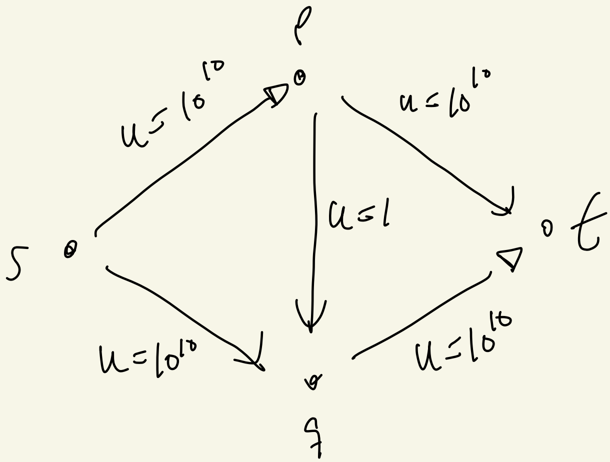
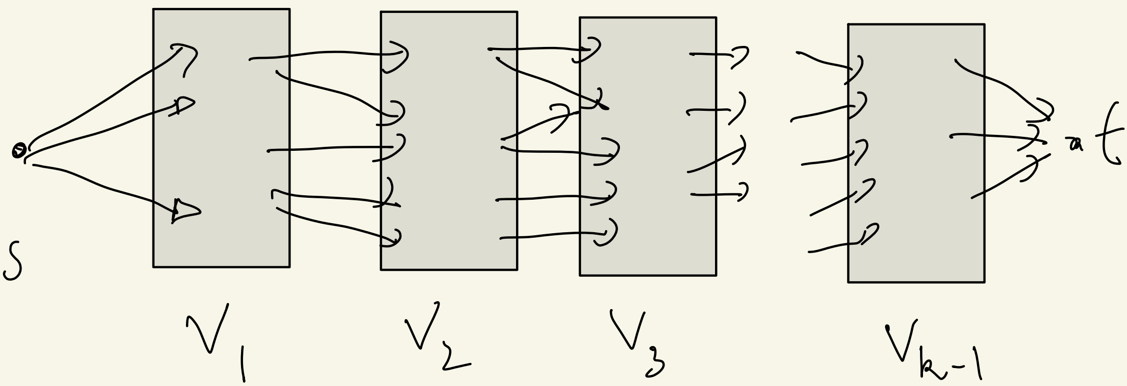
Augmenting along shortest augmenting paths

$N = (V \cup \{s,t\}, A, \ell \equiv 0, u)$



Layered network:



Given $N=(V\cup\{s,t\}, A, \ell\equiv 0, u)$ and an (s,t)-flow x in N

define

$\delta_x(s,t)$ = length of a shortest (s,t)-path in $N(x)$
($=\infty$ if no such path)

$\mathcal{L}N(x)$ = layered s to t subnetwork of $N(x)$ which is defined from the distance classes from s in $N(x)$.

Observation:

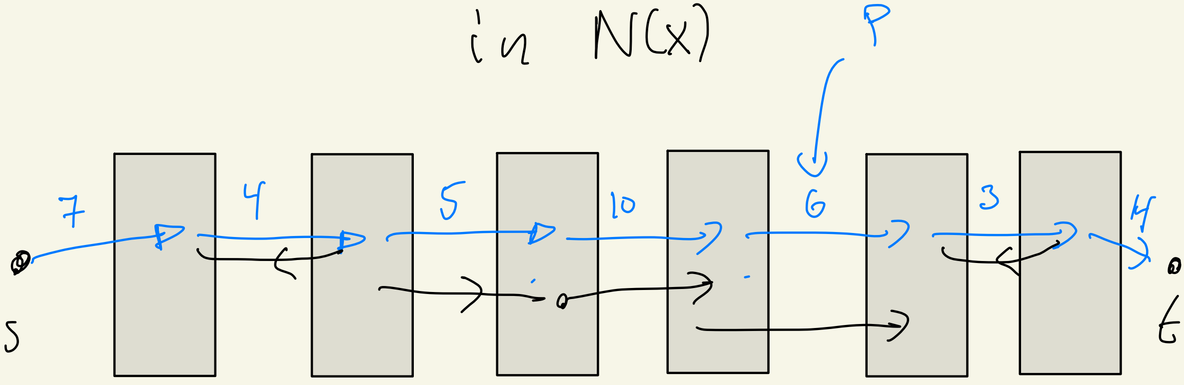
$\partial N(x)$ contains all the shortest (s,t)-paths in $N(x)$.

Lemma 3.6.2

Let x be an (s,t)-flow in N and P a shortest (s,t)-path in $N(x)$ and $x' = x \oplus \delta(P)$.

Then
$$\delta_{x'}(s,t) \geq \delta_x(s,t)$$

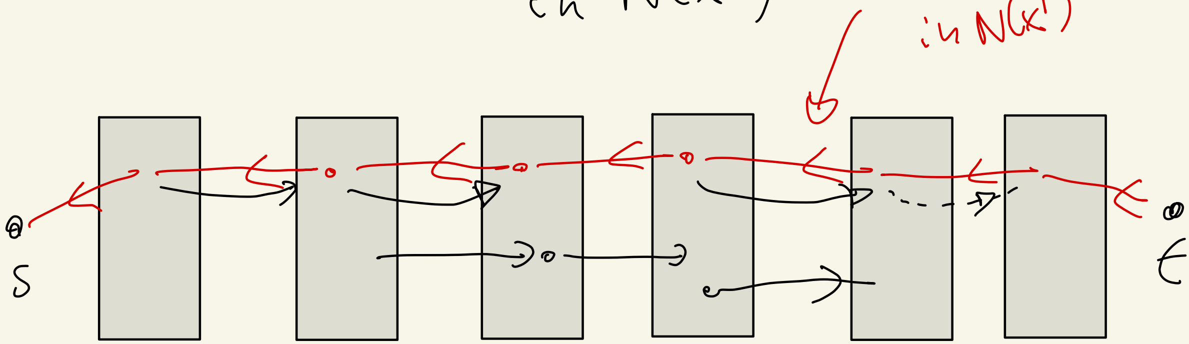
in $N(x)$



$x' = x \oplus \delta(P)$ $\delta(P) = 3$

in $N(x')$

only possible new arcs in $N(x')$



$$\Downarrow$$

$$\delta_{x'}(s,t) \geq \delta_x(s,t)$$

$$d = d_{X \equiv 0}(s,t)$$

Consider phases

phase 0 $\quad d \quad$ augment as long as possible

phase 1 $\quad d+i_1 \quad$ — " —

$\quad d+i_2 \quad$ — " —

$\vdots$

phase $q \quad d+i_q \le n-1 \quad$ — " —

$$i_1 < i_2 < i_3 \cdots < i_q \le n-d-1$$

Max no of augmentations (finding an (s,t)-path in current residual network during a phase is $O(m)$ (at most $2m$)

Conclusion The whole algorithm runs in time $O(nm^2)$

Conclusion The whole algorithm runs in time $O(nm^2)$

at most $n-2$ phases

in each phase we make $O(m)$ augmentations

to find each (s,t)-path we need $O(n+m)$

$O(nm^2)$

(Edmonds-Karp algorithm)

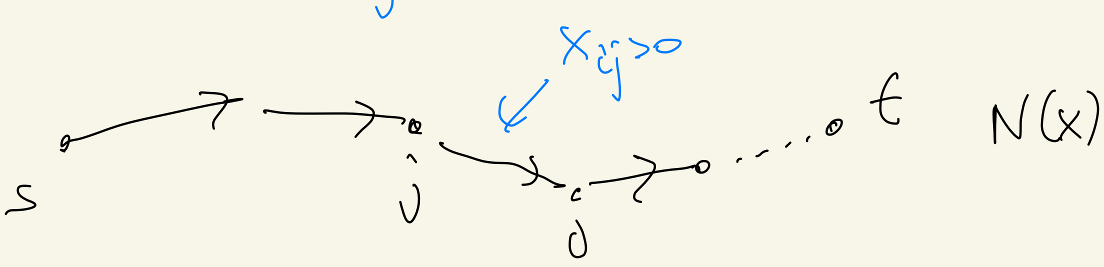
Observation: we don't need to build the layered networks!

Remark on the Edmonds-Karp algorithm:

we may need $\Omega(nm)$ augmentations so the worst case running time is $\Omega(nm^2)$

new definition

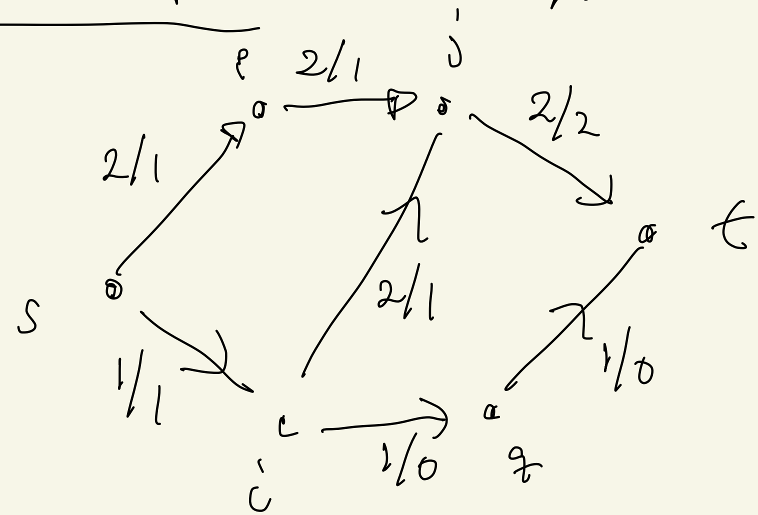
a blocking flow in an network $N = (V \cup \{s,t\}, A, \ell \equiv 0, u)$ is an (s,t)-flow x s.t. that every (s,t)-path in $N(x)$ uses at least one arc ji for which $x_{ij} > 0$ (and hence $x_{ji} = 0$)



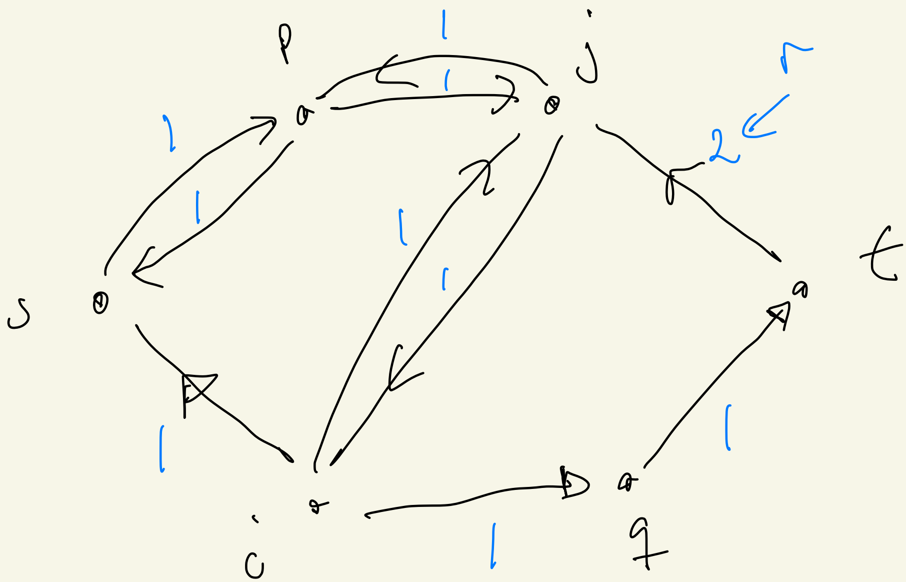
Example

u/x

N:



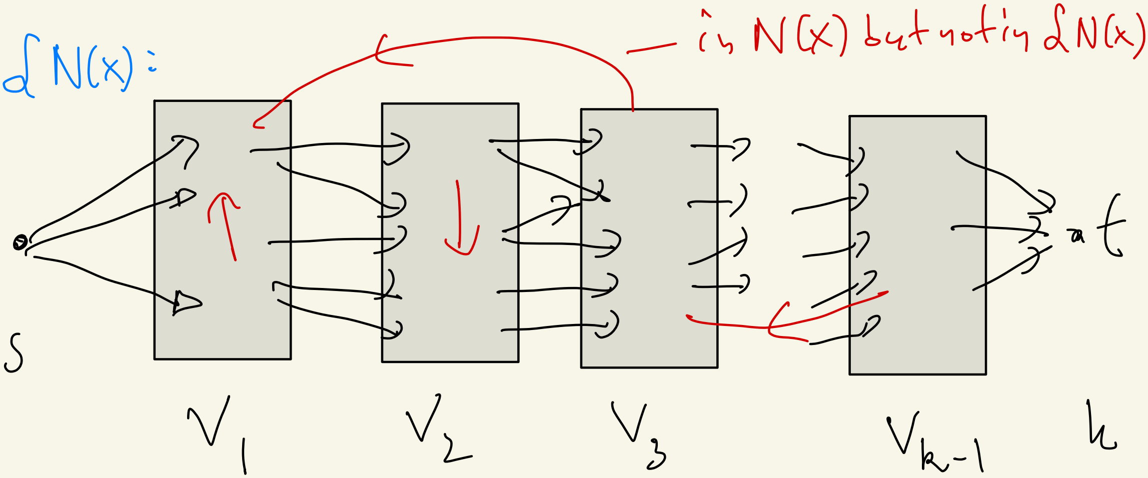
N(x)



only (s,t)-path is

spjiqt

$\ell N(x)$:



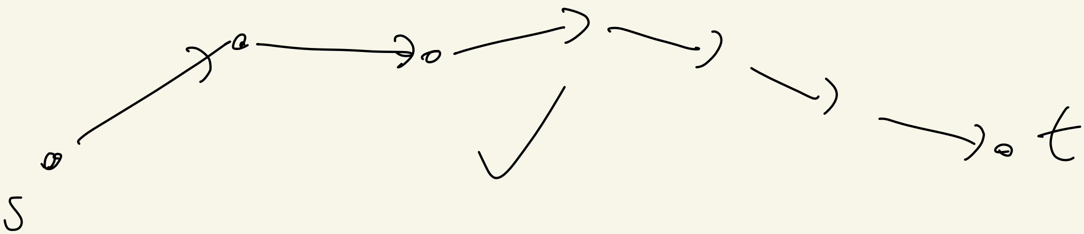
$\ell N(x)$ is a subnetwork of $N(x)$

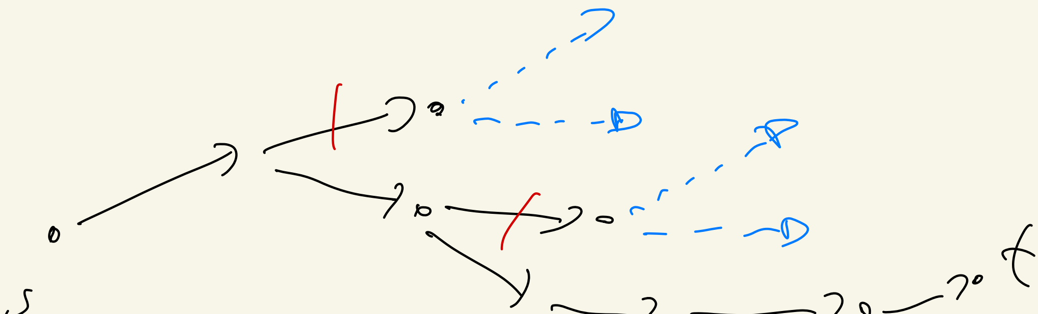
x' is a blocking flow in $\ell N(x)$ if there is no (s,t)-path of length k = dist from s to t in $\ell N(x)(x')$

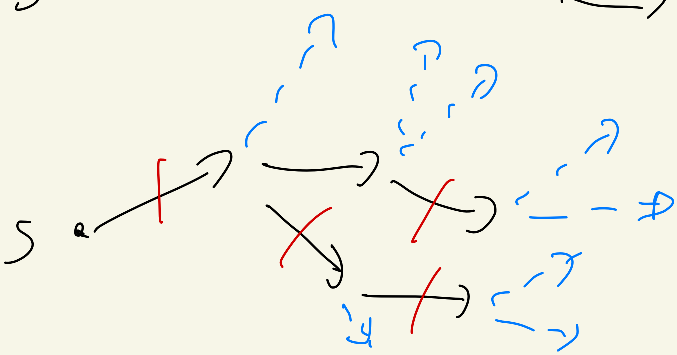
Edmonds-Karp algorithm finds a blocking flow in $\ell N(x)$ in time $O(m^2)$

Dinic's algorithm:

idea find augmenting paths in $\partial N(x)$ via a modified Depth First search

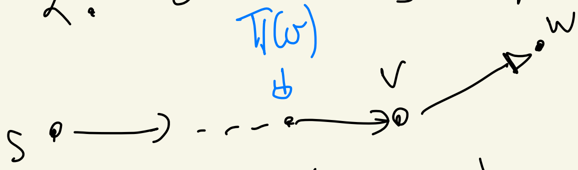


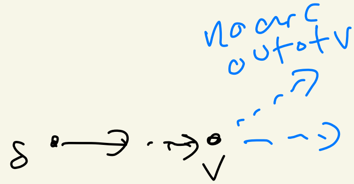




1. start with $x \equiv 0$; $v \leftarrow s$

2. Searching step





go to step 4

- continue step 2 from w unless $w = t$
- if $w = t$ go to step 3

3. using parent pointers $\pi(v)$ find the augmenting path P and $\delta(P)$ and then update capacities along P, delete an arc if it becomes full. go to 2 with $v \leftarrow s$

4. delete all arcs incident to v
set $v \leftarrow \pi(v)$ and go to step 2

5. Return the blocking flow
(if we are in step 2 with

→ go to 5

The algorithm does find a blocking flow in finite time:

we only delete an arc if it cannot be part of a new augmenting path in $\mathcal{D}N(x)$

Thm Dinic's algorithm finds a blocking flow in time $O(nm)$

P:
- we only delete arcs that cannot be part of new augmenting path of length k
- no (s,t)-path at termination.
- For $O(n)$ steps we either find a new augmenting paths or we delete a vertex

after augmenting along new path at least one arc is deleted phan 1 $d = \delta_{x\equiv 0}(s,t)$ $\longrightarrow$ $\tilde{X}_0$ blocking in $\Delta N(x) = \Delta N$

$X_1 \Leftarrow X \oplus \tilde{X}_0$

plan 2 calculate $\Delta N(x_1)$

$\tilde{X}_1$ blocking flow in $\Delta N(X_1)$

$X_2 \Leftarrow X_1 \oplus \tilde{X}_1$

plan 3 calculate $\Delta N(X_2)$

$\tilde{X}_2$ blocking flow in $\Delta N(X_2$

$X_3 \Leftarrow X_2 \oplus \tilde{X}_2$

$\vdots$

phan q calculate $\Delta N(X_{q-1})$

$\tilde{X}_{q-1}$ blocking flow in $\Delta N(x_{q-1})$

max flow $\longrightarrow$ $X_q \Leftarrow x_{q-1} \oplus \tilde{X}_{q-1}$