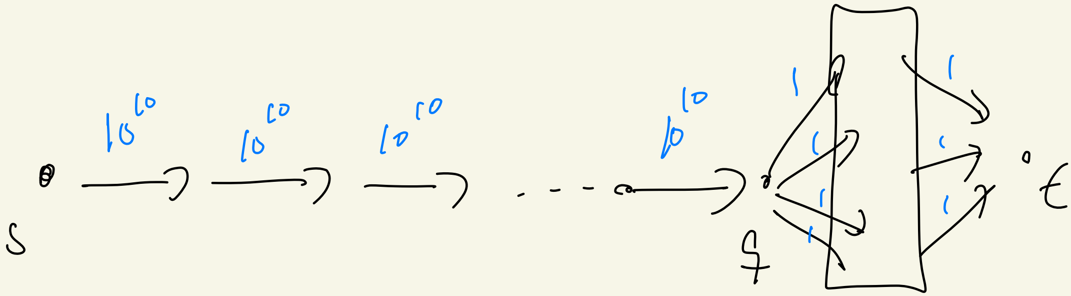
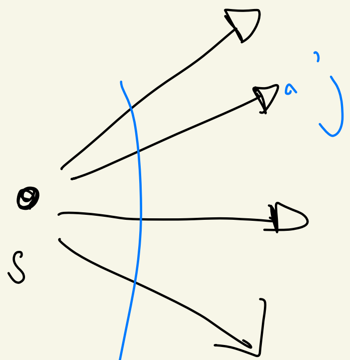


idea: try to send as much as possible out of s:



fill them

$x_{sj} \leftarrow a_{sj}$

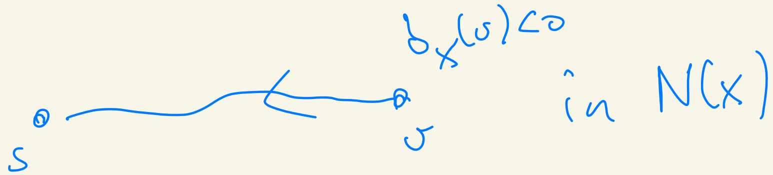
A preflow in $N=(V\cup\{s,t\}, A, \ell\equiv 0, u)$ is a flow x s.t.:

$$b_x(v)\le 0 \quad \forall v\in V-s$$

(so s is the only vertex with positive balance)

Note that <u>every</u> preflow decomposes into path and cycle flows along $P_1,P_2,\dots,P_\alpha,\ C_1,\dots,C_\beta$ where each P_i starts in s and ends in a vertex v with $b_x(v)<0$

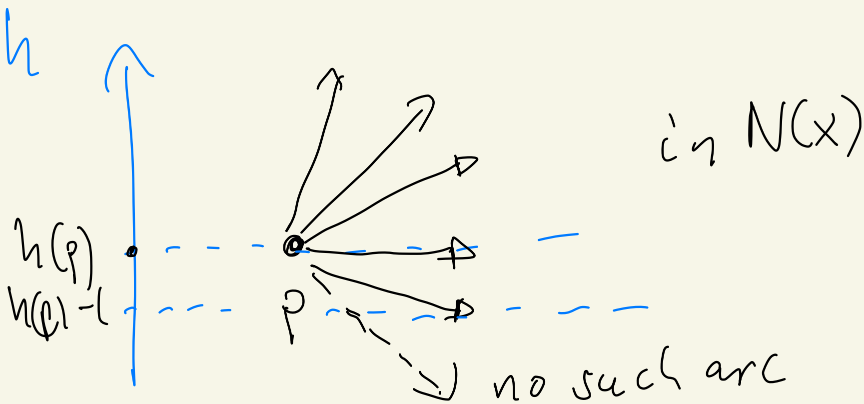
<u>Lemma</u> if x is a preflow and $b_x(v)<0$ then $N(x)$ contains a (v,s)-path



h is a height function wrt x if

$$h(s) = n, \quad h(t) = 0,$$

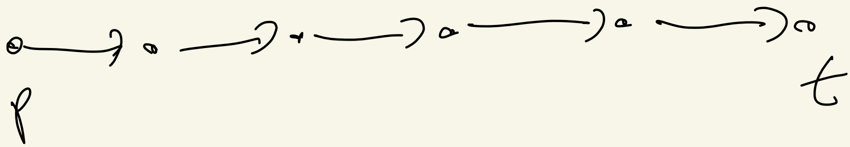
$$h(p) \le h(q) + 1 \quad \forall pq \in N(x) \quad (*)$$



Example of a height function:

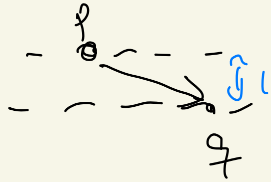
distance to t: $h(v) = dist_{N(x)}(v, t)$

$$dist(p, t) \le dist(q, t) + 1 \quad \forall pq \in N(x)$$



push(pq) precondition $\delta_x(p) < 0$ $p \neq t$

and $h(p) = h(q) + 1$



update $x_{pq} \leftarrow x_{pq} + g$ when (□)

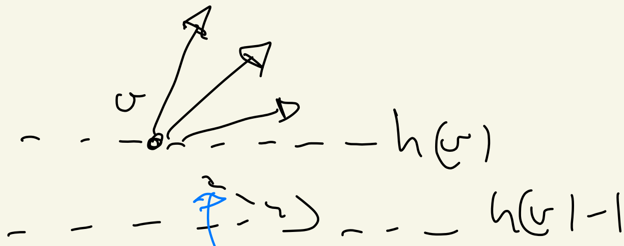
$$g = \min\{-\delta_x(p), r_{pq}\}$$

two possible outcomes of a push(pq)

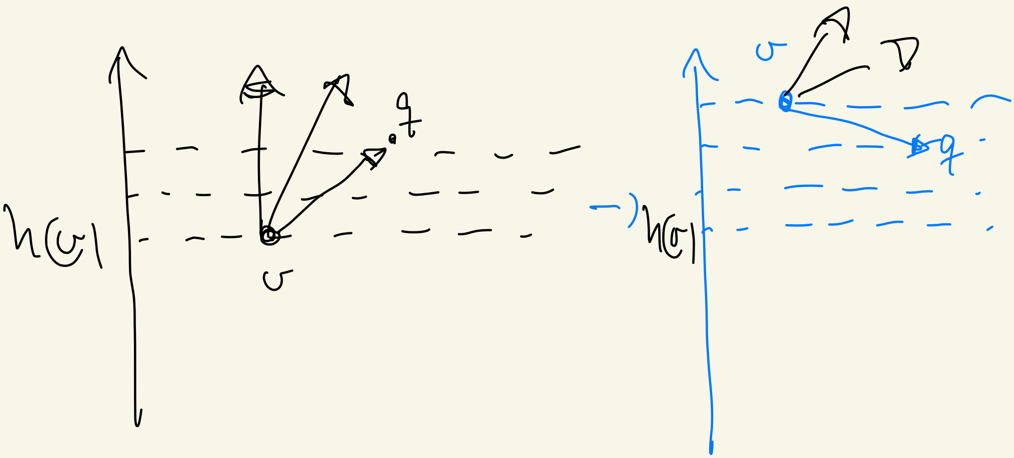
- p becomes balanced (possibly unsaturated push)
- arc $pq \notin N(x)$ after (□)
  (pq became saturated by the push)

lift(v)

precond $\delta_x(v) < 0$



(△) $h(v) \leftarrow \min\{h(z)+1 \mid pz \in N(x)\}$



Note if $\delta_x(v) < 0$
then v has an out-neighbour in $N(x)$
(so (△) is well defined)

reason $\delta_x(v) < 0 \Rightarrow (v,s)$-path in $N(x)$

Generic preflow-push algorithm:

preprocessing

(a) $\forall p \in V \quad h(p) \leftarrow dist_N(p,t)$

(b) $h(s) \leftarrow n$

(c) $x_{sq} \leftarrow u_{sq} \quad \forall sq \in A$

(d) $x_{ij} \leftarrow 0$ for all other arcs

Main loop

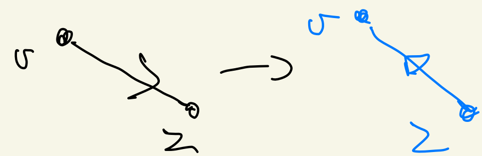
while $\exists v \in V - \{s,t\}$ s.t. $\delta_x(v) < 0$

if $N(x)$ contains an arc vq with $h(v) = h(q)+1$ then push(vq)

else lift(v)

(☐) h remains a height function during the algorithm:

- ok initially as $h(v) = \text{dist}_N(v,t)$
- $h(v)$ is only changed when we lift(v) is applied and this preserves the property
- push(vz) may create the arc $z \to v$



$h(z) = h(v) - 1$

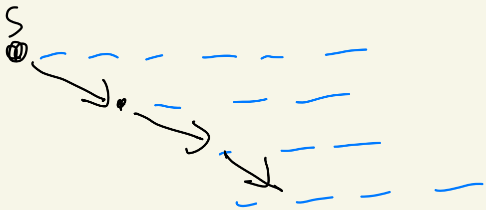
(☐) x remains a preflow

preserved by push operation

(b) If the algorithm terminates, then x is a maximum flow

- at termination $b_x(v) = 0 \quad \forall v \in V - \{s,t\}$
  so x is an (s,t)-flow
- then no (s,t)-path in $N(x)$

$n = h(s)$



- $h(t) = 0$

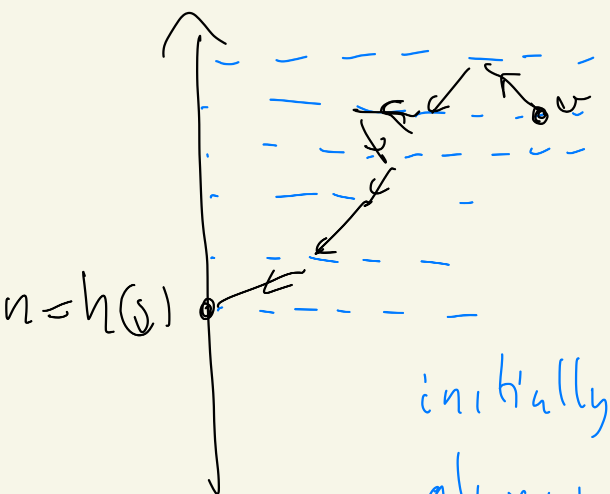
such a path would have n arcs ↯

MFMC thm $\Rightarrow$ x is a max flow.

Claim the algorithm does terminate and it runs at most $O(n^2 m)$ operations.

(A) at most $O(n^2)$ lifts in the algorithm

lift(v) increases $h(v)$ by at least 1 and $h(v) \leq 2n-1$ must still hold since $N(x)$ has a (v,s)-path



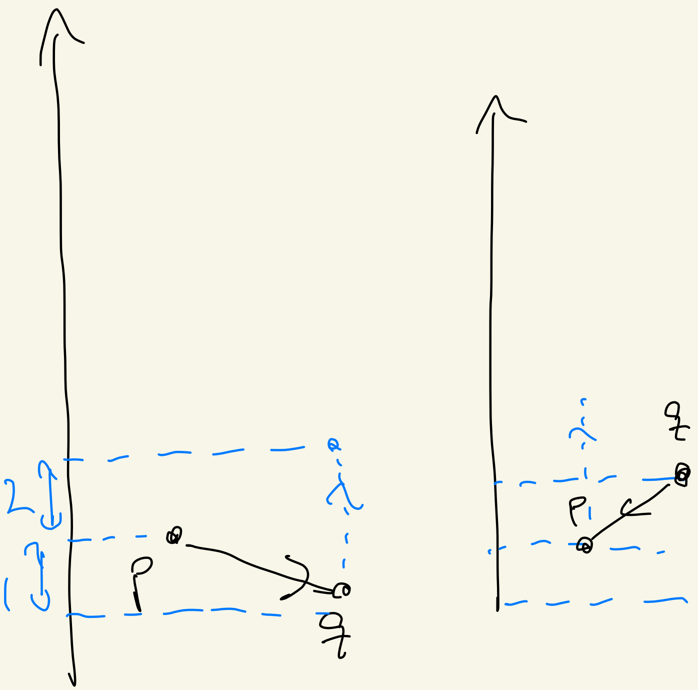
initially $h(v) \geq 1$

always $h(v) \leq 2n-2$

$\Downarrow$ v lifted at most $2n$ times

(3) $O(nm)$ saturating pushes:

bound # times push(pq) is executed for a given arc pq:



$h(q)$ must increase by at least 2 before we can reduce x_{pq}

next time we push from p to q $h(p)$ has increased by at least 2.

as $h(p) \leq 2n-1$ in the whole algorithm we have at most n pushes from p to q

(C) $O(n^2 m)$ unsaturating pushes:

define $\Phi = \sum_{b_X(v)<0} h(v)$ so $\Phi \geq 0$ always

initially $\Phi_0 \leq 2n^2$

contributions to Φ during the algorithm:

- lifts contribute $O(n^3)$ by (A)
- saturating pushes contribute $O(n^2 m)$ by (B) ($O(nm)$ saturating pushes, each contributing $O(n)$)
- Each unsaturating push decreases Φ by at least one!

Conclusion

$\Phi_0 \le 2n^2$ and the total increase in Φ during the algorithm is $O(n^2 m)$

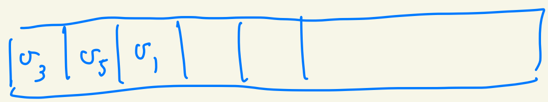
so # nonsaturating pushes is $O(n^2 m)$

So 1) the algorithm terminates

2) using $O(n^2 m)$ operations

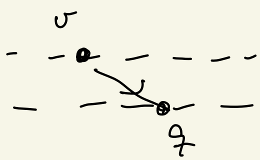
preflow push algorithm uses $O(n^2 m)$ operations

active vertices $(b_x(v) < 0)$



all have $b_x < 0$

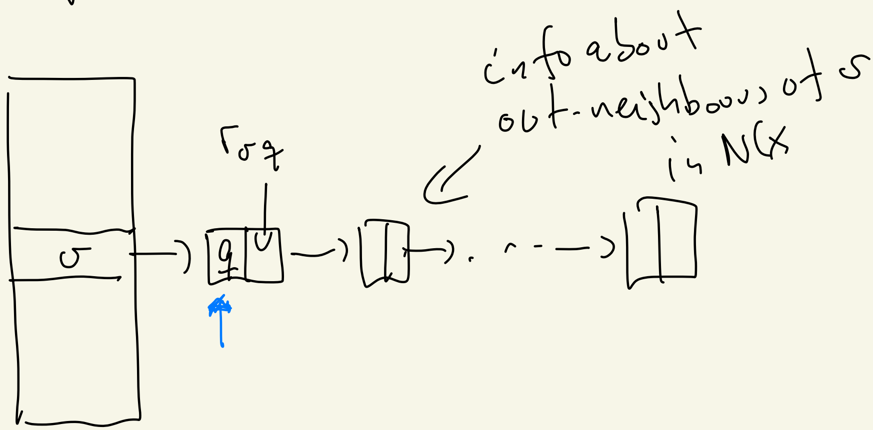
Given v with $b_x(v) < 0$



find q s.t. $h(v) = h(q) + 1$
and $vq \in N(x)$

lift: $\quad h(v) \leftarrow \min\{h(z)+1 \mid vz \in N(x)\}$

keep adjacency list representation of $N(x)$

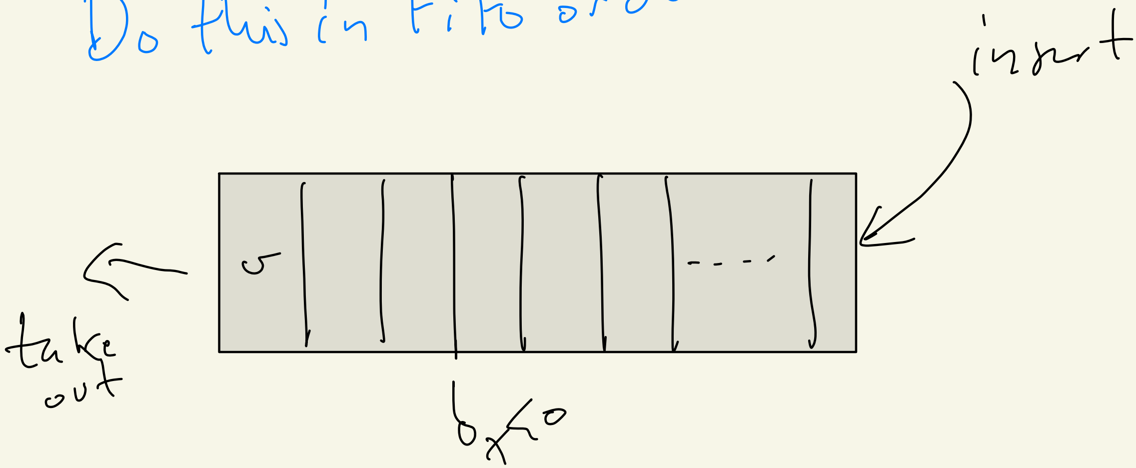


Ahuja 7.7 FIFO preflow push alg

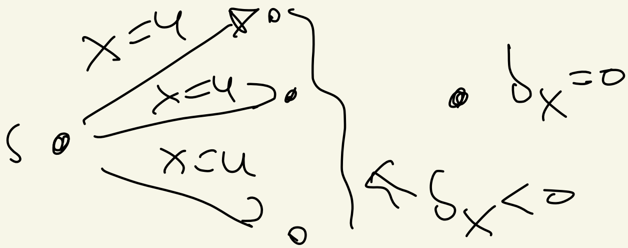
Rule: once v with $\delta_x(v) < 0$ is chosen, we keep pushing from until either $\delta_x(v)$ becomes 0 or we lift v.

(node examination step)

Do this in FIFO order

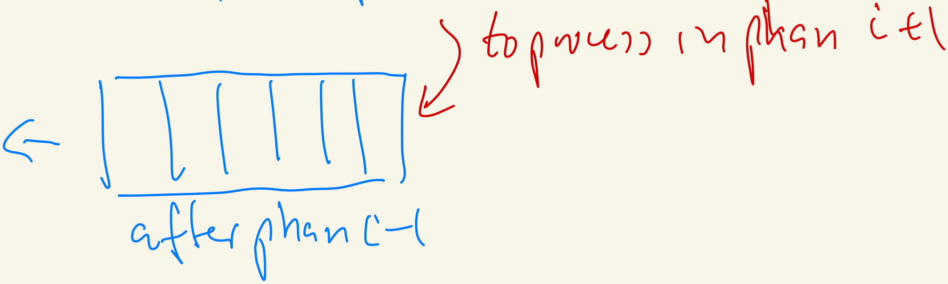


Partition examination operations into phases phase 1 : do node examination for all v that got flow from s in the initialization



phase i : do node examination on all nodes in the list after phase $i-1$

Any node $v \in V$ is processed at most once per phase



<u>Claim</u> there are at most $2n^2+n$ phases in the algorithm

let $\Phi = \max\{h(v) \mid \delta_x(v) < 0\}$

consider the total change of Φ during a phase:

$$\Phi_i - \Phi_{i-1}$$

end of phase i

end of phase $i-1$

Case 1 we performed ≥ 1 lift in phase i.

Then Φ could increase but no more than $2n^2$ over the whole algorithm

Case 2 no lifts in phase i

(each vertex from phase i got balanced during the phase)

Φ will decrease by at least one

(Every vertex in the list when phase i ends have height $< \Phi$

Conclusion $\leq 2n^2 + n$ phases

initially put $h(v) = n$ if no (s,t)-path in $N(x)$

This implies that total number of unsaturating pushes is $O(n^3)$:

Each vertex v is examined at most once per phase and at most one unsaturating push from v in a phase

So since # phases is $O(n^2)$ we do $O(n^3)$ unsaturated pushes.