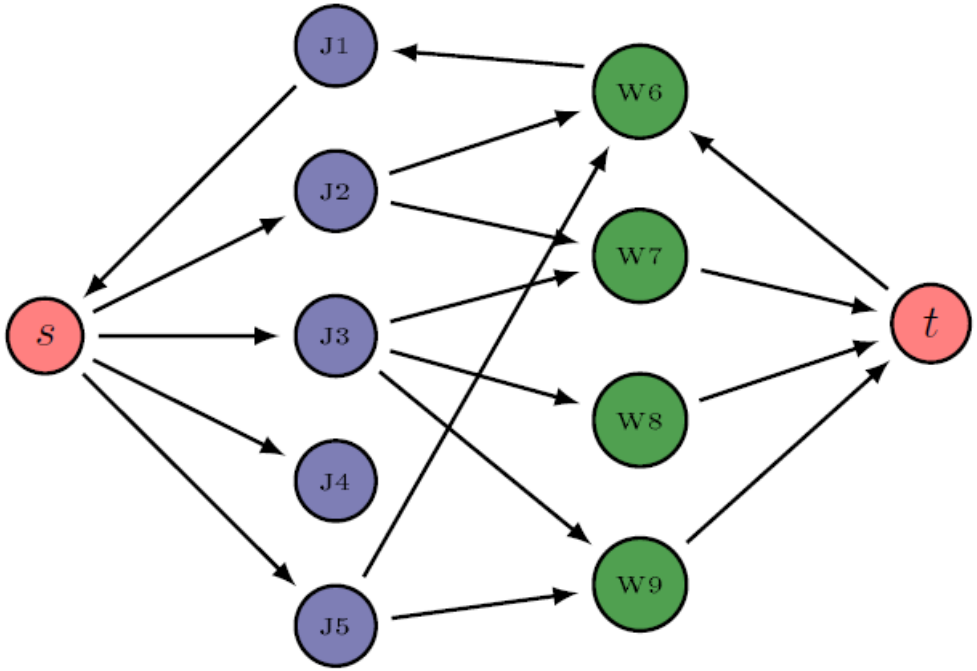


Find a Path

Next...

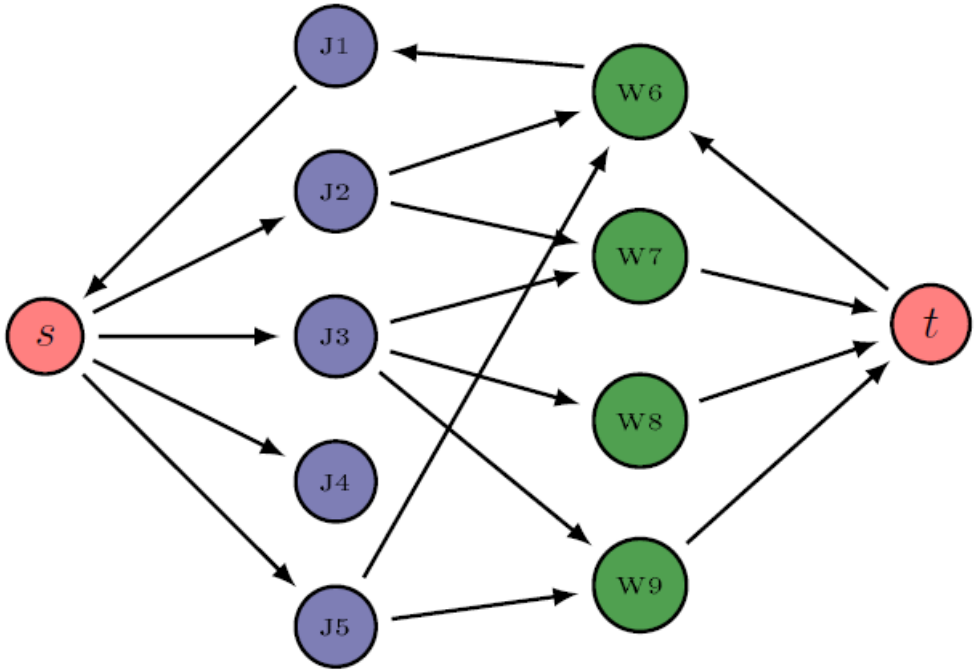


Add the flow

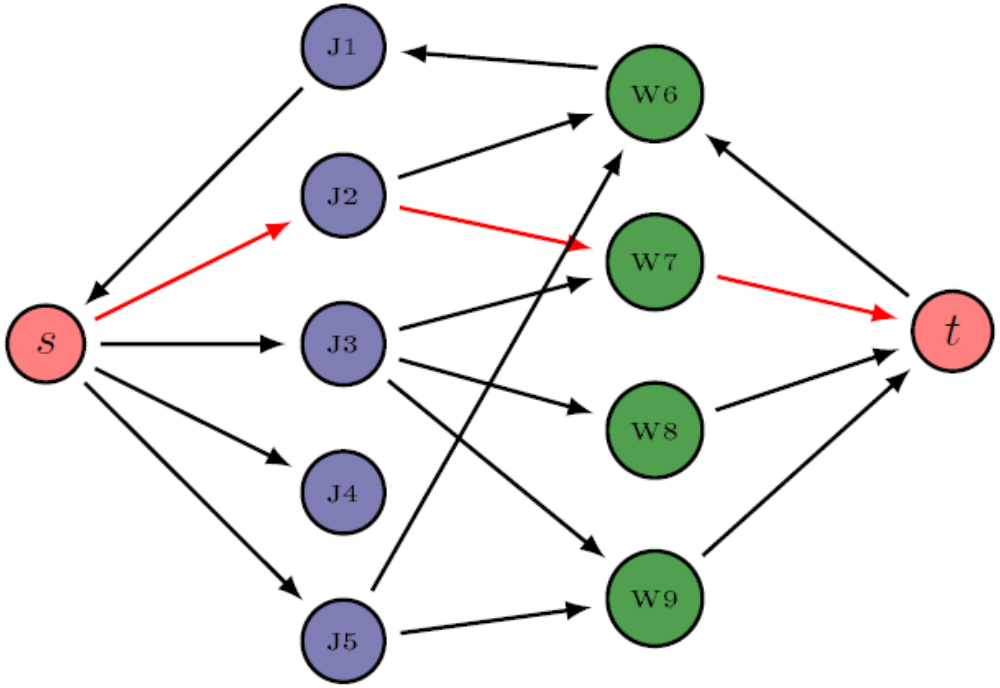


Residual Graph

Next...

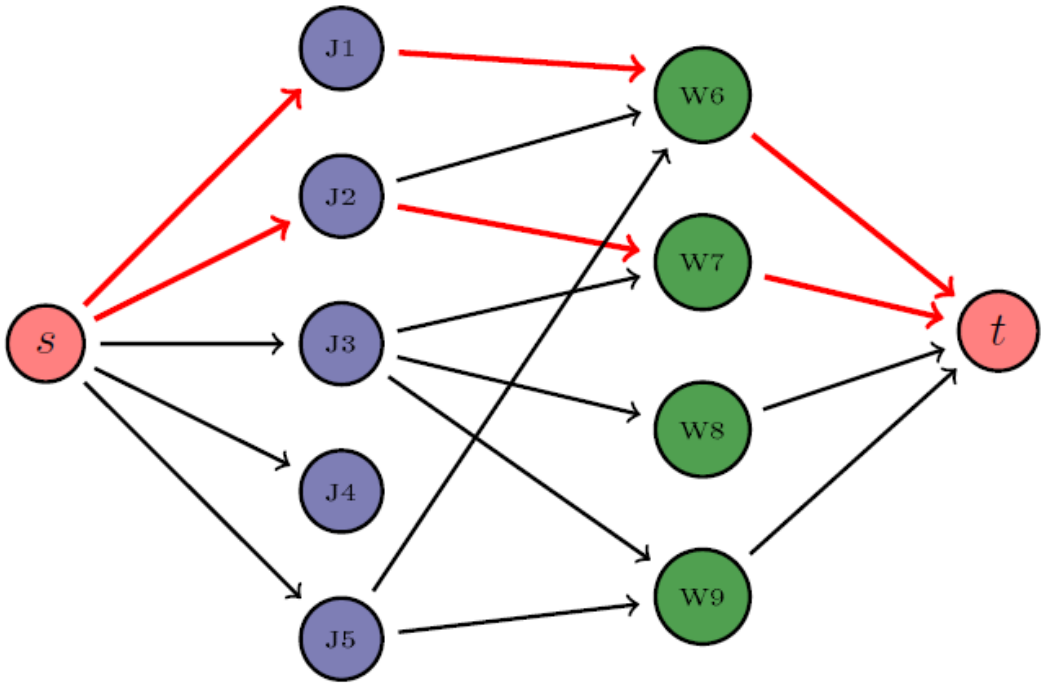


Residual Graph

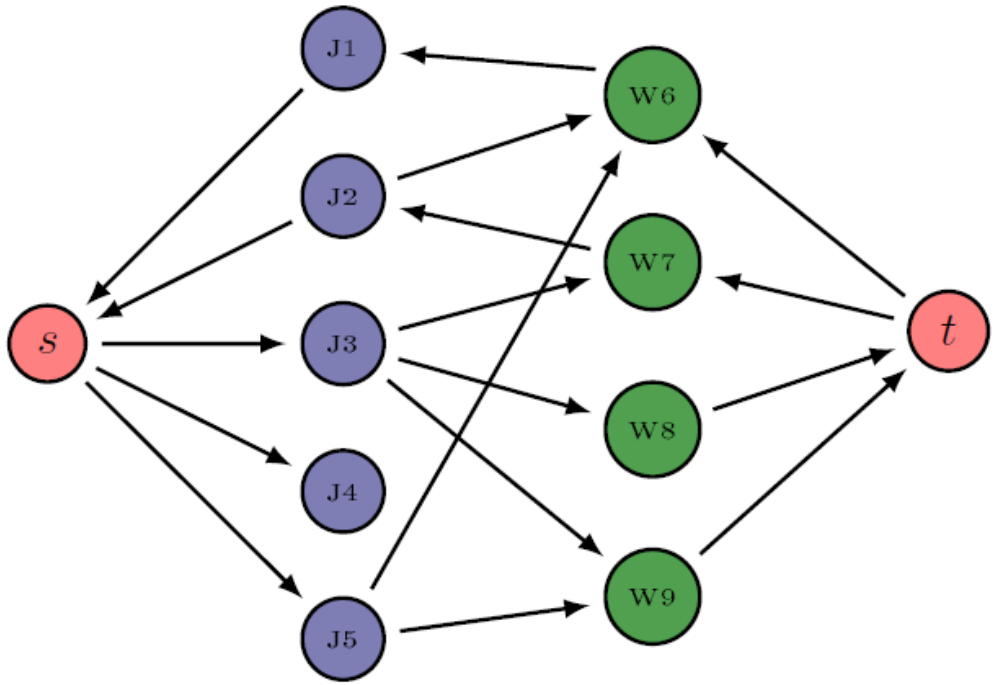


Find a Path

Next ...

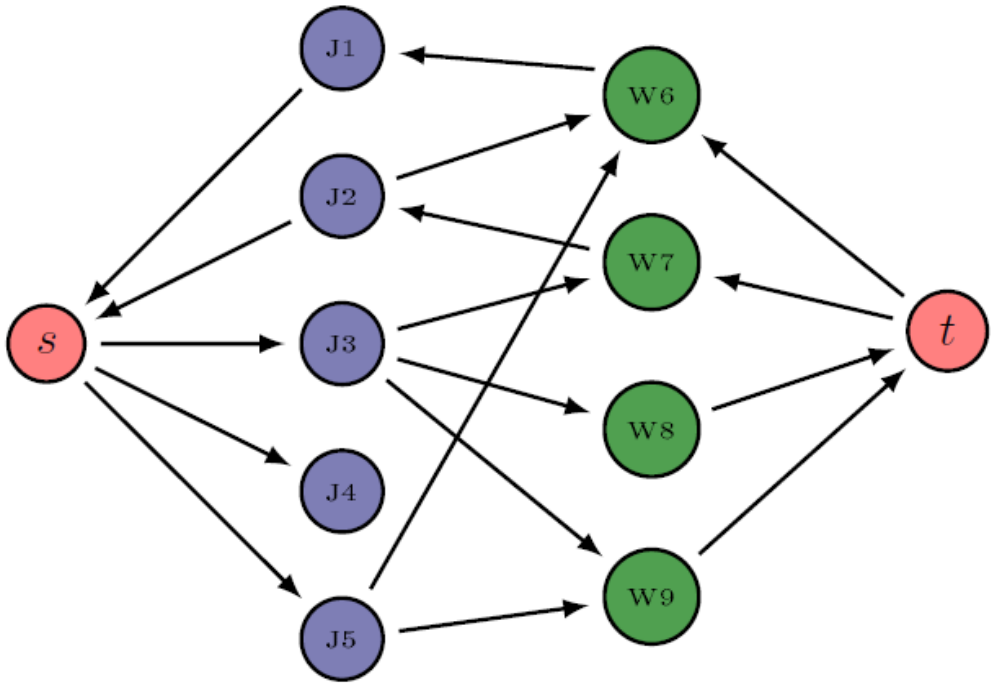


Add the flow

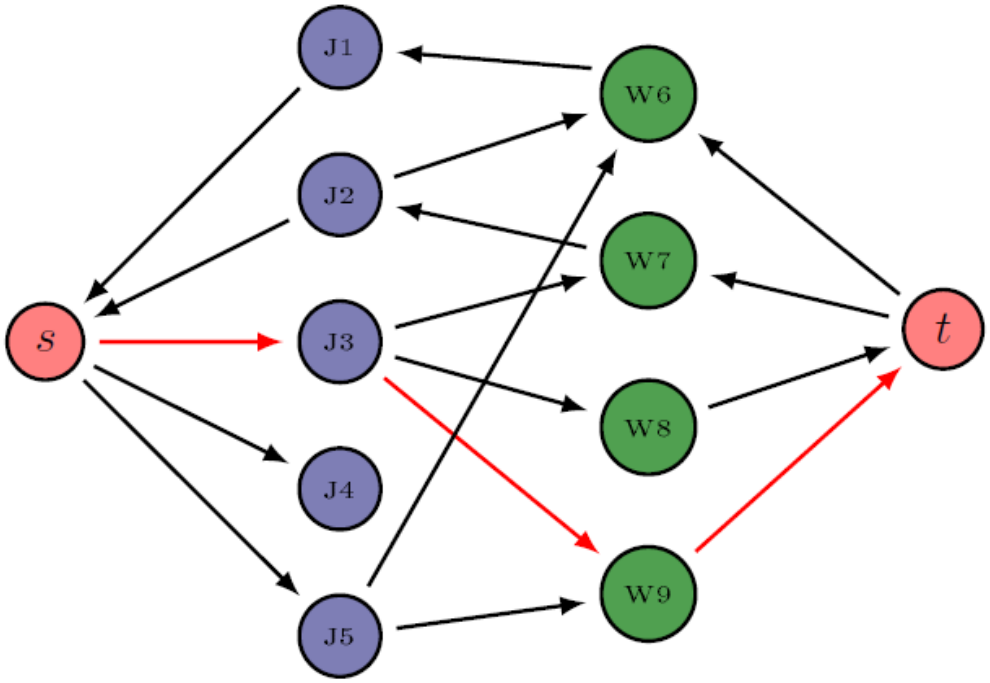


Residual Graph

Next ...

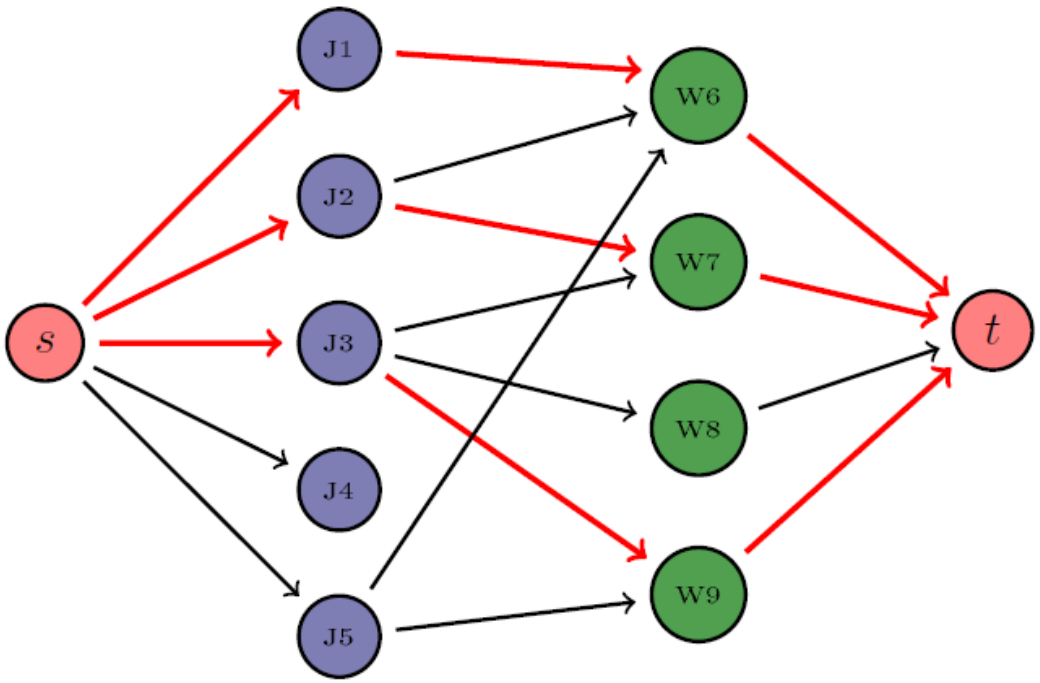


Residual Graph

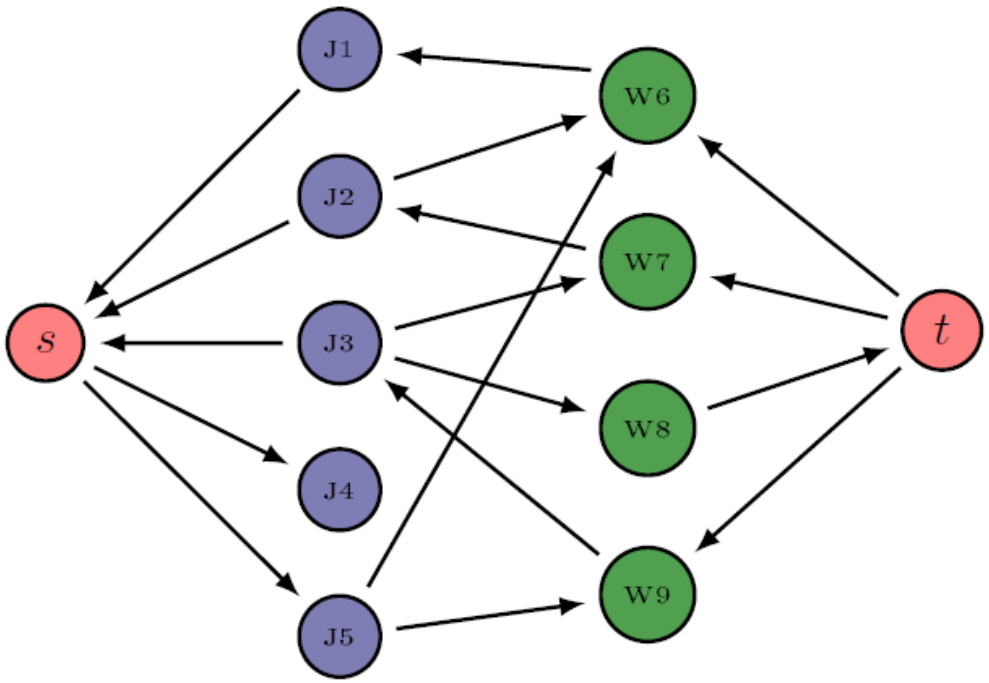


Find a Path

Next ...

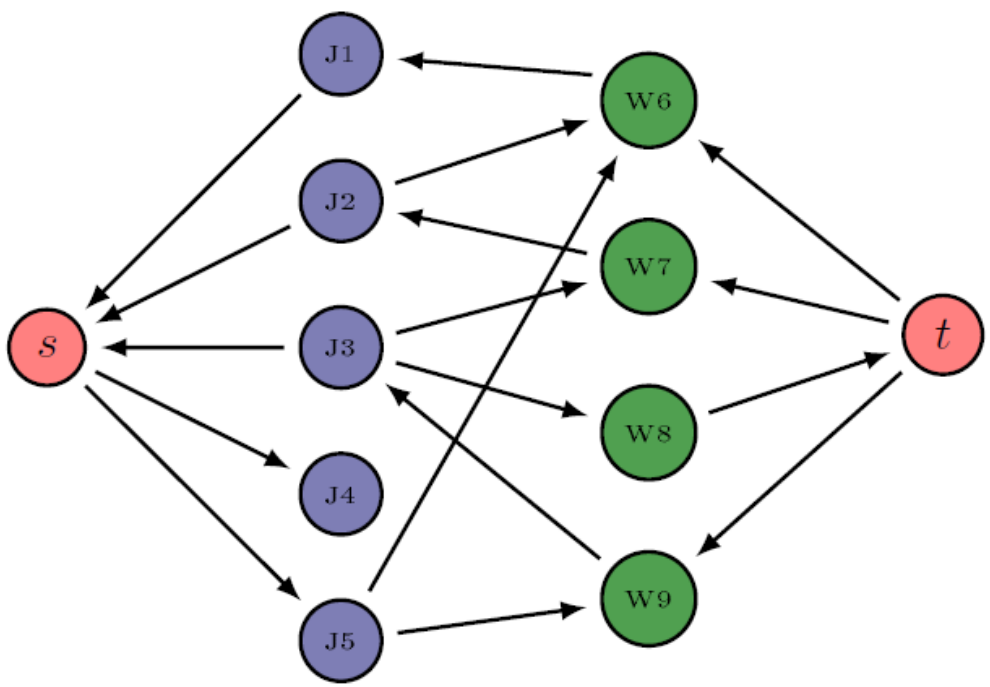


Add the flow

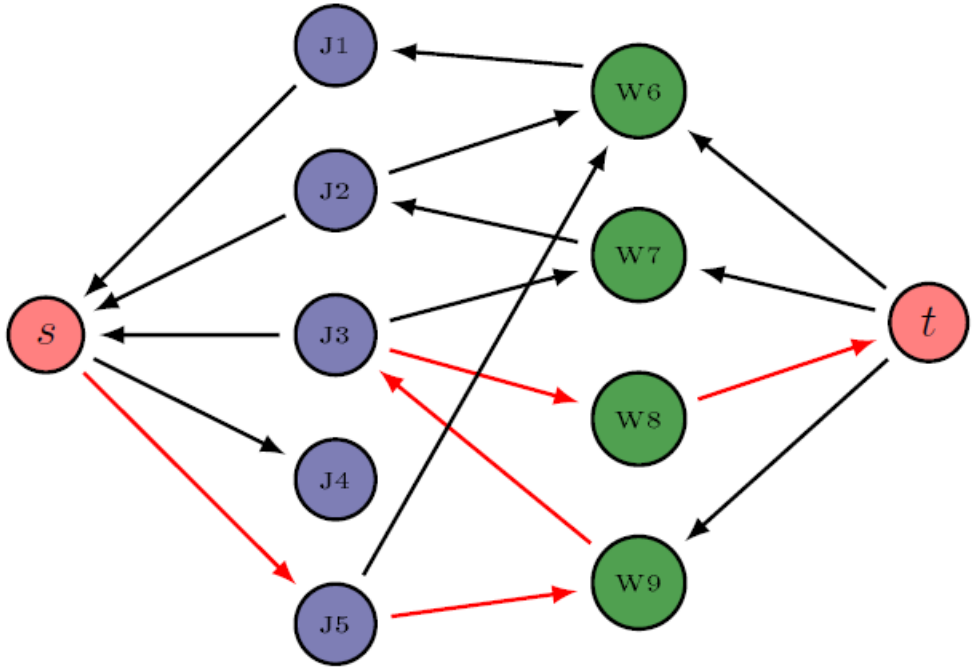


Residual Graph

Next …

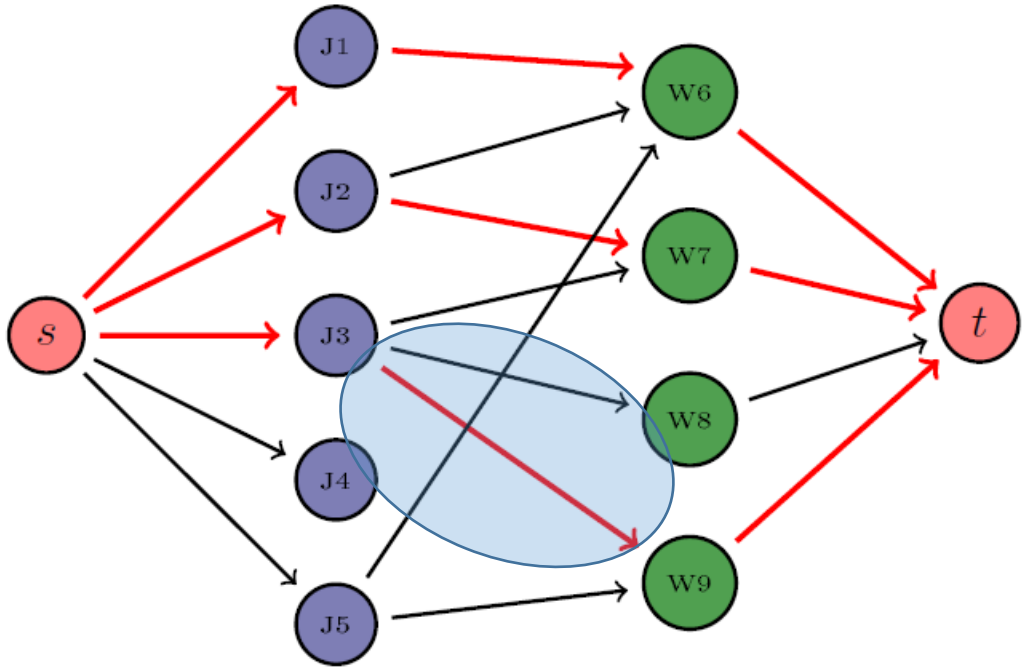


Residual Graph

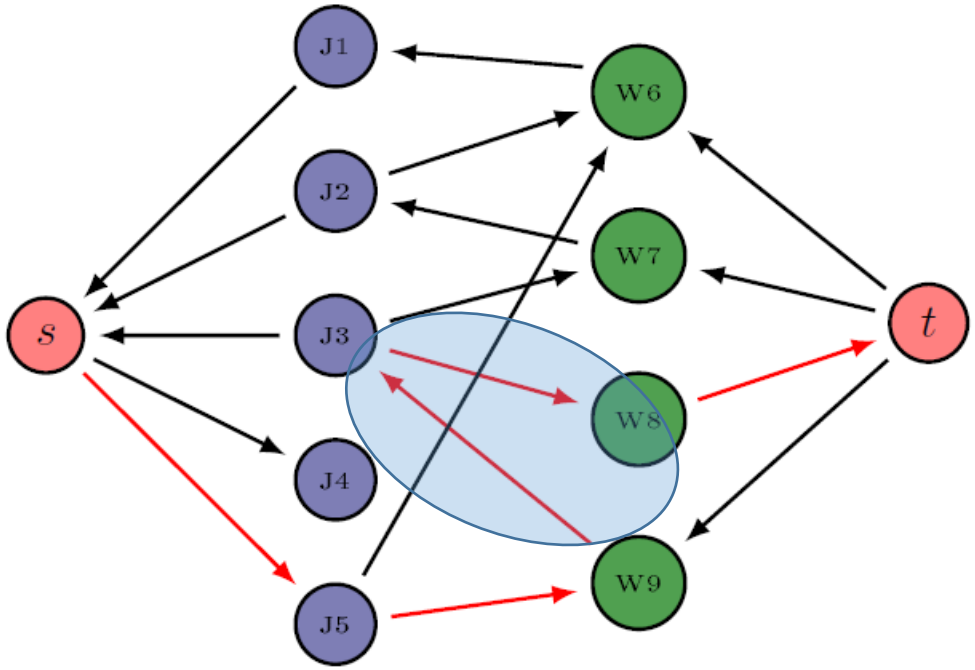


Find a Path

Next… How to add the flow?



Previous Flow



New Flow

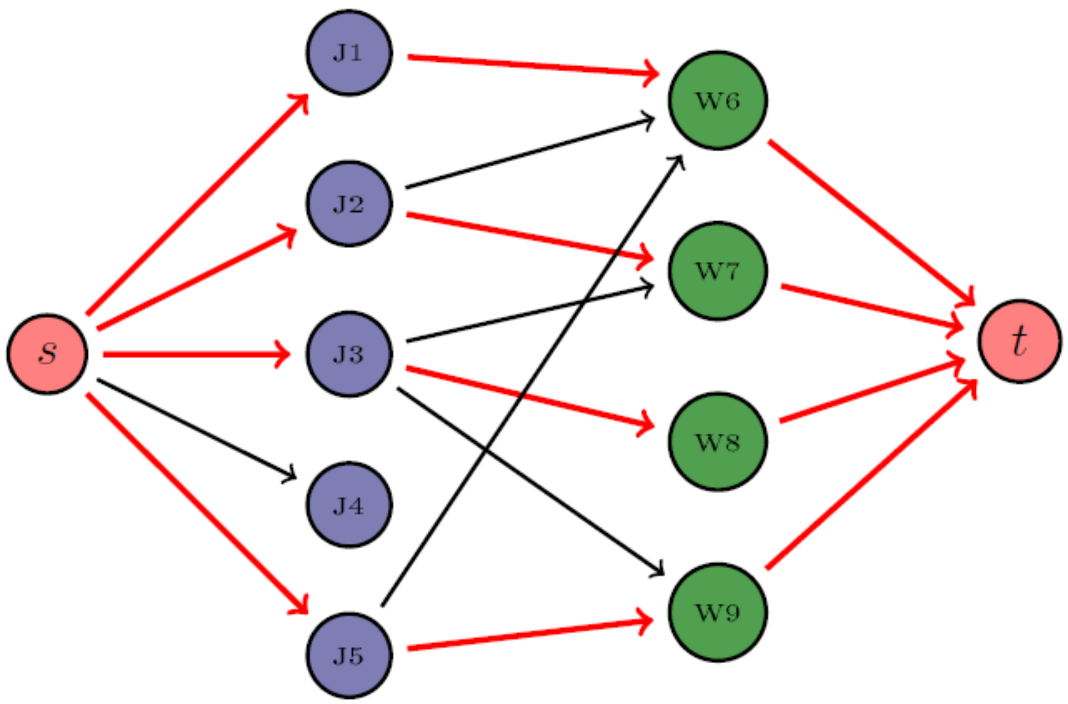
Algorithm 2 Ford-Fulkerson

```
function FORD-FULKERSON(Graph G, Vertex s, Vertex t)
    for each edge (u, v) ∈ E[G] do                          ▷ Initialization of Flows
        f[u, v] ← 0
        f[v, u] ← 0
    end for
    while Finding a path from s to t in G_f do
        c_min(p) ← min{c_f(u, v) : (u, v) ∈ p}
        for each edge (u, v) ∈ p do
            if (u, v) ∈ E then
                f[u, v] ← f[u, v] + c_min(p)                ▷ adding the new flow
            else
                f[v, u] ← f[v, u] − c_min(p)
            end if
            c_f(u, v) ← c_f(u, v) − c_min(p)                ▷ Update Residual Graph, G_f
            c_f(v, u) ← c_f(v, u) + c_min(p)
        end for
    end while
end function
```

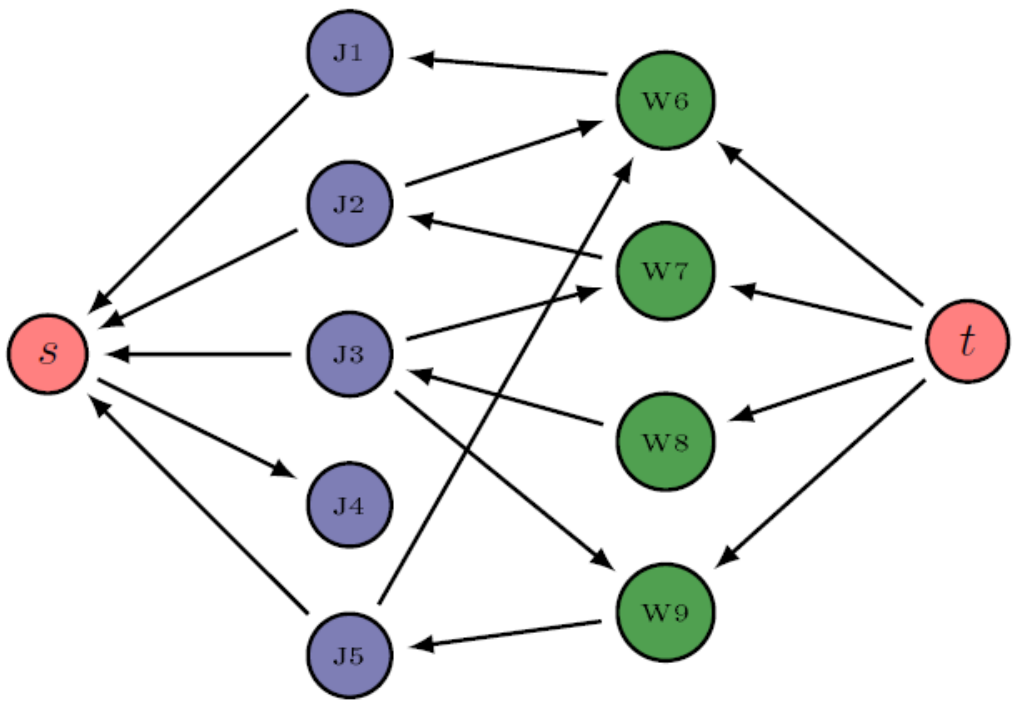
- $c_f(u, v) = c(u, v)$
- $c_f(v, u) = 0$: G_f is the same as the original network G

- G_f *is the residual network*
- $c_{min}(p)$ is always 1 for matching problem here

Next… No new flow. Done!



Add the flow



Residual Graph

Summary

- Ford Fulkerson Method
  - Maximum Flow Problem
  - Maximum Matching Problem
- Finding the augmenting paths
  - BFS or DFS or any graph traversal
- Matching problem is a {0,1} weighted graph

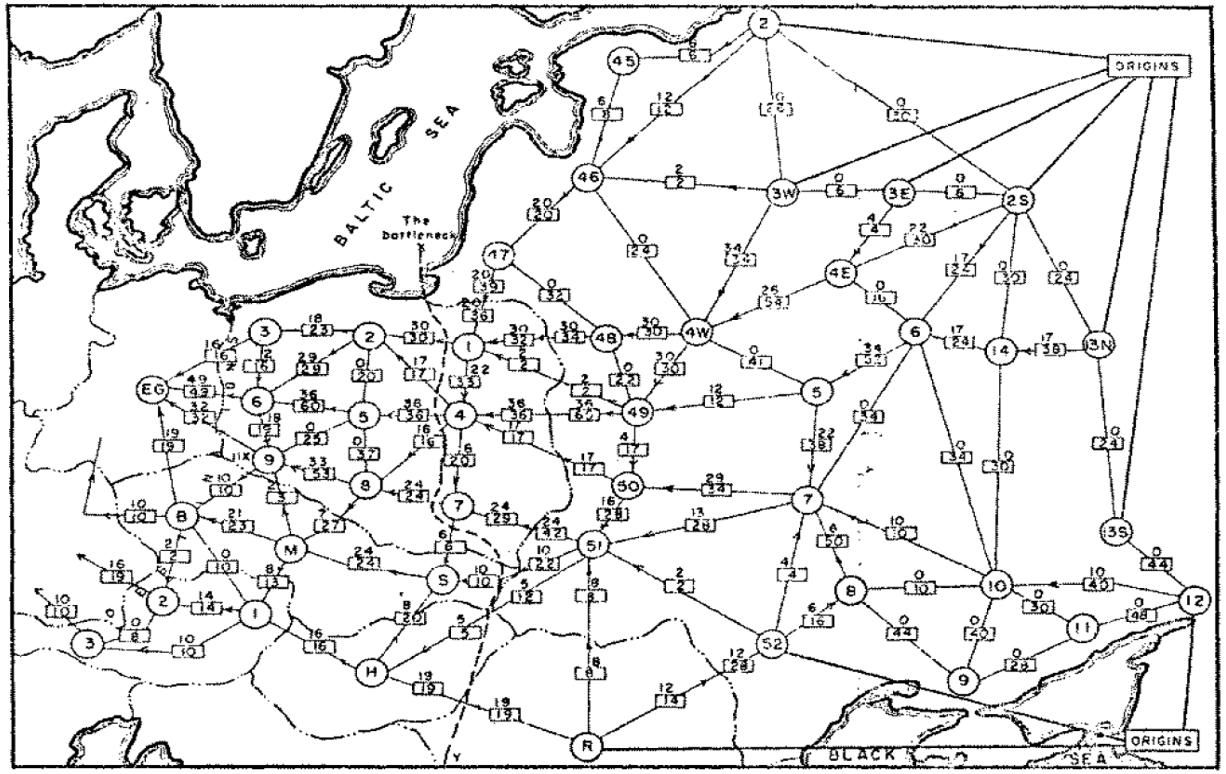
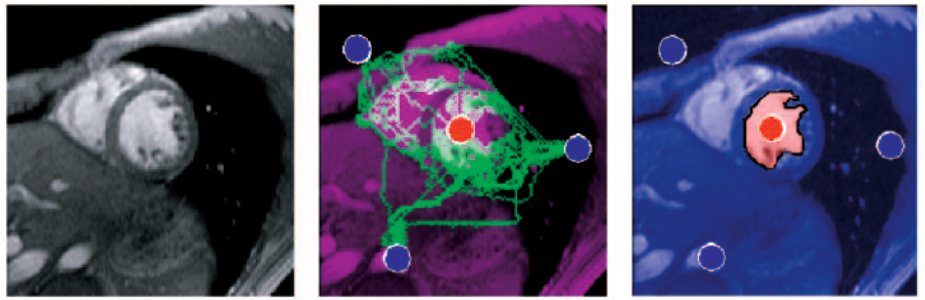


Figure 2

From Harris and Ross [1955]: Schematic diagram of the railway network of the Western Soviet Union and Eastern European countries, with a maximum flow of value 163,000 tons from Russia to Eastern Europe, and a cut of capacity 163,000 tons indicated as "The bottleneck".

Source: On the history of the transportation and maximum flow problems. Alexandar Schrijver in Math Programming, 91:3,2002



(a) (b) (c)

Fig. 3. Graph cut/flow example in the context of image segmentation in Section 4.4. Red and blue seeds are "hard-wired" to the source s and the sink t, correspondingly. As usual, the cost of edges between the pixels (graph nodes) is set to low values in places with high intensity contrast. Thus, cuts along object boundaries in the image should be cheaper. Weak edges also work as "bottlenecks" for a flow. In (b), we show a maximum flow from s to t. In fact, it saturates graph edges corresponding to a minimum cut boundary in (c). (a) Original image. (b) A maximum flow. (c) A minimum cut.

Source: An Experimental Comparison of Min-Cut/ Max-Flow Algorithms for Energy Minimization in Vision
Yuri Boykov and Vladimir Kolmogorov, TPAMI, Vol. 26 No. 9, 2004

Problems for you think about

Suppose you operate a large video streaming platform with hundreds of servers and millions of users. At any given time, thousands of users are currently requesting to watch cat videos, and each user needs to be connected with a server so their video can be streamed. It is important to use the content delivery network efficiently (so that the servers are all mostly operating at a similar load), and it is also important to ensure low latency and quick service to users so they don't leave your platform for some other platform that treats them better. Design an algorithm to assign users to content servers that is fair, efficient, and keeps everyone happy?