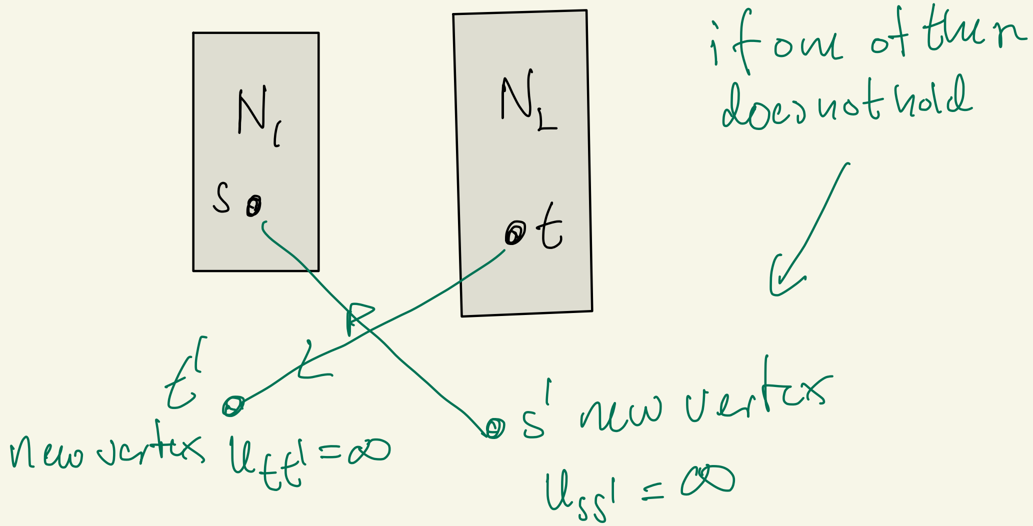
Ahuja 8.3 Flows in bipartite networks

$G = (N_1 \cup N_2, A)$ $\qquad n_i = |N_i|$

assume $n_1 \leq n_2$ and that $s \in N_2$ and $t \in N_1$



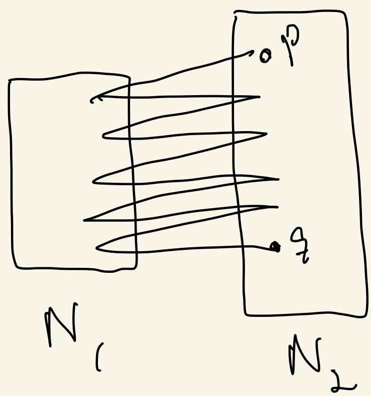
Goal: show that preflow push algorithm performs (much) better than $O(n^2 m)$ when G is bipartite and $n_1 << n_2$

Modifications of generic pfp algorithm

- Initialize $d(s)$ to $d(s) = 2n_1 + 1$ instead of $d(s) \leftarrow n$

(Ahuja uses d for height function)

justification: No path in $N(x)$ has more than $2n_1$ arcs:



Hence $N(x)$ cannot contain an (s,t)-path when we set $d(s) = 2n_1 + 1$

Initialize $d(i)$ to

$$d(i) \Leftarrow \min\{2n_1+1, \mathrm{dist}_N(i,t)\}$$

Lemma 8.3 During the whole algorithm then

we have $d(i) \le 4n_1+1 \quad \forall i \in N_1 \cup N_2$

P: This holds after initialization and if we lift i then there is an (i,s)-path in current $N(x)$

so $d(i) \le 2n_1 + d(s) = 4n_1+1$

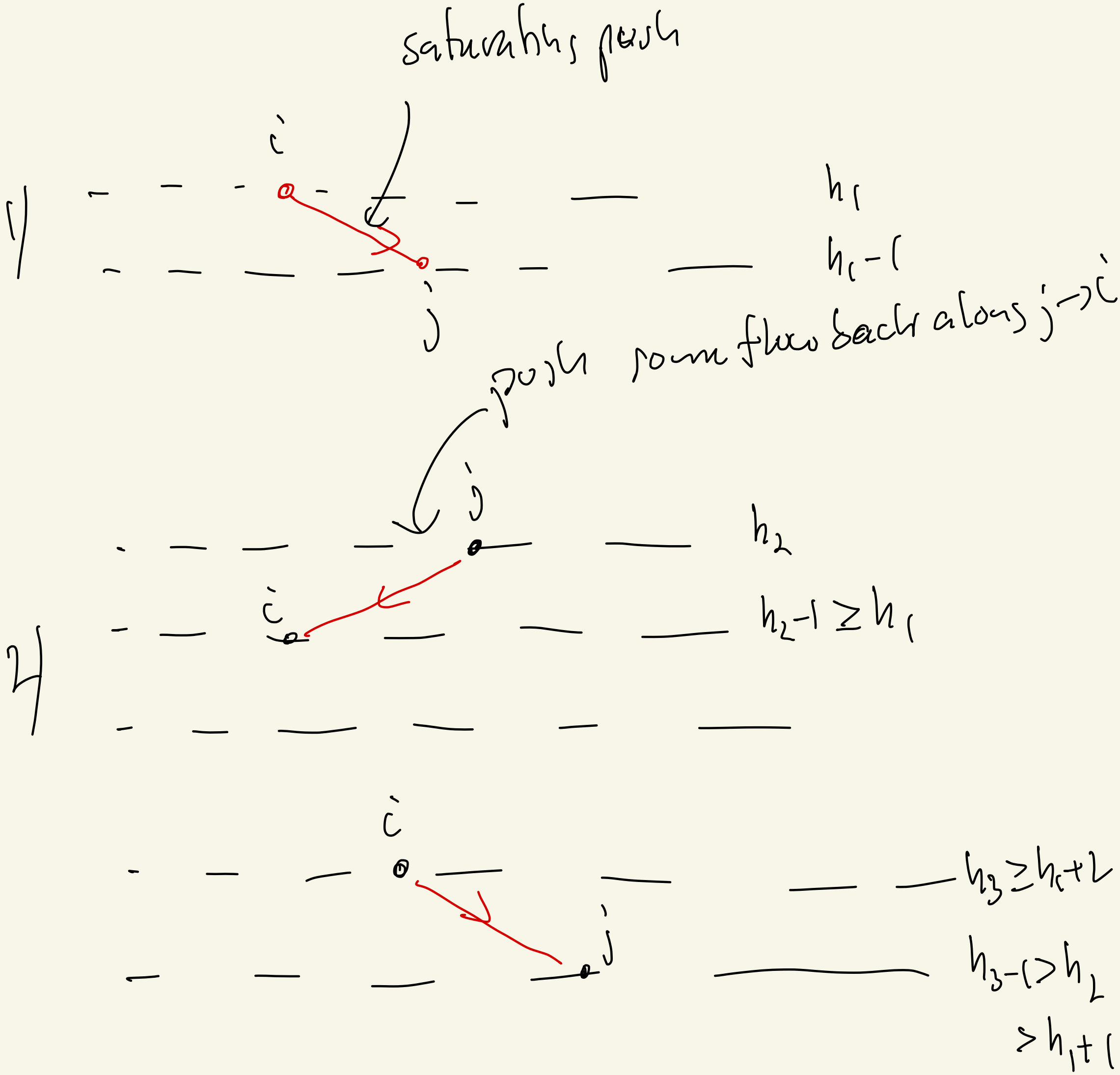
Lemma 8.4

(a) Each $d(i)$ changes $O(n_1)$ times

so total # of lifts is $O(n_1(n_1+n_2))$

which is $O(n_1 n_2)$ since $n_2 > n_1$

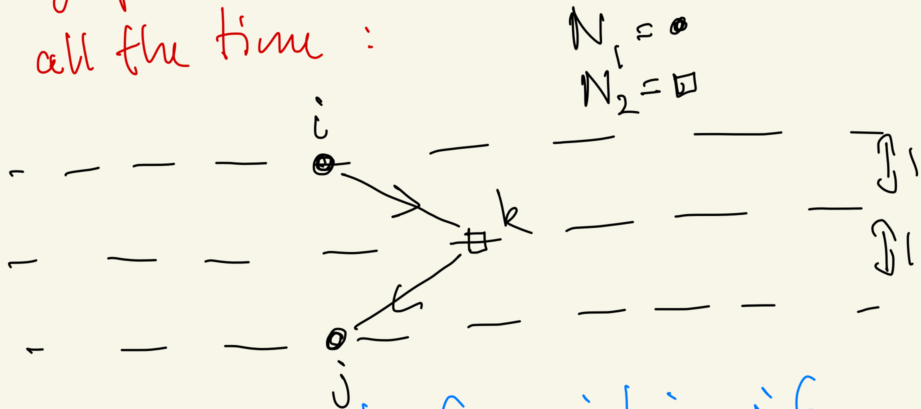
(b) The number of saturating pushes is $O(n_1 m)$



saturating push
1)
i
j
h_1
h_1 - 1
push some flow back along j → i
2)
j
i
h_2
h_2 - 1 ≥ h_1
i
j
h_3 ≥ h_1 + 2
h_3 - 1 > h_2
> h_1 + 1

$O(n_1^2 m)$ algorithm:

- Only allow vertices in N_1 to be active by pushing along paths of length 2 all the time:



So we can push from i to j if $\exists\, k \in N_2$ s.t. $d(i) = d(k)+1 = d(j+2)$ and $ik, kj \in N(x)$

Now a lift operation may involve 2 lifts, one in N_2 and one in N_1

If $b_x(i) < 0$ then

If $\exists\, ik \in A(N(x))$ with $d(i) = d(k)+1$

then if $\exists\, kj \in A(N(x))$ with $d(k) = d(j)+1$

then push $i \to k \to j$

Else lift k

Else lift i

Observation: ok to lift k if no arc kj as above

Reason: Either k is not used later ✓
or k is used to push flow through it and then it must be lifted before we can do this

Lemma 8.5 The algorithm performs $O(n_1^2 m)$ unsaturating pushes

P: same as for generic algorithm

$$\Phi = \sum_{i \text{ active}} d(i) \qquad i \neq s \text{ active} \overset{\text{def}}{\Longleftrightarrow} b_x(i) < 0 \text{ and } i \in N_1$$

a) $d(i) \leq 4n_1 \;\; \forall i \in N_1$ and only vertices in N_1 are active

$\Rightarrow \Phi_0 \leq 4n_1^2$

b) Effect on Φ:

(1) lift $i \in N_1$ total change in Φ during algorithm $O(n_1^2)$

(2) lift $k \in N_2$ no change

(3) saturating push $i \to k, k \to j$ (one or both arcs saturated)
total $O(n_1) \cdot O(n_1 m) = O(n_1^2 m)$

(4) Each unsaturating push $i \to k, k \to j$ decreases Φ by at least 2 as $d(j) = d(i) - 2$

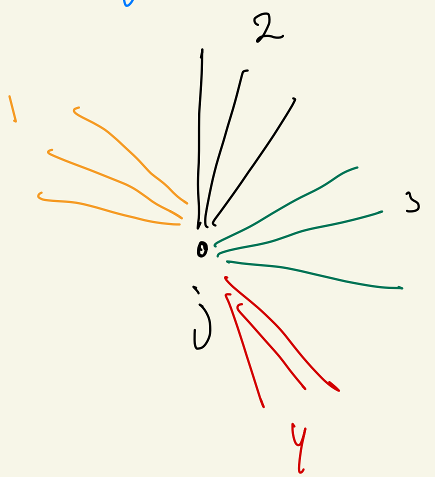
$\Phi \geq 0$ always so $O(n_1^2 m)$ unsaturating pushes and hence algorithm is $O(n_1^2 m)$

Application 8.2 in Ahuja

Network reliability testing

Goal: test each edge ij of $G=(N,E)$ α_{ij} times for given $\{\alpha_{ij} \mid ij \in E\}$

Resource limitation: Each day we can test at most β_j edges incident to vertex j



$\beta_j = 3$

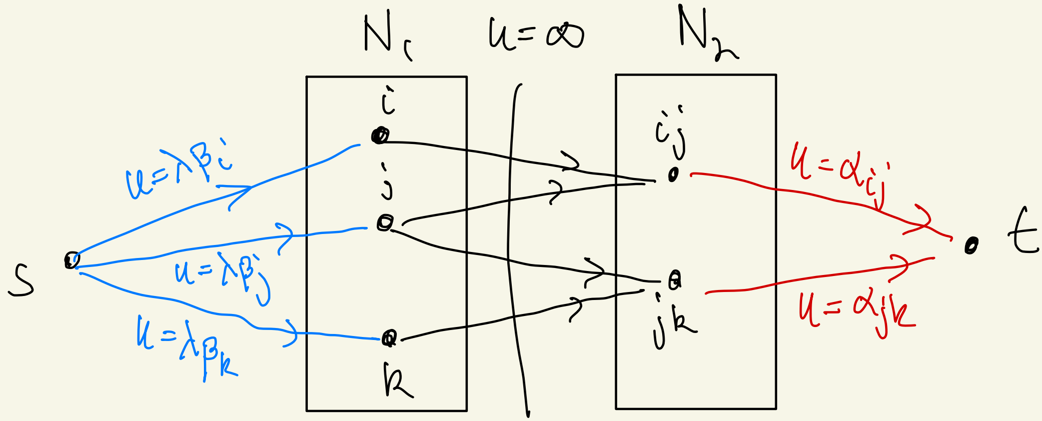
Task: Find a schedule for doing all the tests in a minimum # of days

Remark: a test of ij can be associated to either i or j

Given $G=(N,E)$ form network

$$N = (N_1 \cup N_2 \cup \{s,t\}, A, \ell \equiv 0, u) \text{ where}$$

$$N_1 = N, \quad N_2 = \{ij \mid ij \in E\}$$



Find (via binary search) the minimum integer λ such that all arcs into t are filled by an (s,t)-flow (which is then a maximum flow)

$\lambda = 1, 2, 4, 8 \; \ldots \; 2^k, 2^{k+1}$

$- \quad - \quad - \quad - \qquad - \quad f \quad f \quad - \quad - \quad -$

now find best λ via binary search

in $[2^k, 2^{k+1}]$