
15.082 and 6.855J

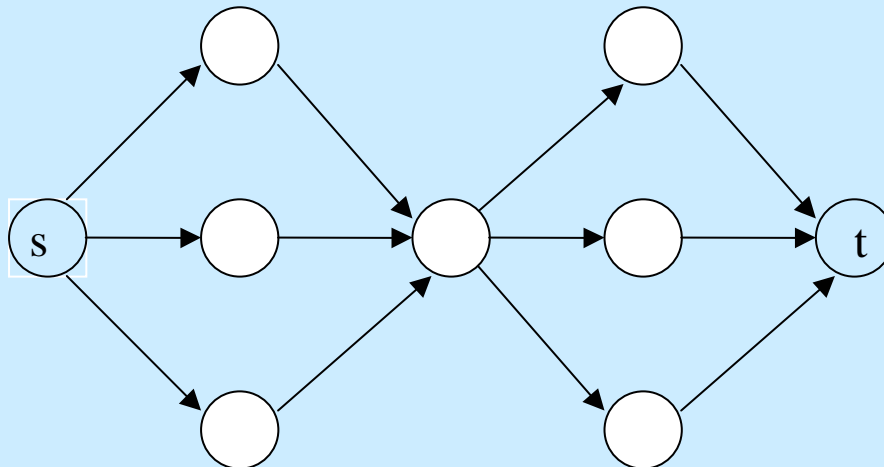
March 6, 2003

Maximum Flows 2

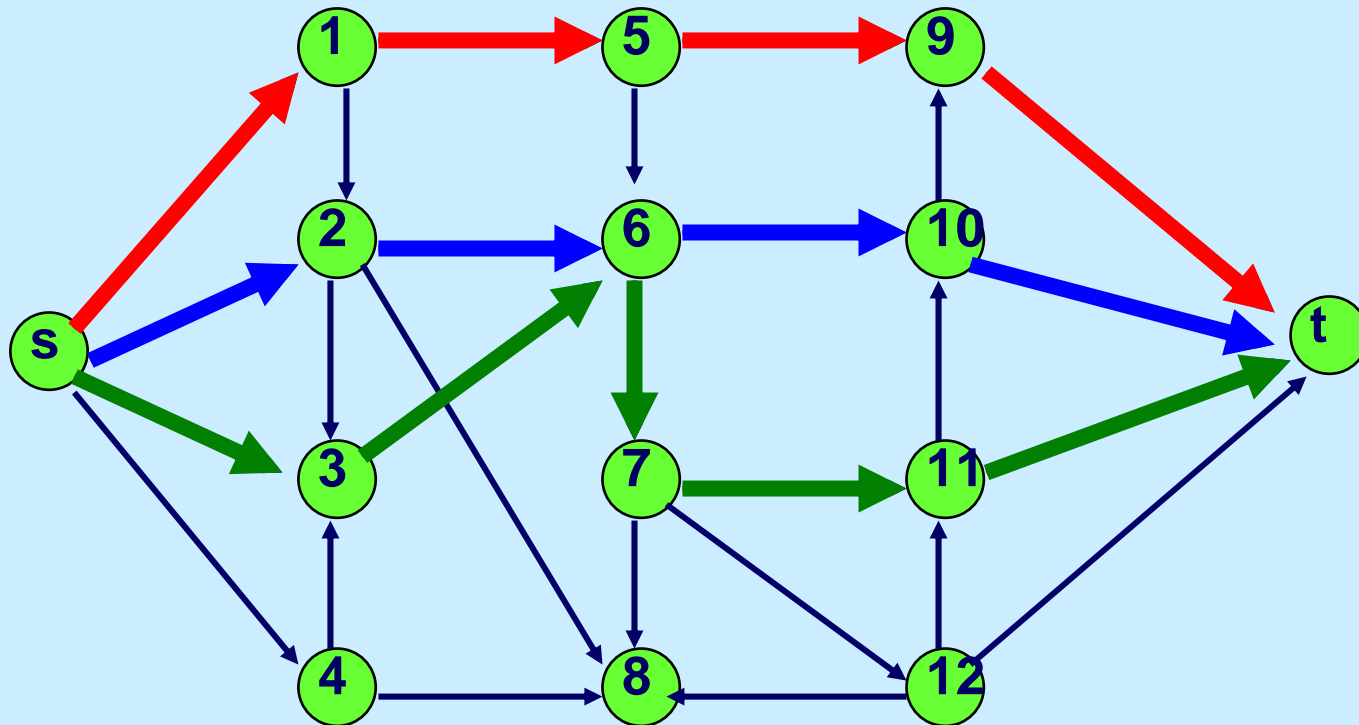
Network Reliability

- ◆ Communication Network
- ◆ What is the maximum number of arc disjoint paths from s to t ?
 - How can we determine this number?

Theorem. *Let $G = (N,A)$ be a directed graph. Then the maximum number of arc-disjoint paths from s to t is equal to the minimum number of arcs upon whose deletion there is no directed s - t path.*



There are 3 arc-disjoint s-t paths



Node disjoint paths

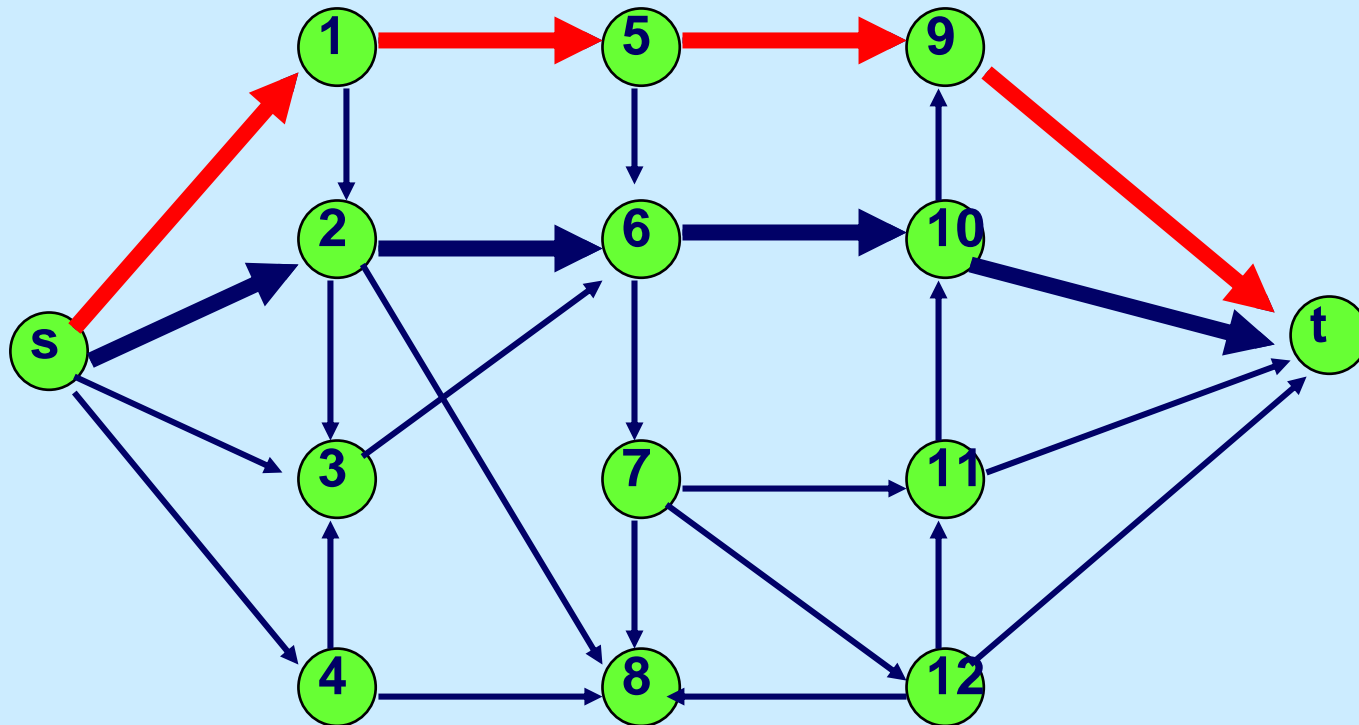
Two s-t paths P and P' are said to be **node-disjoint** if the only nodes in common to P and P' are s and t).

How can one determine the maximum number of node disjoint s-t paths?

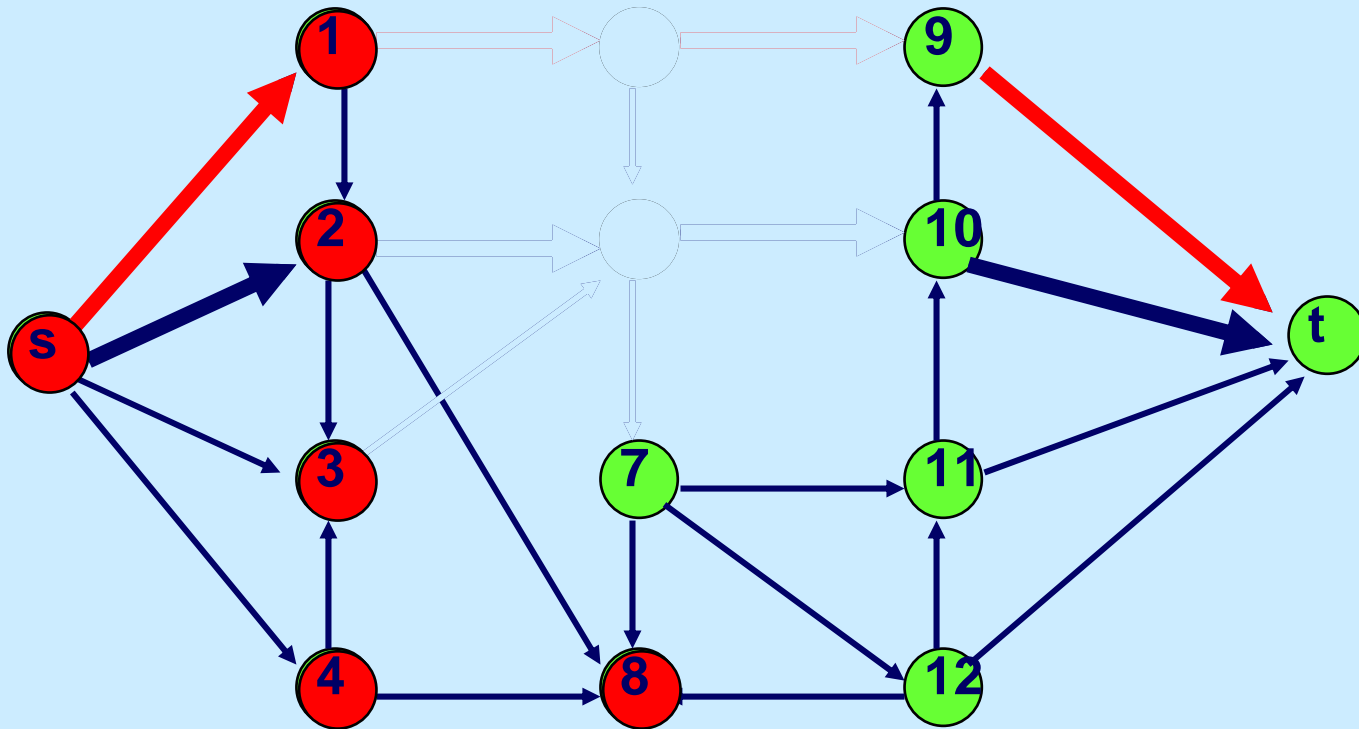
Answer: node splitting

Theorem. Let $G = (N, A)$ be a network with no arc from s to t . The maximum number of node-disjoint paths from s to t equals the minimum number of nodes whose removal from G disconnects all paths from nodes s to node t .

There are 2 node disjoint s-t paths.



Deleting 5 and 6 disconnects t from s?



Let $S = \{s, 1, 2, 3, 4, 8\}$

Let $T = \{7, 9, 10, 11, 12, t\}$

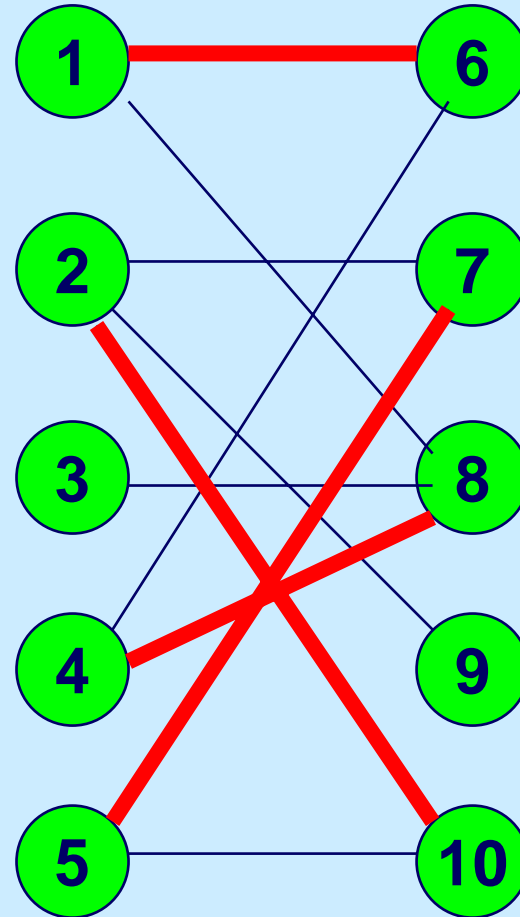
There is no arc directed from S to T .

Matchings

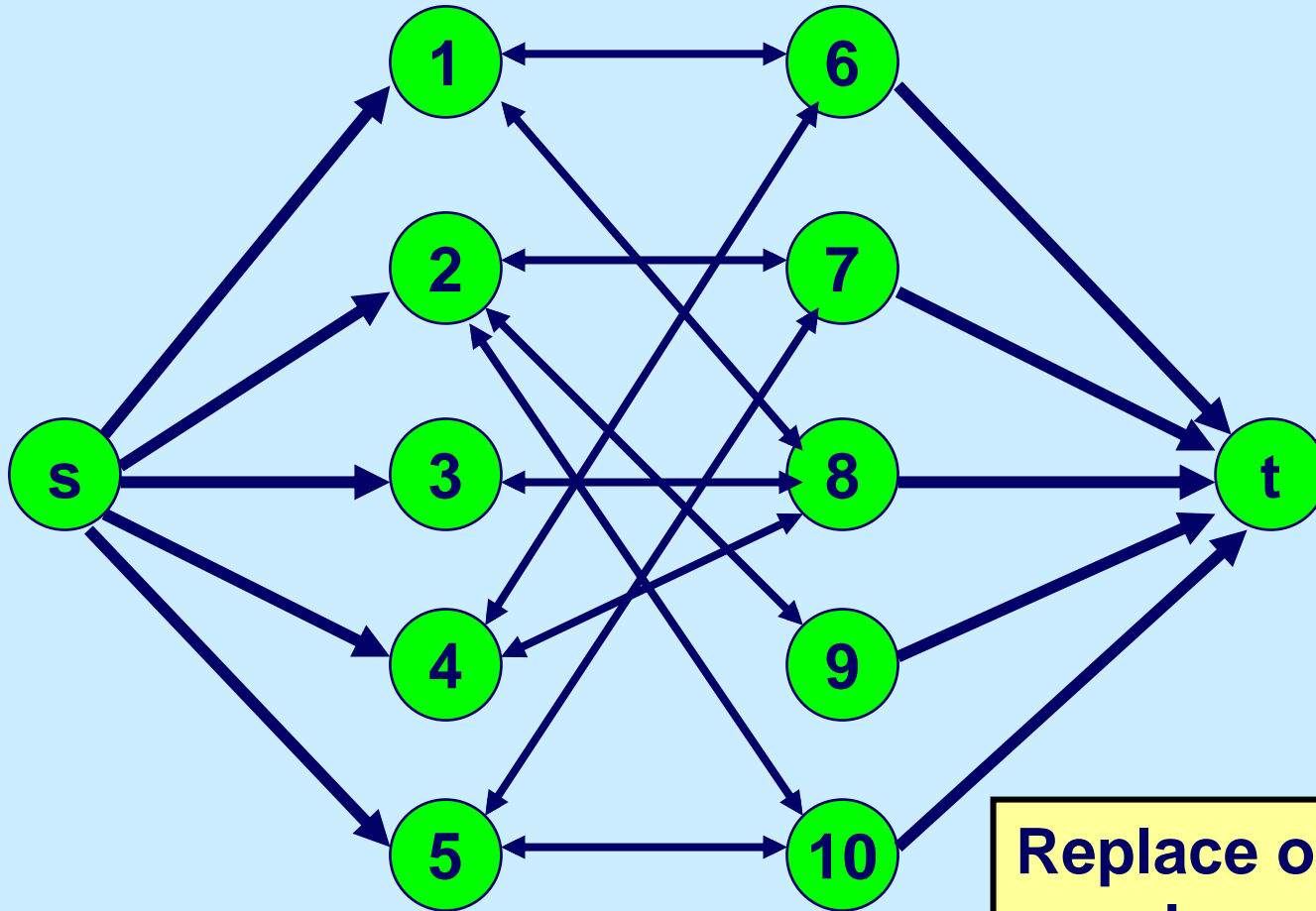
An undirected network $G = (N, A)$ is **bipartite** if N can be partitioned into N_1 and N_2 so that for every arc (i,j) either $i \in N_1$ and $j \in N_2$.

A ***matching*** in N is a set of arcs no two of which are incident to a common node.

Matching Problem: Find a matching of maximum cardinality



Transformation to a Max Flow Problem

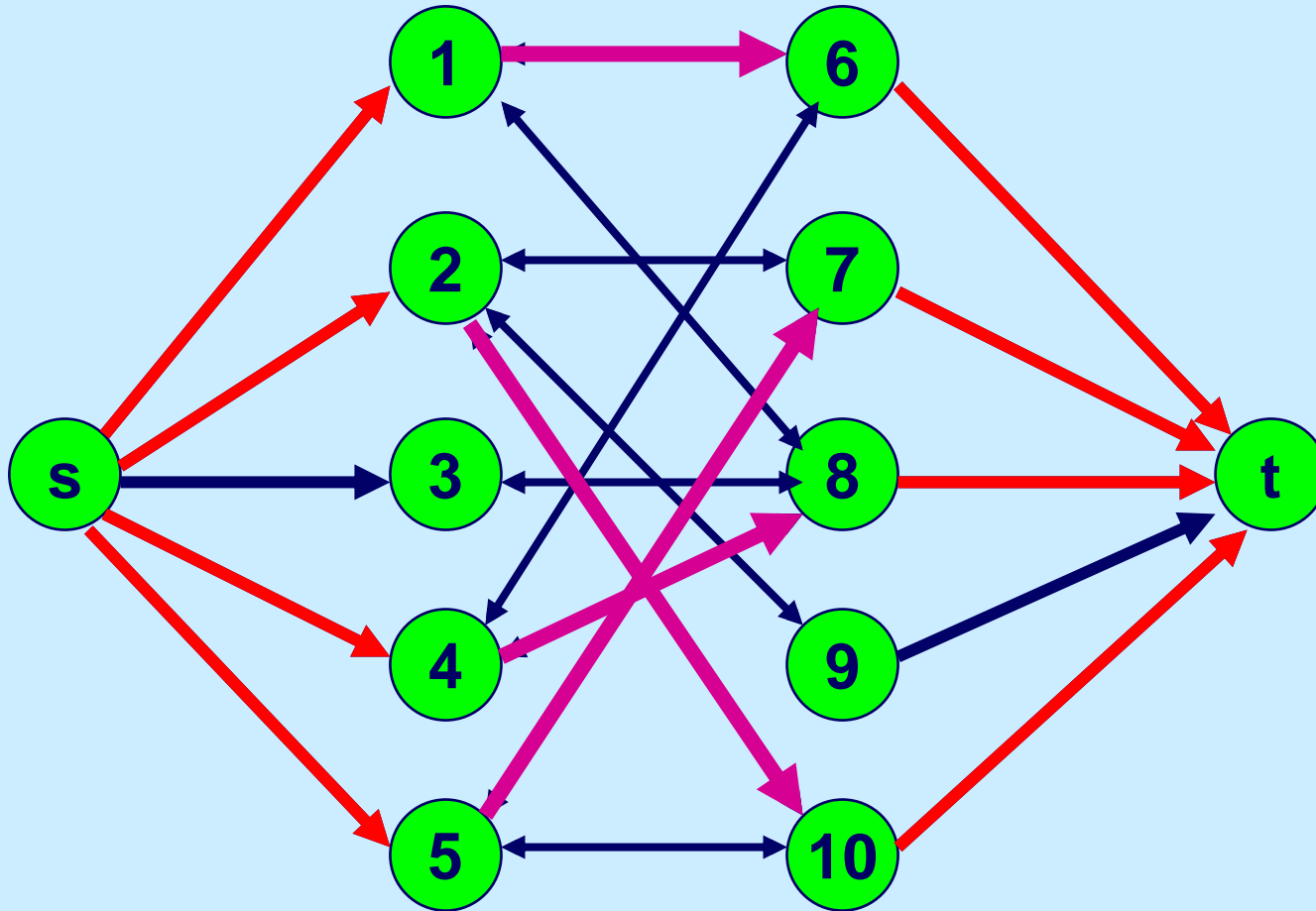


Each arc (s, i) has a capacity of 1.

Each arc (j, t) has a capacity of 1.

Replace original arcs by pairs, and put infinite capacity on original arcs.

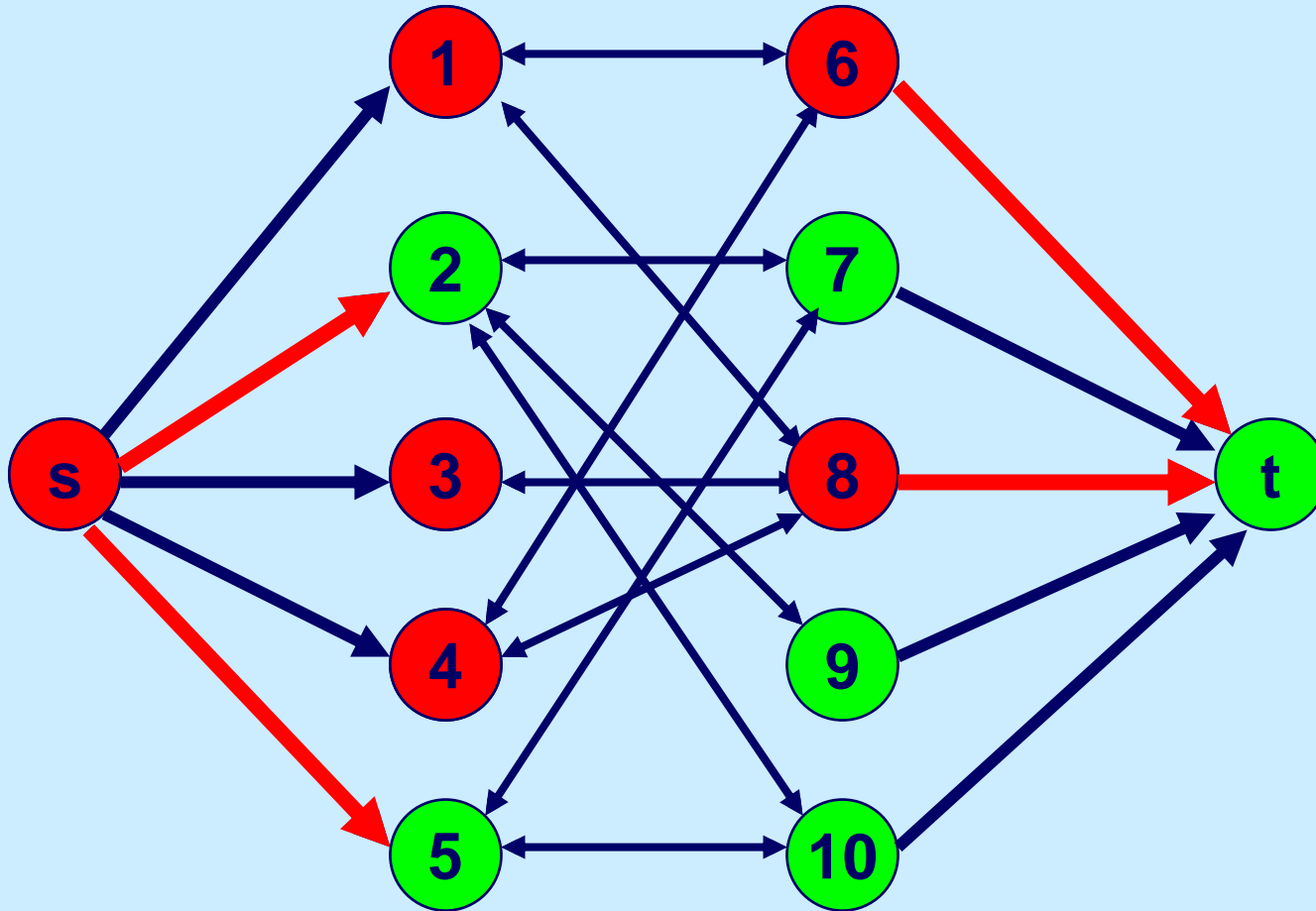
Find a max flow



The maximum s-t flow is 4.

The max matching has cardinality 4.

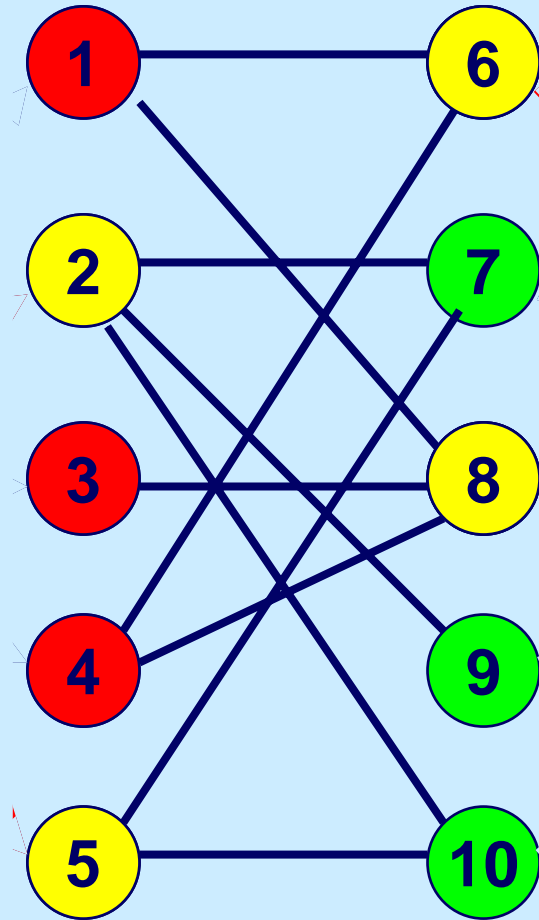
Determine the minimum cut



$S = \{s, 1, 3, 4, 6, 8\}$. $T = \{2, 5, 7, 9, 10, t\}$.

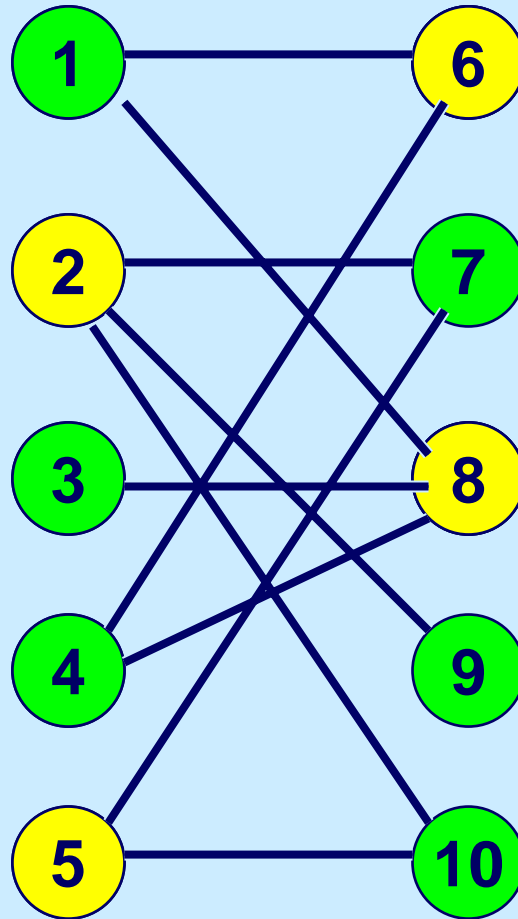
There is no arc from $\{1, 3, 4\}$ to $\{7, 9, 10\}$ or from $\{6, 8\}$ to $\{2, 5\}$. Any such arc would have an infinite capacity.

Interpret the minimum cut



Look at the original nodes incident to the minimum cut.
Every original arc is incident to one of them.

Matching Duality 1



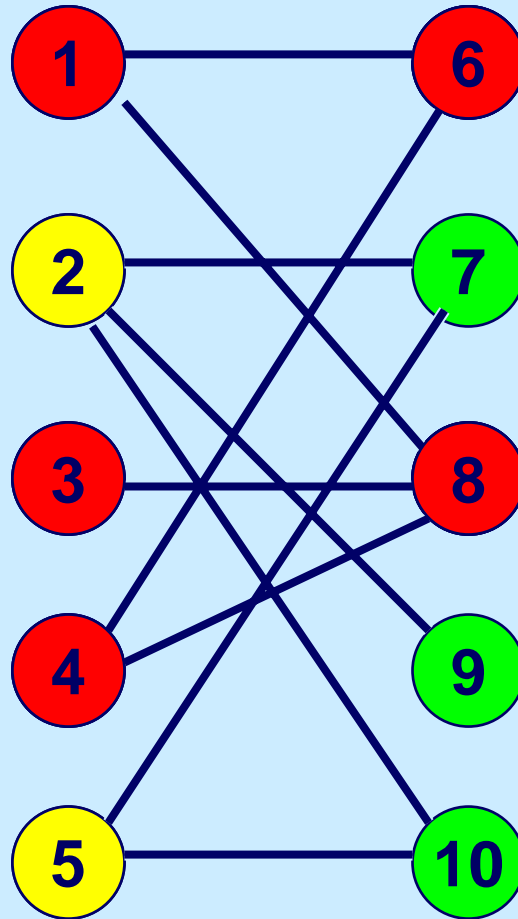
Such a collection of nodes is called a *node cover*

The maximum cardinality of a matching is the minimum number of nodes that “covers” all of the arcs.

Matching Duality 2

Consider a matching problem without a “complete matching.”

Let S = nodes on the left not in the node cover.



Let T = nodes on the right in the node cover.

Claim: $|S| > |T|$ and the nodes in S are adjacent only to nodes in T .

Theorem. *Hall's Theorem.* If there is no perfect matching, then there is a set S of nodes of N_1 such that $|S| > |T|$ where T are the nodes of N_2 adjacent to S .

Review of the Ford-Fulkerson Algorithm

Begin

$x := 0$;

create the residual network $G(x)$;

while there is some directed path from s to t in $G(x)$ **do**

begin

let P be a path from s to t in $G(x)$;

$\delta^* := \delta(P)$;

send δ^* units of flow along P ;

update the r 's;

end

end {the flow x is now maximum}.

Improved Algorithms

The largest augmenting path algorithm:

- **Let P be a path from s to t in $G(x)$ such that δ^* is maximum.**

The shortest augmenting path algorithm:

- **Let P be a path from s to t in $G(x)$ with the fewest number of arcs.**

The Shortest Augmenting Path Algorithm

Overview:

- We will establish the following:
 - We can determine each augmentation in $O(n)$ time if we maintain "distance labels" and can carry out the augmentation in $O(n)$ time.
 - The total time to maintain and update all distance labels is $O(nm)$.
 - The total number of augmentations is $O(nm)$.

Conclusion. The total running time is $O(n^2m)$.

Distance Labels

A *distance label* is a function $d: N \rightarrow Z^+$. A distance label is said to be ***valid*** if it satisfies the following:

$$d(t) = 0.$$

$$d(i) \leq d(j) + 1 \text{ for each } (i,j) \in G(x).$$

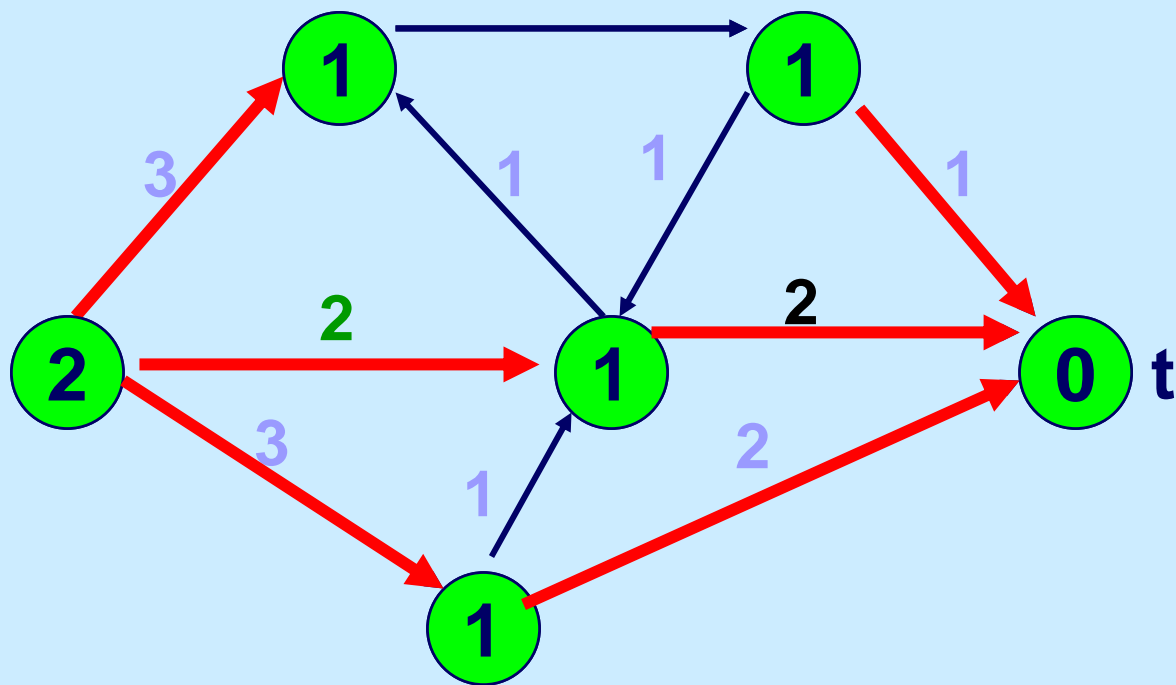
An arc $(i,j) \in G(x)$ is ***admissible*** if $d(i) = d(j) + 1$.

An example of valid distance labels

The distance labels are on the nodes.

All arcs are in the residual network.

The admissible arcs are thick and red.

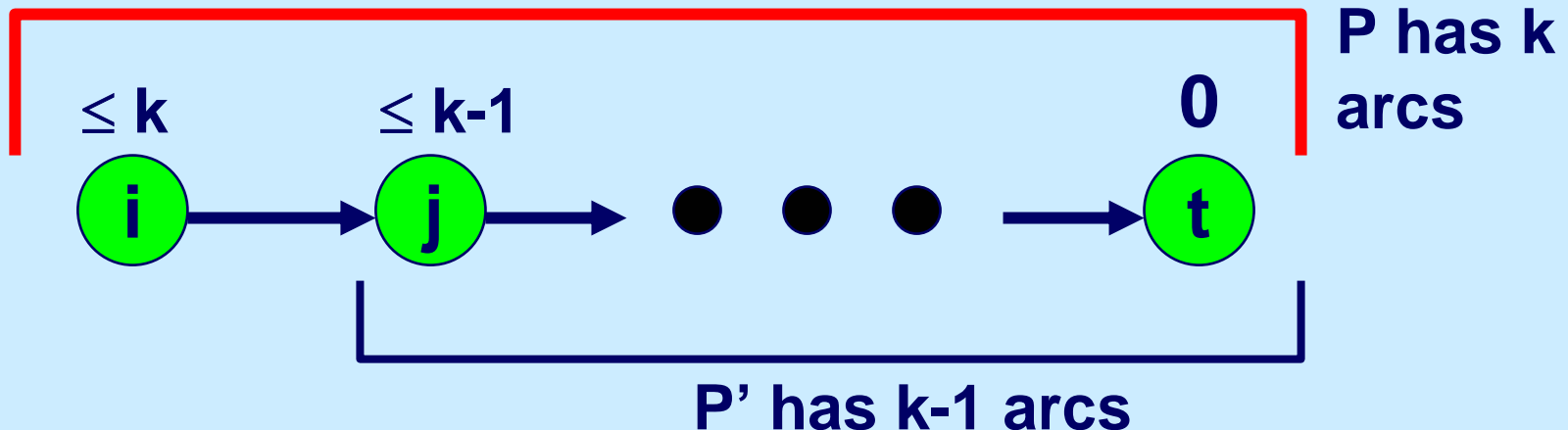


The labels would not be valid if there were an arc from "2" to "0".

More on valid distance labels

Lemma. Let $d(\cdot)$ be a valid distance label. Then $d(i)$ is a lower bound on the distance from i to t in the residual network. (The distance is measured in terms of the number of arcs.)

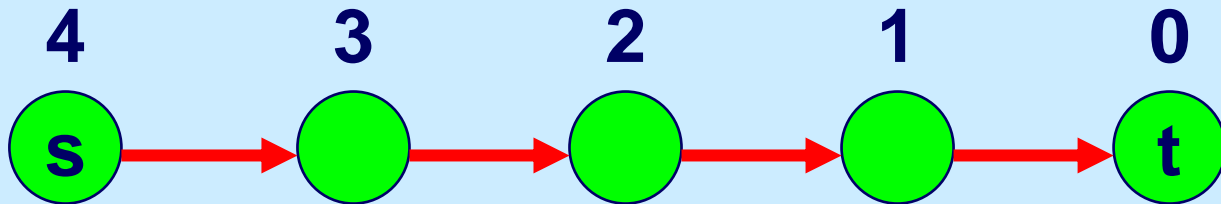
Proof. Let P be any path from i to t in $G(x)$ with k arcs. We claim to show that $d(i) \leq k$. Assume the claim is true for paths of $k-1$ or fewer arcs.



On Finding Paths shortest s-t paths

Lemma. *If there is an admissible path P from s to t , then it is a shortest path.*

Proof. The length of the path is $d(s)$ which is at most the length of the shortest path.



The shortest augmenting path algorithm

```
begin  
  while  $d(s) < n$  do  
    begin  
      if there is a node  $i$  with  $d(i) \leq d(s)$  and no admissible  
        arcs from  $j$  then Relabel( $i$ )  
      else find an admissible path from  $s$  to  $t$  and augment  
        flow along the path  
    end  
  end
```

Procedure Relabel(i)

```
begin  
  if there are no admissible arcs coming out of node  $i$ , then  
     $d(i) := 1 + \min \{ d(j) : r_{ij} > 0 \}$ ;  
  if  $d(s) > n-1$ , then quit;  
end
```

**Shortest augmenting
path animation**

Comments on the run time analysis

- ◆ **Bound the relabels, and the time for relabels**
 - $O(n^2)$ relabels, $O(nm)$ time.
- ◆ **Bound the number of augmentations, and the time to carry out the augmentations**
 - $O(nm)$ augmentations
 - $O(n^2m)$ arcs in augmentations
 - $O(n^2m)$ time.
- ◆ **Bound the time spent looking for augmentations.**
 - $O(n^2m)$ time spent identifying the arcs in augmentations.

Bounding the number of relabels.

Claim: after a relabel of node i , the distances are still valid, and the distance label of node i **strictly increased**.

Claim: Once $d(i) > n-1$, there is no path from node i to the sink node t , and so one can ignore node i subsequently.

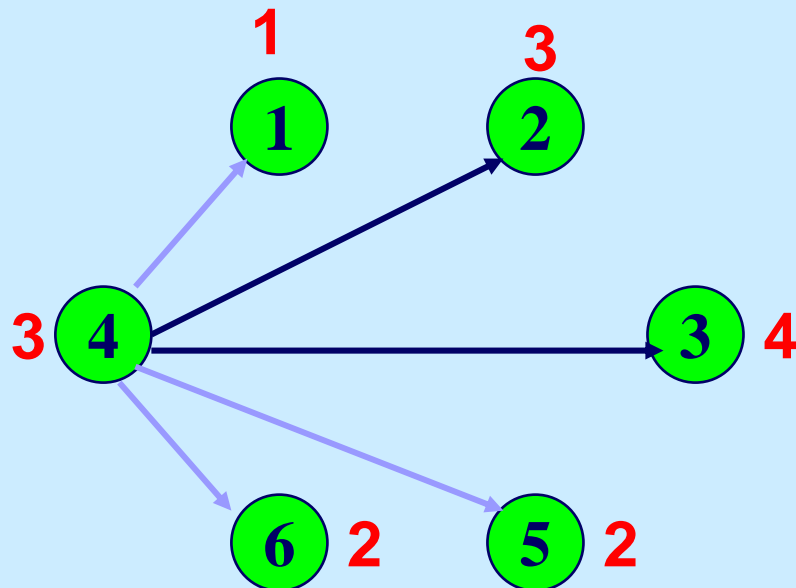
Conclusion: There can be at most n relabels of node i , and at most n^2 relabels in total.

Bounding the time for relabels

Tail	Head	Res. Cap	Admissible ?
4	1	0	No
4	2	1	No
4	3	4	No
4	5	0	No
→ 4	6	0	No

Maintain a current arc for each adjacency list.

Scan through $A(4)$.
 $d(3) := 4$



Each arc in $A(4)$ is scanned once per relabel, at most n times over all relabels.

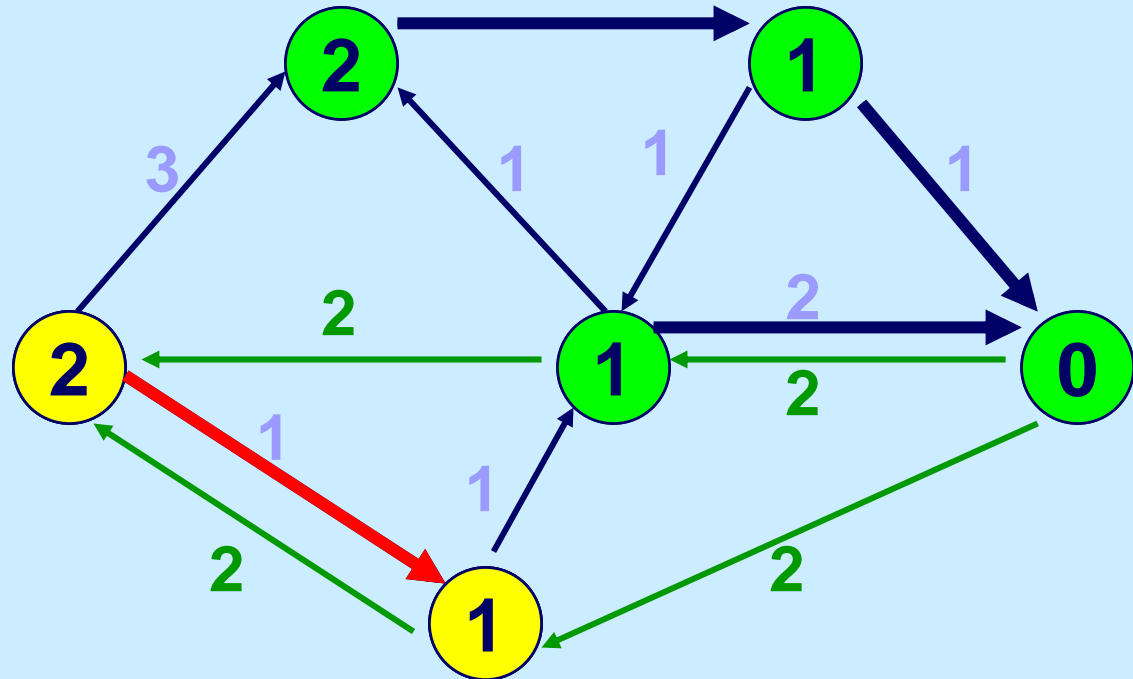
Total time for relabels:
 $O(nm)$.

Bounding the Number of Augmentations

- ◆ If an augmentation uses up the residual capacity of an arc, then the arc is said to be **saturated**.
- ◆ At least one arc is saturated at each augmentation.
- ◆ If arc (i,j) is saturated, then it is not admissible until flow is sent from j to i , and this cannot happen until $d(j)$ increases. (see next slide)
- ◆ **Conclusion:** each arc is saturated at most n times.
- ◆ **Corollary.** There are $O(nm)$ augmentations.
- ◆ The number of arcs in these augmentations is $O(n^2m)$.

Time spent looking for augmentations

We need to find admissible arcs, and know when they do not exist.



Start with s and do a depth first search using admissible arcs.

If there are no admissible arcs from i , then $\text{relabel}(i)$ and reverse along the path leading to i .

Bounding number of arcs in paths

Each arc added to a path either ends up being reversed or ends up in an augmentation.

$O(n^2m)$ arcs in augmentation

$O(n^2)$ arcs in reversals, since a reversal immediately follows a relabel.

$O(n^2m)$ arcs added to paths in total.

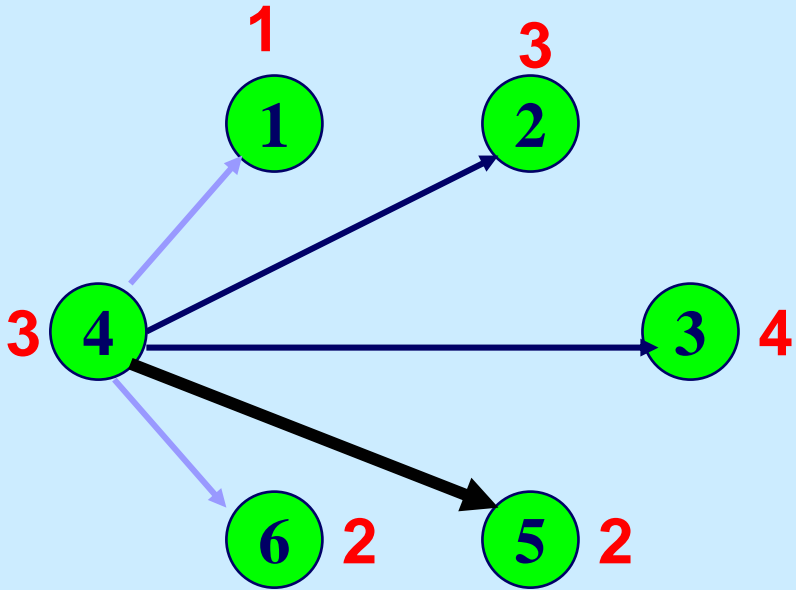
Last step: finding admissible arcs

Tail	Head	Res. Cap	Admissible ?
4	1	0	No
4	2	1	No
4	3	4	No
4	5	2	Yes
4	6	0	No



Scan arcs in $A(4)$ looking for an admissible arc.

key observation: if $(4, j)$ is not admissible, it cannot be admissible again until after node 4 is relabeled.



So, current arc is moved at most $|A(4)|$ times between relabels of node 4.

Summary

Applications of Maximum Flow, including implications of the max flow min cut theorem.

The shortest augmenting path algorithm has $O(nm)$ augmentations, and takes $O(n^2m)$ time.

Use of distance labels to identify how to send flow.

Next lecture: an algorithm that does not rely on augmenting paths.

The Capacity Scaling Algorithm

- ◆ For any fixed value Δ , let $G(x, \Delta)$ be the arcs in $G(x)$ with capacity at least Δ .
- ◆ A flow x is called Δ -maximum if there is no augmenting path of size Δ or more.
- ◆ Subroutine $\text{ImproveApprox}(x, \Delta)$. It takes a flow that is Δ -maximum and outputs a flow that is $\Delta/2$ -maximum.

$\text{ImproveApprox}(x, \Delta)$

begin

$\Delta := \Delta/2$;

 while there is a path from s to t in $G(x, \Delta)$ do

 begin

 find a path P from s to t in $G(x, \Delta)$;

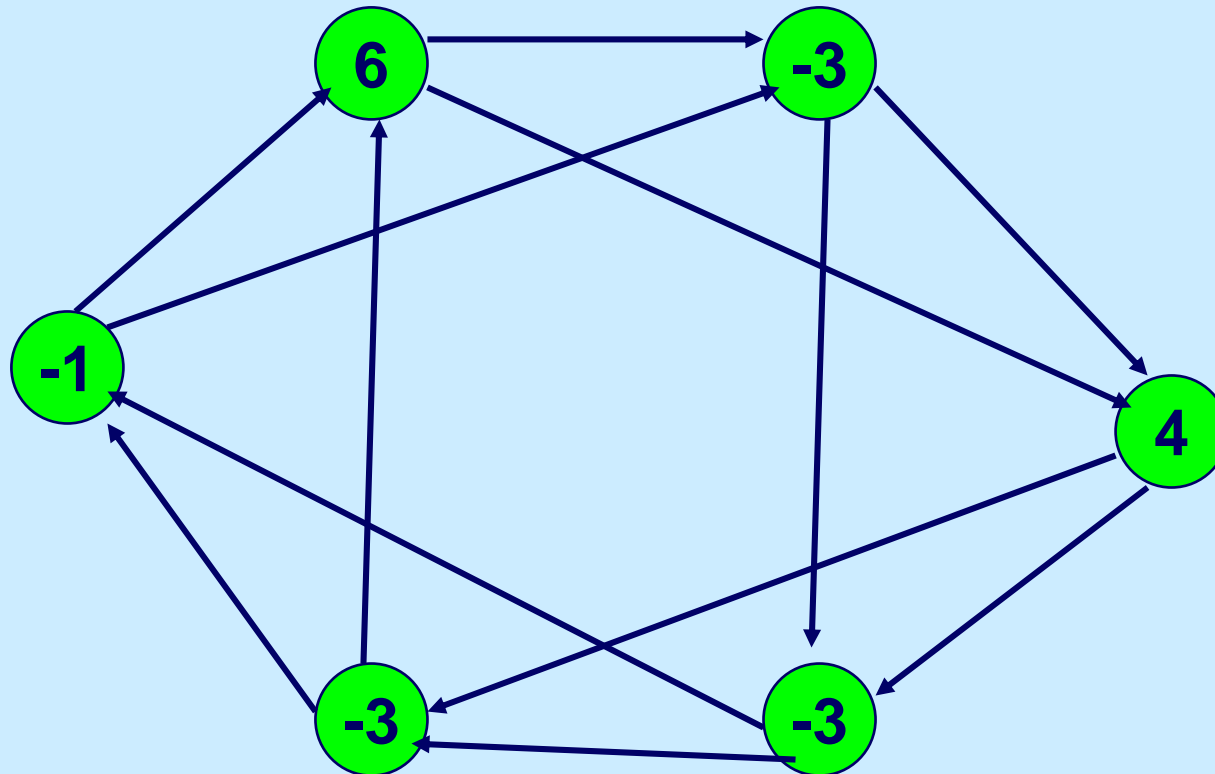
 augment flow along P ;

 update data structures;

 end

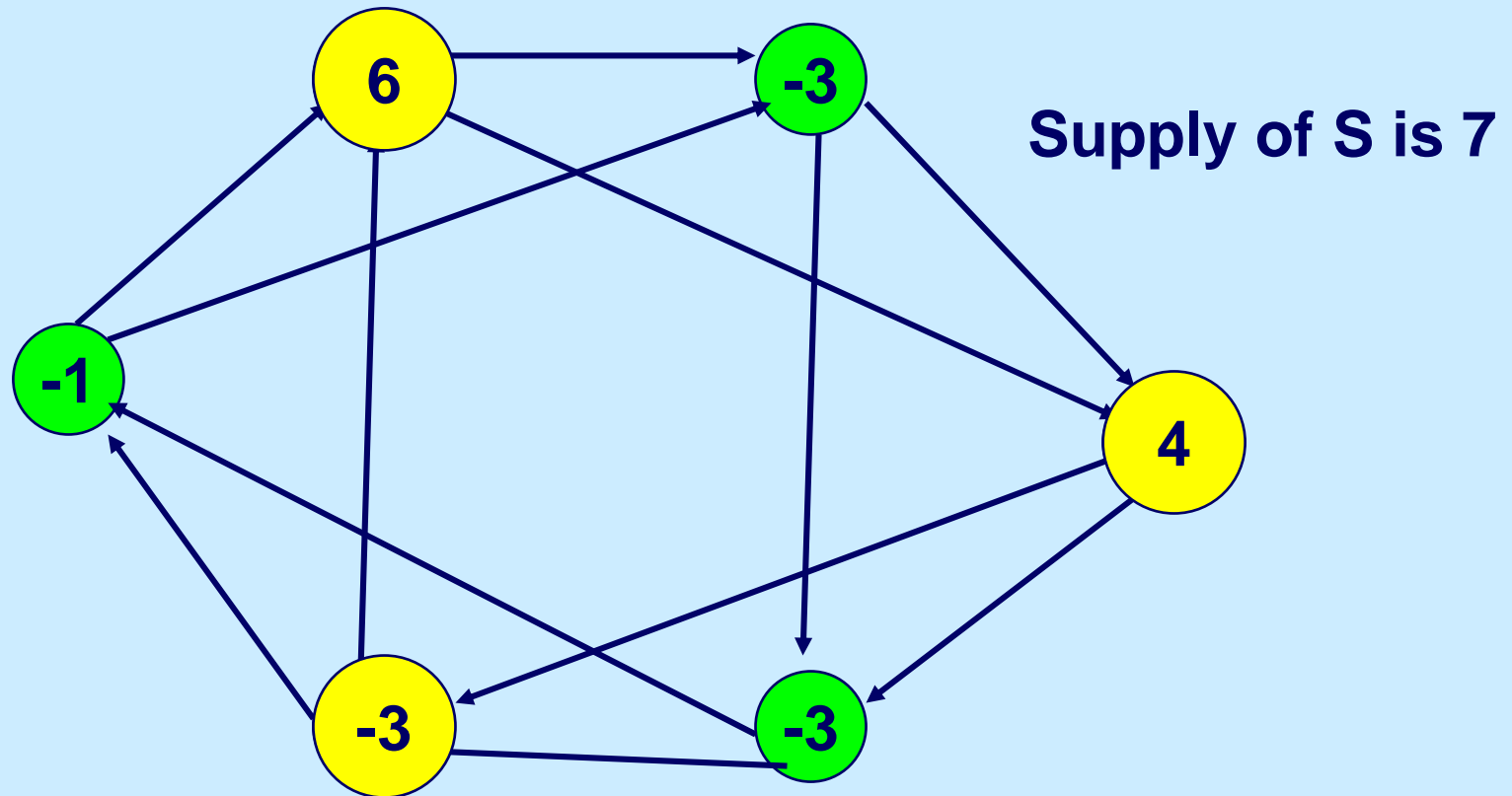
end

When is there a feasible flow?



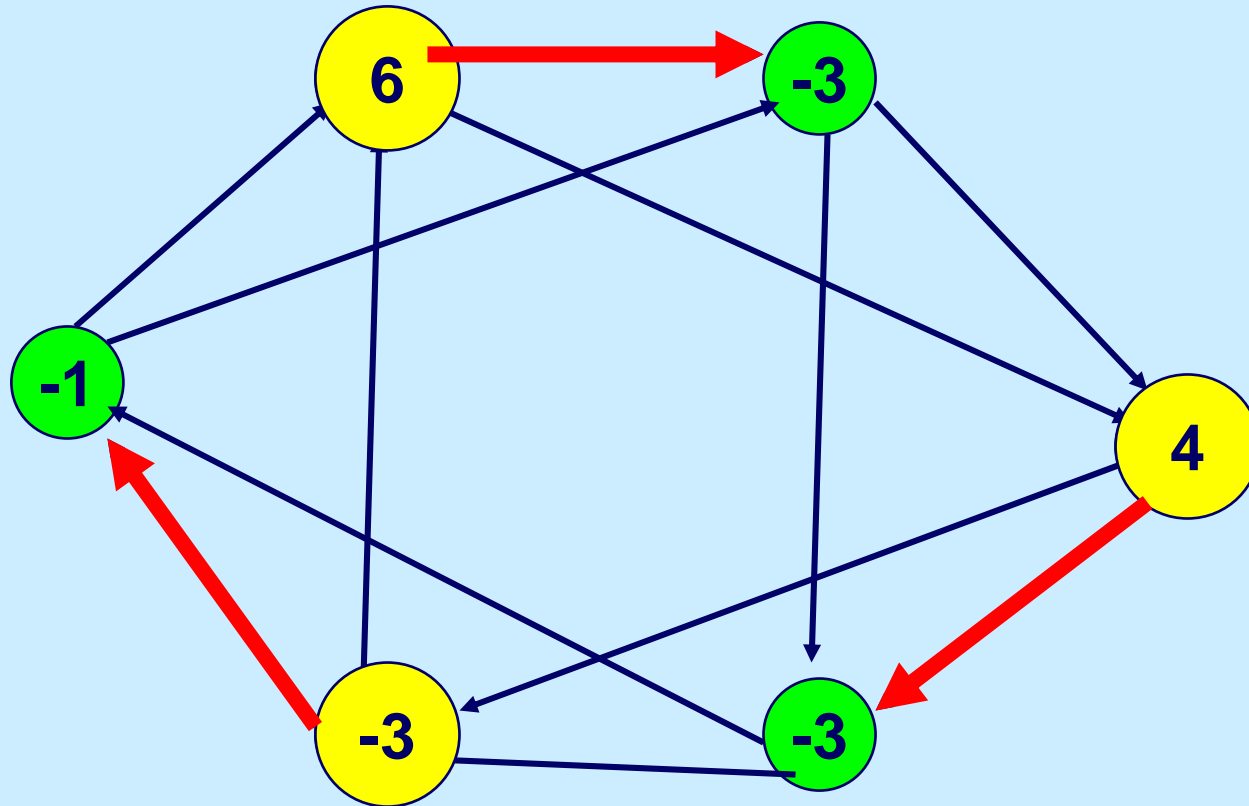
Suppose all arcs have a capacity of 2, and that node numbers are supplies/demands. Then there is no feasible flow.

Infeasibility Theorem



Infeasibility Theorem. *Either there is a feasible flow, or there is a cut (S, T) such that: the capacity of $(S, T) < \text{supply of } S$*

Infeasibility Theorem

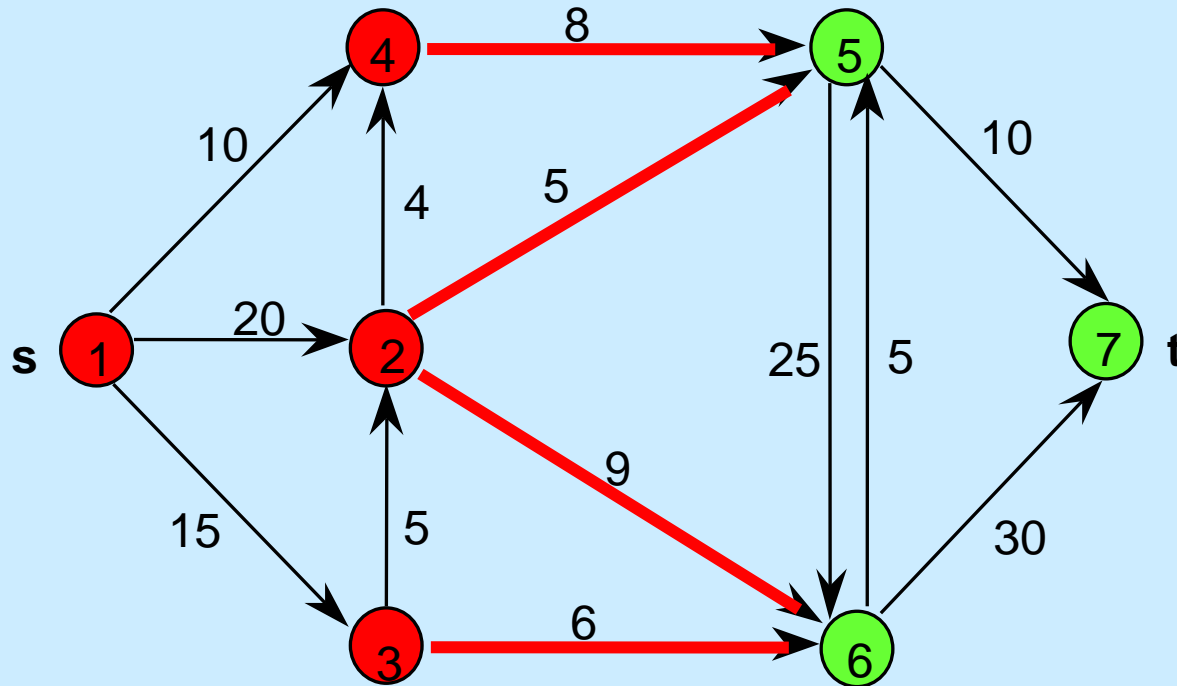


Supply of S is 7.

The capacity of $(S, N \setminus S) = 6$.

More on Capacity Scaling

This is the residual network at the end of the scaling phase when $\Delta = 10$.



How many augmentations can there be from s to t when Δ is reduced from 10 to 5?

Analysis of Capacity Scaling

There are $O(\log U)$ scaling phases.

- ◆ **Initially Δ is at most $2U$.**
- ◆ **Δ is halved at each scaling phase**
- ◆ **We can stop when Δ is 1.**

The running time per scaling phase is $O(m^2)$.

- ◆ **Each scaling phase has $O(m)$ augmentations.**
- ◆ **The time per augmentation is $O(m)$.**

The total running time is $O(m^2 \log U)$