15.082J & 6.855J & ESD.78J
October 14, 2010

Maximum Flows 2

Review of the Ford-Fulkerson Algorithm

```
x := 0;
create the residual network G(x);
while there is some directed path from s to t in G(x) do
    let P be a path from s to t in G(x);
    ™* := ™(P);
    send ™* units of flow along P;
    update the r's;
```

Max-Flow Min-Cut. Let x^* be the final flow with flow value v^*.

Let $S^* = \{j \in N : s \rightarrow j \text{ in } G(x^*)\}$. Let $T^* = N \backslash S^*$.

Then x^* is a max flow, and (S^*, T^*) is a minimum cut,

and $v^* = \mathrm{CAP}(S^*, T^*)$

Overview of this lecture

1. Applications of max flow and min cut

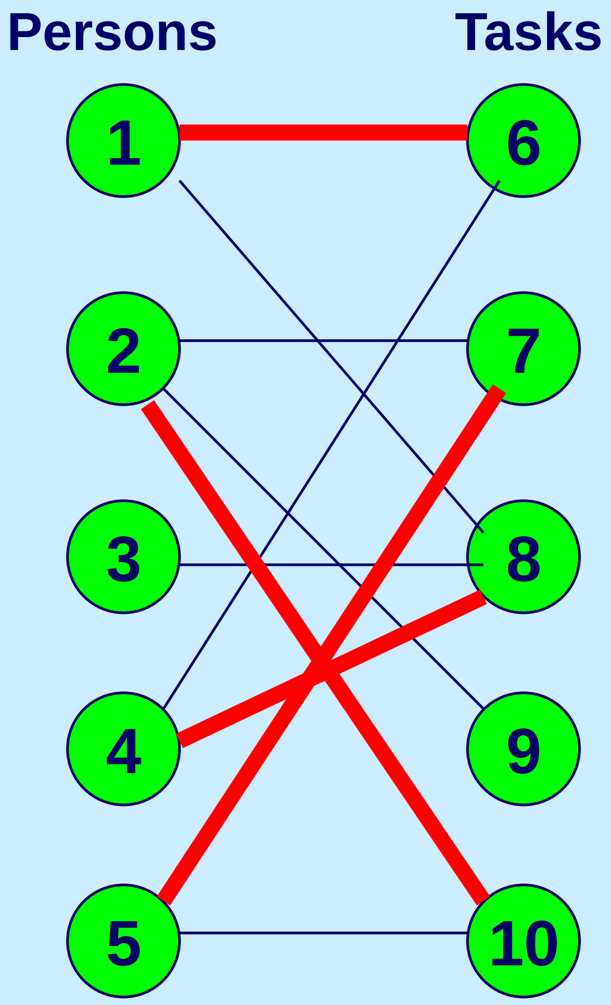
2. Speedups of the max flow augmenting path algorithm

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), $i \in N_1$ and $j \in N_2$.

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

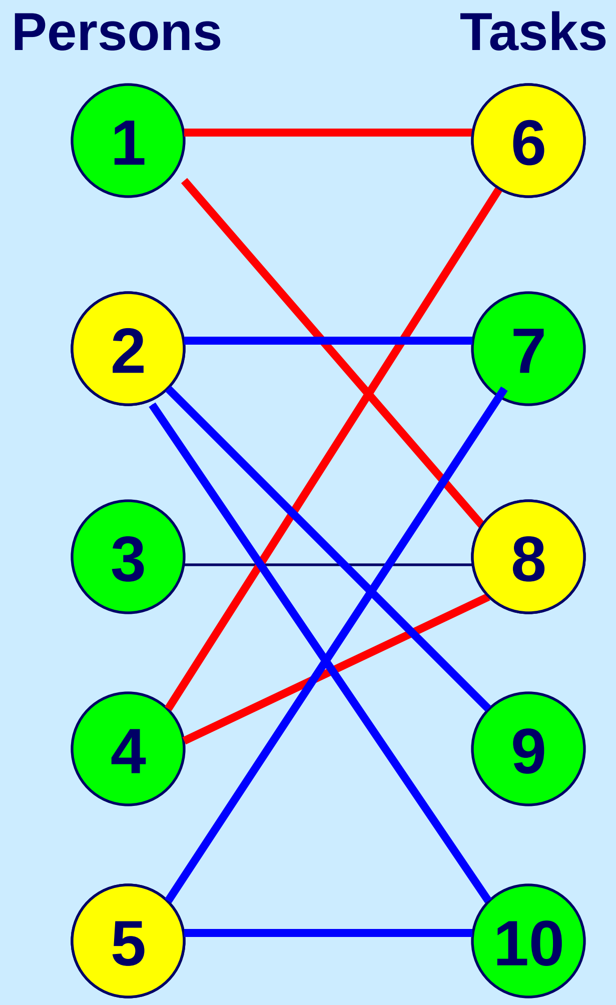
Matching Problem: Find a matching of maximum cardinality



Node Covers

A node cover is a subset S of nodes such that each arc of G is incident to a node of S.

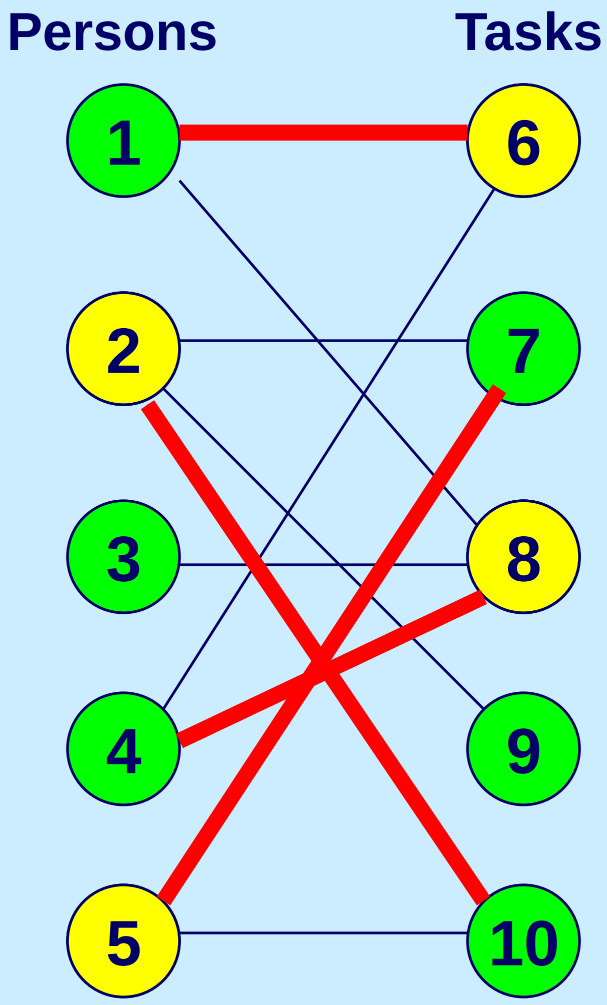
Node Cover Problem: Find a node cover of minimum cardinality.



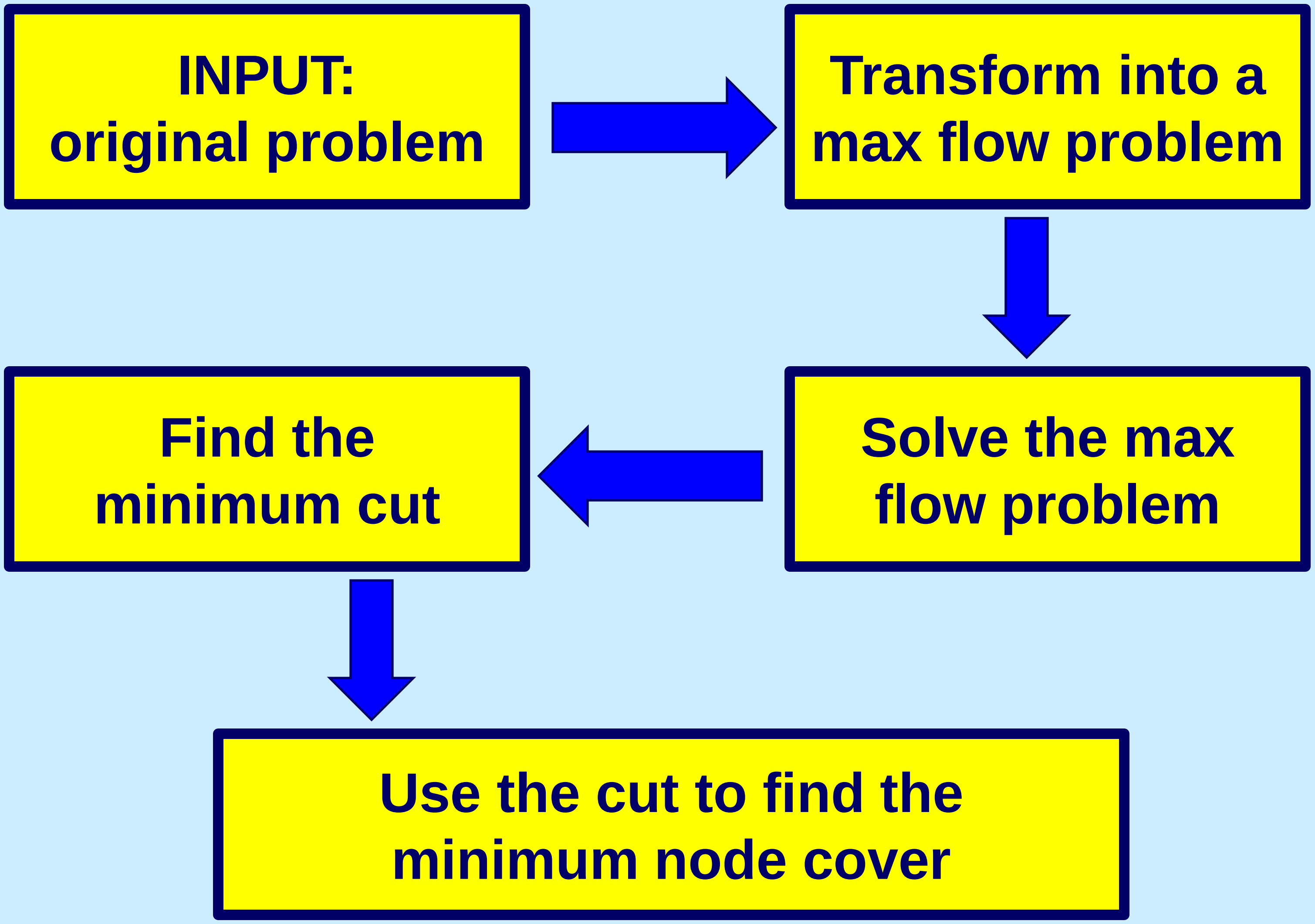
Matching Duality Theorem

Theorem. **König-Egerváry. The maximum cardinality of a matching is equal to the minimum cardinality of a node cover. (Proof in 4 slides)**

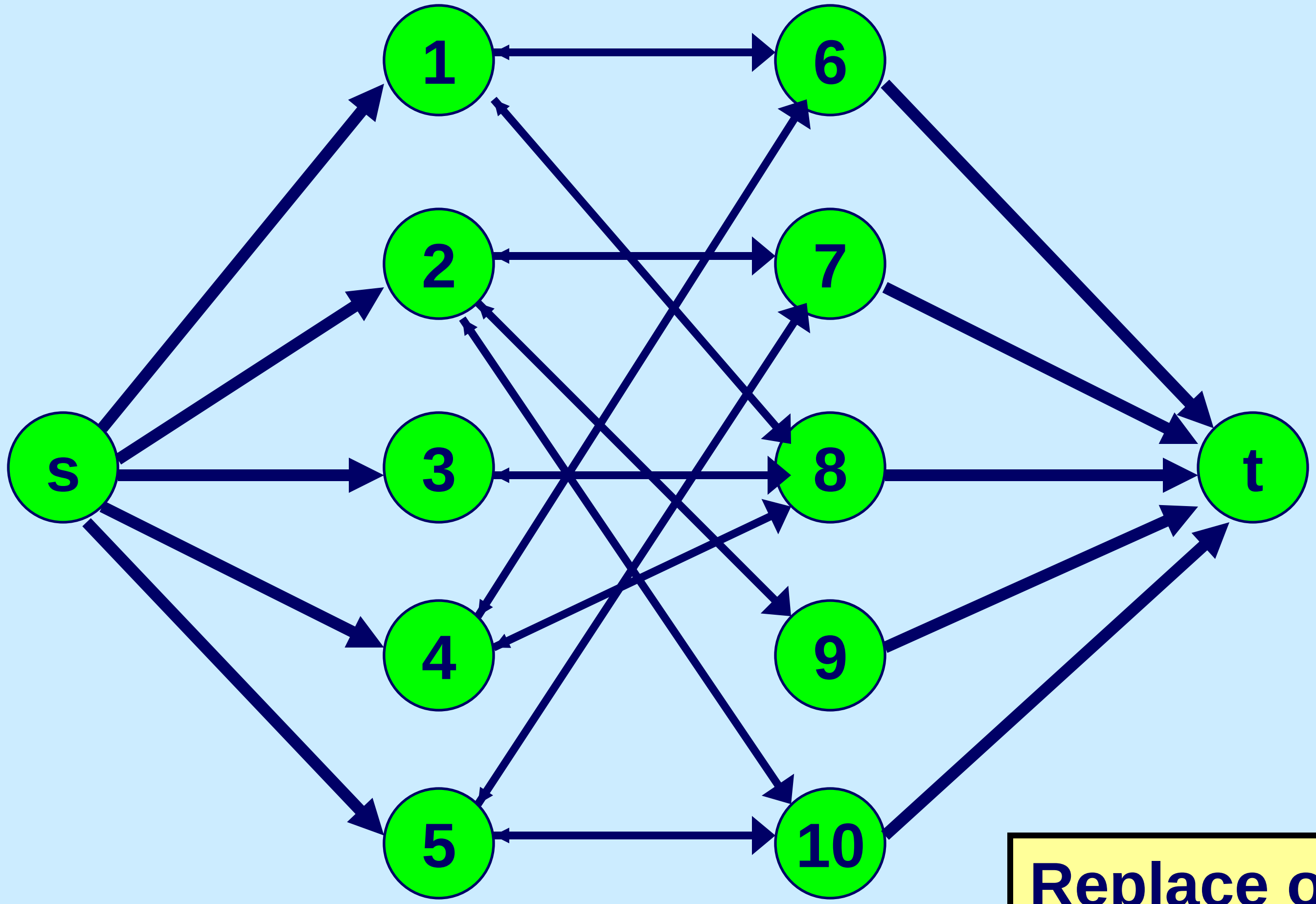
Note. **Every node cover has at least as many nodes as any matching because each matched edge is incident to a different node of the node cover.**



How to find a minimum node cover



Solving the Matching Problem as a Max Flow Problem

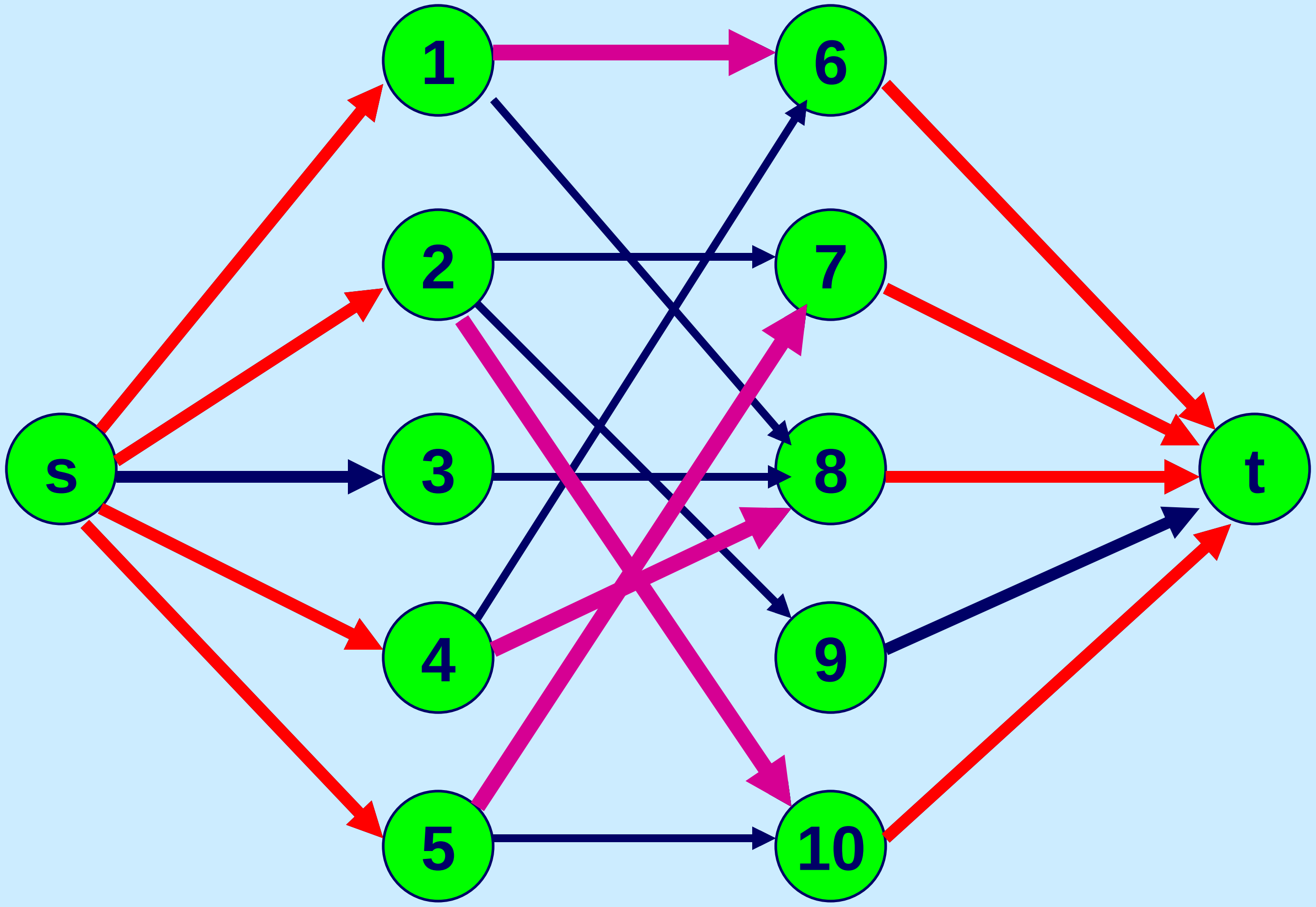


Each arc (s, i) has a capacity of 1.

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

Replace original arcs by directed arcs with infinite capacity.

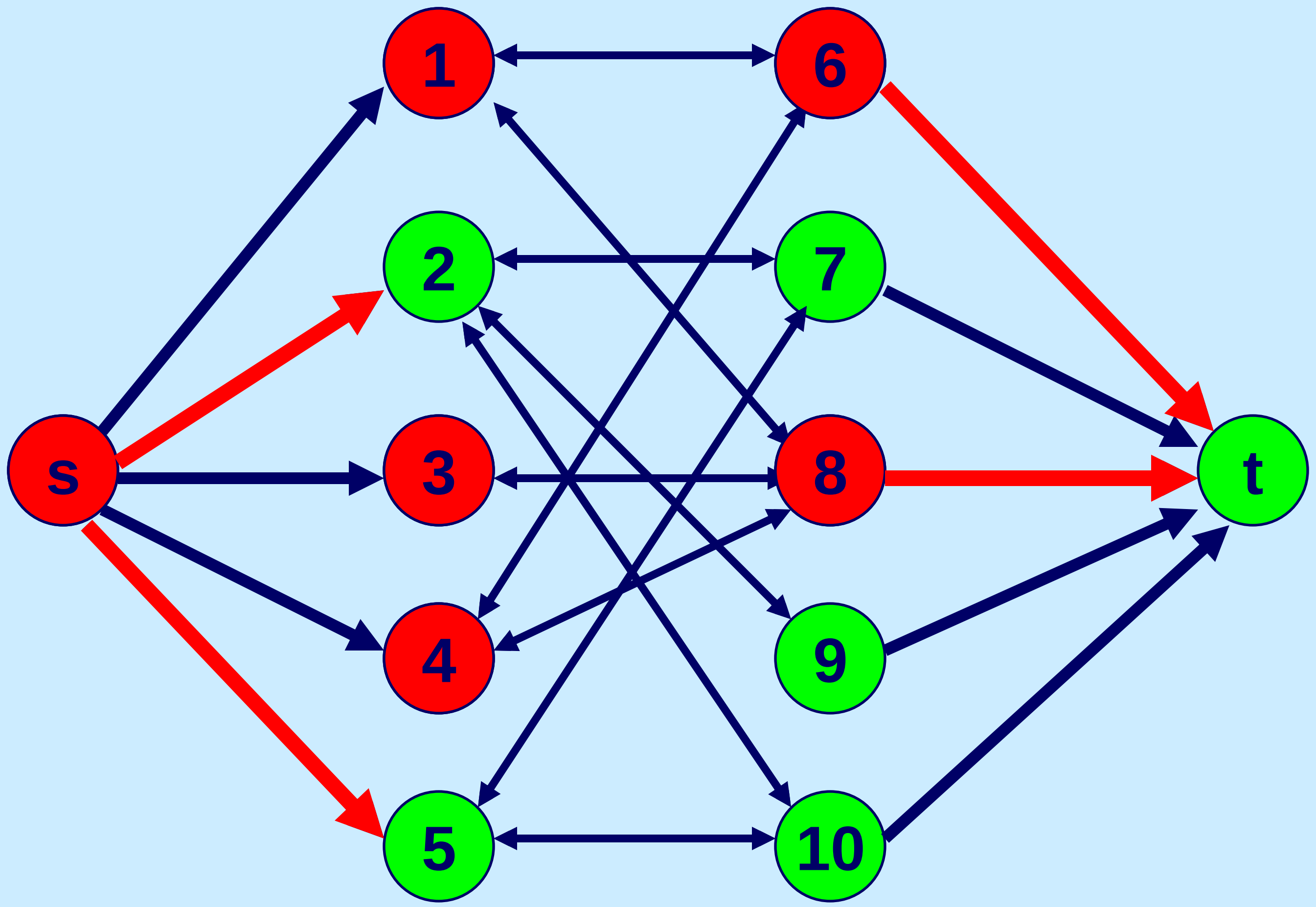
Find a max flow



The maximum s-t flow is 4.

The max matching has cardinality 4.

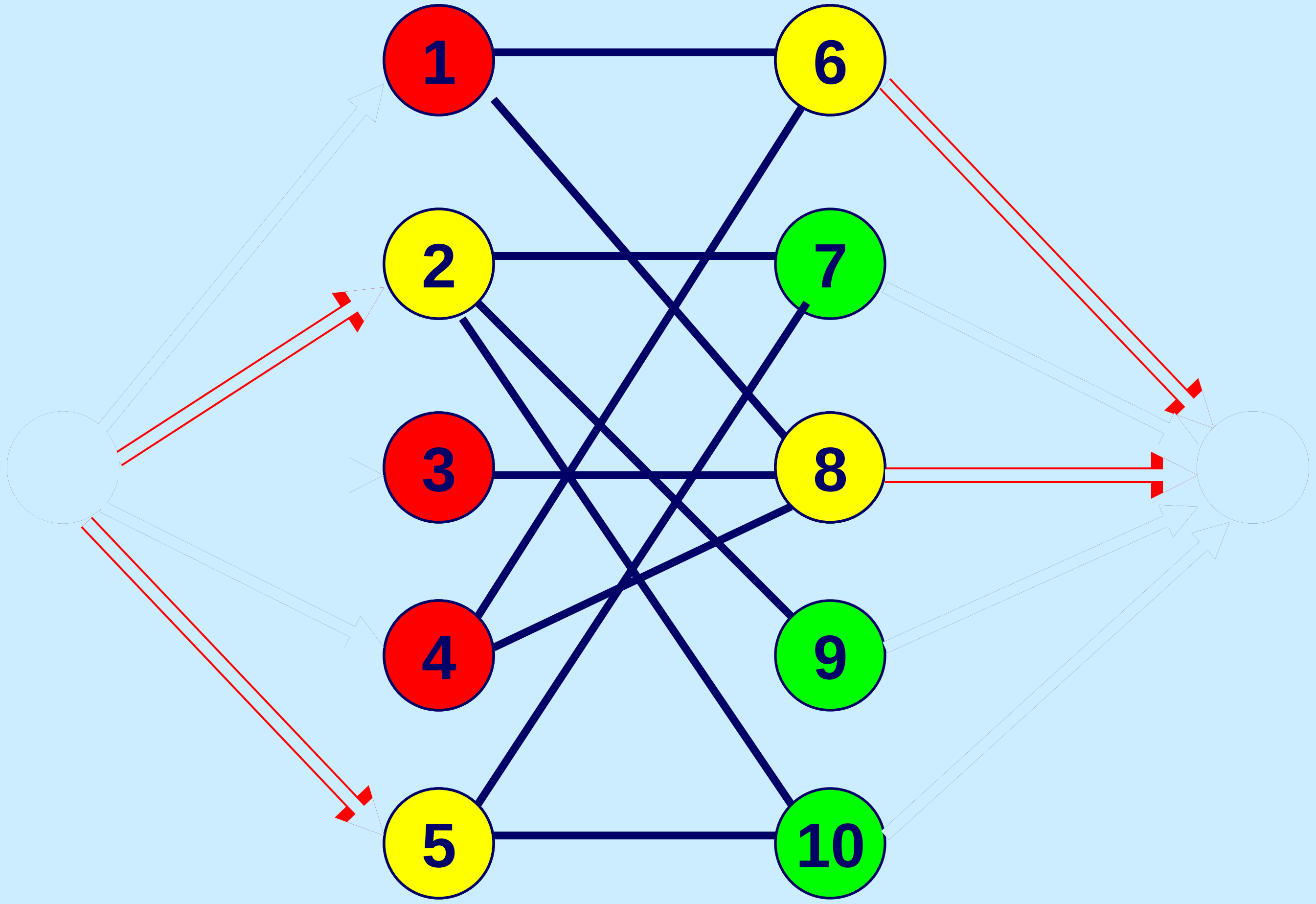
Determine the minimum cut



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

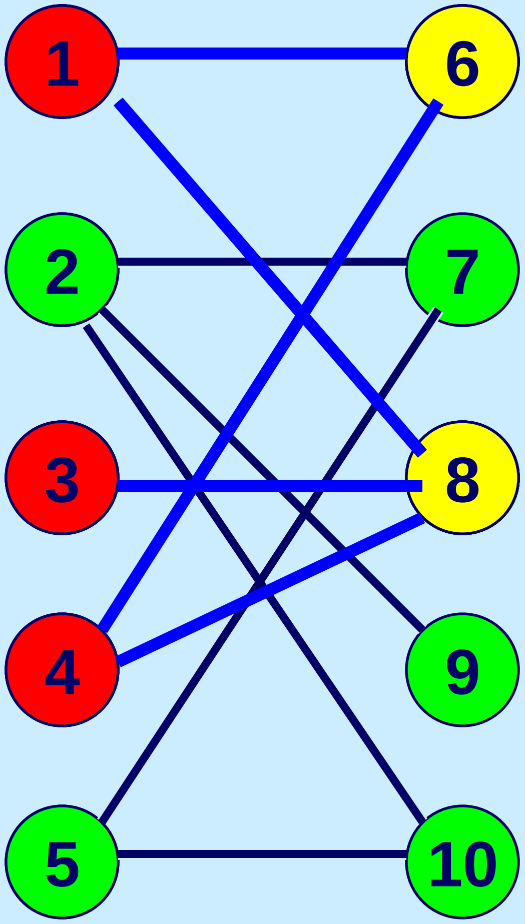
There is no arc from {1, 3, 4} to {7, 9, 10} or from {6, 8} to {2, 5}.   Any such arc would have an infinite capacity.

Find the min node cover



The minimum node cover is the set of nodes incident to the arcs across the cut.   Max-Flow Min-Cut implies the duality theorem for matching.

Philip Hall's Theorem

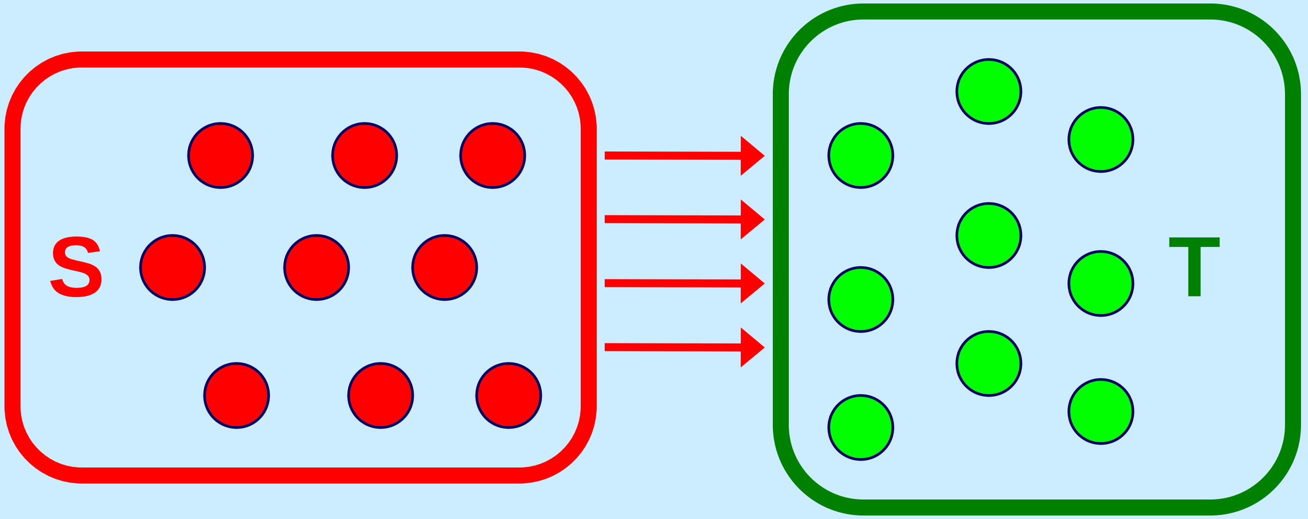


***Theorem**. Hall's Theorem. If there is no perfect matching, then there is a set S of nodes of N_1 such that |S| > |T| where T are the nodes of N_2 adjacent to S.*

Generalization of Hall's Theorem: Feasibility for min cost flows

Let G = (N, A) be a network

- b_j = supply/demand for node i. $\sum_i b_i = 0$.
- u_{ij} = upper bound on flow in (i, j)
- assume that all lower bounds are 0



T = N\S

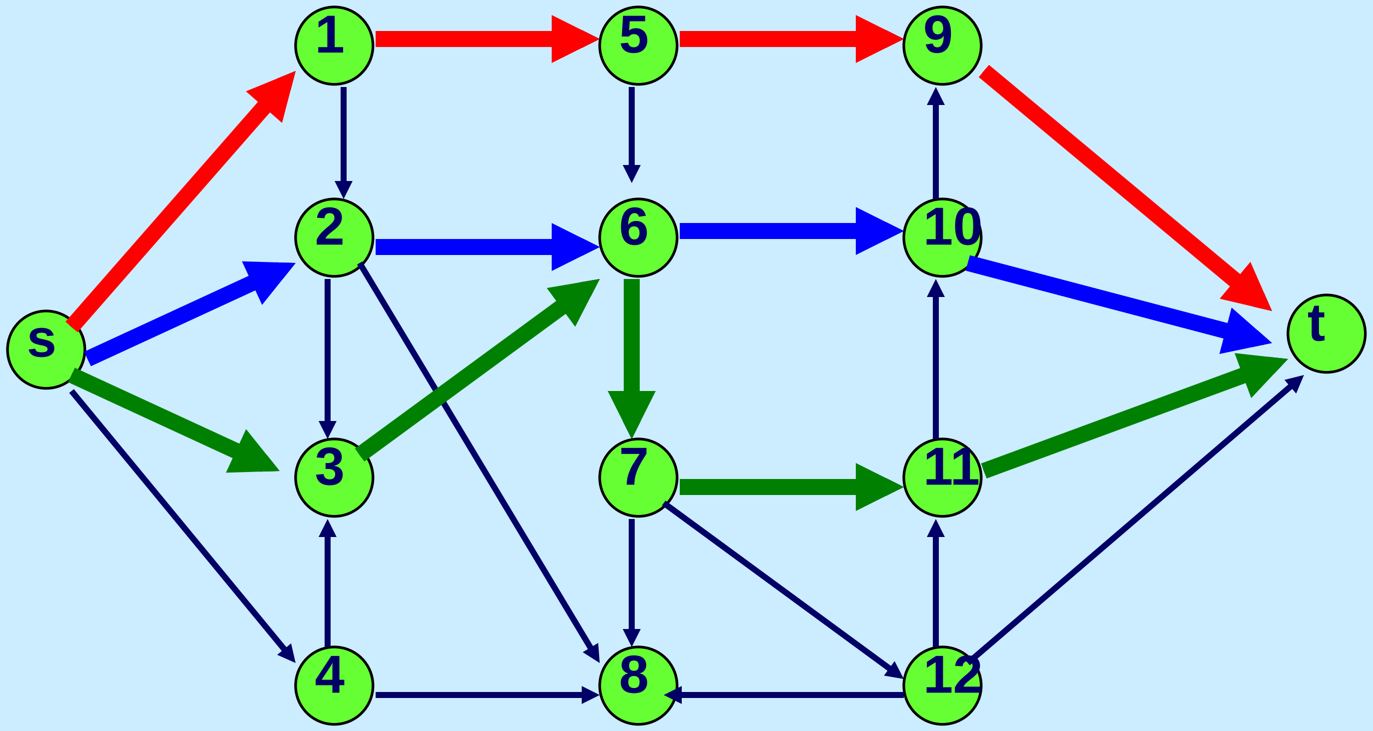
Theorem: Either there is a feasible flow in G or else there is a subset S of nodes such that $\sum_{i \in S} b_i > \text{CAP}(S, T)$.

Network Reliability

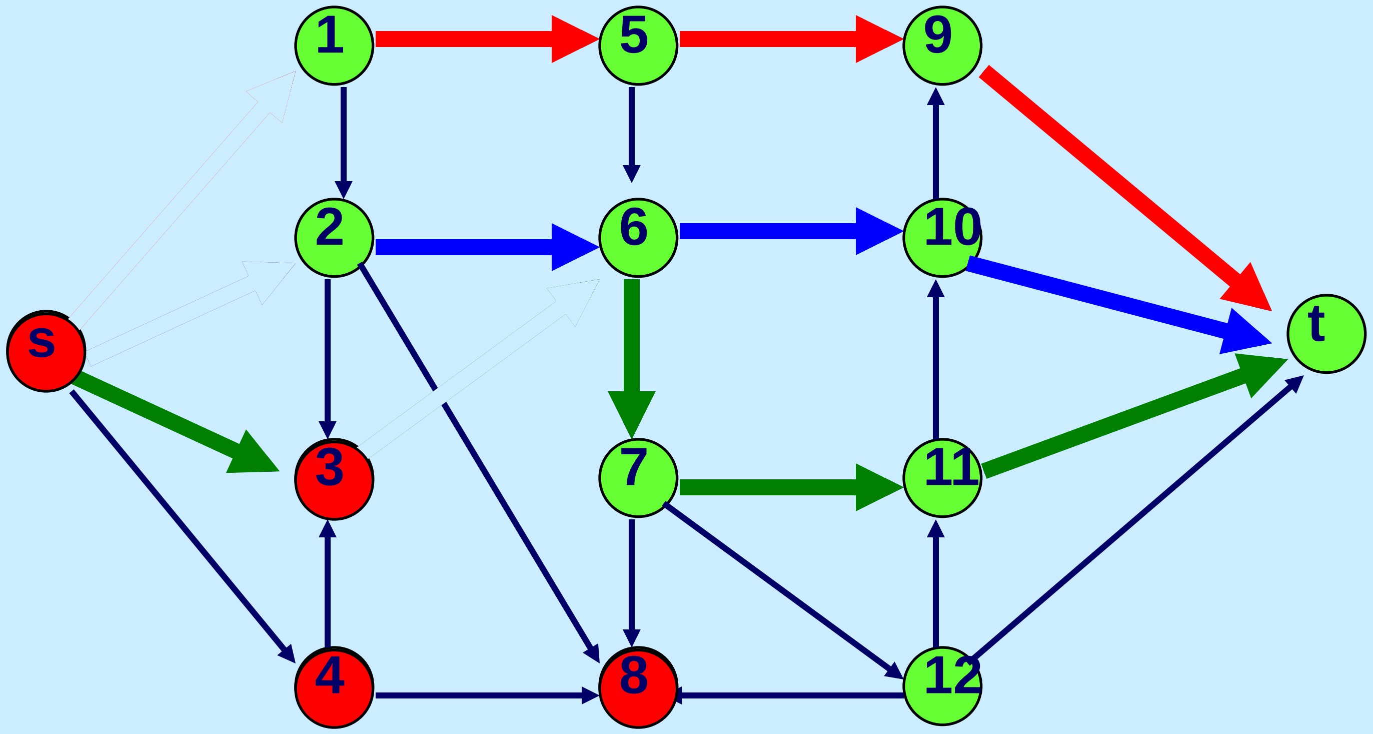
- Communication Network
- What is the maximum number of arc disjoint paths from s to t?
  - How can we determine this number?

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.

Node disjoint paths

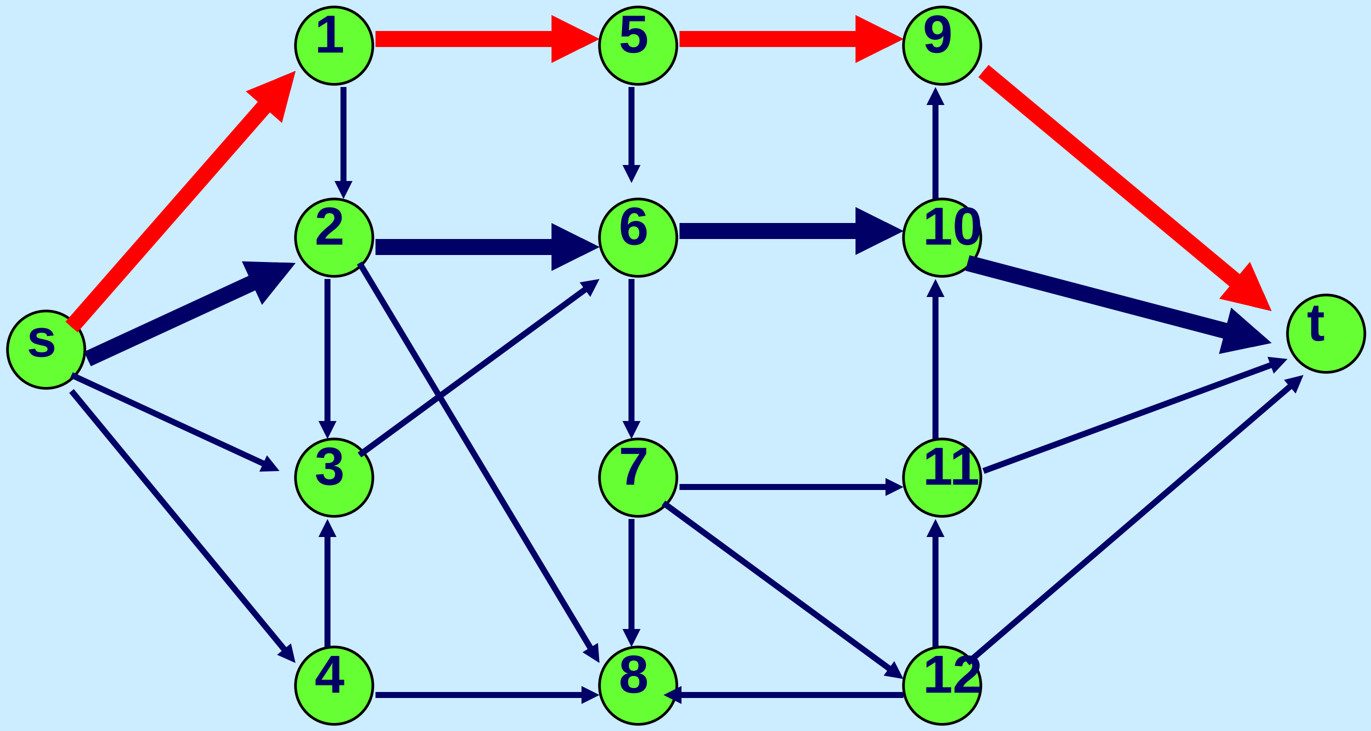
Two s-t paths P and P' f are said to be ***node-disjoint*** if the only nodes in common to P and P' are s and t).

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

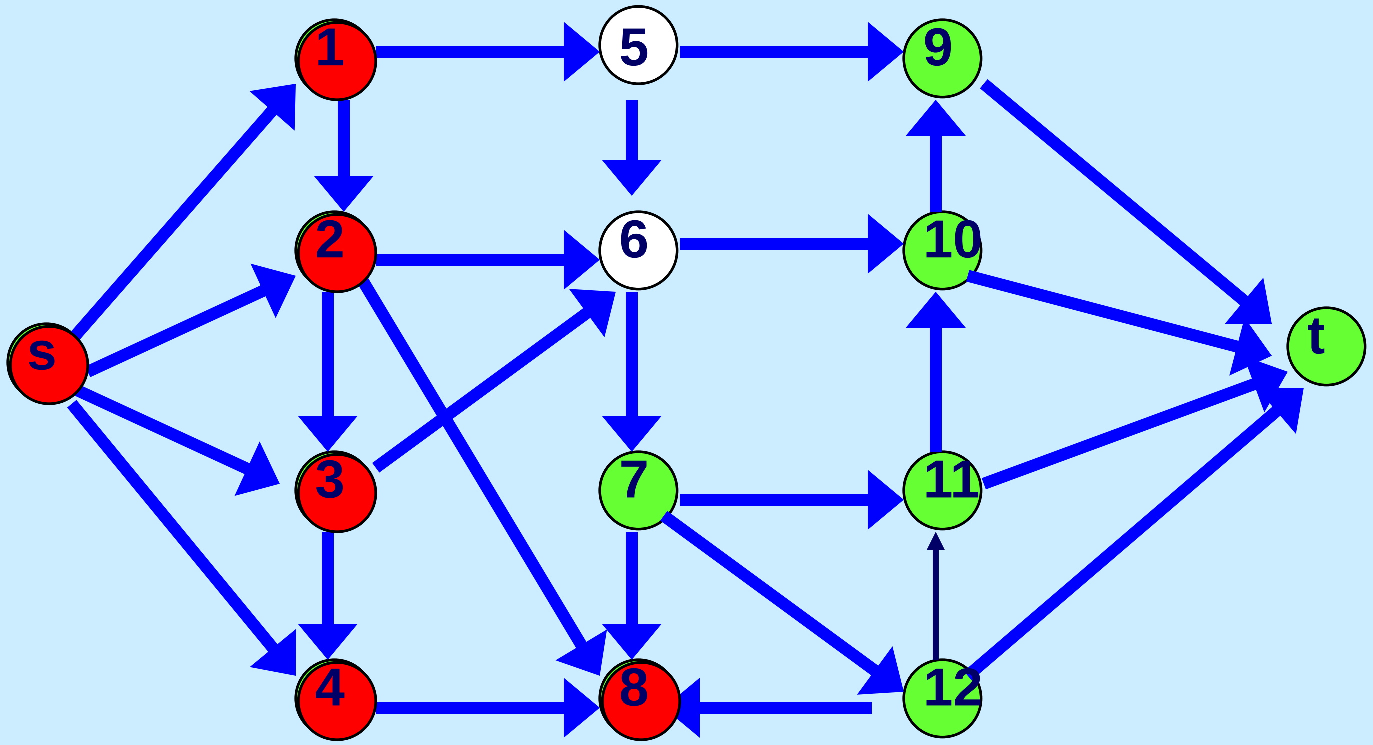
Answer: node splitting

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 in N\{s,t} 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.

Mental Break

What did ancient Egyptians shave to mourn the death of their cats?

Their eyebrows.

In what country are the ruins of Troy located?

Turkey

At the height of its power (around 400 BCE), Sparta had 25,000 citizens. How many slaves did it have?

500,000.

Mental Break

In ancient Rome, being born with a crooked nose was considered a sign. What was it a sign of?

Leadership.

The Roman emperor Caligula gave a special honor to his horse. What was the honor?

He made his horse a senator.

How long did it take for the great wall of China to be built.

Around 1900 years. From 5th century BC to the 16th century. The wall is around 3900 miles long. It is not visible to the human eye from space.

Speedups of the augmenting path algorithm

1. **Shortest augmenting path algorithm: always augment along the path in G(x) with the fewest number of arcs.-**

2. **Largest augmenting path algorithm: always augment along a path in G(x) with the greatest capacity.**

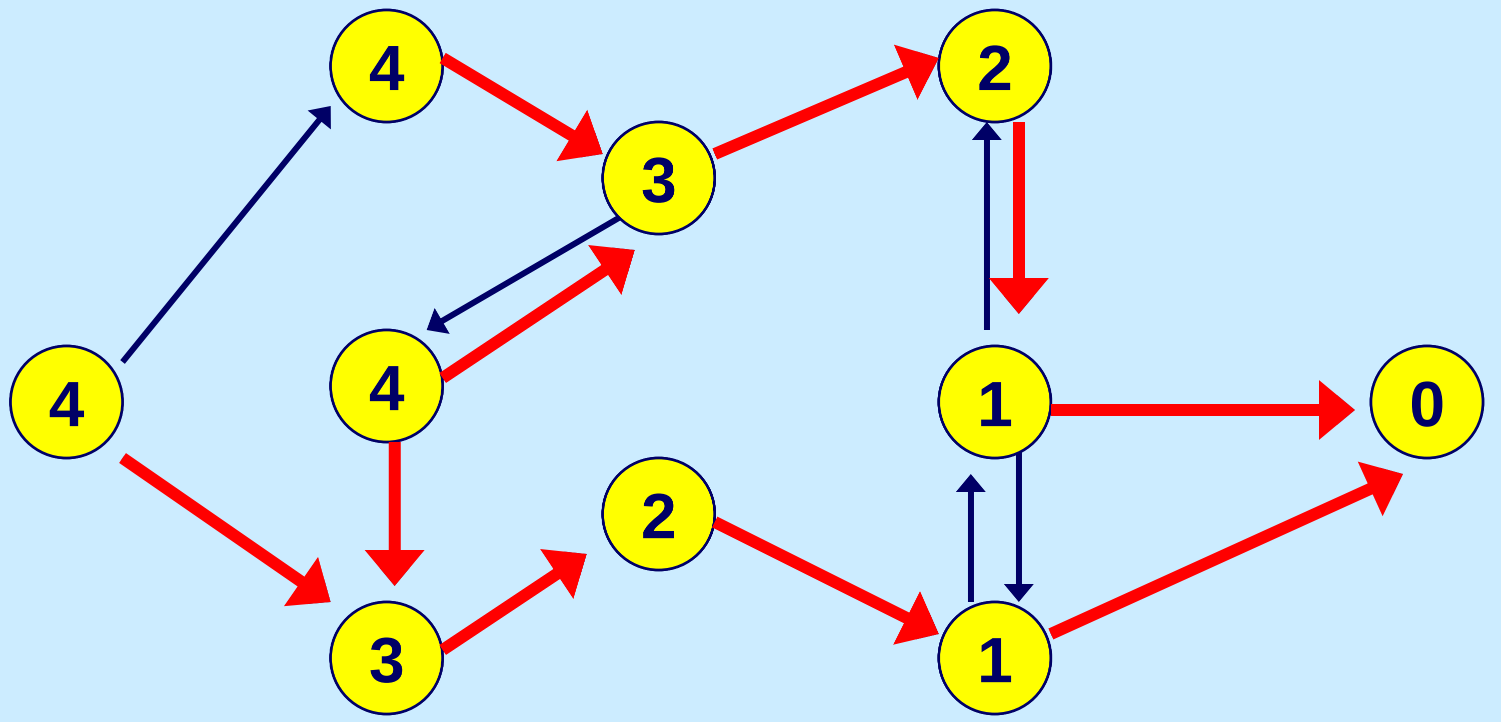
The shortest augmenting path algorithm

```
x := 0;
create the residual network G(x);
while there is some directed path from s to t in G(x) do
    let P be a path from s to t in G(x) with the fewest number of arcs;
    ™* := ™(P);
    send ™* units of flow along P;
    update the r's;
```

Theorem. The shortest augmenting path algorithm determines a maximum flow in O(nm) augmentations.

This algorithm can be implemented to run in $O(n^2m)$ time.

Distance Labels: Let d(i) be the length of the shortest path from i to t in G(x)



FACT 1: If (i, j)∈ G(x), then d(i) ≤ d(j) + 1.

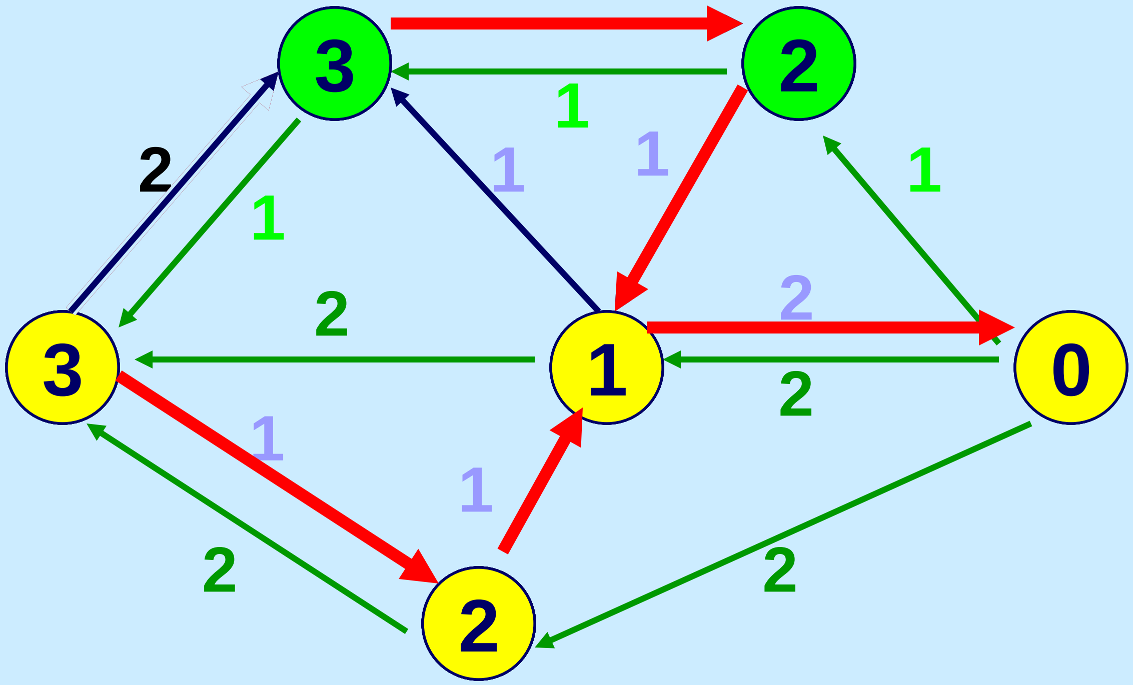
FACT 2: Arc (i, j) is on a shortest path from i to t if and only if d(i) = d(j) + 1.

FACT 3: d(t) = 0; d(i) < n for all i s.t. i ➔ t in G(x).

Valid Arcs and Saturating Pushes

An arc (i, j) ∈ G(x) is **valid** if $d(i) = d(j) + 1$.

FACT: If P is an augmenting path, then every arc of P is valid.



Suppose δ units of flow are sent along P. The augmentation **saturates** arc (i, j) ∈ P if $r_{ij} = \delta$.

The number of augmenting paths

Theorem. The number of augmenting paths for the shortest augmenting path algorithm is O(nm).

Fact. In every augmenting path, at least one arc (i, j) is saturated.

Lemma 1. Arc (i, j) and its reversal (j, i) can be saturated at most n/2 times each. (To be proved later.)

Proof of theorem. Let a_{ij} be the number of times that arc (i, j) is saturated. Let A be the number of augmentations.

Then $A \leq \sum_{(i,j)\in A} 2a_{ij} \leq \sum_{(i,j)\in A} n \leq nm$.

Proof of Lemma 1. Each arc (i, j) is saturated fewer than n times.

Lemma 2. Let d be the distance labels an iteration where arc (i, j) is saturated. Suppose that d' is the vector of distance labels at a subsequent iteration where (i, j) is saturated. Then $n > d'(i) \geq d(i) + 2$.

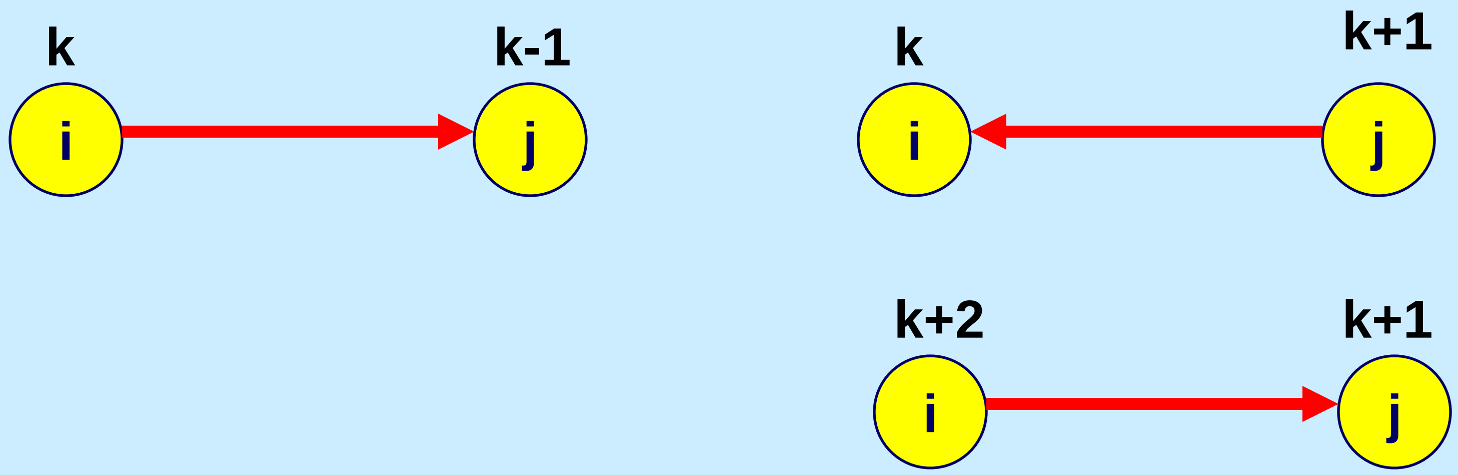
(to be proved on next slide).

Proof of Lemma 1 from Lemma 2. Before (i, j) can be saturated again, its distance label will increase by at least 2. Since $0 < d(i) < n$, the distance label can increase at most n/2 times.

Proof of Lemma 2.

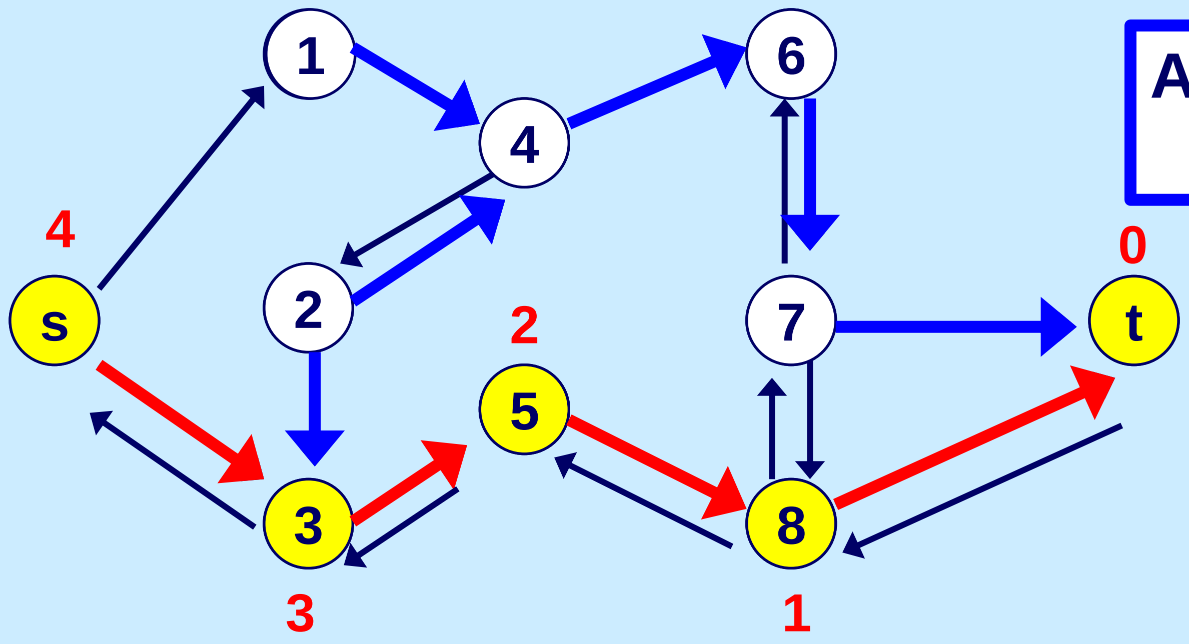
Lemma 3. Let d be the distance labels at some iteration, and let d' be the distance labels at a subsequent iteration. Then d'(i) ≥ d(i) for all i ∈ N. (to be proved on next slide)

Proof of Lemma 2 from Lemma 3. Suppose that (i, j) is saturated when d(i) = k. There is no more flow in (i, j) until flow is sent in (j, i) at which point the distance label of j is k+1. But flow cannot be returned in (i, j) until it is valid again, and the distance label is at least k+2.



Proof of Lemma 3.

Assume that Lemma 3 is false. Let d be the distance labels at some iteration. Let P be the augmenting path. After the augmentation, the reversals of arcs in P are in the residual network. But adding these arcs to G(x) does not decrease any distance. And deleting arcs of P cannot decrease a distance label.



Augmenting path:
s-3-5-8-t.

Largest Augmenting Path Algorithm

Theorem. (Edmonds-Karp). Suppose that one augments along the augmentation with the largest residual capacity. Then the maximum flow is determined after O(m log U) augmenting paths.

Running time: The time to find the maximum augmenting path is O(m log m). (Why?)

Thus the total running time is O(m^2 log m log U)

Finding the largest augmenting path

Step 1. Sort the capacities in the residual network. Suppose that sorted capacities are $c_1, c_2, \ldots, c_{2m}$.

Step 2. Let $G_j(x')$ be the residual network as restricted to arcs with capacity at least c_j.

Step 3. Use binary search to find the largest value j so that there is a path from s to t in $G_j(x')$.

Notation for the proof of the theorem: Let v* be the maximum flow value out of s. Let v_k be the amount of flow out of s immediately prior to the k-th augmentation.
Let a_k be the capacity of the k-th augmentation.

Lemma 4 and Lemma 5.

Lemma 4. For all k, $a_k \geq (v^* - v_k)/m$.

Lemma 5. For all k, there exists $k' < k + 2m$ such that $a_{k'} < a_k/2$.

Proof of Theorem from Lemma 5. The largest initial augmentation is at most U.

Then $a_k \leq U/2$ for some $k \leq 2m$.

Then $a_k \leq U/4$ for some $k \leq 4m$.

One can show using induction that

$a_k < 1$ for some $k \leq 2m(\lceil \log U \rceil + 1)$.

But capacities are always integer valued. So, there are at most 2m (log U+1) augmentations.

Proofs of Lemmas 4 and 5

Proof of Lemma 4. Let x' be the arc flows prior to the k-th iteration. Then the max flow out of s in G(x') is (v* - v').

Let y be the maximum flow in G(x'). The flow decomposition of y has most m paths from s to t. The sum of these flows is (v* - v') and so the maximum of the capacities of these paths is at least (v* - v')/m.

Proof of Lemma 5. Suppose that the theorem is false. Suppose that $a_j > a_k/2$ for j = k to k+2m. The total amount of flow sent from s during these iterations is greater than $2m(a_k/2) > v^* - v_k$.

This is impossible, and so the lemma is true.

Geometric convergence arguments

Lemma 5. Suppose that any algorithm for a maximization problem has objective value v_k at iteration k, and let v* be the optimum value. Suppose there is some positive integer B such that for all k,

$(v_{k+1} - v_k) \geq (v^* - v_k)/B.$ Then for all k:

$(v_{k+2B} - v_k) > (v^* - v_k)/2.$

Proof by contradiction. Let $a_k = v_{k+1} - v_k$. Suppose that $(v_{k+2B} - v_k) \leq (v^* - v_k)/2$. By assumption:

for each j from k to k+2B,

$a_j = (v_{j+1} - v_j) \geq (v^* - v_j)/B \geq (v^* - v_k)/2B$. Then

$$v_{k+2B} = v_k + \sum_{j=0}^{2B-1} a_j \geq v_k + \sum_{j=0}^{2B-1} \left(v^* - v_k\right)/2B \geq v^*.$$

Summary

- Applications of max-flow min-cut
- Analysis of Shortest Augmenting Path Algorithm
- Distance labels
- Analysis of Largest Augmenting Path Algorithm
- Geometric convergence arguments.