15.082J and 6.855J and ESD.78J
October 21, 2010

Max Flows 4

Overview of today's lecture

- Scaling Algorithms
- Potential function analysis
- The Excess Scaling Algorithm
  - $O(n^2 \log U)$ non-saturating pushes, where $U = 1 + \max\{u_{ij} : (i, j) \in A\}$
  - $O(nm + n^2 \log U)$ running time.
- A proof that Highest Level Preflow Push uses $O(n^2m^{1/2})$ non-saturating pushes.

Scaling Algorithms

1. Define a concept called Δ-optimal, where Δ is some positive integer, and where a 1-optimal solution is optimal for the original problem.

2. Develop a subroutine that efficiently determines Δ_0-optimum solution where

   Δ_0 is some (possibly large) power of 2

3. Develop a subroutine Improve-Approx that transforms a Δ-optimal solution into a Δ/2-optimal solution.

Generic Scaling Algorithm

```
Δ := 2^K  for some selected
   value K
determine a Δ-optimal
   solution x
while Δ > 1 do
    y := ImproveApprox(x, Δ)
    x := y
    Δ := Δ/2
```

The Capacity Scaling Algorithm

- A flow x is called ***Δ-maximum*** if there is no augmenting path in G(x) of capacity ⊗ or more.
  - Note. If $\Delta \geq U$, then x = 0 is Δ-maximum.
    $U = 1 + \max \{u_{ij} : (i, j) \in A\}$

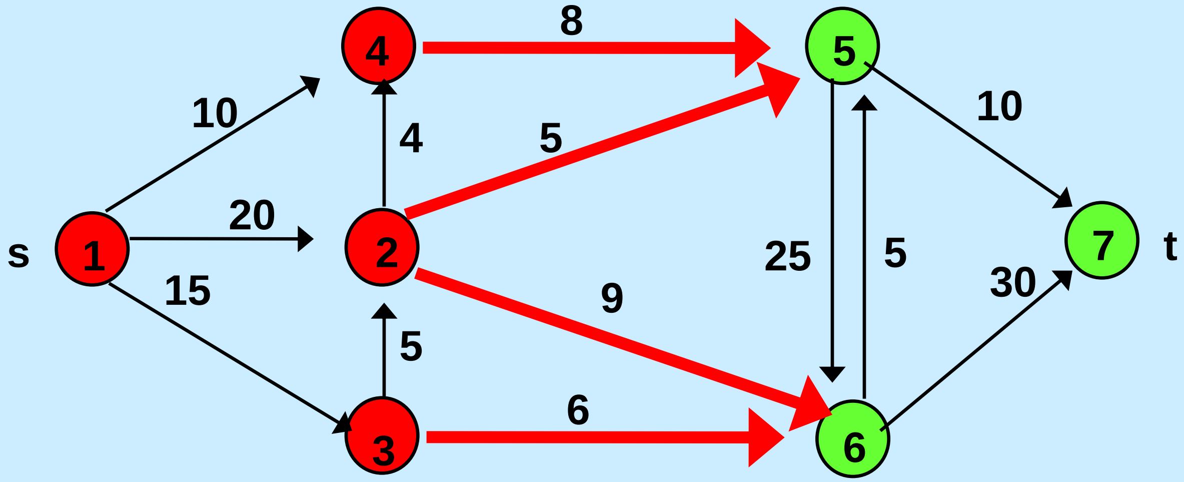
- Subroutine ***ImproveApprox(x,Δ)***: takes a flow that is ⊗-maximum and outputs a flow that is ⊗/2-maximum.

- We refer to a path in G(x) as a **Δ-augmenting** if it is an s-t path whose capacity is at least Δ.

```
ImproveApprox(x,Δ)
    while there is a Δ/2-augmenting path in G(x) do
        find a Δ/2-augmenting path P in G(x);
        augment flow along P;
        update residual capacities and data structures;
```

Analysis of Capacity Scaling

Lemma. At the beginning of the Δ-scaling phase, there is a an s-t cut (S, T) such that the capacity of each arc (i, j) from S to T is less than Δ. S = { j : there is a path P of capacity ≥ Δ from s to j}



$\boxed{\Delta = 10}$

The residual capacity of (S, T) is less than mΔ.

Analysis of Capacity Scaling

Corollary. The number of augmentations per scaling phase is at most 2m.

Proof. Each augmentation reduces the residual capacity of the cut by at least $\Delta/2$.

Lemma. The number of times that ImproveApprox is called is at most $\lceil \log U \rceil$

Proof. Initially $\Delta = 2^K$, where $K = \lceil \log U \rceil$

At each subsequent iteration Δ is halved.

The algorithm stops when $\Delta = 1$.

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

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

The running time can be improved to $O(nm \log U)$

A preview

Next: an algorithm that is based on scaling excesses rather than capacities.

Based on preflow-push

At the Δ-scaling phases, all excesses are less than Δ.

A Review of 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

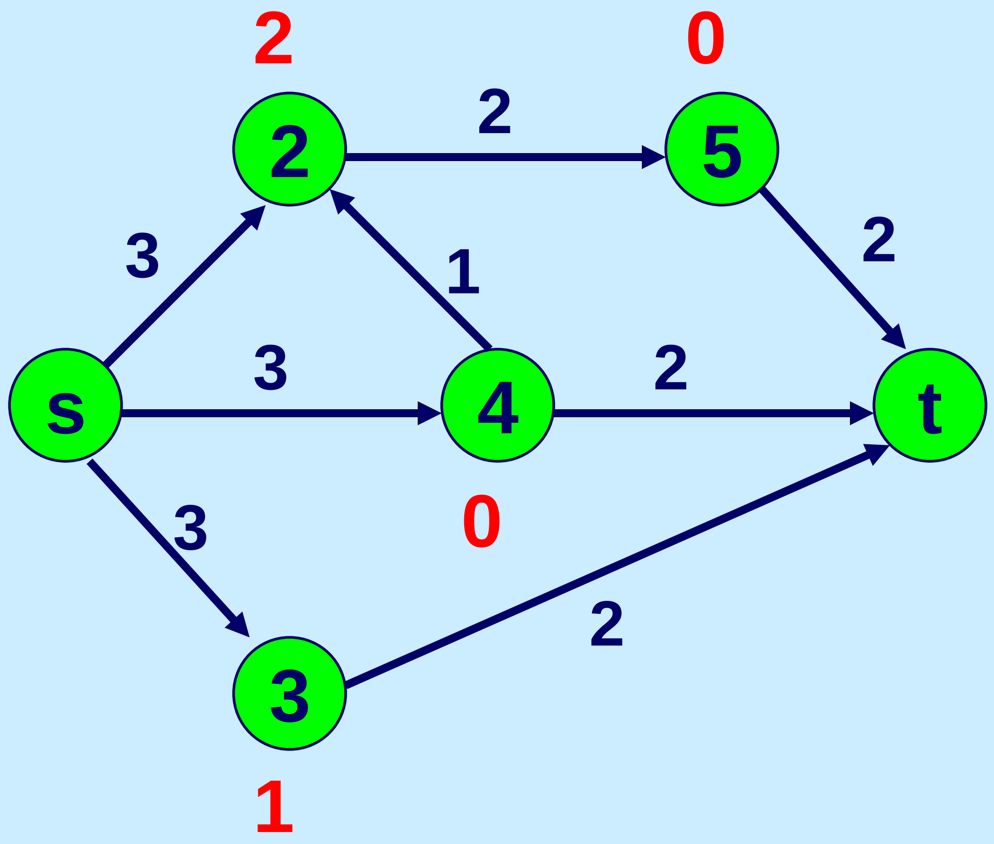
$$e(i) = \sum_{j \in N} x_{ji} - \sum_{j \in N} x_{ij} \geq 0,$$

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 ≠ s, t is the flow in minus the flow out.

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

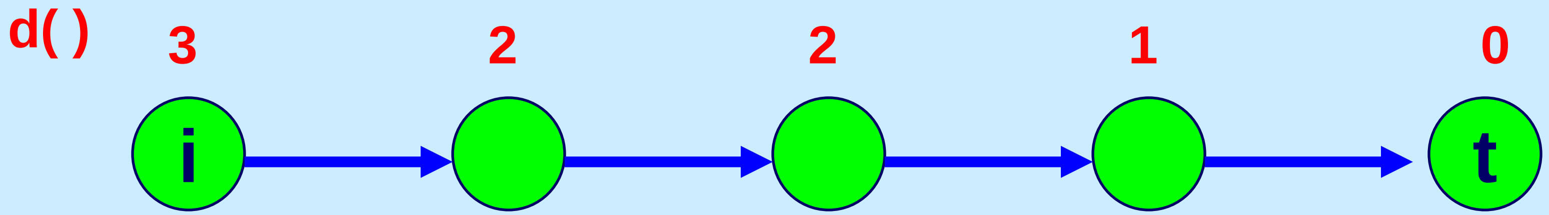
Distance Labels

Distance labels **d( ) 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( ) be a valid distance label. Then d(i) is a lower bound on the distance from i to t in the residual network.***



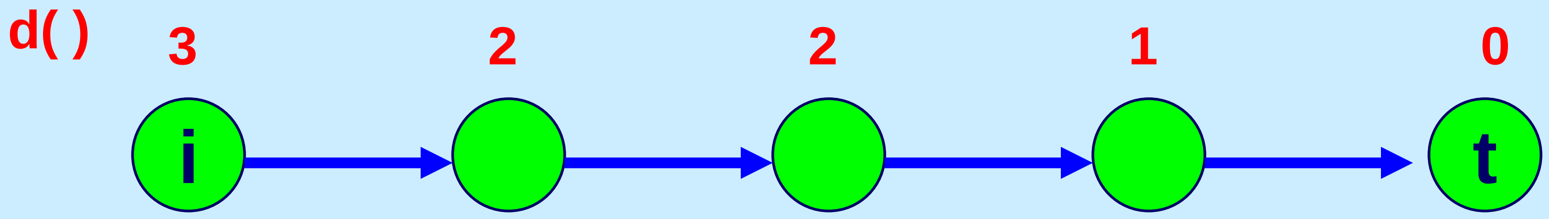
P = the shortest path from i to t in G(x)

Distance labels and gaps

We say that there is a gap at a distance level k ($0 < k < n$) if there is no node with distance label k.

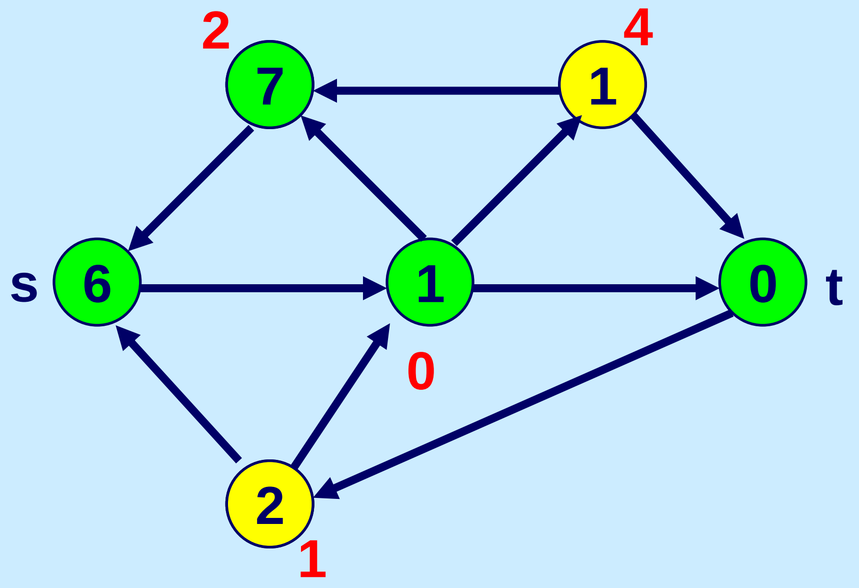
Lemma. Suppose there is a gap at distance level k. Then for any node j with $d(j) > k$, there is no path from j to t in the residual network.

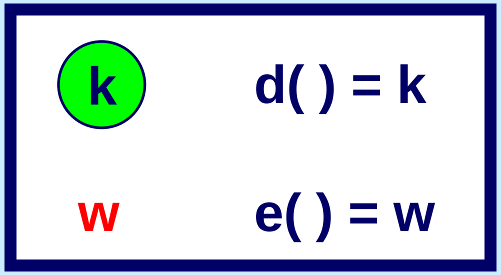
Proof. The shortest path from j to t would have to pass through a node whose distance level is k.



P = the shortest path from i to t in G(x)

Active nodes in the residual network





A node j in G\{s} is active if:

- e(j) > 0 and
- there is no gap at a distance level less than d(j)

The preflow push algorithm will push flow from active nodes “towards the sink”, relying on d( ).

Goldberg-Tarjan Preflow Push Algorithm

```
Procedure Preprocess
    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;
```

```
Algorithm PREFLOW-PUSH;
    preprocess;
    while there is an active node i do
        select an active node i;
        push/relabel(i);
    convert the max preflow into a max flow
```

Note: the "while loop" ends when there are no active nodes; i.e., if e(j) > 0, then d(j) is above a gap.

The Excess Scaling Algorithm

- A preflow x is called ***Δ-maximum*** if $e(j) < \Delta$ for all $j \neq s, t$.
    - Note. If $\Delta \geq U$, then the preflow after the preprocess step is Δ-maximum.

- If a preflow is 1-maximum and if $d(s) = n$, then the preflow is a maximum flow.

- Subroutine ***ImproveApprox(x,Δ)***: takes a preflow x that is ⊗-maximum and outputs a preflow that is ⊗/2-maximum.

Excess Scaling Algorithm

```
Δ := 2^K  where K = ⌈ log U ⌉
Preprocess
while Δ > 1 do
    y := ImproveApprox(x, Δ)
    x := y
    Δ := Δ/2
convert the maximum
  preflow x into a maximum
  flow
```

Improve Approx

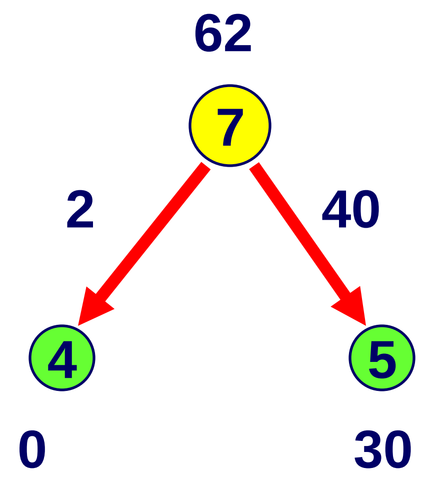
We say that a node is **Δ-active** if

1. $e(i) \geq \Delta$
2. node i is not above a gap If a node i is above a gap, then there is no path from i to t in G(x).

Subroutine *ImproveApprox(x,Δ)*

```
while the G(x) has a Δ/2-active node j do
    among Δ/2-active nodes, choose i with
        minimum distance label
    perform push/relabel(i) where the amount
        pushed in (i, j) is  min ($\Delta/2$, $r_{ij}$)
```

Δ = 64

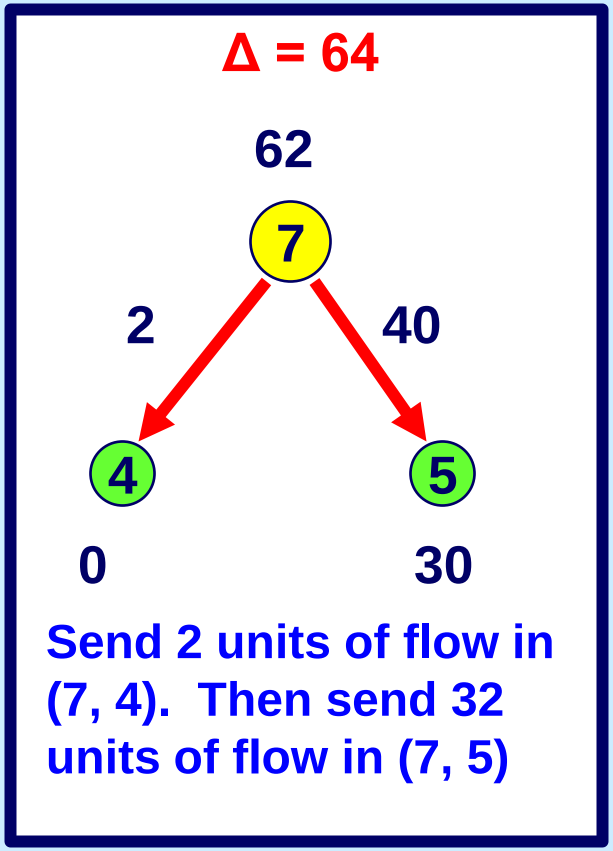


Send 2 units of flow in (7, 4). Then send 32 units of flow in (7, 5)

Lemmas about pushing

Lemma 1. Throughout ImproveApprox, $e(j) < \Delta$ for all active nodes j.

Proof. If we push in (i, j), then $e(j) < \Delta/2$ before the push, and the amount pushed is at most $\Delta/2$.



Send 2 units of flow in (7, 4). Then send 32 units of flow in (7, 5)

Lemma 2. Each non-saturating push sends exactly $\Delta/2$ units of flow.

Proof. The amount pushed in (i, j) is min $(\Delta/2, r_{ij})$.

Analysis of the Excess Scaling Algorithm

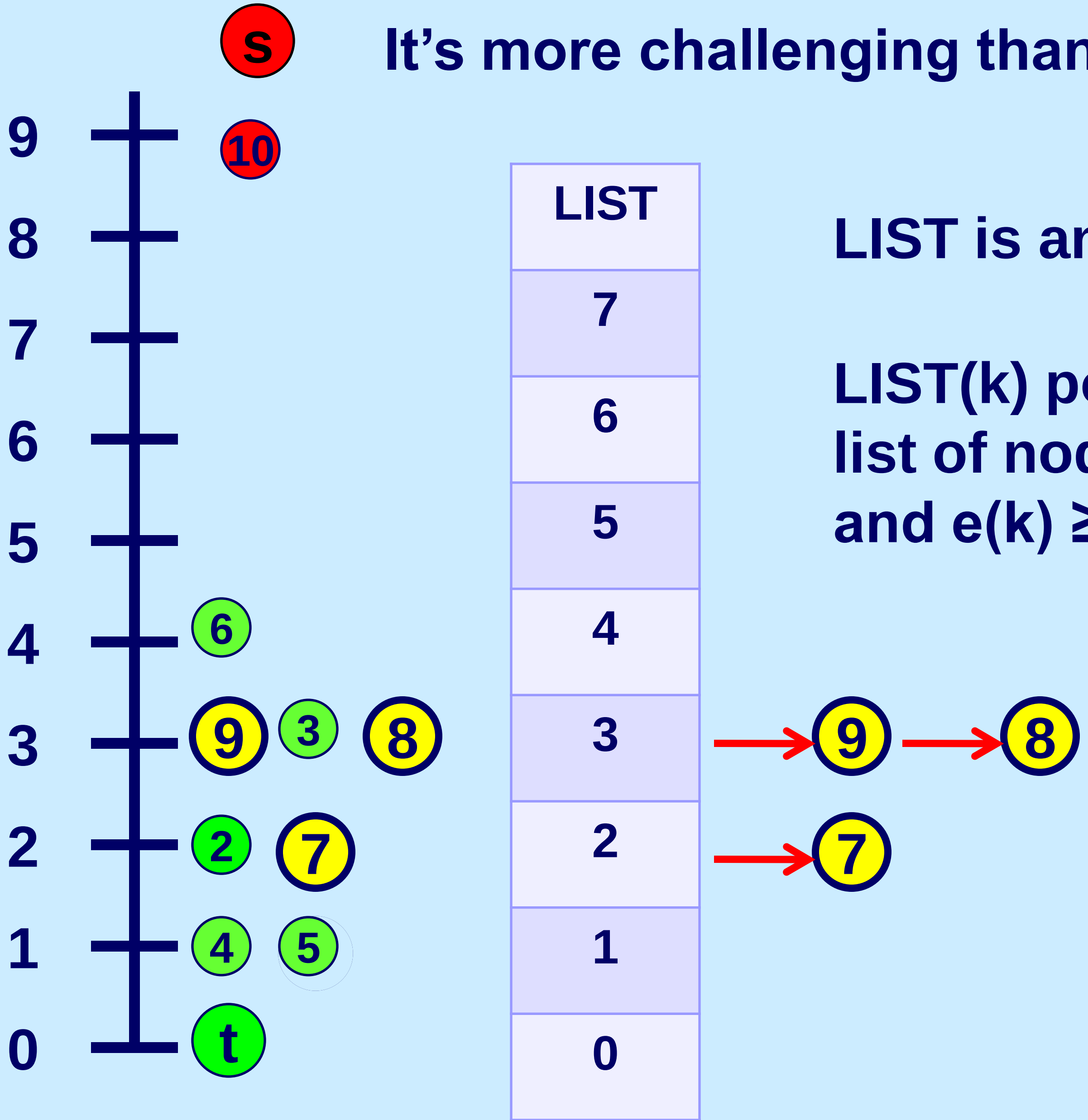
Theorem. The Excess Scaling Algorithm finds a maximum flow in $O(nm + n^2 \log U)$ steps.

Proof. We have already shown the following in the analysis of preflow-push algorithms.

1. If it terminates, it terminates with a max flow
2. The time spent in all steps other than nonsaturating pushes is O(nm).
3. What remains to be proved:
   - A $\Delta/2$-active node with lowest distance label can be selected in O(1) steps.
   - The number of nonsaturating pushes is $O(n^2 \log U)$. For this we will rely on a new potential function.

Selecting Δ-active nodes efficiently

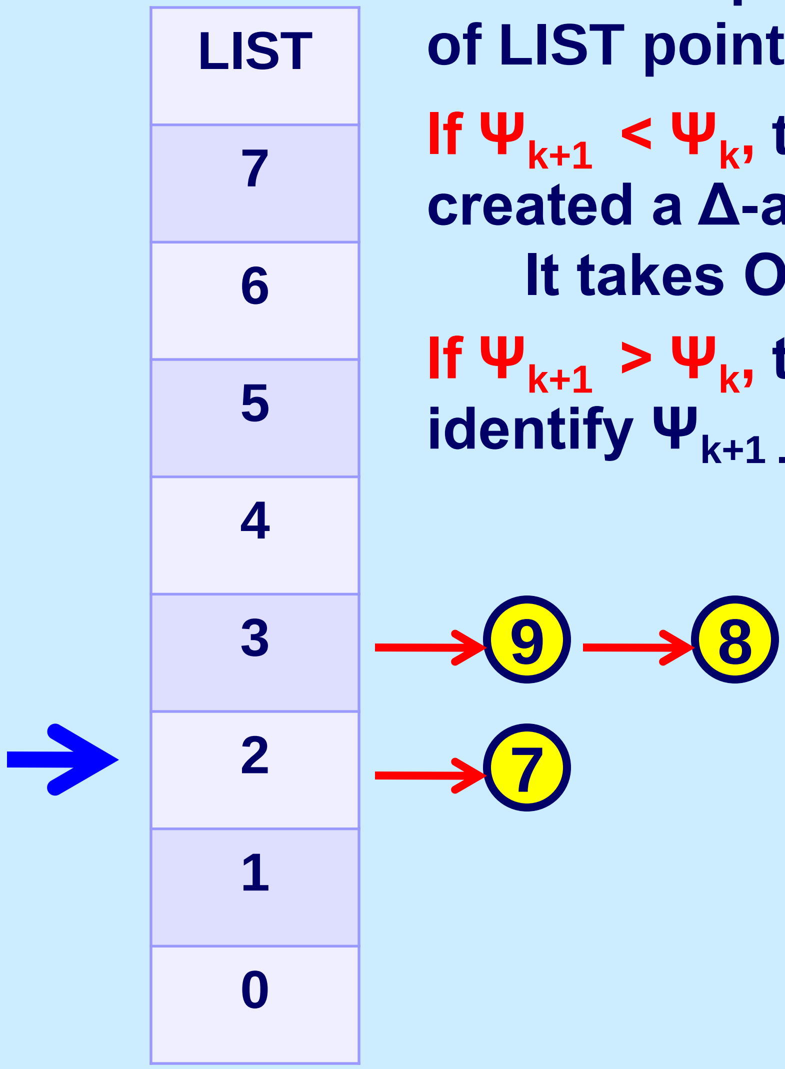
It’s more challenging than one might guess.



LIST is an array of size n.

LIST(k) points to a linked list of nodes i with d(i) = k, and e(k) ≥ Δ/2

Selecting Δ-active nodes efficiently



Maintain a pointer to LIST. Let Ψ_k be the index of LIST pointed to at an iteration k.

If $\Psi_{k+1} < \Psi_k$, then the push at the k-th iteration created a Δ-active node at level $\Psi_{k+1} = \Psi_k - 1$.

It takes O(1) steps to identify Ψ_{k+1}.

If $\Psi_{k+1} > \Psi_k$, then it takes $O(\Psi_{k+1} - \Psi_k)$ steps to identify Ψ_{k+1}.

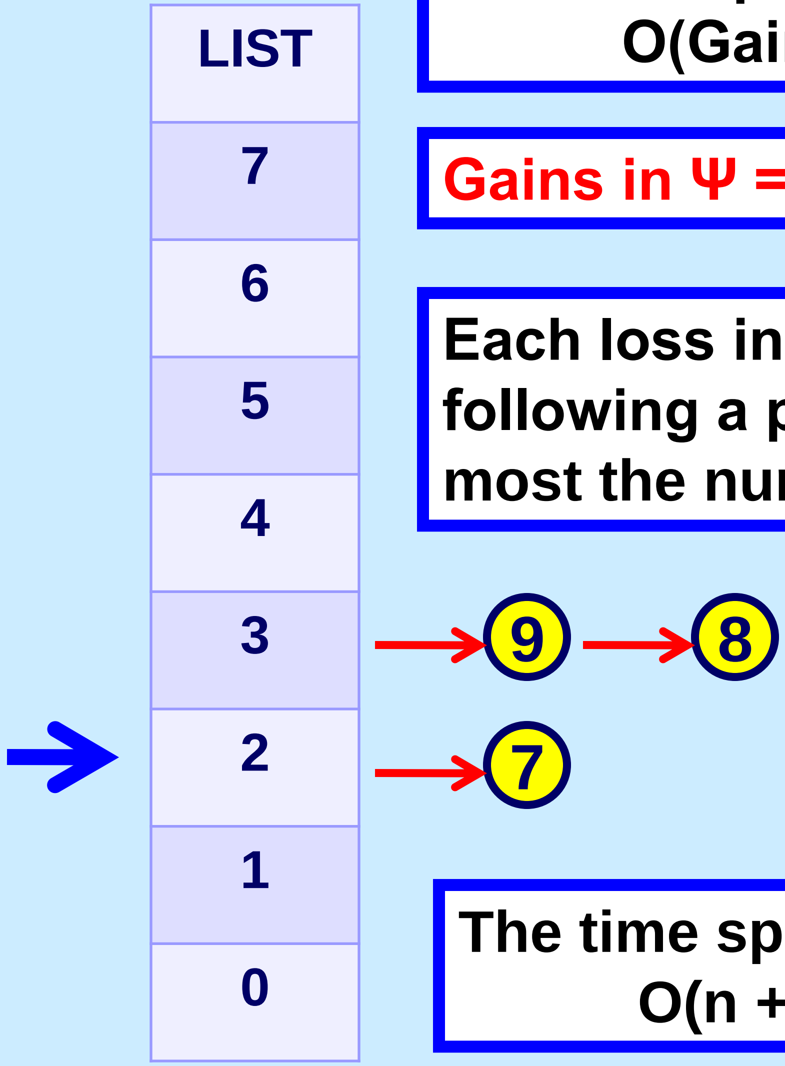
View Ψ_k as a potential function.

Let $\delta_k = \Psi_{k+1} - \Psi_k$.

Then $\delta_k \geq -1$.

Losses in potential =
Initial Potential +
+ Gains in potential
- Final potential

Selecting Δ-active nodes efficiently



The time spent scanning list =
O(Gains in Ψ + Losses in Ψ)

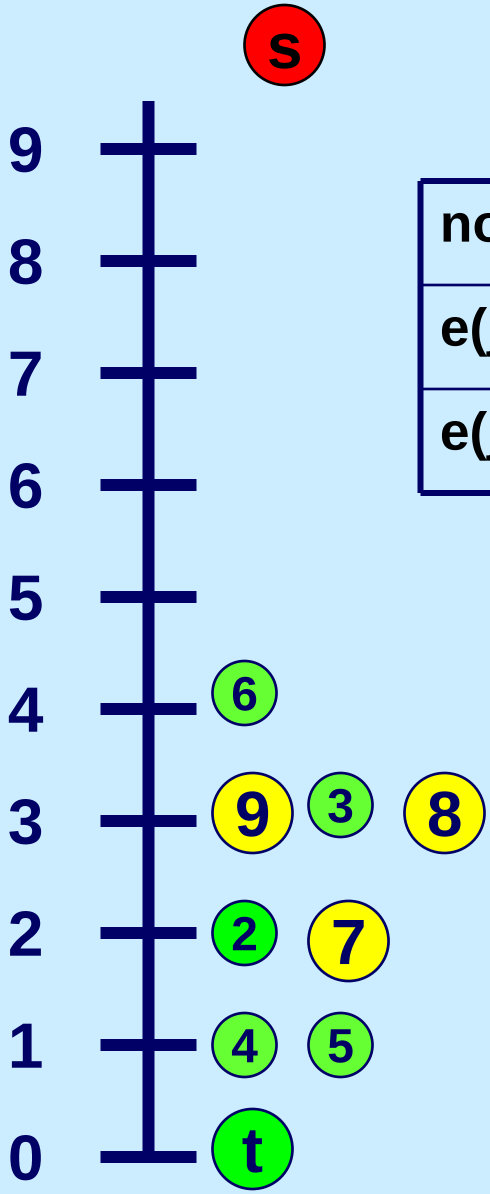
Gains in Ψ = Final Ψ + Losses in Ψ – Initial Ψ

Each loss in Ψ is exactly 1, and it occurs following a push. So, the losses in Ψ are at most the number of pushes.

Gains in Ψ ≤
n + number of pushes

The time spent scanning list =
O(n + number of pushes)

A potential function for bounding NSAT



$\Phi = \sum_{j \in N} e(j)\, d(j) / \Delta$

node	2	3	4	5	6	7	8	9
e(j)	4	0	1	0	5	35	55	40
e(j)d(j)	8	0	1	0	20	70	165	120

$\Phi = 384 / 64 = 6$

The potential is the "gravitational potential" measured in units of Δ

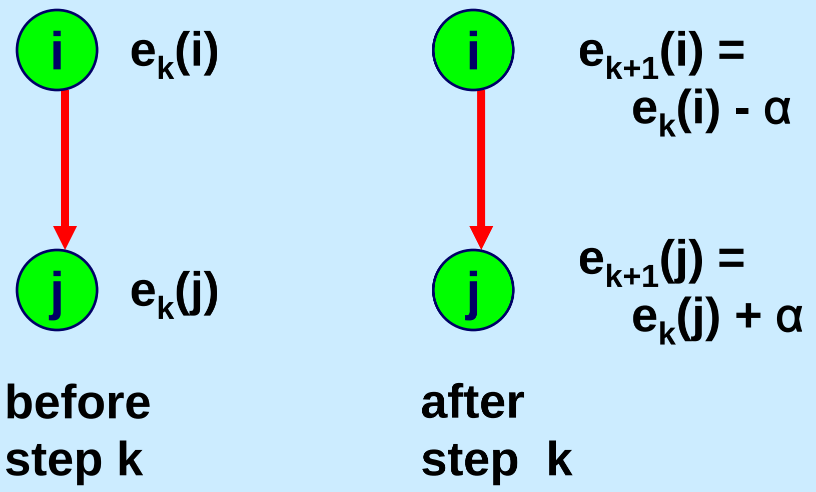
Losses in $\Phi = \Phi_0 +$ Gains in $\Phi - \Phi_f$

What if the k-th iteration is a push?

Every push decreases Φ

Suppose that α units of flow are sent in (i, j) at the k-th iteration.

$$\Phi_k = \sum_{j \in N} e_k(j)\, d_k(j) / \Delta$$



i
j
$e_k(i)$
$e_k(j)$
before step k
i
j
$e_{k+1}(i) = e_k(i) - \alpha$
$e_{k+1}(j) = e_k(j) + \alpha$
after step k

$$\delta_k = \Phi_{k+1} - \Phi_k = [\, e_{k+1}(i)\, d_{k+1}(i) + e_{k+1}(j)\, d_{k+1}(j)$$

$$- \; e_k(i)\, d_k(i) - e_k(j)\, d_k(j) \,] / \Delta$$

$$= [\, -\alpha\, d_k(i) + \alpha\, d_k(j) \,] / \Delta = -\alpha / \Delta.$$

$d_k(\,) = d_{k+1}(\,)$

$d_k(u) = d_k(v) + 1$

A Nonsat push at step k sends $\Delta/2$ units of flow.

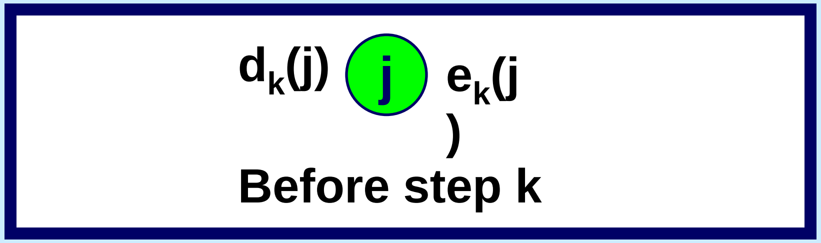
In this case, $\delta_k = -\frac{1}{2}$.

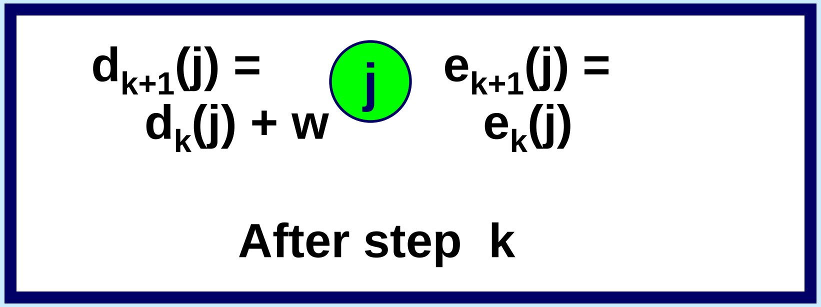
What if the k-th iteration is a relabel?

Every relabel increases Φ.

$\Phi_k = \sum_{j \in N} e_k(j)\, d_k(j) / \Delta$

Suppose that $d_{k+1}(j) = d_k(j) + w$.





$$\delta_k = \Phi_{k+1} - \Phi_k = [\, e_{k+1}(j)\, d_{k+1}(j) - e_k(j)\, d_k(j)\,]/\Delta$$

$$= [\, e_k(j)\,(d_k(j) + w) - e_k(j)\, d_k(j)\,]/\Delta$$

$$= e_k(j)\, w / \Delta \le w$$

Increasing d(j) by w, increases Φ by at most w.

Bounding NSAT

$$\Phi_k = \sum_{j \in N} e_k(j)\, d_k(j) / \Delta$$

NSAT(Δ) = number of nonsat pushes in Δ-scaling phase

$$\text{NSAT}(\Delta)/2 \leq \text{Losses in } \Phi = \Phi_0 + \text{Gains in } \Phi - \Phi_f$$

$$\text{NSAT}(\Delta)/2 \leq n^2 + n^2 - 0 = O(n^2)$$

Theorem. **The total number of nonsaturating pushes over all scaling phases is $O(n^2 \log U)$.**

Mental Break

You are entitled to receive something if you bring a raccoon’s head to the town hall in Henniker, NH. What is it?

$10.

In 1980, a Las Vegas hospital suspended workers because of what they were betting on. What was it?

They were betting when patients would die.

In what year did Christians begin celebrating December 25 as Jesus Christ’s birthday?

440 C.E.

Mental Break

The NERF ball is a popular children's toy.

What does NERF stand for?

Nothing.

In advertising displays that include a clock, what time is most frequently given?

10:10.

Babe Ruth kept something under his hat to keep cool. What was it?

A cabbage leaf. He changed it every 2 innings.

Highest Level Pushing

The highest level pushing algorithm refers to the special case of the Goldberg-Tarjan preflow push algorithm in which pushes are from an active node with maximum distance label.

Theorem. The running time of the highest level pushing algorithm is $O(n^2m^{.5})$.

Note: Selecting an active node with highest distance level is carried out similarly to selecting a Δ-active node at the lowest level.

It remains to prove that the number of nonsaturating pushes is $O(n^2m^{.5})$.

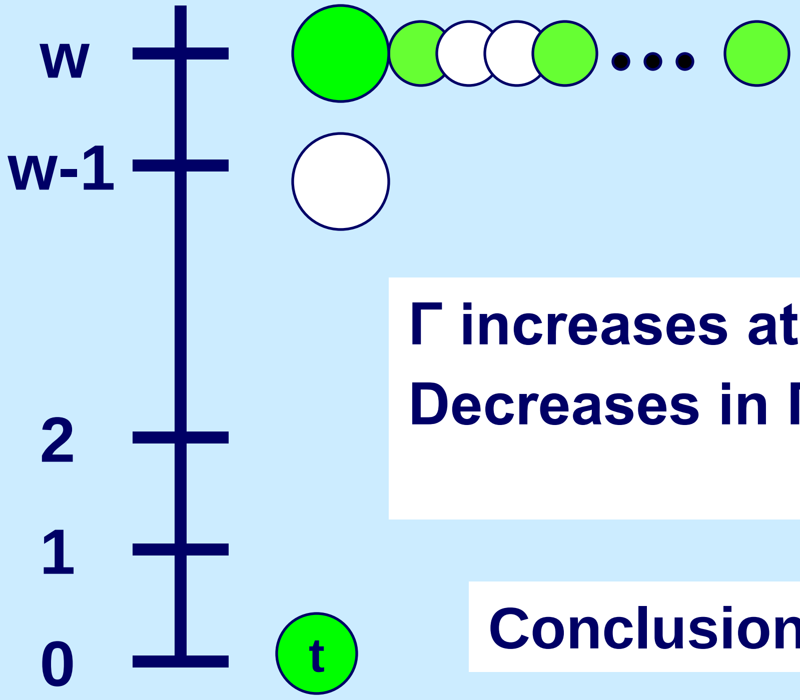
- The analysis is involved, but it is much easier than the analysis in the text.

Phases

A **phase** consists of a consecutive sequence of pushes from nodes at the same level.

Theorem. The number of phases is $O(n^2)$.

Proof. Let Γ be the highest level of an active node. It is a potential function. A phase ends in one of two ways:



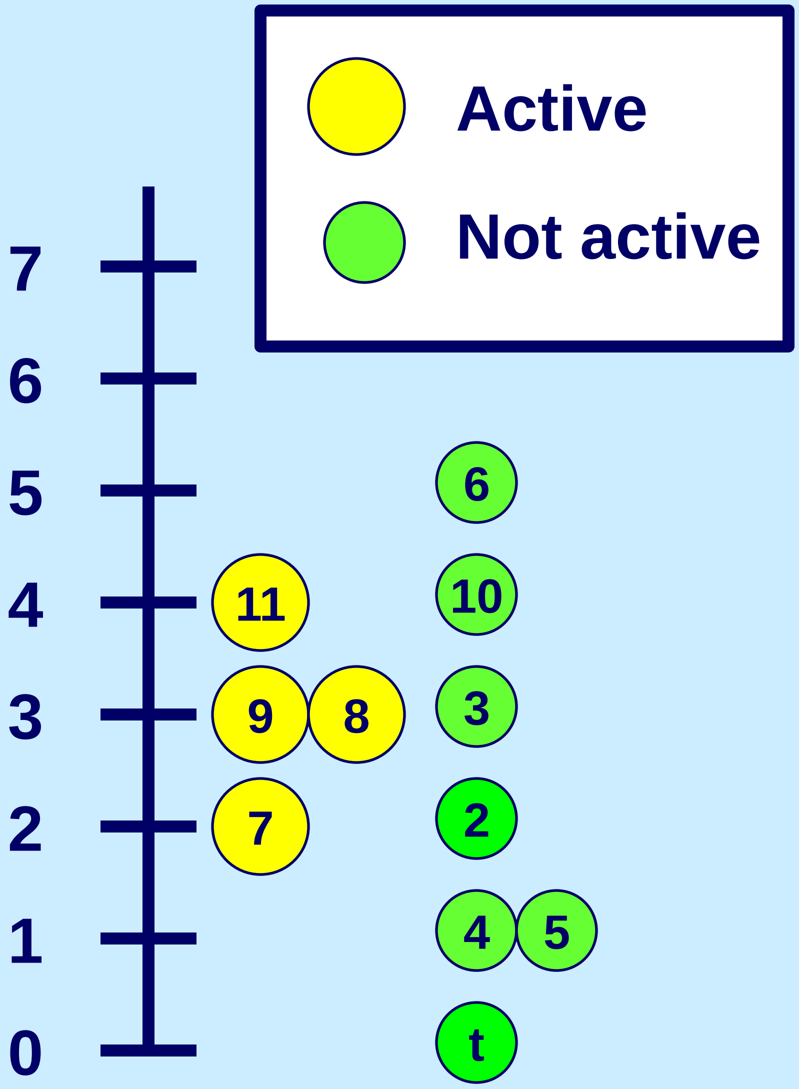
1. Γ increases (because of a relabel)
2. Γ decreases (no more active nodes at level w)

Γ increases at most n^2 . (bound on relabels)

$$\text{Decreases in } \Gamma = \Gamma_0 + \text{Increases in } \Gamma - \Gamma_f$$
$$\leq n + n^2 - 0 = n^2$$

Conclusion: there are $O(n^2)$ phases.

A potential function for highest level pushing.



Nodes 7, 8, 9, and 11 are active.

$\Phi(j)$ is the number of active nodes i with $d(i) \geq d(j)$.

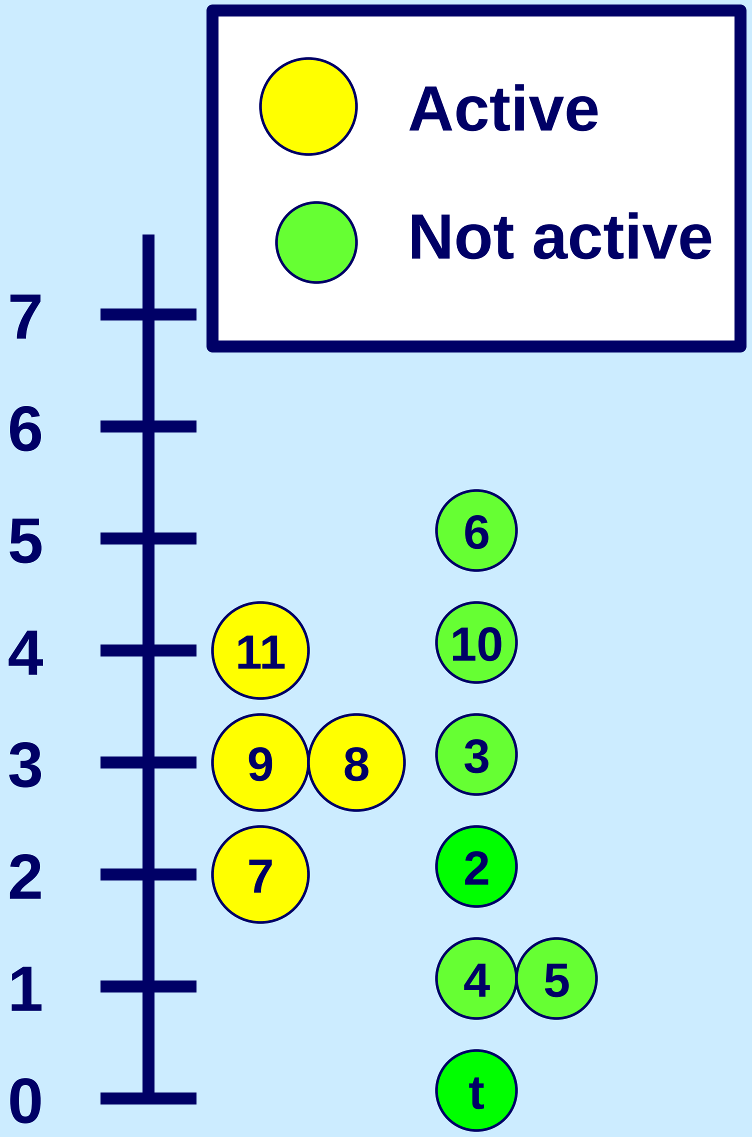
e.g., $\Phi(6) = 0$, $\Phi(10) = 1$

$\Phi(3) = 3$, $\Phi(2) = 4$

$$\Phi = \sum_{j \in N} \Phi(j)$$

$$\Phi = 0 + 1 + 3 + 4 \times 4 = 20$$

The effect of a node becoming active



Suppose that node i is made active.

Φ(j) increases by 1 if d(j) ≤ d(i).

Conclusion: if a node is made active and if there are no other changes in potential, then Φ increases by less than n.

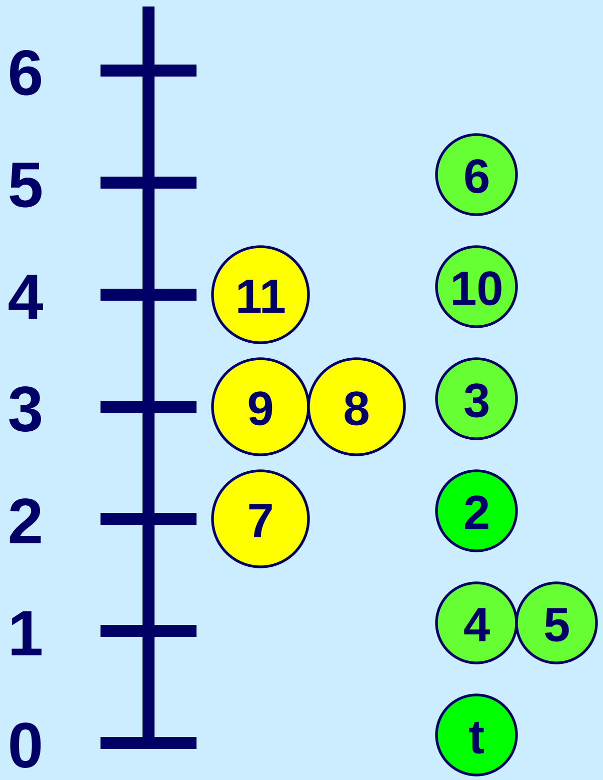
Bounds on δ_k

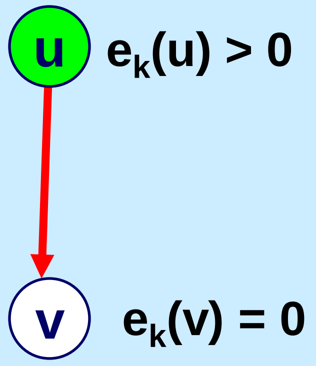
Consider the case that the k-th step is a saturating push in arc (u,v).

$\Phi_k(j)$ is the number of active nodes i with $d(i) \geq d(j)$.

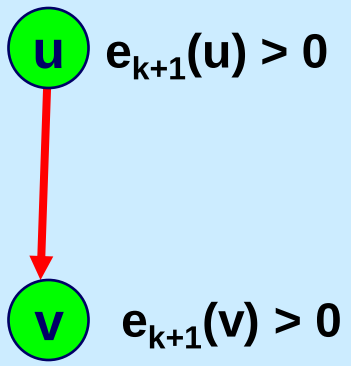
$\Phi_k = \sum_{j \in N} \Phi_k(j)$

$d_{k+1} = d_k$





lowest possible
Φ before step k



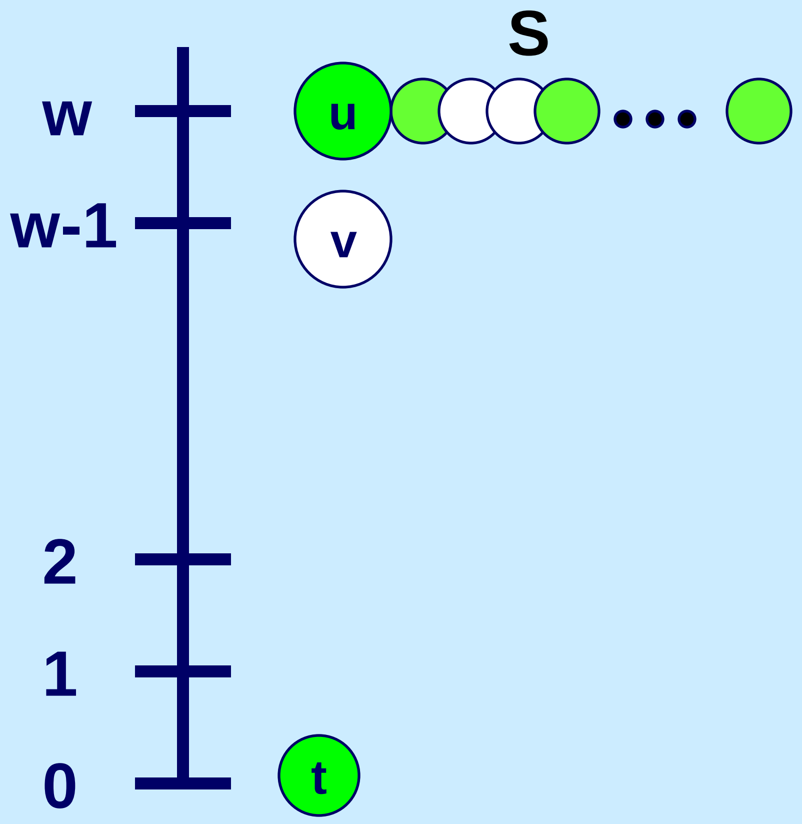
highest possible
Φ after step k

$\delta_k = \Phi_{k+1} - \Phi_k < n$

Bounds on δ_k

Consider the case that the k-th step is a non-saturating push in arc (u,v).

Let S be the set of nodes at level d(u) = w



🟢 u $e_k(u) > 0$ ⚪ u $e_{k+1}(u) = 0$

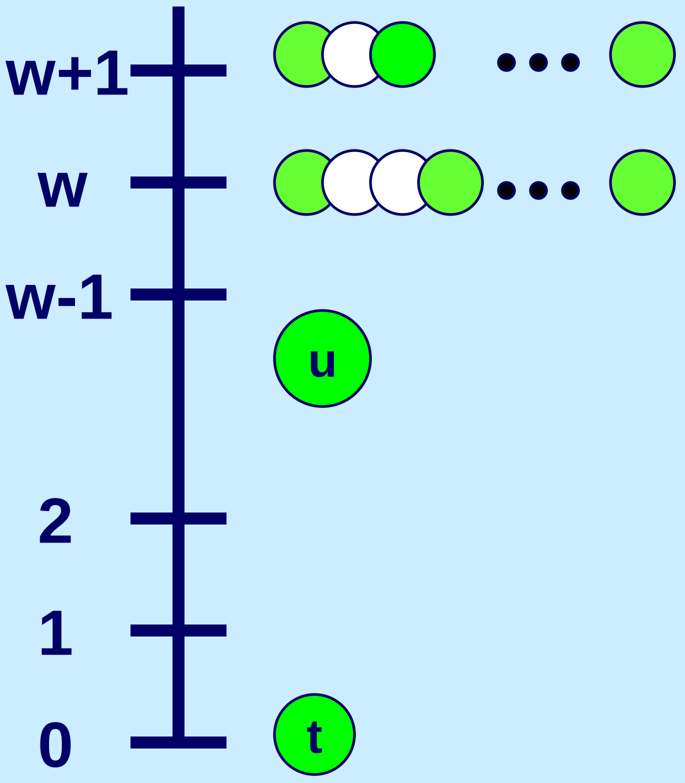
for $j \in S$, $\Phi_{k+1}(j) = \Phi_k(j) - 1$

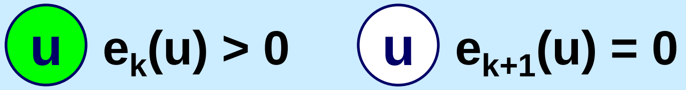
for $j \notin S$, $\Phi_{k+1}(j) \le \Phi_k(j)$

$\delta_k = \Phi_{k+1} - \Phi_k \le -|S|$

Bounds on δ_k

Consider the case that the k-th step is a relabel of u.





$e_k(u) > 0$ $e_{k+1}(u) = 0$

$$\boxed{\text{for } j \in N,\ \Phi_{k+1}(j) \le \Phi_k(j) + 1}$$

$$\boxed{\delta_k = \Phi_{k+1} - \Phi_k < n}$$

Bounding NSAT

$$\text{Losses in } \Phi = \Phi_0 + \text{Gains in } \Phi - \Phi_f$$
$$\leq n^2 + n^2m + n^3 - 0 = O(n^2m)$$

We say that a phase is large if it has at least K nonsaturating pushes. Otherwise, it is small. The number of nonsaturating pushes is NSAT-large + NSAT-small.

NSAT-small $\leq K \times$ number of phases $= O(Kn^2)$

NSAT-large $\times K \leq$ Losses $= O(n^2m)$
NSAT-large $\leq$ Losses/K $= O(n^2m/K)$

Choose $K = m^{.5}$.
Then NSAT $= O(Kn^2 + n^2m/K) = O(n^2m^{.5})$.

Review of Lecture

Scaling techniques are useful when it is quicker to solve an optimization problem starting from the optimal solution of a closely related problem.

- capacity scaling
- excess scaling

Potential functions are useful when the total running time is less than the bounds obtained by adding up running times bounds for each step. It permits a kind of global analysis.

- time to select active nodes in excess scaling
- NSAT pushes for excess scaling
- number of phases for highest level pushing alg.
- NSAT pushes for highest level pushing alg.