Network Design and Optimization course

Lecture 5

Alberto Ceselli
alberto.ceselli@unimi.it

Dipartimento di Tecnologie dell'Informazione
Università degli Studi di Milano

November 7, 2011

Università degli Studi di Milano

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 $\delta^*$ units of flow along P;
        update the r's;
    end
end {the flow x is now maximum}.
```

Limits of Ford-Fulkerson

- Computational complexity of $O(nmU)$;
- bad behaviour also on simple instances;
- may not converge to optimal solutions when data is irrational.

How to improve Ford-Fulkerson

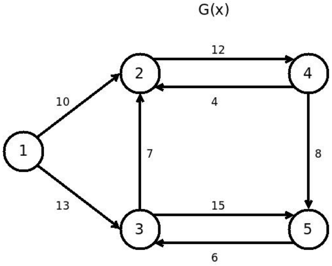
Some ways to improve them:

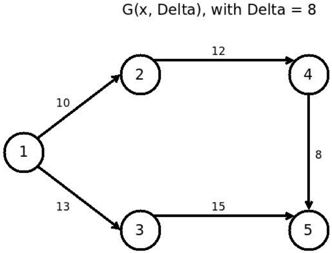
- augmenting in *large* increments of flow (capacity scaling algorithms),
- using a combinatorial strategy to choose augmenting paths (shortest augmenting path algorithms),
- relax flow conservation constraints in intermediate steps of the algorithm (preflow-push algorithms).

Preflow-push yield $O(nm + n^2 \log U)$ time complexity.

Δ-Residual network

Given a flow x and a parameter Δ, let us define as Δ-residual network $G(x, \Delta)$ the *subgraph* obtained from the residual network by *removing arcs of residual capacity less than* Δ.





Capacity scaling algorithm

```
BEGIN
$x := 0$
$\Delta := 2^{\lfloor \log U \rfloor}$
while $\Delta \geq 1$ do
begin
    while $G(x, \Delta)$ contains a path from $s$ to $t$ do
    begin
        identify a path $P$ in $G(x, \Delta)$
        $\delta := \min\{r_{ij} : (i, j) \in P\}$
        augment $\delta$ units of flow along $P$ and update $G(x, \Delta)$
    end
    $\Delta := \Delta/2$
end
END
```

Complexity of capacity scaling

We can prove that capacity scaling solves the maximum flow problem

- within $O(m \log U)$ augmentations,
- using $O(m^2 \log U)$ time overall.

Basic observations

Observation 1: $\Delta := 2^{\lfloor \log U \rfloor}$ in the beginning, and halves at every iteration (scaling phase), $\Rightarrow$ at most $\log U$ scaling phases are performed.
Observation 2: when $\Delta == 1$ (i.e. during the last iteration) $G(x, \Delta) == G(x)$, $\Rightarrow$ capacity scaling outputs an optimal flow.

More observations

Observation 3: capacity scaling performs at most $2m$ augmentations per scaling phase.
Proof:

- Let x' be the flow after scaling phase with $\Delta = \widetilde{\Delta}$, and let v' be its value,
- let S be the set of nodes reachable from s in $G(x', \widetilde{\Delta})$,

More observations

Observation 3: capacity scaling performs at most $2m$ augmentations per scaling phase.
Proof:

- Let x' be the flow after scaling phase with $\Delta = \widetilde{\Delta}$, and let v' be its value,
- let S be the set of nodes reachable from s in $G(x', \widetilde{\Delta})$,
- then $[S, \bar{S}]$ forms an s-t cut

More observations

Observation 3: capacity scaling performs at most $2m$ augmentations per scaling phase.
Proof:

- Let x' be the flow after scaling phase with $\Delta = \widetilde{\Delta}$, and let v' be its value,
- let S be the set of nodes reachable from s in $G(x', \widetilde{\Delta})$,
- then $[S, \bar{S}]$ forms an s-t cut
- $\rightarrow$ the residual capacity of each arc in the cut is $< \widetilde{\Delta}$,

More observations

Observation 3: capacity scaling performs at most $2m$ augmentations per scaling phase.
Proof:

- Let x' be the flow after scaling phase with $\Delta = \widetilde{\Delta}$, and let v' be its value,
- let S be the set of nodes reachable from s in $G(x', \widetilde{\Delta})$,
- then $[S, \bar{S}]$ forms an s-t cut
- $\rightarrow$ the residual capacity of each arc in the cut is $< \widetilde{\Delta}$,
- $\rightarrow$ the residual capacity of the cut is $\leq m\widetilde{\Delta}$

More observations

Observation 3: capacity scaling performs at most $2m$ augmentations per scaling phase.

Proof:

- Let x' be the flow after scaling phase with $\Delta = \widetilde{\Delta}$, and let v' be its value,
- let S be the set of nodes reachable from s in $G(x', \widetilde{\Delta})$,
- then $[S, \bar{S}]$ forms an s-t cut
- $\rightarrow$ the residual capacity of each arc in the cut is $< \widetilde{\Delta}$,
- $\rightarrow$ the residual capacity of the cut is $\leq m\widetilde{\Delta}$
- $\rightarrow$ $v^* \leq v' + m\widetilde{\Delta}$ (weak duality)

More observations

Observation 3: capacity scaling performs at most $2m$ augmentations per scaling phase.
Proof:

- Let x' be the flow after scaling phase with $\Delta = \widetilde{\Delta}$, and let v' be its value,
- let S be the set of nodes reachable from s in $G(x', \widetilde{\Delta})$,
- then $[S, \bar{S}]$ forms an s-t cut
- $\rightarrow$ the residual capacity of each arc in the cut is $< \widetilde{\Delta}$,
- $\rightarrow$ the residual capacity of the cut is $\leq m\widetilde{\Delta}$
- $\rightarrow$ $v^* \leq v' + m\widetilde{\Delta}$ (weak duality)
- $\rightarrow$ at most $m\widetilde{\Delta}$ units of flow remain to be carried in the next scaling phases.

More observations

Observation 3: capacity scaling performs at most $2m$ augmentations per scaling phase.
Proof:

- Let x' be the flow after scaling phase with $\Delta = \widetilde{\Delta}$, and let v' be its value,
- let S be the set of nodes reachable from s in $G(x', \widetilde{\Delta})$,
- then $[S, \bar{S}]$ forms an s-t cut
- $\rightarrow$ the residual capacity of each arc in the cut is $< \widetilde{\Delta}$,
- $\rightarrow$ the residual capacity of the cut is $\leq m\widetilde{\Delta}$
- $\rightarrow$ $v^* \leq v' + m\widetilde{\Delta}$ (weak duality)
- $\rightarrow$ at most $m\widetilde{\Delta}$ units of flow remain to be carried in the next scaling phases.
- each augmentation in the next scaling phase carries at least $\widetilde{\Delta}/2$ units of flow

More observations

Observation 3: capacity scaling performs at most $2m$ augmentations per scaling phase.
Proof:

- Let x' be the flow after scaling phase with $\Delta = \widetilde{\Delta}$, and let v' be its value,
- let S be the set of nodes reachable from s in $G(x', \widetilde{\Delta})$,
- then $[S, \bar{S}]$ forms an s-t cut
- $\rightarrow$ the residual capacity of each arc in the cut is $< \widetilde{\Delta}$,
- $\rightarrow$ the residual capacity of the cut is $\leq m\widetilde{\Delta}$
- $\rightarrow v^* \leq v' + m\widetilde{\Delta}$ (weak duality)
- $\rightarrow$ at most $m\widetilde{\Delta}$ units of flow remain to be carried in the next scaling phases.
- each augmentation in the next scaling phase carries at least $\widetilde{\Delta}/2$ units of flow
- $\rightarrow$ at most $2m$ augmentations can be performed

More observations

Observation 4: each scaling phase requires at most $O(m)$ time

- $O(m)$ time to find an augmenting path (depth first search),
- $O(m)$ time to update the residual network.

So, overall $2m\lfloor \log U \rfloor$ iterations, each of cost $O(m)$
$\rightarrow O(m^2\lfloor \log U \rfloor)$.

Shortest augmenting path algorithm

See Orlin's slides (cut and paste ahead!)

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:

$d(t) = 0$.

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

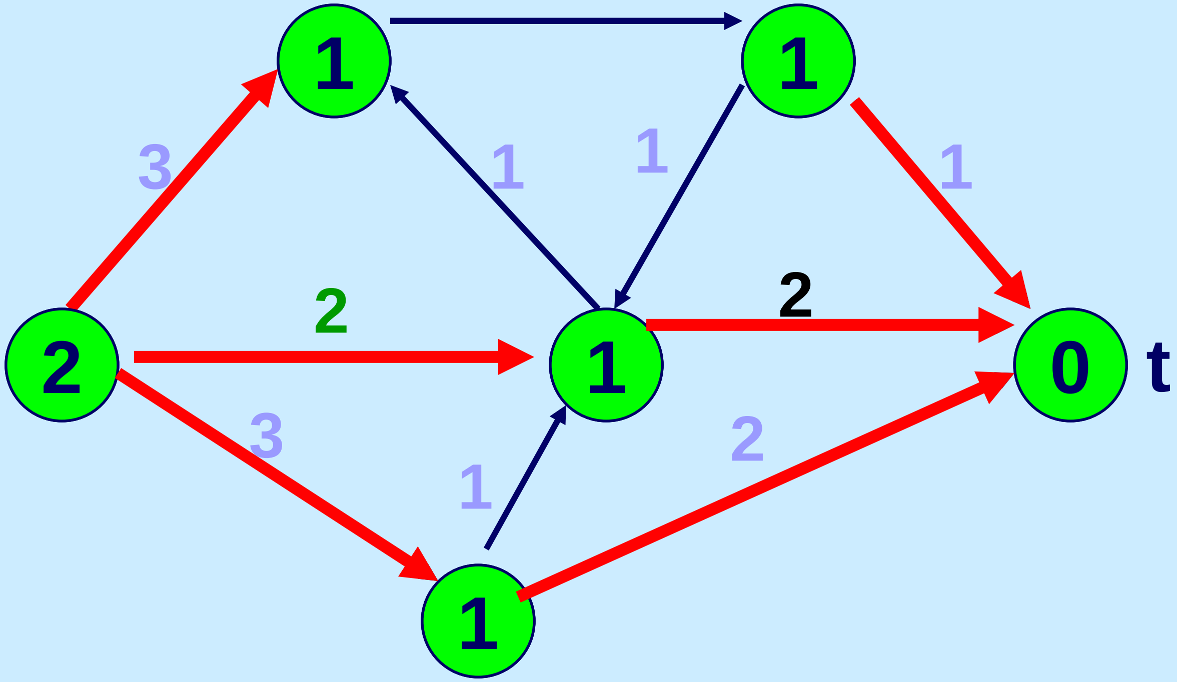
An arc $(i,j) \in G(x)$ is ***admissible*** if $d(i) = d(j) + 1$.

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.

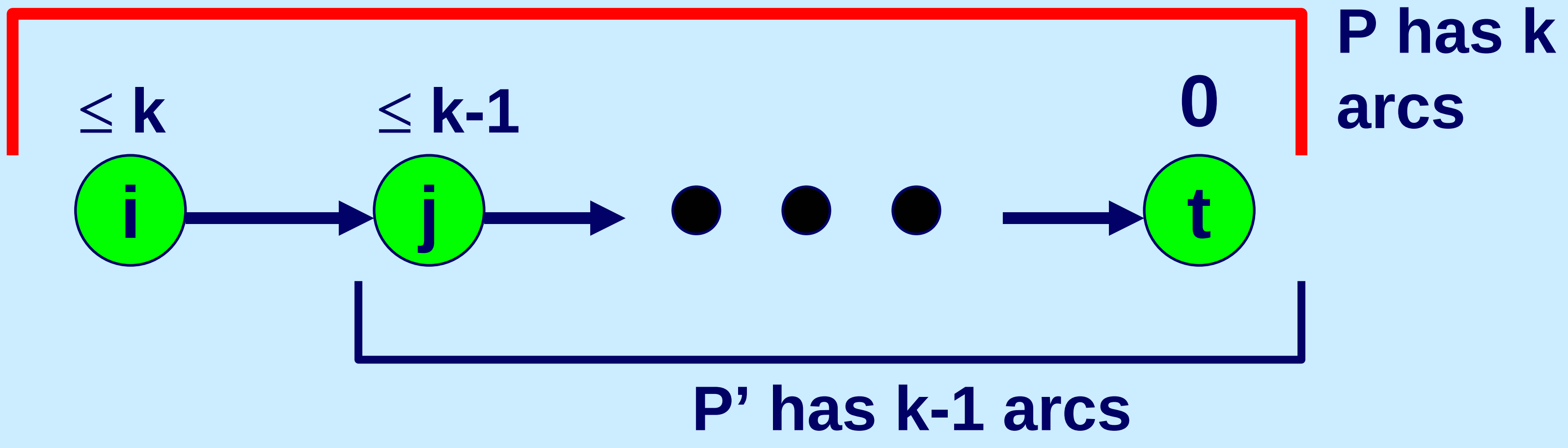


The labels would not be valid if there were an arc from “2” to “0”.

More on valid distance labels

Lemma. *Let d( ) 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.)*

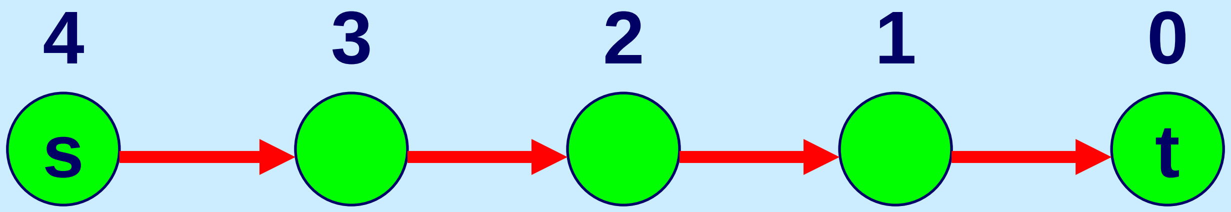
Proof. Let P be any path from i to t in G(x) with k arcs. We claim to show that d(i) ≤ k. Assume the claim is true for paths of k-1 or fewer arcs.



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  with d(i) ≤ 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.

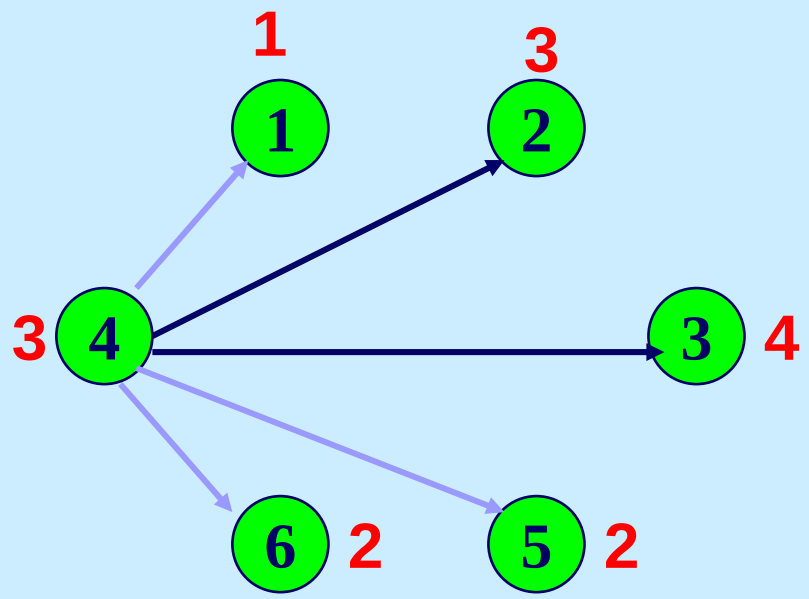
Bounding the time for relabels

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:

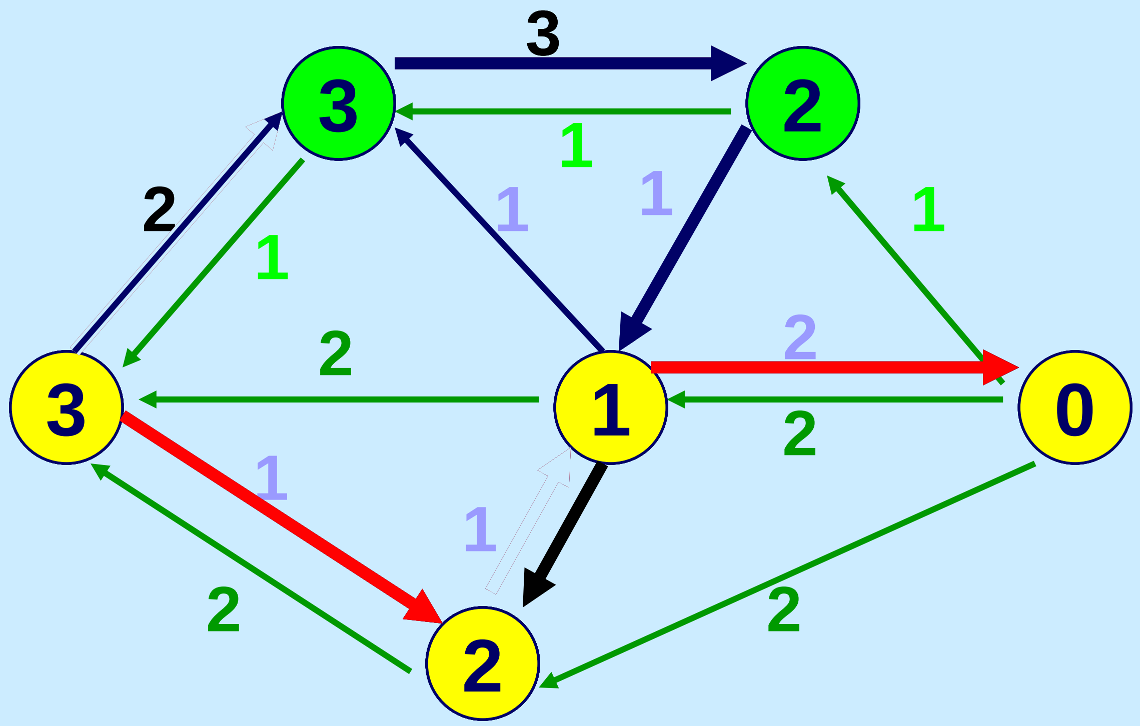
O(nm).

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 $O(n^2m)$.

(2,1) is saturated in the augmentation



After the saturation, arc (2,1) is deleted from G(x).

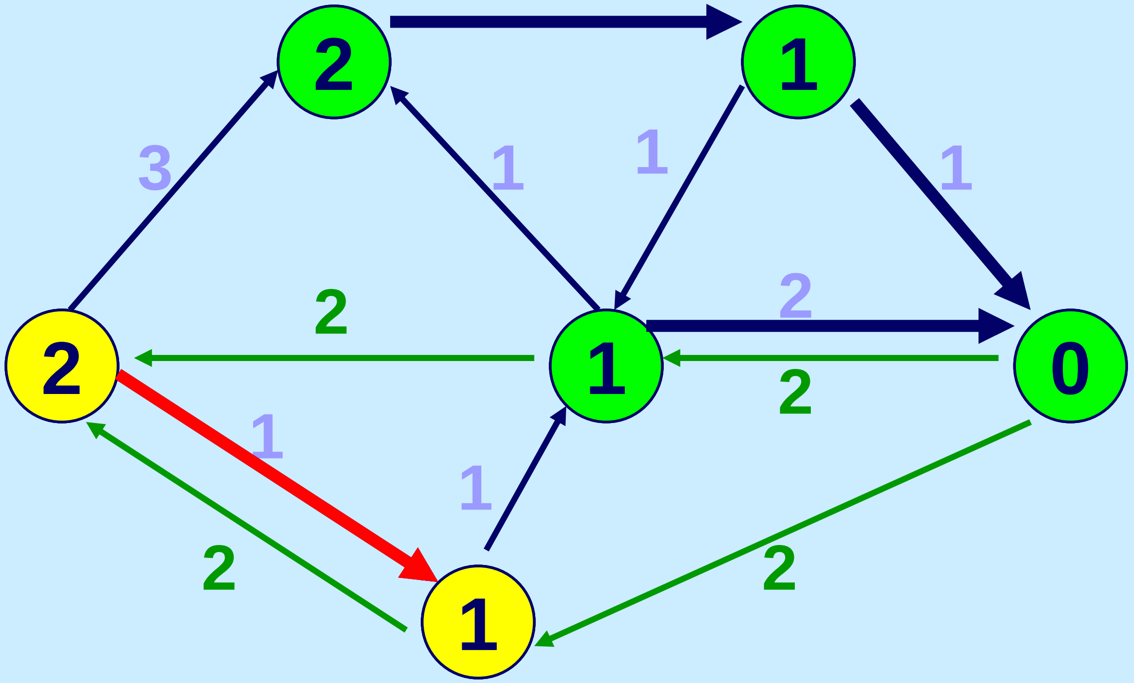
It doesn't get added until there is flow in (1,2)

But for that, the distance label must increase from 1 to 3.

And to send flow back, the distance label must increase from 2 to 4.

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 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.

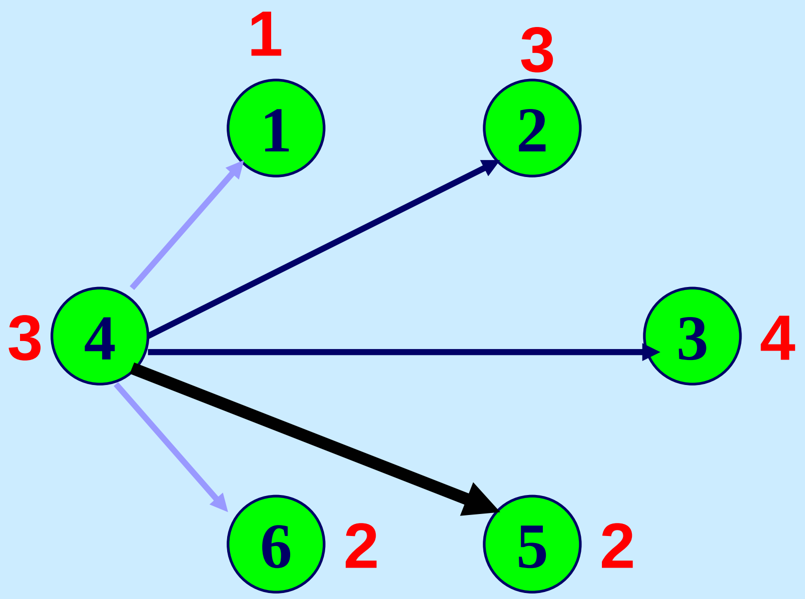
$O(n^2m)$ arcs in augmentation

$O(n^2)$ 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.

Network reliability

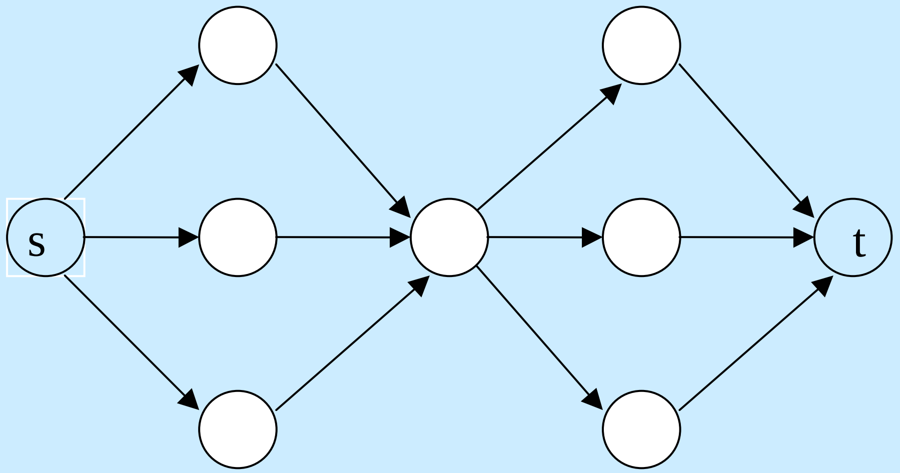
Given a network and two clients of this network

- what is the number of *link-disjoint paths* between these two clients?
- how can we find them?

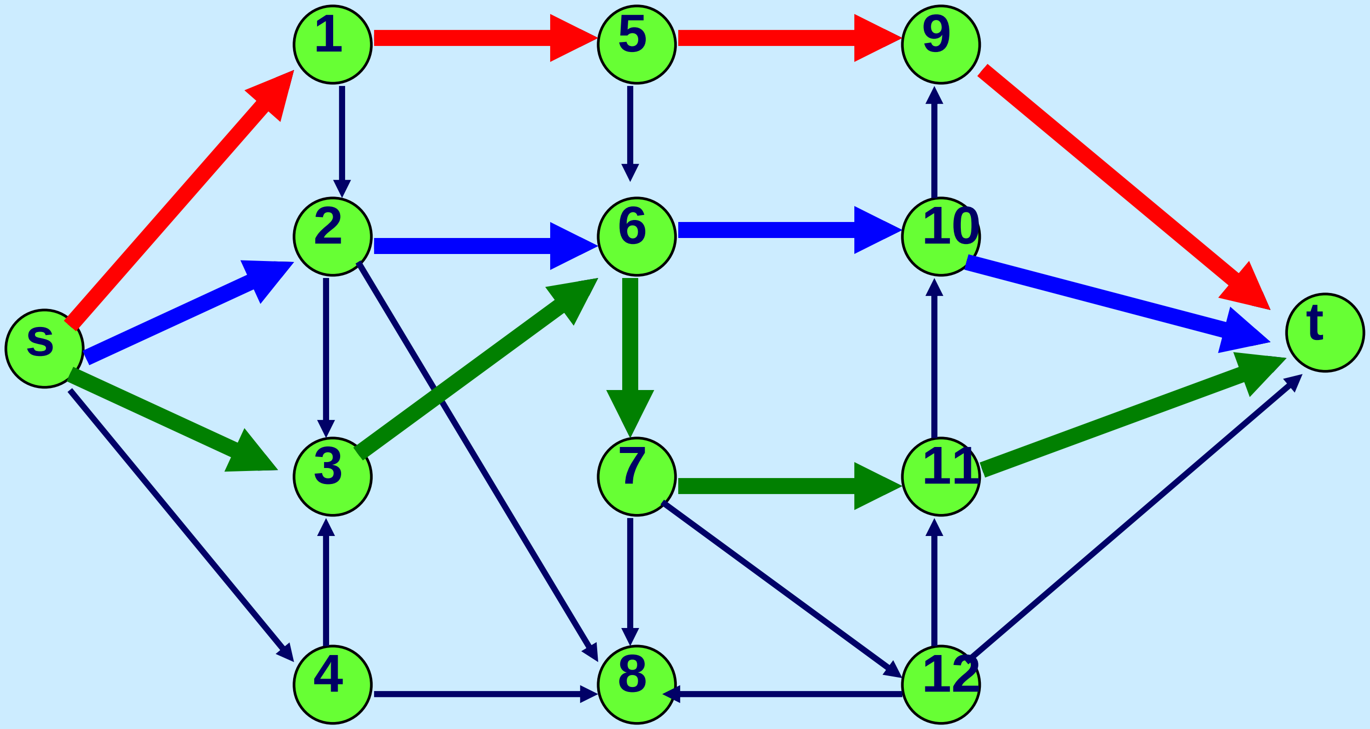
Network Reliability

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

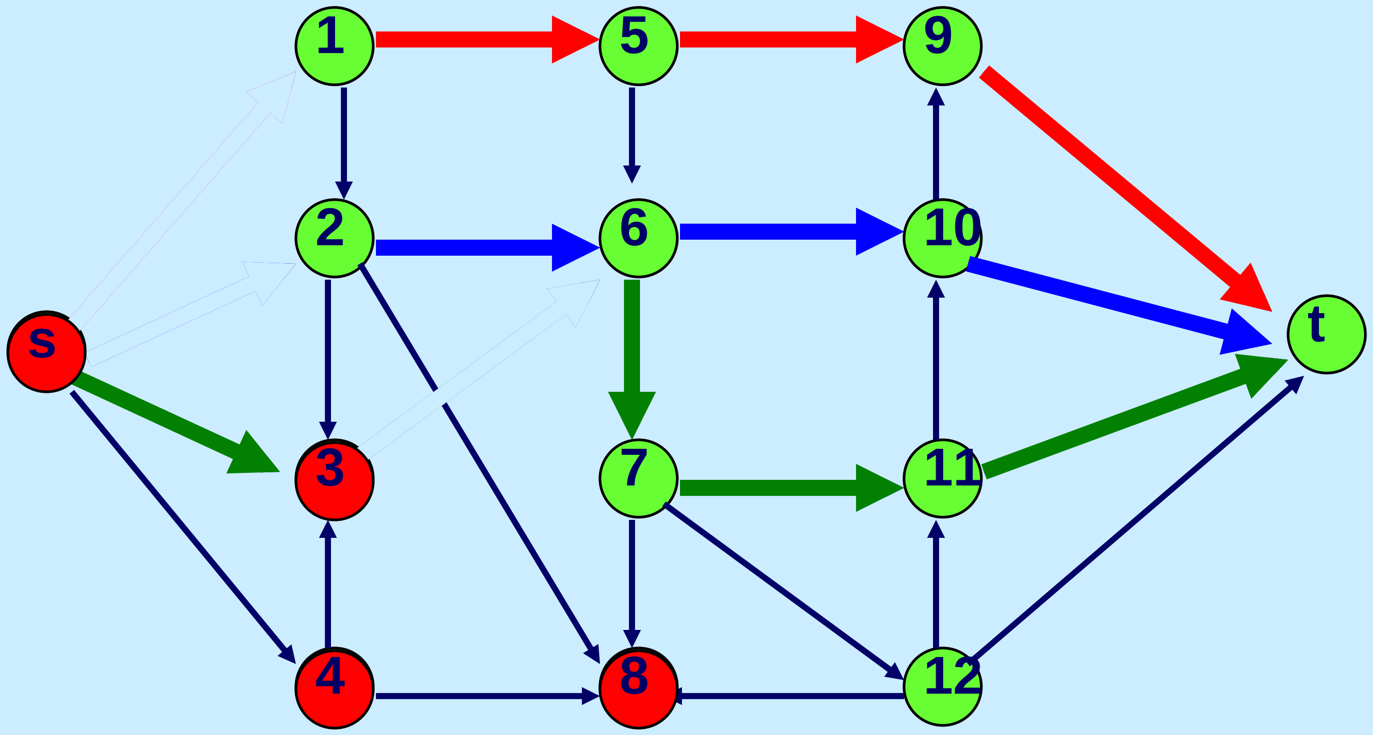
Theorem. *Let G = (N,A) be a directed graph. Then the maximum number of arc-disjoint paths from s to t is equal to the minimum number of arcs upon whose deletion there is no directed s-t path.*



There are 3 arc-disjoint s-t paths



Deleting 3 arcs disconnects s and t



Let S = {s, 3, 4, 8}.   The only arcs from S to T = N\S are the 3 deleted arcs.

Network reliability

Given a network and two clients of this network

- what is the number of *node-disjoint paths* between these two clients?
- how can we find them?

Maximum matching

Given

- a set of computers and a set of tasks,
- for each computer, the subset of tasks that can be performed on it
- each computer can perform at most 1 task at a time
- each task can be performed at most by 1 computer at a time
- find the allocation of tasks to computers that maximizes the number of tasks performed.

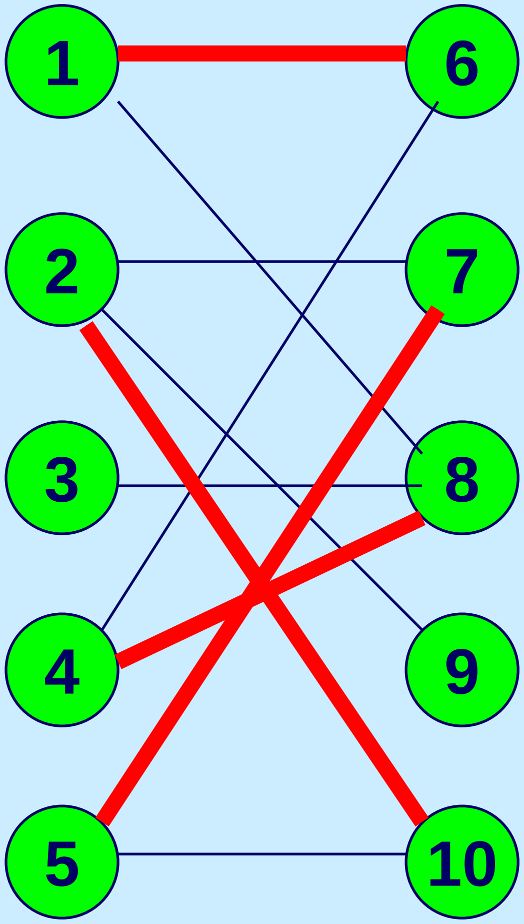
Università degli Studi di Milano

Matchings

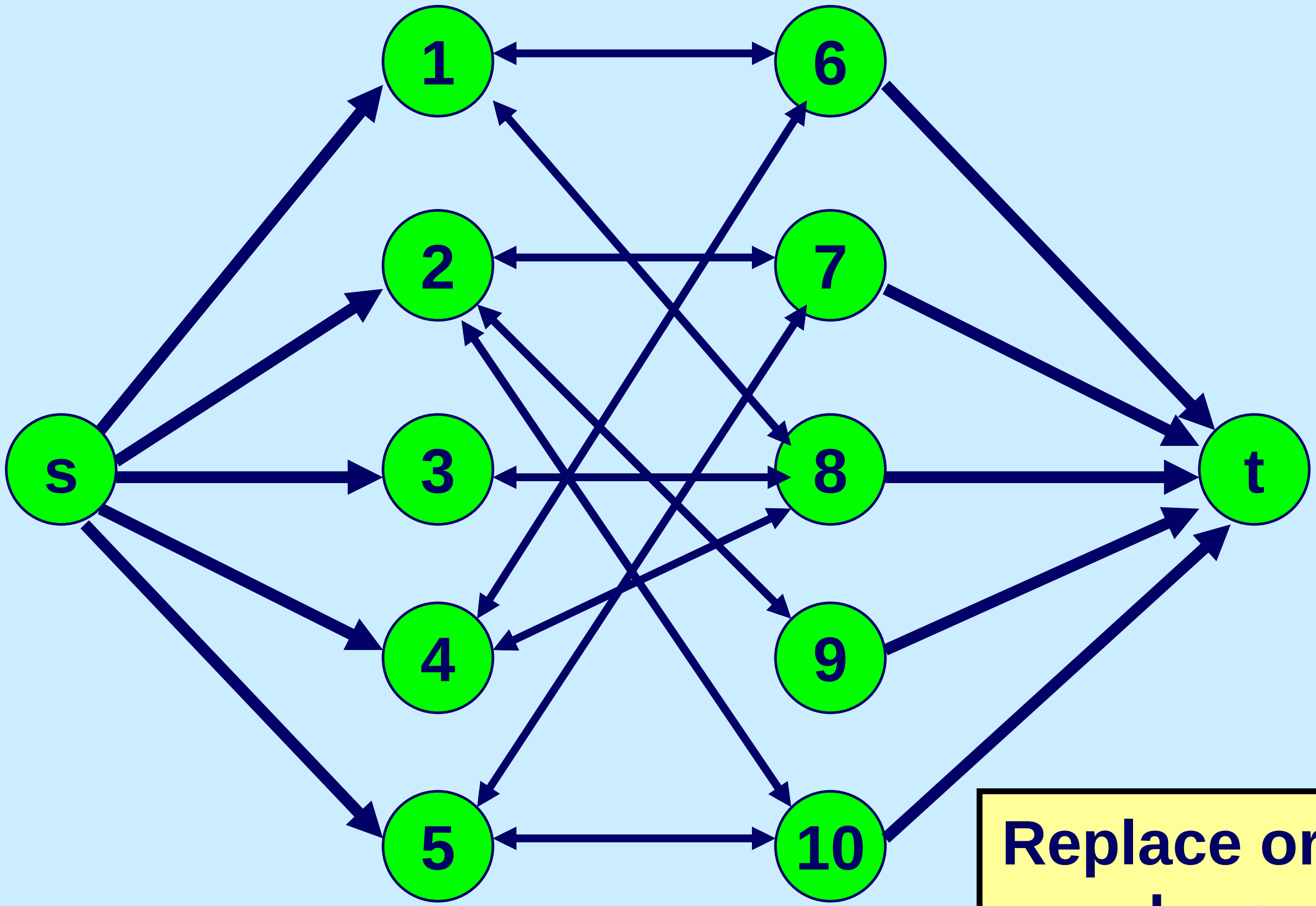
An undirected network G = (N, A) is ***bipartite*** if N can be partitioned into N_1 and N_2 so that for every arc (i,j) either i $\in$ N_1 and j $\in$ N_2.

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

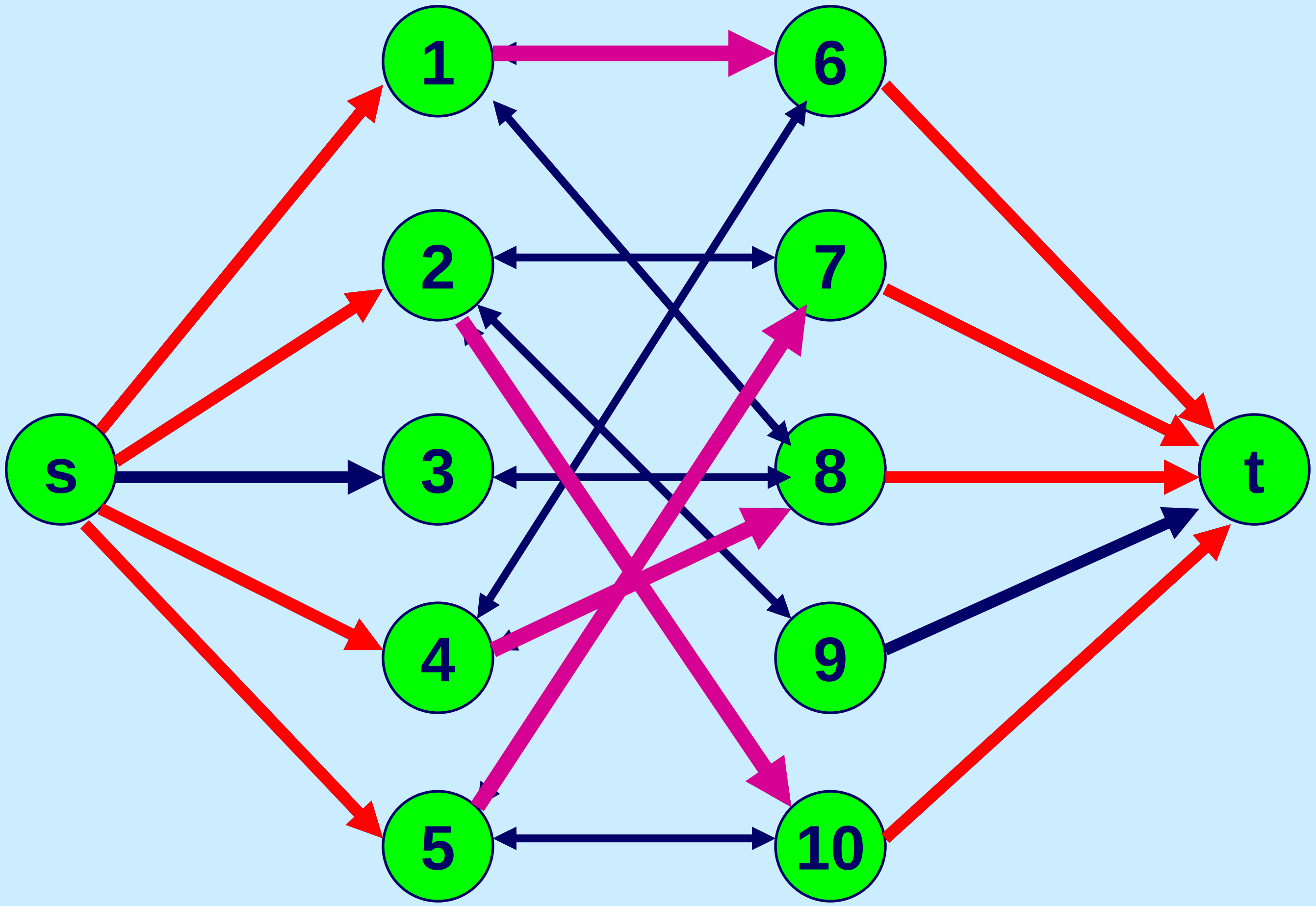


Replace original arcs by pairs, and put infinite capacity on original arcs.

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

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

Find a max flow



The maximum s-t flow is 4.

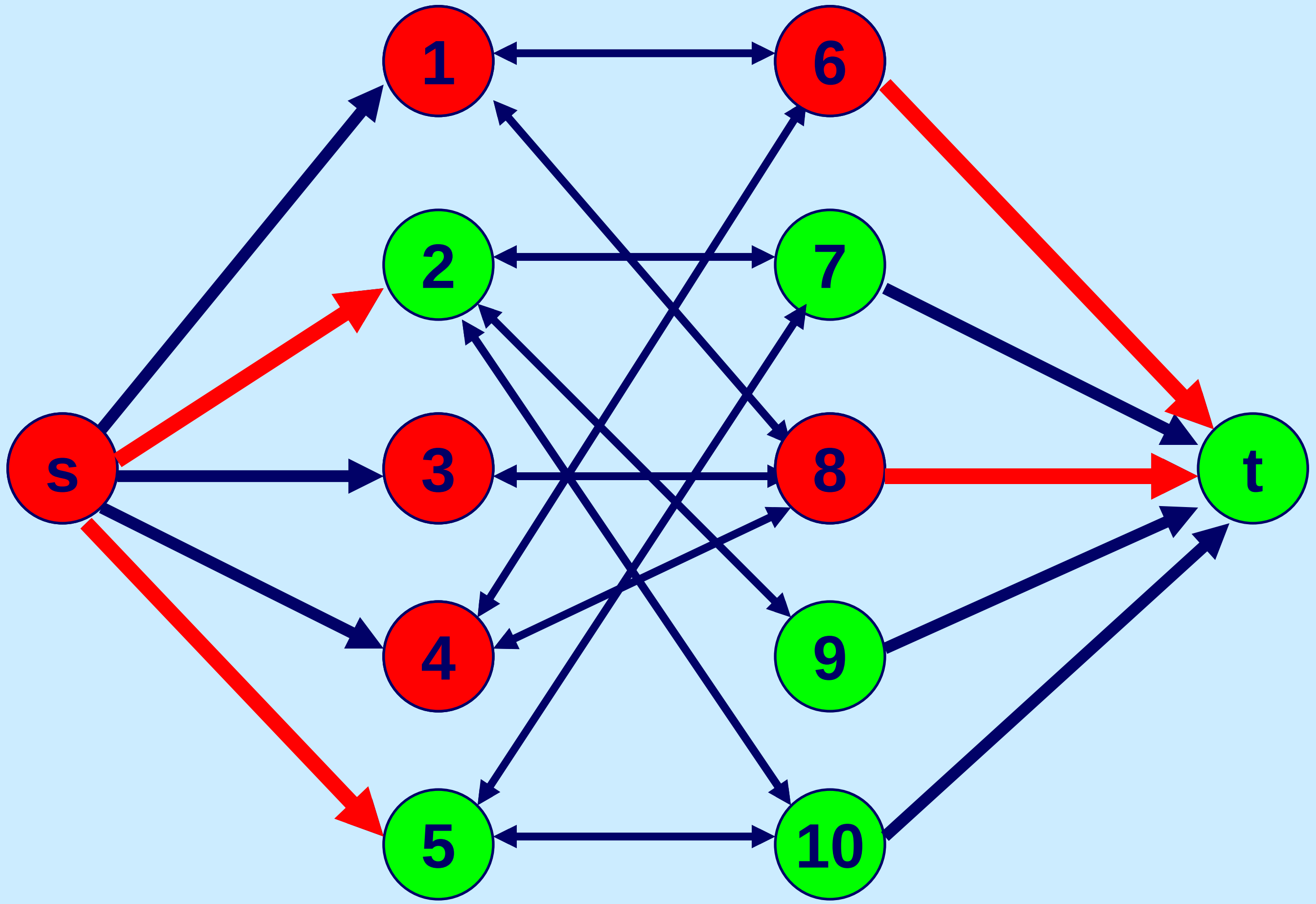
The max matching has cardinality 4.

Network surveillance

Given

- two networks
- a set of links connecting them
- install surveillance software on some computers to check connection links
- so that each link is under surveillance
- and the number of software licenses is minimized

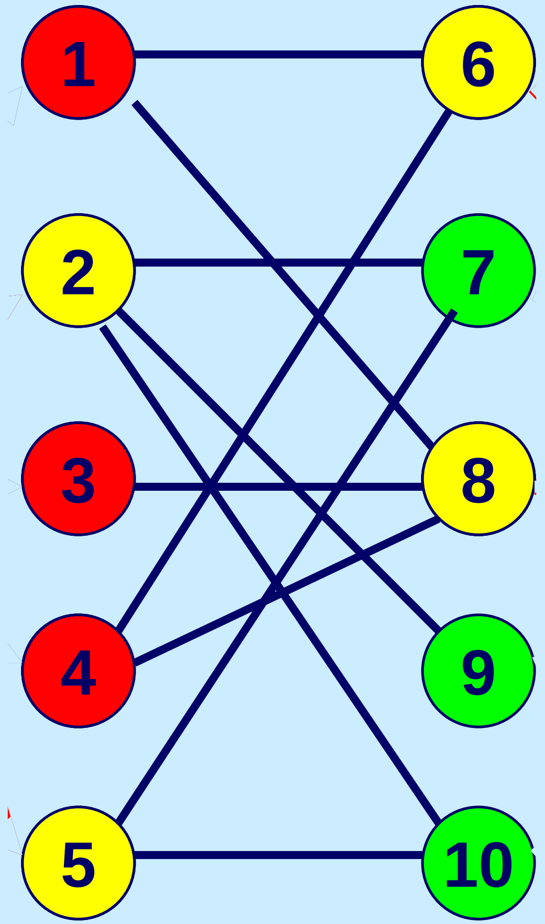
Determine the minimum cut



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

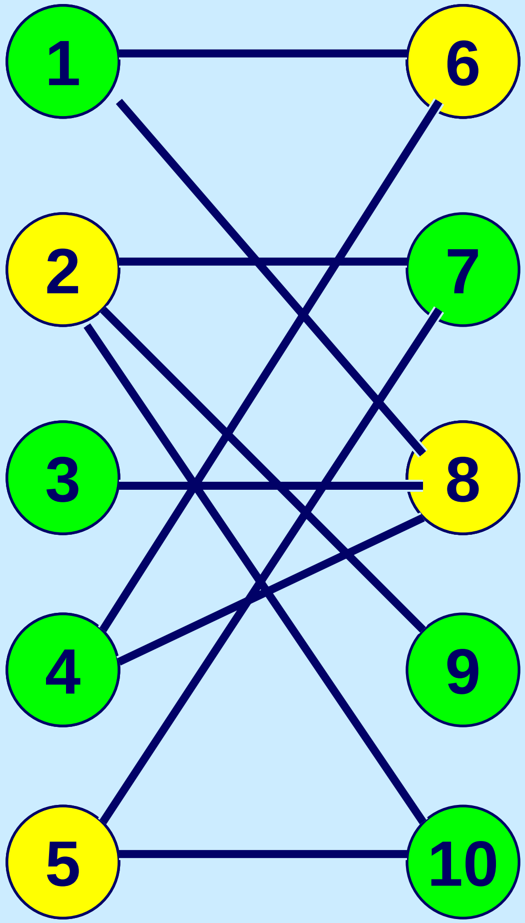
There is no arc from {1, 3, 4} to {7, 9, 10} or from {6, 8} to {2, 5}. Any such arc would have an infinite capacity.

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.