
15.082 and 6.855J
March 11, 2003

Max Flows 3
Preflow-Push Algorithms

Review of Augmenting Paths

At each iteration: maintain a flow x

Let $G(x)$ be the residual network

At each iteration, find a path from s to t in $G(x)$.

In the shortest augmenting path algorithm, we kept distance labels $d(\cdot)$, and we sent flow along the shortest path in $G(x)$.

Preflows

At each intermediate stages we permit more flow arriving at nodes than leaving (except for s)

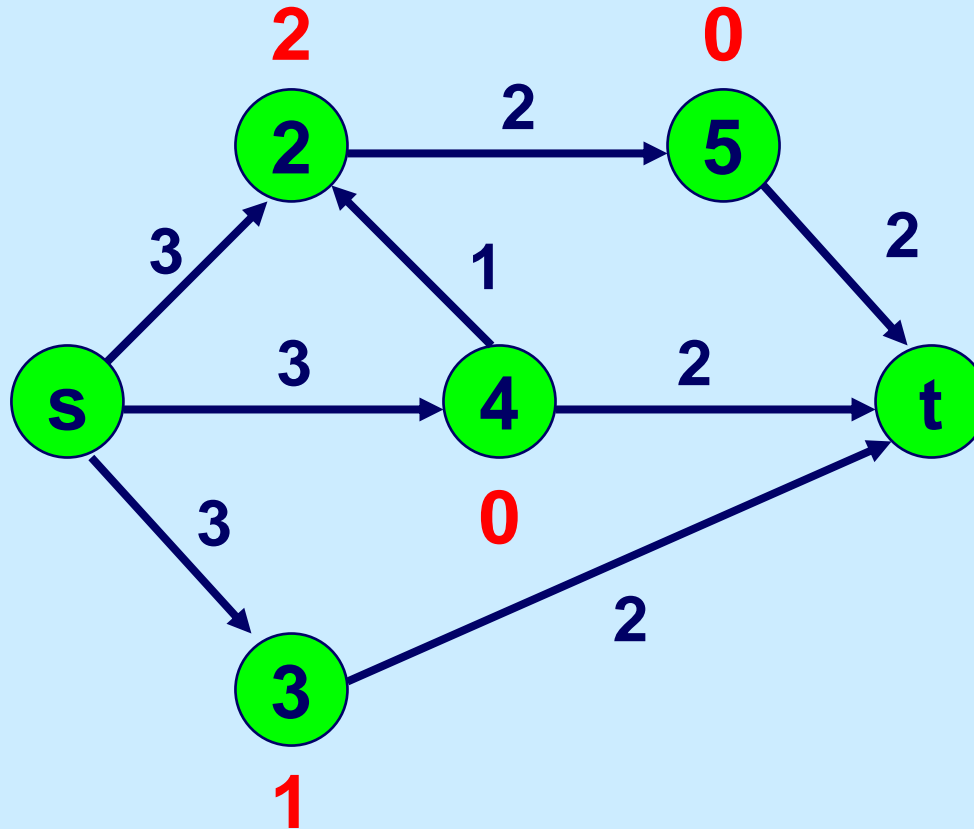
A **preflow** is a function $x: A \rightarrow R$ s.t. $0 \leq x \leq u$ and such that

$$e(i) = \sum_{j \in N} x_{ji} - \sum_{j \in N} x_{ij} \geq 0,$$

for all $i \in N - \{s, t\}$.

i.e., $e(i) =$ **excess** at $i =$ net excess flow into node i .
The excess is required to be nonnegative.

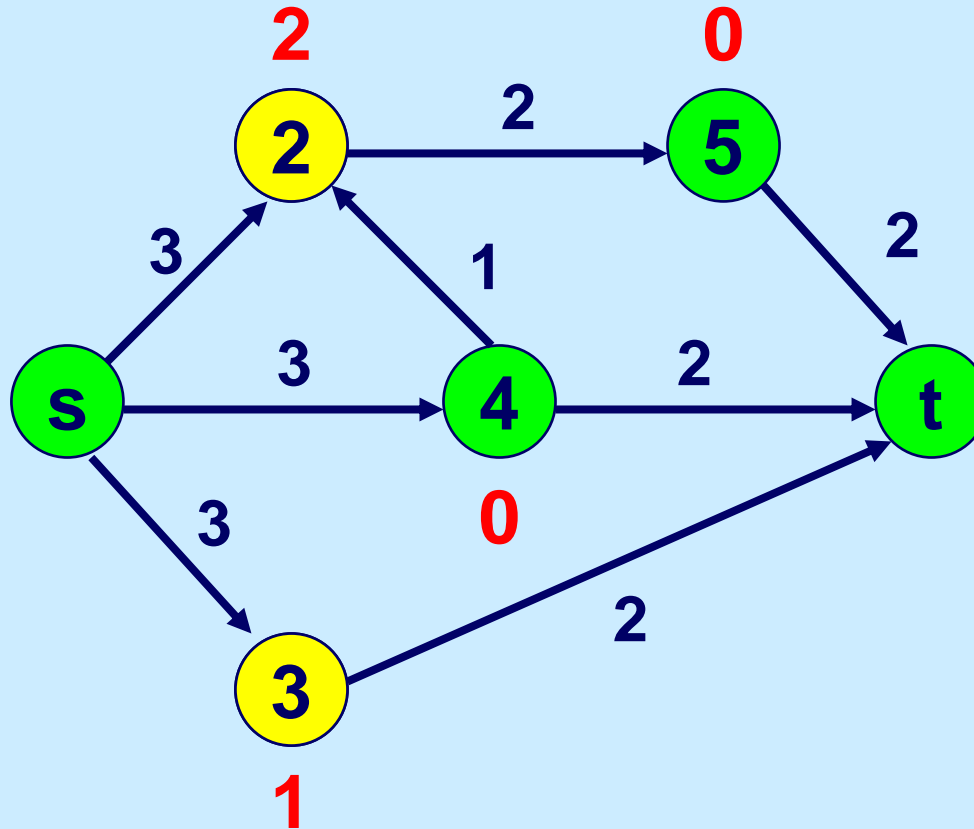
A Feasible Preflow



The **excess** $e(j)$ at each node $j \neq s, t$ is the flow in minus the flow out.

Note: total excess = flow out of s minus flow into t.

Active nodes



Nodes with positive excess are called **active**.

The preflow push algorithm will try to push flow from active nodes towards the sink, relying on $d(\cdot)$.

Review of Distance Labels

Distance labels $d(\cdot)$ are **valid** for $G(x)$ if

- i. $d(t) = 0$
- ii. $d(i) \leq d(j) + 1$ for each $(i,j) \in G(x)$

Defn. An arc (i,j) is **admissible** if $r_{ij} > 0$
and $d(i) = d(j) + 1$.

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.

Push/Relabel, the fundamental subroutine

Suppose we have selected an active node i .

Procedure Push/Relabel(i)

begin


if the network contains an admissible arc (i,j) then

push $\delta := \min\{e(i), r_{ij}\}$ units of flow from i to j ;

else replace $d(i)$ by $\min\{d(j) + 1 : (i,j) \in A(i) \text{ and } r_{ij} > 0\}$

end;

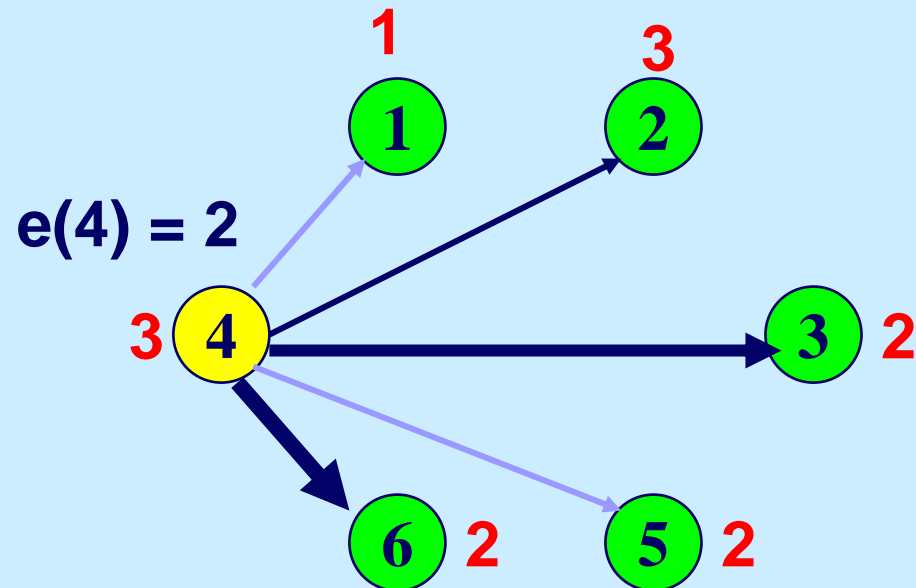
Pushing using current arcs

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

Suppose that node 4 is active, and has excess.

Scan arcs in $A(4)$ one at a time using “Current Arc” till an admissible arc is found.

Push on (4,3)



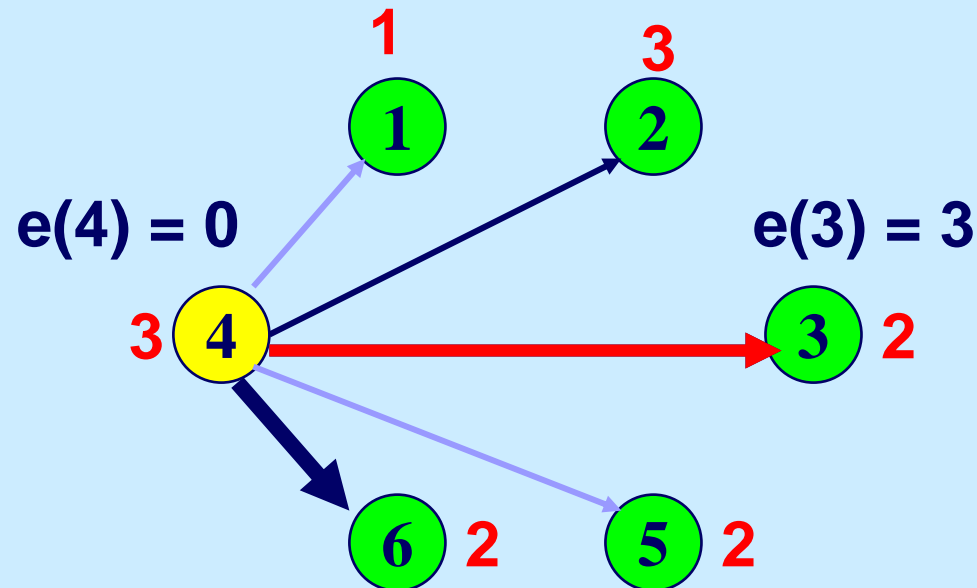
Pushing on (4,3)

	Tail	Head	Res. Cap	Admissible ?	
	4	1	0	No	Push on (4,3)
	4	2	1	No	
→	4	3	2	Yes	
	4	5	0	No	
	4	6	2	Yes	

Send $\min(e(4), r_{43}) = 2$ units of flow.

Update the residual capacities and excesses.

For the next push from node 4, start with arc (4,3).



Goldberg-Tarjan Preflow Push Algorithm

Procedure Preprocess

begin

$x := 0;$

compute the exact distance labels $d(i)$ for each node;

$x_{sj} := u_{sj}$ for each arc $(s,j) \in A(s)$; $d(s) := n$;

end

Algorithm PREFLOW-PUSH;

begin

preprocess;

while there is an active node i **do**

begin

select an active node i ;

push/relabel(i);

end;

end;

**Preflow Push
Animation**

Preview of Results on Preflow Push

The preflow push algorithm is superb both in theory and in practice.

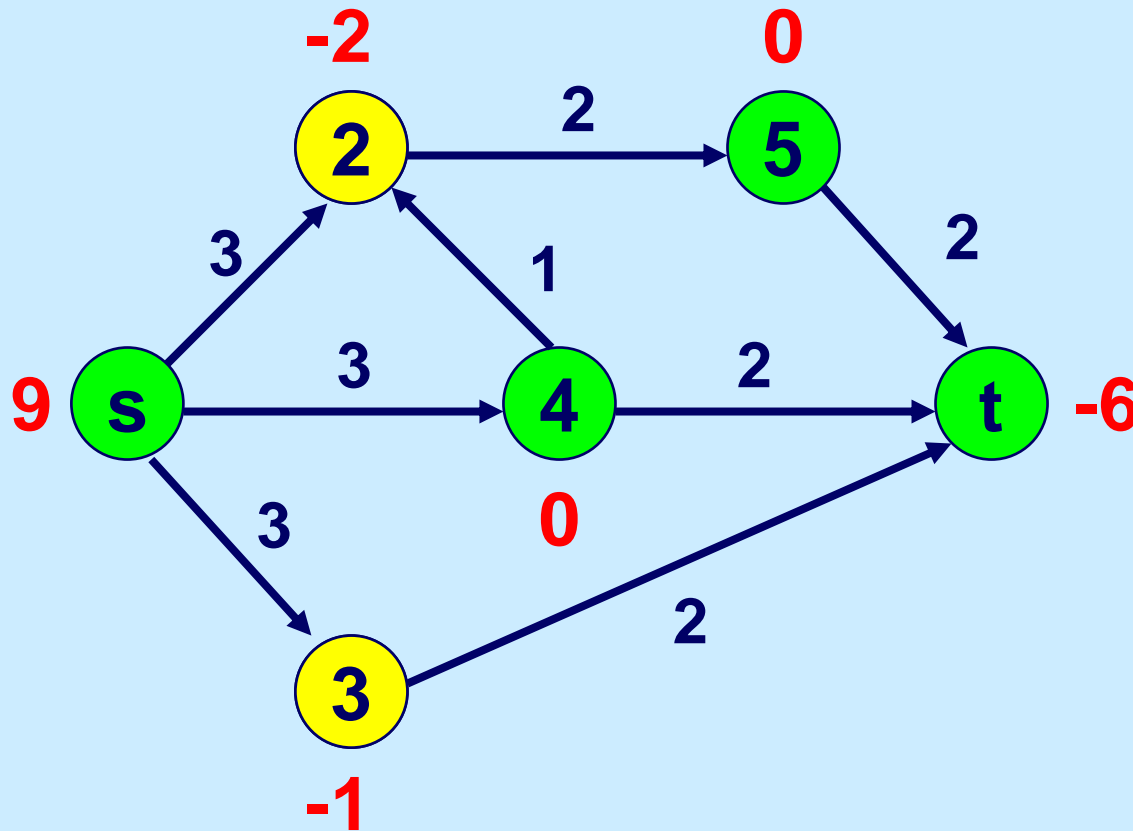
To prove:

1. $d(j) < 2n$ throughout the algorithm
2. The algorithm terminates with a maximum flow
3. The number of steps excluding non-saturating pushes is $O(nm)$
4. The number of non-saturating pushes is $O(n^2m)$.

Further improvements are possible.

A lemma needed for the time bounds

Lemma 7.11. *At each stage of the algorithm, there is a path in $G(x)$ from each node i with $e(i) > 0$ to node s .*



Here is the same preflow as before

As a flow, the “excesses” are negative supplies.

Proof. Apply flow decomposition to the flow x .

2 units in s-2-5-t;

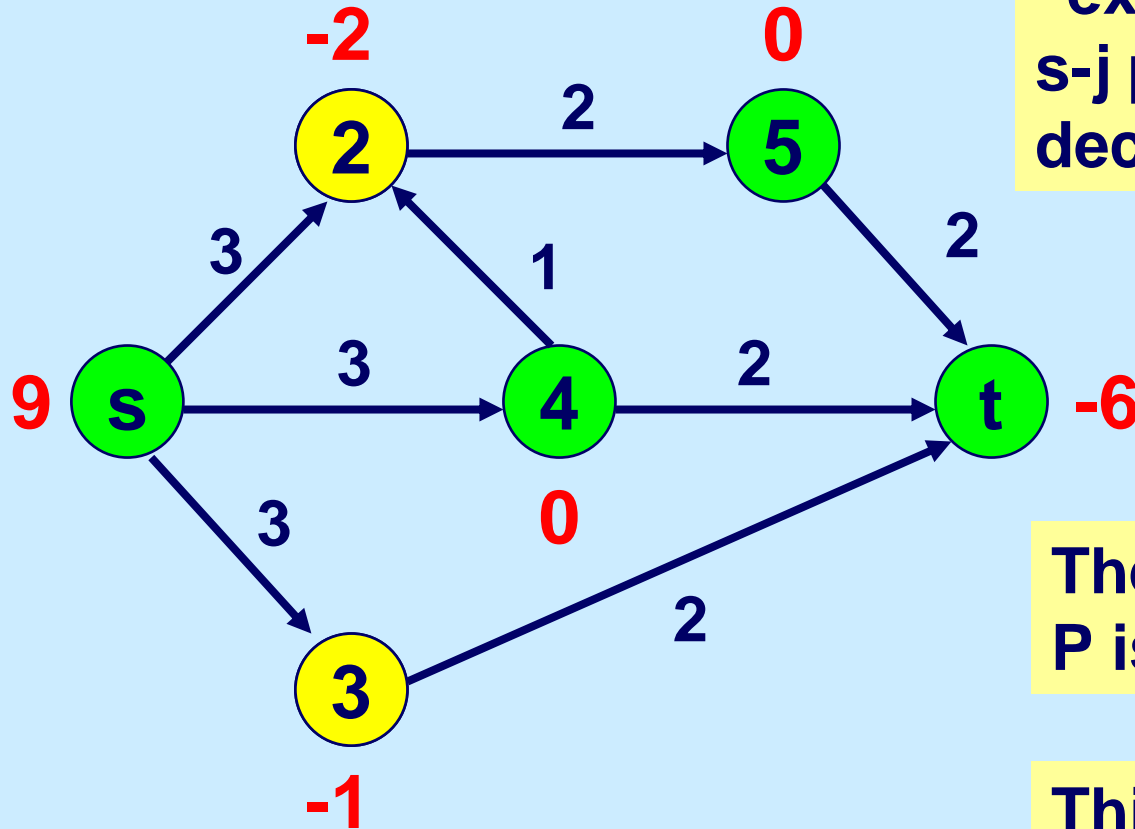
1 unit in s-2

1 unit in s-3

2 units in s-4-t

1 unit in s-4-2

2 units in s-3-t



For each node j with “excess” there is an s - j path P in the flow decomposition.

The reversal of P is in $G(x)$.

This completes the proof.

Bounding $d(j)$

Lemma 7.12. *For each node j with excess, $d(j) \leq 2n - 1$.*

Proof. Let $f(j)$ be the length of the shortest path in $G(x)$ from node j to node s . then
$$d(j) \leq d(s) + f(j) \leq n + (n-1) = 2n - 1.$$

Lemma 7.13. *Each node is relabeled fewer than $2n$ times, and so the total number of relabels is fewer than $2n^2$.*

Theorem. *The preflow push algorithm is finite and terminates with the maximum s-t flow.*

Proof. The algorithm is finite because the distance labels can increase at most $2n$ times each.

The algorithm ends with a flow (as opposed to preflow) because if there were any active node, the algorithm would not end.

At the end, $d(s) = n$. Since $d(s)$ is a lower bound on the shortest s-t path in $G(x)$, there is no s-t path in $G(x)$, and the flow is maximum.

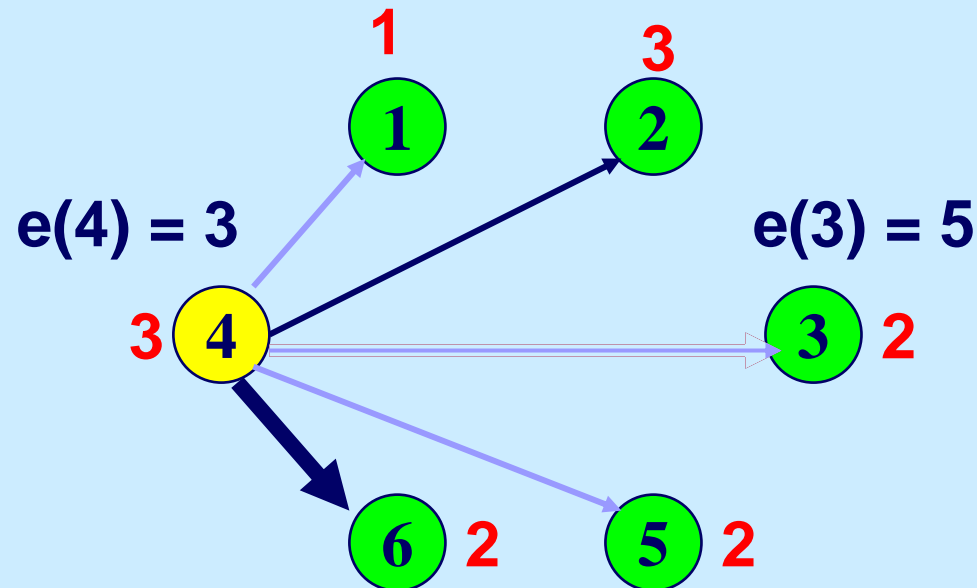
Bounding the time for relabels

Time to relabel.

- To relabel node j , one needs to scan each arc of $A(j)$
- So, each arc gets scanned at most $2n$ times
- Total relabel time is $O(nm)$

Bounding the remaining steps

	Tail	Head	Res. Cap	Admissible ?	
	4	1	0	No	Push Flow from node 4.
	4	2	1	No	
→	4	3	0	Yes	
	4	5	0	No	
	4	6	9	Yes	



Send $\min(e(4), r_{43}) = 4$ units of flow.

Update the residual capacities and excesses.

In a saturating push in (i,j) the flow is r_{ij} .

Computation time

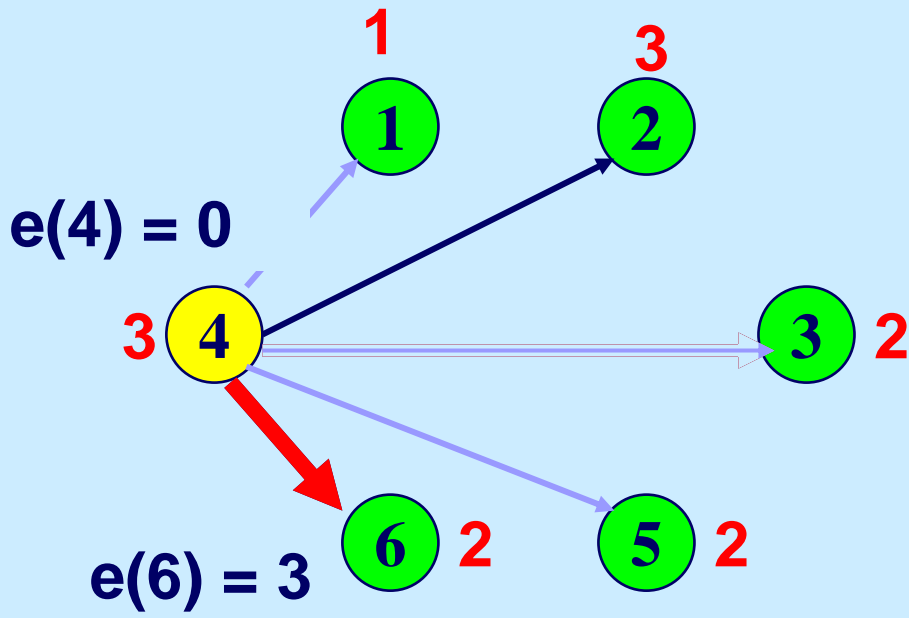
Moving “current arc” for i . $|A(i)|$ steps between relabels. This run time is no more than the time for a relabel of node i .

Saturating pushes for i . At most $|A(i)|$ between relabels of i . Once an arc is saturated, the current arc will increase by 1.

But the bottleneck are the *non-saturating* pushes.

Bounding the remaining steps

	Tail	Head	Res. Cap	Admissible ?	
	4	1	0	No	Push Flow from node 4.
	4	2	1	No	
	4	3	0	No	
	4	5	0	No	
→	4	6	6	Yes	



Send $\min(e(4), r_{46}) = 3$ units of flow.

Update the residual capacities and excesses.

Note: **CurrentArc** will stay on node (4,6)

Review of run time analysis

Time to select active nodes is $O(1)$ per push since we can maintain a set of active nodes.

We bounded relabels and relabel time

We bounded all time in pushing, except for non-saturating pushes

- **Time to advance “currentArc” is bounded by the time to relabel**
- **Time for saturating pushes is bounded by the time to advance “currentArc”**

Non-saturating pushes really are the bottleneck.

- **We will use a different analysis technique to bound them**

Potential Functions -- accounting version.

Example 1. Consider a joint bank account held by persons A and B.

The account starts with an initial deposit of \$1,000.

- A only makes withdrawals, and each is for \$10 or more.
- B makes deposits and withdrawals. Each deposit is for at most \$100. B has made at most 20 visits to the bank, and at most one deposit at each visit.
- The account never goes below \$100.

What is the maximum number of times that A has made withdrawals?

Example 2

Consider a joint bank account held by persons A, B, and C

The account starts with an initial deposit of at most n^2 .

- A only makes withdrawals, each for 1 or more.
- B makes fewer than n^2 deposits. Each deposit is for at most 1.
- C makes fewer than nm withdrawals and deposits. Each deposit is for at most n .
- The account never goes below 0.

What is the maximum number of withdrawals for A?

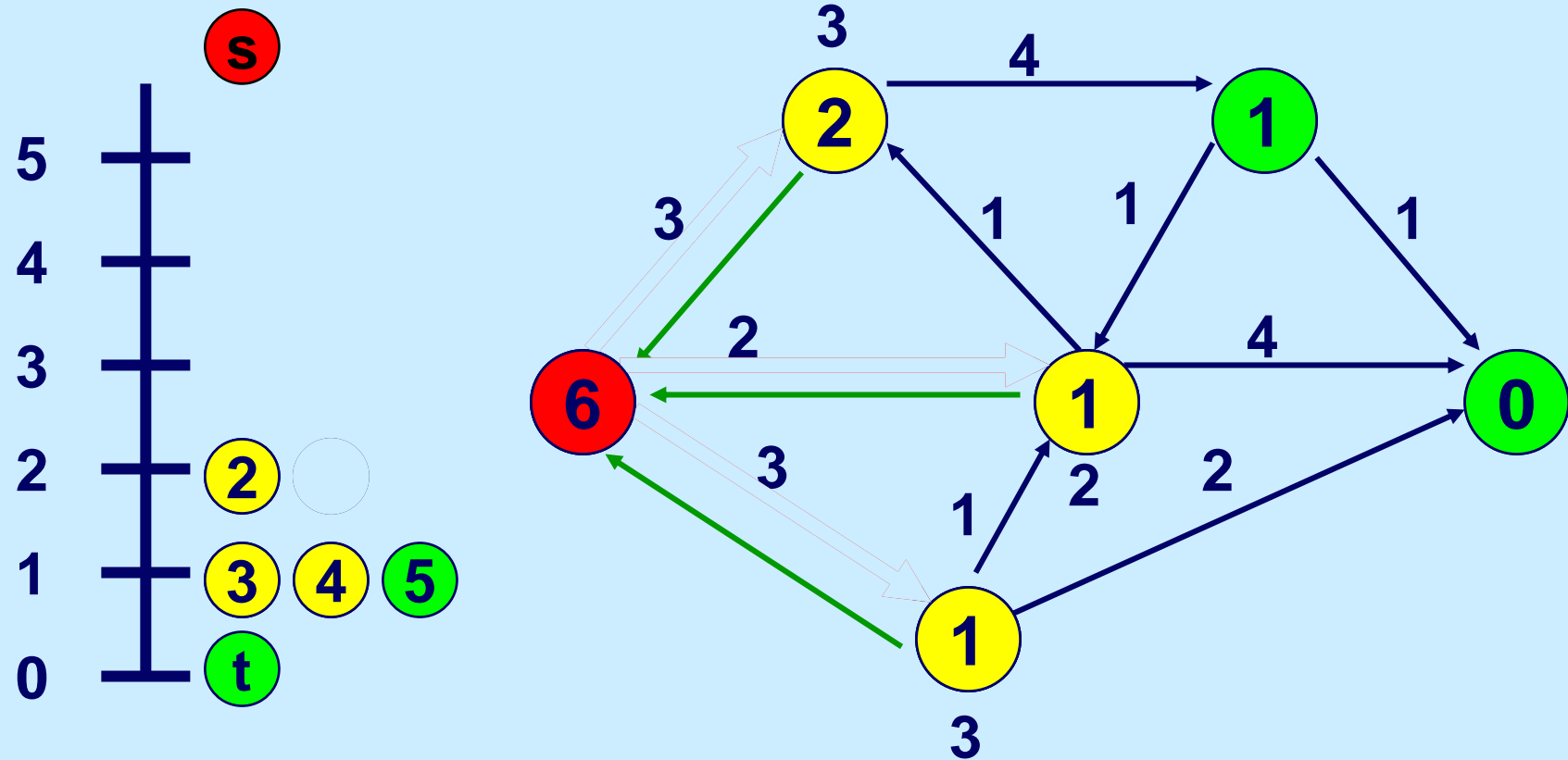
A Potential Function for Preflow-Push

Let $F = \sum_{j \text{ active}} d(j)$.

We call F a *potential function*.

It plays the role of the account level (total dollars) in the shared bank account in the previous example.

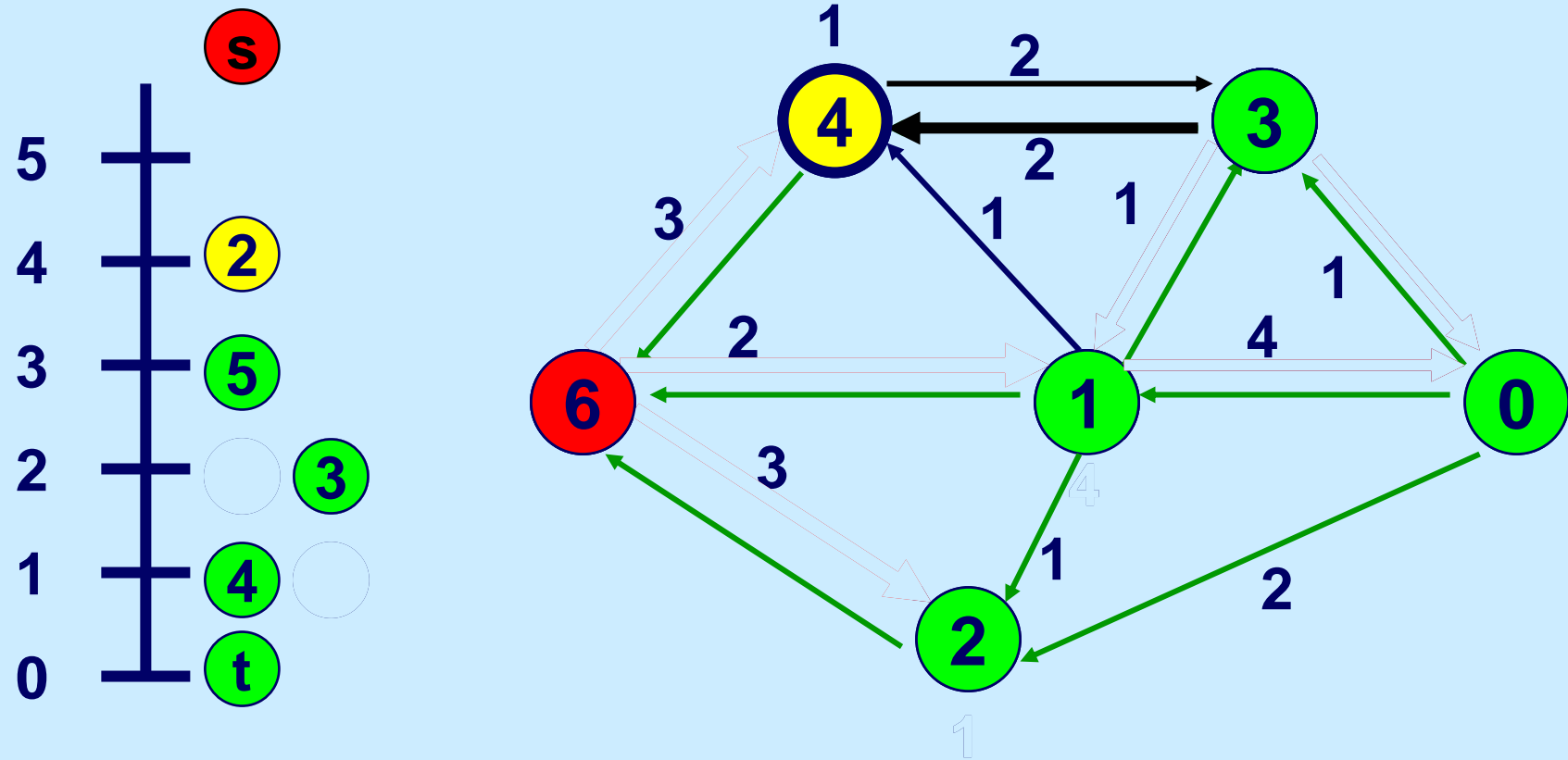
Initial value of $F = \sum_{j \text{ active}} d(j)$.



Add up distance labels of active node. $1 + 1 + 2$

In general, F is bounded initially by n^2 .

Effect of relabels on $F = \sum_{j \text{ active}} d(j)$



Consider the relabel of node 2.

F increases from 2 to 4. Each unit increase in $d(4)$ increases F by 1.

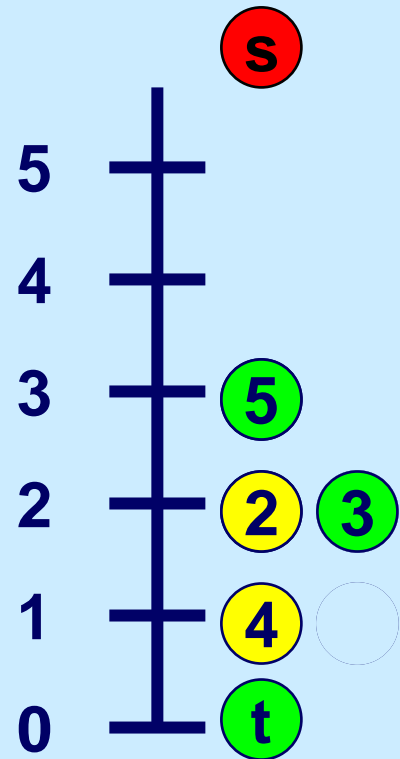
More on relabels

Recall: a “deposit” is an increase in F . The value of the deposit is the increase in F .

$d(4) < 2n$. So, the total “deposits” over all iterations from relabeling node 4 is at most $2n$.

The total “deposits” from relabeling all nodes is at most $2n^2$.

More on non-saturating pushes



Recall: a withdrawal is a decrease in F .

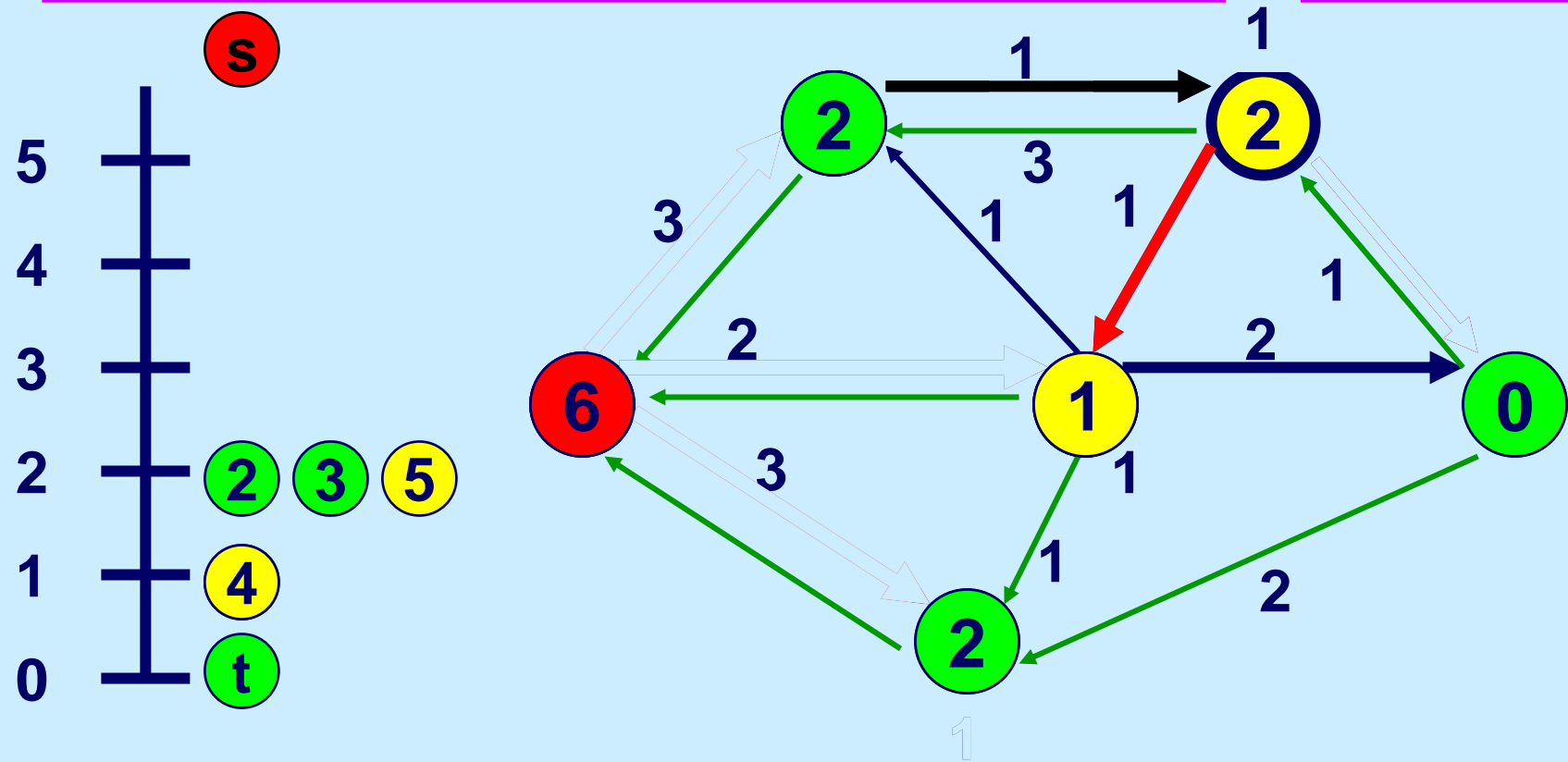
If there is a non-saturating push in $(5, 2)$, then node 5 is active before the push and inactive afterwards. Node 2 is active after the push.

Either F decreases by $d(5) - d(2) = 1$ or else F decreases by $d(5)$.

Conclusion: every non-saturating push leads to a withdrawal of at least 1.

Recall: The entire point of this analysis is to bound the number of non-saturating pushes.

Effect on F of saturating pushes



Consider a saturating push from node 5 to node 4.

In this case, node 4 becomes active.

F increases by $d(4) = 1$.

More on Saturating Pushes

Recall: a “deposit” is an increase in F . The value of the deposit is the increase in F .

A saturating push in $(5, 4)$ can lead to a deposit. The max deposit occurs when node 4 becomes active and node 5 stays active. The increase would be $d(4)$

The max deposit from a push in arc (i,j) is $d(j) < 2n$. There are at most nm saturating pushes.

Conclusion: the total deposits from saturating pushes is at most $2n^2m$.

Putting it all together

$$F = \sum_{j \text{ active}} d(j). \quad \text{Initially, } F < n^2. \quad \text{At end, } F = 0.$$

Deposit: an increase in F

Withdrawal: a decrease in F .

Each non-saturating push is a withdrawal of at least 1.

The total deposits from relabels is at most $2n^2$.

The total deposits from saturating pushes is at most $2n^2m$.

Conclusion: the number of non-saturating pushes is at most $n^2 + 2n^2 + 2n^2m = O(n^2m)$.

Summary

- ◆ **Preflow-Push Algorithm**
- ◆ **Relies on pushes in a single arc and on relabels**
- ◆ **Number of relabels is $O(n^2)$**
- ◆ **Number of pushes is $O(n^2m)$**
- ◆ **Bottleneck operation: non-saturating pushes.**
- ◆ **Use of potential functions to prove time bounds.**