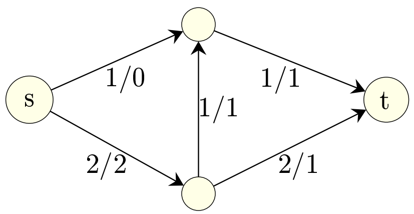
The Maximum Flow Problem

Input:

- a directed graph $G = (V, E)$, source node $s \in V$, sink node $t \in V$
- edge capacities $cap : E \to I\!R_{\geq 0}$



Goal:

- compute a flow of maximal value, i.e.,
- a function $f : E \to I\!R_{\geq 0}$ satisfying the capacity constraints and the flow conservation constraints

(1) $\quad 0 \leq f(e) \leq cap(e) \quad$ for every edge $e \in E$

(2) $\quad \sum_{e;\, target(e)=v} f(e) = \sum_{e;\, source(e)=v} f(e) \quad$ for every node $v \in V \backslash \{s, t\}$

- and maximizing the net flow into t.

Some Notation and First Properties

- the excess of a node v: $excess(v) = \sum_{e;\, target(e)=v} f(e) - \sum_{e;\, source(e)=v} f(e)$
- in a flow: all nodes except s and t have excess zero.
- the value of a flow $= val(f) = excess(t)$

Clearly: the net flow into t is equal to the next flow out of s.

Lemma 1 $excess(t) = -excess(s)$

The proof is short and illustrates an important technique

$$excess(s) + excess(t) = \sum_{v \in V} excess(v) = 0$$

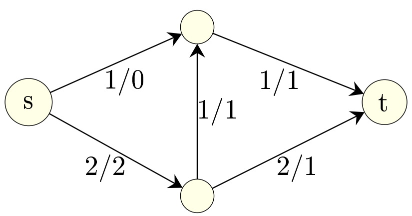
- the first equality holds since $excess(v) = 0$ for $v \neq s, t$.
- the second equality holds since the flow accross any edge $e = (v, w)$ appears twice in this sum
  - positively in $excess(w)$ and negatively in $excess(v)$

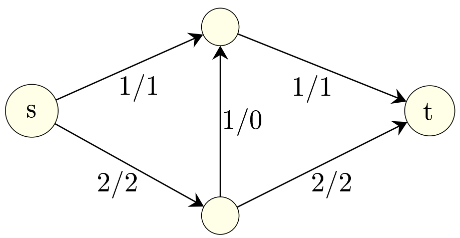
Cuts

- a subset S of the nodes is called a cut. Let $T = V \setminus S$
- S is called an (s,t)-cut if $s \in S$ and $t \in T$.
- the *capacity* of a cut is the total capacity of the edges leaving the cut,

$$cap(S) = \sum_{e \in E \cap (S \times T)} cap(e).$$

- a cut S is called *saturated* if $f(e) = cap(e)$ for all $e \in E \cap (S \times T)$ and $f(e) = 0$ for all $e \in E \cap (T \times S)$.





Cuts and Flows

Lemma 2 *For any flow f and any (s,t)-cut*

- $val(f) \leq cap(S)$.
- *if S is saturated,* $val(f) = cap(S)$.

Proof: We have

$$\begin{aligned} val(f) &= -excess(s) = -\sum_{u \in S} excess(u) \\ &= \sum_{e \in E \cap (S \times T)} f(e) - \sum_{e \in E \cap (T \times S)} f(e) \leq \sum_{e \in E \cap (S \times T)} cap(e) \\ &= cap(S). \end{aligned}$$

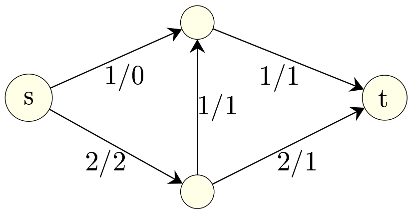
For a saturated cut, the inequality is an equality. ∎

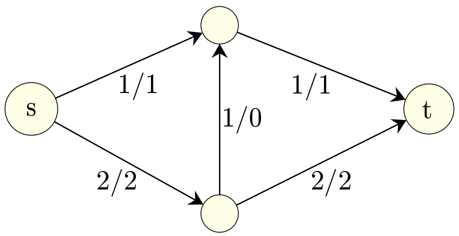
Remarks:

- A saturated cut proves the maximality of a flow.
- For every maximal flow there is a saturated cut proving its maximality ($\Longrightarrow$)

The Residual Network

- let f be a flow in $G = (V, E)$
- the *residual network* G_f captures possible changes to f
  - same node set as G
  - for every edge $e = (v, w)$ up to two edges e' and e'' in G_f
    - if $cap(e) < f(e)$, we have an edge $e' = (v, w) \in G_f$
      residual capacity $r(e') = cap(e) - f(e)$.
    - if $f(e) > 0$, we have an edge $e'' = (w, v) \in G_f$
      residual capacity $r(e'') = f(e)$.
- two flows and the corresponding residual networks



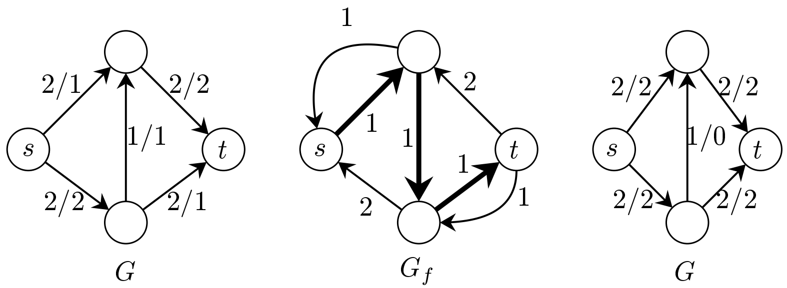


The Max-Flow-Min-Cut Theorem

Theorem 1 *Let f be an (s,t)-flow, let G_f be the residual network with respect to f, and let S be the set of nodes that are reachable from s in G_f.*

a) *If $t \in S$ then f is not maximum.*

b) *If $t \notin S$ then S is a saturated cut and f is maximum.*



An illustration of part a)

Max-Flow-Min-Cut: The Proof of Part a)

If t is reachable from s in G_f, f is not maximal

- Let p be any simple path from s to t in G_f
- Let δ be the minimum residual capacity of any edge of p. Then $\delta > 0$.
- We construct a flow f' of value $val(f) + \delta$. Let (see Figure on preceding slide)

$$f'(e) = \begin{cases} f(e) + \delta & \text{if } e' \text{ is in } p \\ f(e) - \delta & \text{if } e'' \text{ is in } p \\ f(e) & \text{if neither } e' \text{ nor } e'' \text{ belongs to } p. \end{cases}$$

- f' is a flow and $val(f') = val(f) + \delta$.

a path in G_f: $s \longrightarrow v_1 \longrightarrow v_2 \longrightarrow v_3 \longrightarrow v_4 \longrightarrow v_5 \longrightarrow t$

the corresponding path in G:

Max-Flow-Min-Cut: The Proof of Part b)

If t cannot be reached from s in G_f, f is maximal.

- Let S be the set of nodes reachable from s and let $T = V \setminus S$.
- There is no edge (v, w) in G_f with $v \in S$ and $w \in T$.
- Hence
  - $f(e) = cap(e)$ for any e with $e \in E \cap (S \times T)$ and
  - $f(e) = 0$ for any e with $e \in E \cap (T \times S)$
- Thus S is saturated and f is maximal.

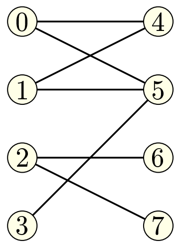
G_f G

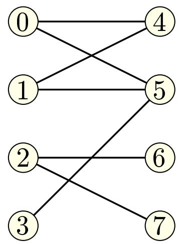
The Ford-Fulkerson Algorithm

- start with the zero flow, i.e., $f(e) = 0$ for all e.
- construct the residual network G_f
- check whether t is reachable from s.
  - if not, stop
  - if yes, increase flow along an augmenting path, and iterate
- each iteration takes time $O(n+m)$
- if capacities are arbitrary reals, the algorithm may run forever
- integral capacities, say in $[0 \mathinner{..} C]$, v^* = value of the maximum flow $\le nC$
  - all flows constructed are integral (and hence final flow is integral)
    - Proof by induction: if current flow is integral, residual capacities are integral and hence next flow is integral
  - every augmentation increases flow value by at least one
  - running time is $O((n+m)v^*)$; this is good if v^* is small

Bipartite Matching

- given a bipartite graph $G = (A \cup B, E)$, find a maximal matching
- matching M, a subset of the edges, no two of which share an endpoint
- reduces easily to network flow
  - add a source s, edges (s,a) for $a \in A$, capacity one
  - add a sink t, edges (b,t) for $b \in B$, capacity one
  - direct edges in G from A to B, capacity $+\infty$
  - integral flows correspond to matchings
  - Ford-Fulkerson takes time $O(nm)$ since $v^* \le n$, can be improved to $O(\sqrt{n}m)$





The Theorem of Hall

Theorem 2 *A bipartite graph $G = (A \cup B, E)$ has an A-perfect matching (= a matching of size $|A|$) iff for every subset $A' \subset A$, $|\Gamma(A')| \ge |A'|$, where $\Gamma(A')$ is the set of neighbors of the nodes in A'.*

condition is clearly necessary; we need to show sufficiency

- assume that there is no A-perfect matching
- then flow in the graph defined on preceding slide is less than $|A|$
- and hence minimum cut has capacity less than $|A|$.
- consider a minimum (s,t)-cut (S,T).
- let $A' = A \cap S$, $A'' = A \cap T$, $B' = B \cap S$, $B'' = B \cap T$
- no (!!!) edge from A' to B'' and hence $\Gamma(A') \subseteq B'$
- flow $= |B'| + |A''| < |A| = |A'| + |A''|$
- thus $|B'| < |A'|$

A Theoretical Improvement for Integral Capacities

- modify Ford-Fulkerson by always augmenting along a flow of maximal residual capacity
- essentially replaces v^* by $m \log v^*$ in time bound, good for large v^*
- practical value is minor, but proof method is interesting
- **Lemma 3** *Max-res-cap-path can be determined in time $O(m \log m)$.*
- **Lemma 4** *$O(m + m \log\lceil v^*/m \rceil)$ augmentations suffice*
- **Theorem 3** *running time becomes: $T = O((m + m \log\lceil v^*/m \rceil) m \log m)$*

Lemma 5 *Max-res-cap-path can be determined in time $O(m \log m)$.*

- sort the edges of G_f in decreasing order of residual capacity
- let e_1, e_2, ... , $e_{m'}$ be the sorted list of edges
- want to find the minimal i such that $\{e_1, \ldots, e_i\}$ contains a path from s to t
- for fixed i we can test existance of path in time $O(n + m)$
- determine i by binary search in $O(\log m)$ rounds.

Lemma 6 $O(m + m \log\lceil v^*/m\rceil)$ *augmentations suffice*

- a flow can be decomposed into at most m paths
  - start with a maximal flow f
  - repeatedly construct a path from s to t, saturate it, and subtract from f
- augmentation along max-res-cap-path increases flow by at least $1/m$ of dist to v^*
- let g_i be the diff between v^* and the flow value after the i-th iteration
- $g_0 = v^*$
- if $g_i > 0$, $g_{i+1} \le g_i - \max(1, g_i/m) \le \min(g_i - 1, (1 - 1/m)g_i)$
- $g_i \le (\frac{m-1}{m})^i g_0$ and hence $g_i \le m$ if i is such that $(\frac{m-1}{m})^i g_0 \le m$.
- this is the case if $i \ge \log_{m/(m-1)}(v^*/m) = \frac{\log(v^*/m)}{\log m/(m-1)}$
- $\log(m/(m-1)) = \log(1 + 1/(m-1)) \ge 1/(2m)$ for $m \ge 10$
- number of iterations $\le m + 2m \log(v^*/m)$

Dinic's Algorithm (1970), General Capacities

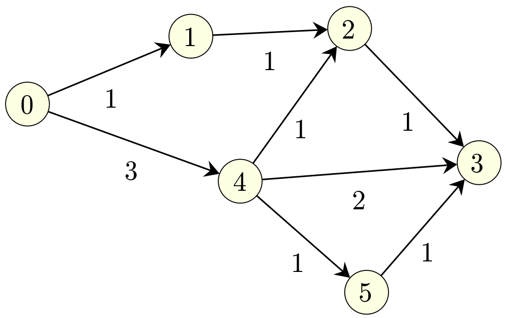
- start with the zero flow f
- construct the layered subgraph L_f of G_f
- if t is not reachable from s, stop
- construct a blocking flow f_b in L_f and augment to f, repeat

- in L_f nodes are on layers according to their BFS-distance from s and only edges going from layer i to layer $i+1$ are retained
- L_f is constructed in time $O(m)$ by BFS
- blocking flow: a flow which saturates one edge on every path from s to t

- the number of rounds is at most n, since the depth of L_f grows in each round (without proof, but see analysis of # of saturating pushes in preflow-push alg)
- a blocking flow can be computed in time $O(nm)$
- $T = O(n^2m)$

An Example Run of Dinic's Algorithm

I will illustrate the sequence of residual graphs and residual level graphs.



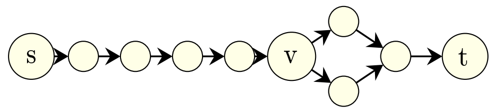
The Computation of Blocking Flows

- maintain a path p starting at s, initially $p = \epsilon$, let $v = tail(p)$
- if $v = t$, increase f_b by saturating p, remove saturated edges, set p to the empty path (**breakthrough**)
- if $v = s$ and v has no outgoing edge, stop
- if $v \neq t$ and v has an outgoing edge, **extend** p by one edge
- if $v \neq t$ and v has no outgoing edge, **retreat** by removing last edge from p.

- running time is $\#_{extends} + \#_{retreats} + n \cdot \#_{breakthroughs}$
- $\#_{breakthroughs} \leq m$, since at least one edge is saturated
- $\#_{retreats} \leq m$, since one edge is removed
- $\#_{extends} \leq \#_{retreats} + n \cdot \#_{breakthroughs}$, since a retreat cancels one extend and a breakthrough cancels n extends
- running time is $O(m + nm) = O(nm)$

Preflow-Push Algorithms

- f is a preflow (Karzonov (74)): $excess(v) \geq 0$ for all $v \neq s, t$
- residual network with respect to a preflow is defined as for flows
- Idea: preflows give additional flexibility



- manipulate a preflow by operation $push(e, \delta)$
  - Preconditions:
    - e is residual, i.e., $e = (v, w) \in E_f$
    - v has excess, i.e, $excess(v) > 0$
    - δ is feasible, i.e, $\delta \leq \min(excess(v), res_f(e))$
  - Action: push δ units of flow from v to w
    - decrease $excess(v)$ by δ, increase $excess(w)$ by δ, modify f and adapt E_f (remove e if it now saturated, add its reversal)
- **Question:** Which push to make?
- **Answer:** push towards t, but what does this mean?

The Level Function (Goldberg/Tarjan)

- a simple and highly effective notion of "towards t"
- arrange the nodes on levels, $d(v)$ = level number of $v \in I\!N$
- at all times: $d(t) = 0$, $d(s) = n$
- call an edge $e = (v, w)$ *eligible* iff $e \in E_f$ and $d(w) < d(v)$
- and only push across eligible edges, i.e., from higher to lower level

Question: What to do when v has positive excess but no outgoing eligible edge?

Answer: lift it up, i.e., increase $d(v)$ by one (relabel v)

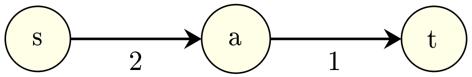
The Generic Push-Relabel Algorithm

```
set f(e) = cap(e) for all edges with source(e) = s;
set f(e) = 0 for all other edges;
set d(s) = n and d(v) = 0 for all other nodes;

while there is a node v ≠ s,t with positive excess
{ let v be any such node node;
  if there is an eligible edge e = (v,w) in G_f
  { push δ across e for some δ ≤ min(excess(v), res_cap(e)); }
  else
  { relabel v; }
}
```

- obvious choice for δ: $\delta = \min(excess(v), res\_cap(e))$
- push with $\delta = res\_cap(e)$ *saturating push*
- push with $\delta < res\_cap(e)$ *non-saturating push*
- need to bound the number of relabels and the number of pushes

A Sample Run



and here comes the sequence of residual graphs (residual capacities are shown)

No Steep Edges

an edge $e = (v, w) \in G_f$ is called *steep* if $d(w) < d(v) - 1$, i.e., if it reaches down by two or more levels.

Lemma 7 *The algorithm maintains a preflow and does not generate steep edges. The nodes s and t stay on levels 0 and n, respectively.*

Proof:

- the algorithm maintains a preflow by the restriction on δ
- after initialization: edges in G_f go sidewards or upwards
- when v is relabeled, no edge in G_f out of v goes down. After relabeling, edges out of v go down at most one level.
- a push across an edge $e = (v, w) \in G_f$ may add the edge (w, v) to G_f. This edge goes up.
- s and t are never relabeled

∎

The Maximum Level Stays Below $2n$

Lemma 8 *If v is active then there is a path from v to s in G_f. No distance label ever reaches $2n$.*

Proof: Let S be the set of nodes that are reachable from v in G_f and let $T = V \setminus S$. Then

$$\sum_{u \in S} excess(u) = \sum_{e \in E \cap (T \times S)} f(e) - \sum_{e \in E \cap (S \times T)} f(e),$$

There is no edge $(v, w) \in G_f$ with $v \in S$ and $w \notin S$. Thus, $f(e) = 0$ for every $e \in E \cap (T \times S)$. We conclude $\sum_{u \in S} excess(v) \leq 0$.

Since s is the only node whose excess may be negative and since $excess(v) > 0$ we must have $s \in S$.

Assume that a node v is moved to level $2n$. Since only active nodes are relabeled this implies the existence of a path (and hence simple path) in G_f from a node on level $2n$ to s (which is on level n). Such a path must contain a steep edge. ∎

Partial Correctness

Theorem 4 *When the algorithm terminates, it terminates with a maximum flow.*

Proof: When the algorithm terminates, all nodes different from s and t have excess zero and hence the algorithm terminates with a flow. Call it f.

In G_f there can be no path from s to t since any such path must contain a steep edge (since s is on level n, t is on level 0). Thus, f is a maximum flow by the max-flow-min-cut theorem. ∎

In order to prove termination, we bound the number of relabels, the number of saturating pushes and the number of non-saturating pushes.

The former two quantities are easily bounded.

We have to work harder for the number of non-saturating pushes.

On the Number of Relabels and Saturating Pushes

Lemma 9 *There are at most $2n^2$ relabels and at most nm saturating pushes.*

Proof:

- no distance label ever reaches $2n$.
- therefore, each node is relabeled at most $2n$ times
- the number of relabels is therefore at most $2n^2$.

- a saturating push across an edge $e = (v, w) \in G_f$ removes e from G_f.
- **Claim:** v has to be relabeled at least twice before the next push across e and hence there can be at most n saturating pushes across any edge.
  - only a push across e^{rev} can again add e to G_f.
  - for this to happen w must be lifted by two levels, ...

∎

On the Number of Non-Saturating Pushes: Scaling

```
/* scaling push-relabel algorithm (Ahuja-Orlin) for integral capacities */
set f(e) = cap(e) for all edges with source(e) = s and f(e) = 0 for all other edges;
set d(s) = n and d(v) = 0 for all other nodes;
set Δ = 2^⌈log max_e cap(e)⌉;

while ( Δ > 1 )
{ while there is a node v ≠ s,t with excess(v) ≥ Δ/2
  { let v be the lowest (!!!) such node;
    if there is an eligible edge e = (v,w) in G_f
    { push δ across e for δ = min(Δ/2, res_cap(e)); }
    else
    { relabel v; }
  }
  Δ = Δ/2;
}
```

- excesses are bounded by Δ, i.e., at all times and for all $v \neq t$: $excess(v) \leq \Delta$
- a non-saturing push moves $\Delta/2$ units of flow

On the Number of Saturating Pushes in Ahuja-Orlin

Lemma 10 *The number of non-saturating pushes is at most $4n^2 + 4n^2\lceil \log U \rceil$, where U is the largest capacity*

We use a potential function argument (let $V' = V \setminus \{s, t\}$)

$$\Phi = \sum_{v \in V'} d(v) \frac{excess(v)}{\Delta}$$

- $\Phi \geq 0$ always, $\Phi = 0$ initially
- total decrease of $\Phi \leq$ total increase of Φ
- a relabel increases Φ by at most one
- every push decreases Φ
- a non-saturating push decreases Φ by 1/2
- a change of Δ increases Φ by at most $2n^2$
- Δ is changed $\lceil \log U \rceil$ times
- $(1/2)\#_{non\ sat\ pushes} \leq$ total decrease $\leq$ total increase $\leq 2n^2 + 2n^2\lceil \log U \rceil$

On the Number of Sat Pushes in the Generic Algorithm

- pushes are made as large as possible, i.e., $\Delta = \min(excess(v), res\_cap(e))$
- a non-saturating push deactivates the source of the push
- (persistence) when an active node v is selected, pushes out of v are performed until either v becomes inactive (because of a non-saturating push out of v) or until there are no eligible edges out of v anymore. In the latter case v is relabeled.
- we study three rules for the selection of active nodes.

***Arbitrary*:** an arbitrary active node is selected.
$\#_{non\ sat\ pushes} = O(n^2 m)$, Goldberg and Tarjan

***FIFO*:** the active nodes are kept in a queue and the first node in the queue is always selected. When a node is relabeled or activated the node is added to the rear of the queue, $\#_{non\ sat\ pushes} = O(n^3)$, Goldberg.

***Highest-Level*:** an active node on the highest level, i.e., with maximal d-value is selected, $\#_{non\ sat\ pushes} = O(n^2\sqrt{m})$, Cheriyan and Maheshwari

The Arbitrary Rule

Lemma 11 *When the Arbitrary-rule is used, the number of non-saturating pushes is $O(n^2 m)$.*

Proof:

$$\Phi = \sum_{v \in V'; v \text{ is active}} d(v).$$

- $\Phi \geq 0$ always, and $\Phi = 0$ initially.
- a non-saturating push decreases Φ by at least one, since it deactivates the source of the push (may activate the target)
- a relabeling increases Φ by one.
- a saturating push increases Φ by at most $2n$, since it may activate the target
- total increase of $\Phi \leq n^2 + nm2n = n^2(1 + 2m)$
- $\#_{non\ sat\ pushes} \leq$ total increase of Φ

∎

The FIFO Rule

- active nodes are in a queue, head of queue is selected for pushing/relabeling
- relabeled and activated nodes are added to the rear of the queue
- we split the execution into phases
- first phase starts at the beginning of the execution
- a phase ends when all nodes that were active at the beginning of the phase have been selected from the queue
- each node is selected at most once in each phase: $\#_{non\ sat\ pushes} \leq n \cdot \#_{phases}$

Lemma 12 *When the FIFO-rule is used, the number of phases is $O(n^2)$.*

Proof: Use $\Phi = \max\{d(v)\ ;\ v \text{ is active }\}$

- $\Phi \geq 0$ always, and $\Phi = 0$ initially.
- a phase containing no relabel operation decreases Φ by at least one, since all nodes on the hightest level become inactive.
- a phase containing a relabel operation increases Φ by at most one, since a relabel increases the highest level by at most one.

∎

Lemma 13 *When the Highest-Level-rule is used, $\#_{non\ sat\ pushes} = O(n^2\sqrt{m})$.*

Warning: Proof in Ahuja/Magnanti/Orlin is wrong, proof here Cheriyan/M

- let $K = \sqrt{m}$. For a node v, let $d'(v) = |\{w; d(w) \leq d(v)\}|/K$.
- potential function $\Phi = \sum_{v; v \text{ is active}} d'(v)$.
- execution is split into phases
- phase = all pushes between two consecutive changes of $d^* = \max\{d(v)\ ;\ v \text{ is active }\}$
- phase is *expensive* if it contains more than K non-sat pushes, *cheap* otherwise.

We show:

(1) The number of phases is at most $4n^2$.
(2) The number of non-saturating pushes in cheap phases is at most $4n^2K$.
(3) $\Phi \geq 0$ always, and $\Phi \leq n^2/K$ initially.
(4) A relabeling or a sat push increases Φ by at most n/K.
(5) A non-saturating push does not increase Φ.
(6) An expensive phase with $Q \geq K$ non-sat pushes decreases Φ by at least Q.

(1) The number of phases is at most $4n^2$.
(2) The number of non-saturating pushes in cheap phases is at most $4n^2K$.
(3) $\Phi \geq 0$ always, and $\Phi \leq n^2/K$ initially.
(4) A relabeling or a sat push increases Φ by at most n/K.
(5) A non-saturating push does not increase Φ.
(6) An expensive phase with $Q \geq K$ non-sat pushes decreases Φ by at least Q.

- Suppose that we have shown (1) to (6).
- (4) and (5) imply total increase of $\Phi \leq (2n^2 + mn)n/K$
- above + (3): total decrease can be at most this number plus n^2/K
- $\#_{non\ sat\ pushes\ in\ expensive\ phases} \leq (2n^3 + n^2 + mn^2)/K$.
-

above + (2)	$\#_{non\ sat\ pushes} \leq (2n^3 + n^2 + mn^2)/K + 4n^2K$
since $n \leq m$:	$\#_{non\ sat\ pushes} \leq 4mn^2/K + 4n^2K = 4n^2(m/K + K)$
$K = \sqrt{m}$:	$\#_{non\ sat\ pushes} \leq 8n^2\sqrt{m}$.

(1) The number of phases is at most $4n^2$: we have $d^* = 0$ initially, $d^* \geq 0$ always, and only relabels increase d^*. Thus, d^* is increased at most $2n^2$ times, decreased no more than this, and hence changed at most $4n^2$ times.

(2) The number of non-saturating pushes in cheap phases is at most $4n^2K$: follows immediately from (1) and the definition of a cheap phase.

(3) $\Phi \geq 0$ always, and $\Phi \leq n^2/K$ initially: obvious

(4) A relabeling or a sat push increases Φ by at most n/K: follows from the observation that $d'(v) \leq n/K$ for all v and at all times.

(5) A non-saturating push does not increase Φ: observe that a non-sat push across an edge (v, u) deactivates v, activates u (if it is not already active), and that $d'(u) \leq d'(v)$.

(6) An expensive phase with $Q \geq K$ non-sat pushes decreases Φ by at least Q: consider an expensive phase containing $Q \geq K$ non-sat pushes. d^* is constant during a phase and hence all Q non-saturating pushes must be out of nodes at level d^*. The phase is finished either because level d^* becomes empty or because a node is moved from level d^* to level $d^* + 1$. In either case, we conclude that level d^* contains $Q \geq K$ nodes at all times during the phase. Thus, each non-saturating push in the phase decreases Φ by at least one (since $d'(u) \leq d'(v) - 1$ for a push from v to u).