CS6130 : Advanced Graph Algorithms

Back to maxflow computation

Push Relabel algorithm

↳ basis for fastest flow computation

↳ a different method

↳ interesting analysis

Computing a Max Flow

Ford Fulkerson based method

Inv: always maintain a valid flow

Stopping Cond: no s-t path in the residual n/w.

Computing a Max Flow

Push Relabel

~~Ford Fulkerson based~~ method

Inv: always maintain ~~a valid flow~~ no s-t path in G_f

Stopping Cond: ~~no s-t path in the residual n/w.~~ when f is a flow.

Preflow : Definition

$f : E \rightarrow R^+$

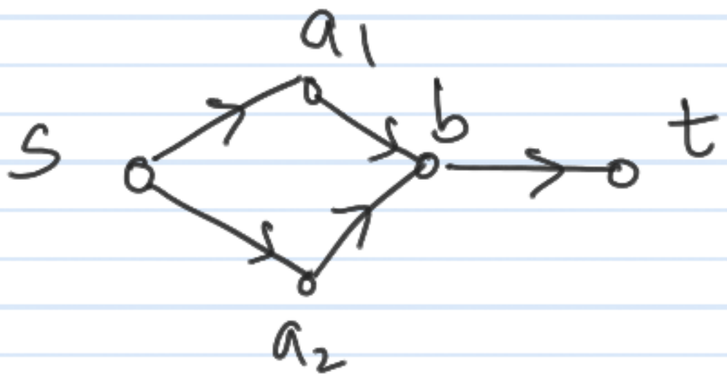
capacity constraint : $0 \leq f(e) \leq c(e)$. $\forall e \in E$

preflow condition : $f_{in}(v) \geq f_{out}(v)$

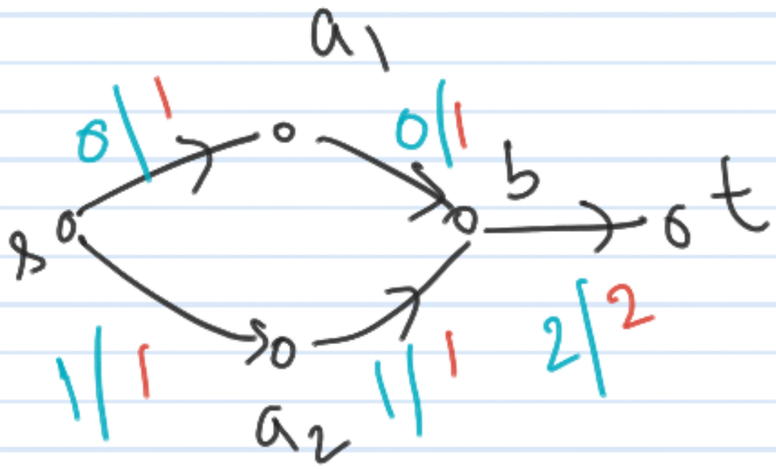
for every v except s

$excess(v) = f_{in}(v) - f_{out}(v)$

Preflow : examples

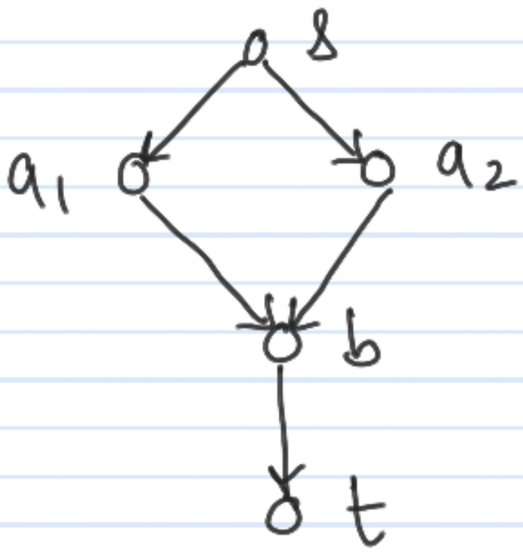


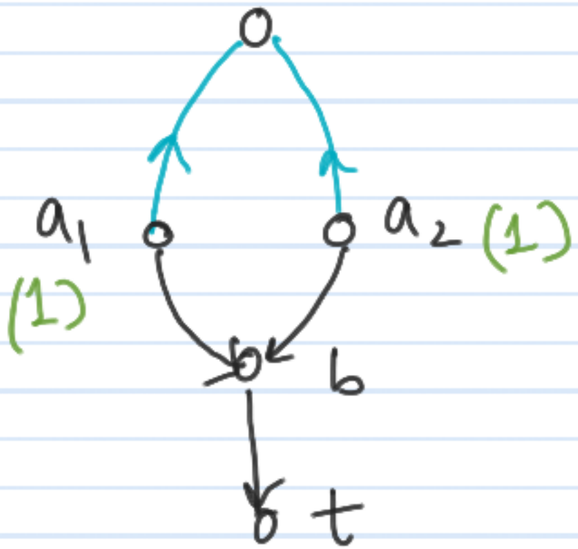
$f(e) = 1$ for all e

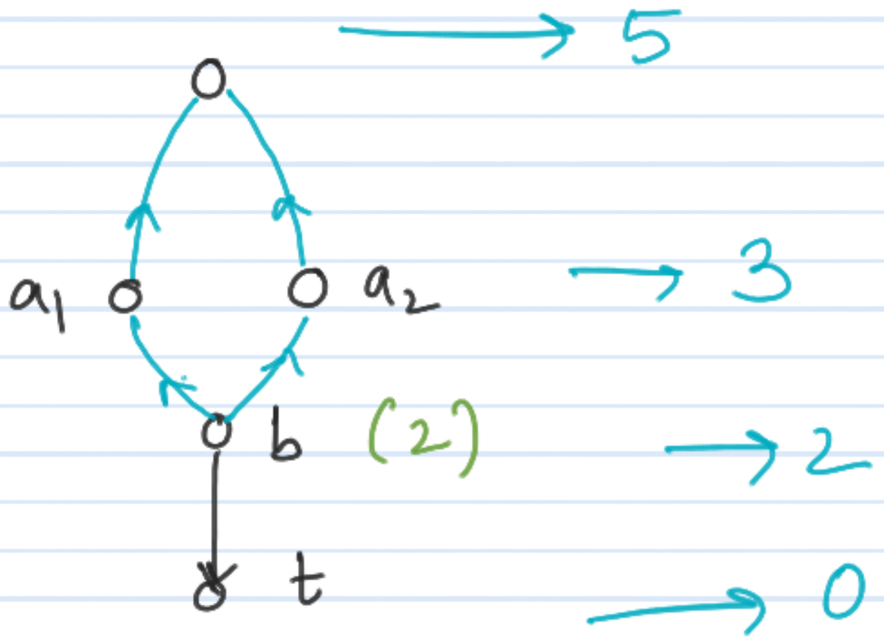


✗ not a preflow.

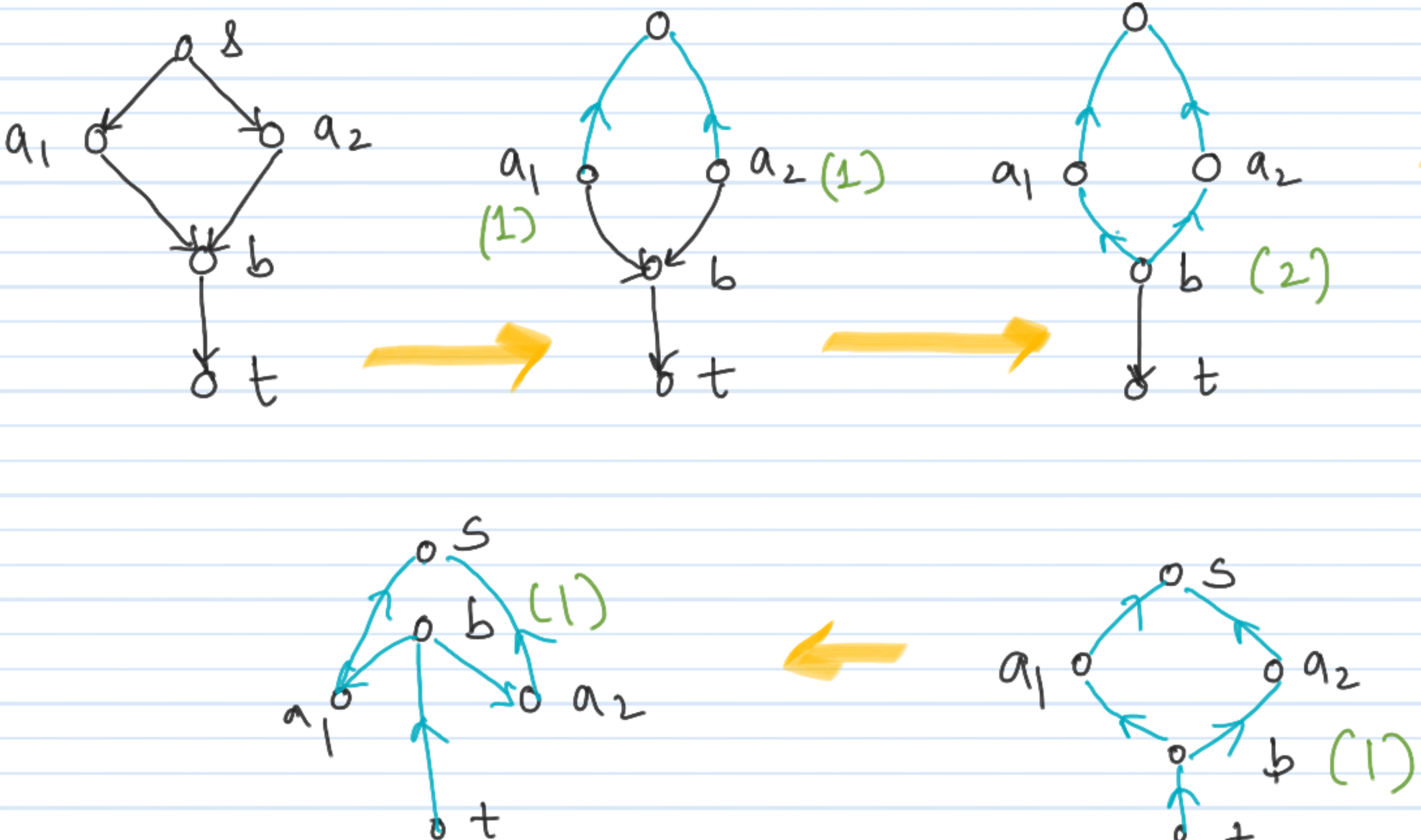
Push Relabel : Intuition and example







Push Relabel : Intuition and example



Push Relabel : Height Function.

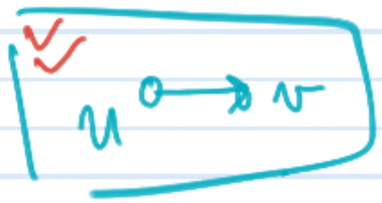
$h(v) : V \rightarrow Z^{\geq 0}$

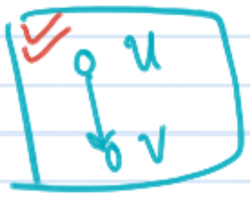
1) $h(s) = n$
2) $h(t) = 0$

} they will remain at these heights throughout

3) if $(u,v) \in E_f$ then

$h(u) \leq h(v) + 1$





Properties of residual n/w and algo

(A) No steep downward edges

(B) Steep upward edges can be present

(C) Heights are non decreasing

Push Relabel Algo

Init Preflow

for every $e = (s, u)$

$f(s,u) = c(s,u)$

$excess(u) = f(s,u)$

$excess(s) = -\sum_{(s,w) \in E} c(s,w)$

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

$ht(u) = 0 \quad \forall u \in V$

Push

Applies if $ex(u) > 0$

$\exists$ an eligible (u,v)

$\Delta = \min(c_f(u,v), ex(u))$

if $(u,v) \in E$

$f(u,v) += \Delta$

else

$f(v,u) -= \Delta$

adjust excess of u & v

Relabel

Applies if $ex(u) > 0$ and no elig. edge exists

$ht(u) = 1 + \min(ht(v))$ $(u,v) \in E_f$

Push Relabel Algo

- Init Preflow
- while ∃ an applicable push or relabel op
  - apply the particular operation

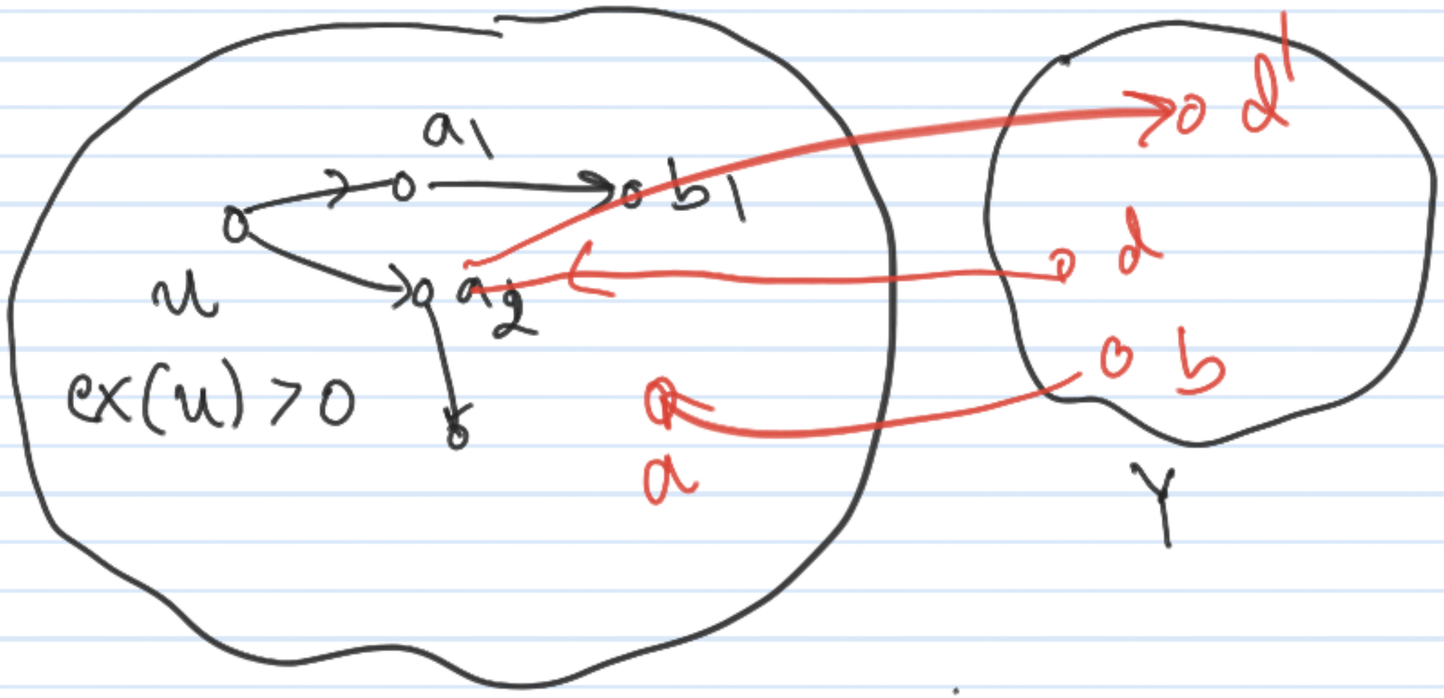
Invariants :

(1) ∄ no s-t path throughout the algo

(2) The push and relabel maintain height func property

(3) height of a vertex never decreases

Path from overflowing vertex to s



where is s?

Presence of $b \to a$ edge carrying flow implies $a \to b$ in residual n/w

contradiction

$$\sum_{a \in X} ex(a) > 0$$

X: reachable in G_f from u s.t $ex(u) > 0$

$$= \sum_{a \in X} \left(\sum_{b \in V} f(b,a) - \sum_{b \in V} f(a,b) \right)$$

$$= \sum_{a \in X} \sum_{b \in Y} f(b,a) - \sum_{a \in X} \sum_{b \in Y} f(a,b)$$

Bound on # of push and relabel operations

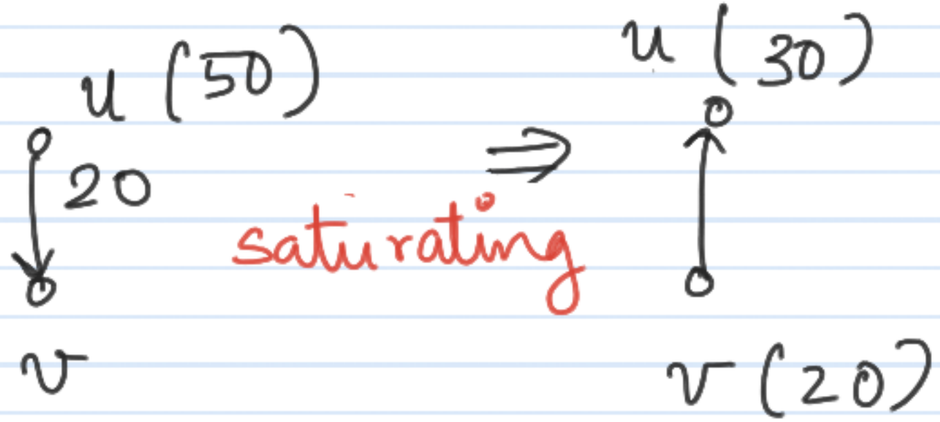
Relabel : max $O(n^2)$ operations

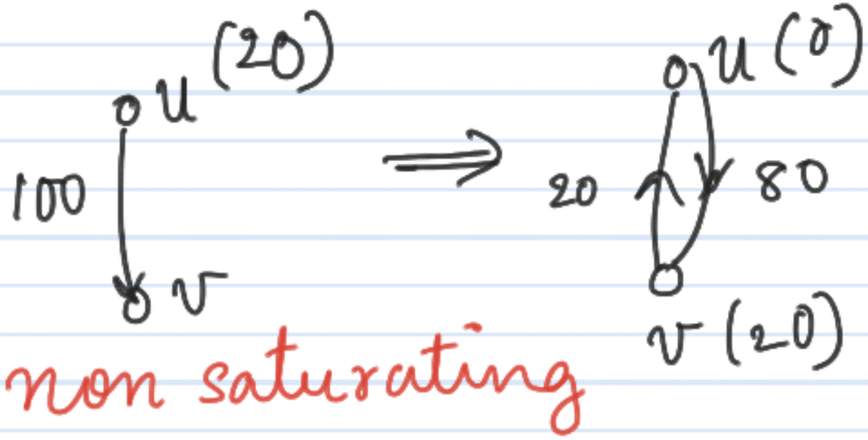
Push

↳ saturating

↳ non saturating

in case of a tie call it non saturating





Bounding # of saturating pushes

→ after saturating push edge (u, v) disappears from residual $n(w)$.

- If we send flow again along (u, v) relabels must have happened.

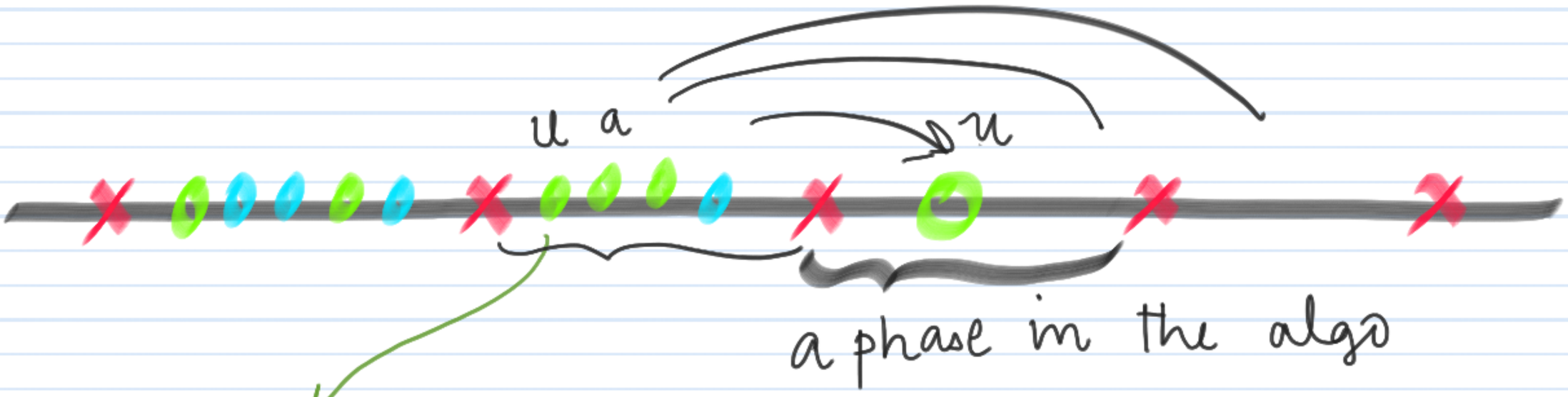
# of saturating pushes = $O(mn)$

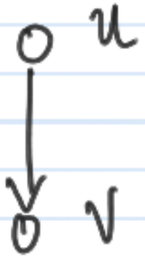
Bounding # of non saturating pushes

A small change in algo.

- pick highest vertex with excess(·) > 0

View the algo from the relabels lens





: saturating push.

: non sat. push.

: relabel

u is drained after the push. can there be any push on u in this phase?

Summary of Push Relabel Algo

(1) A "local" method which maintains invariant "no s-t path" and terminates when preflow turns to a flow.

(2) Established correctness and termination
↳ To do running time

(3) We used the highest vertex modification (CLRS)
↳ checkout other heuristics, what abt w/o heuristic?

Homework Problem

Input : $(S \cup C, E)$ → $(s,c) \in E \Rightarrow$ s is interested in c and c can take s.

S: students, C: courses

Quota: for every $c \in C$ $q(c) =$ max # of students c can accomodate

Additional constraints: for every $c \in C$

$$N(c) = P_1 \cup P_2 \cup P_3 \cdots \cup P_k$$

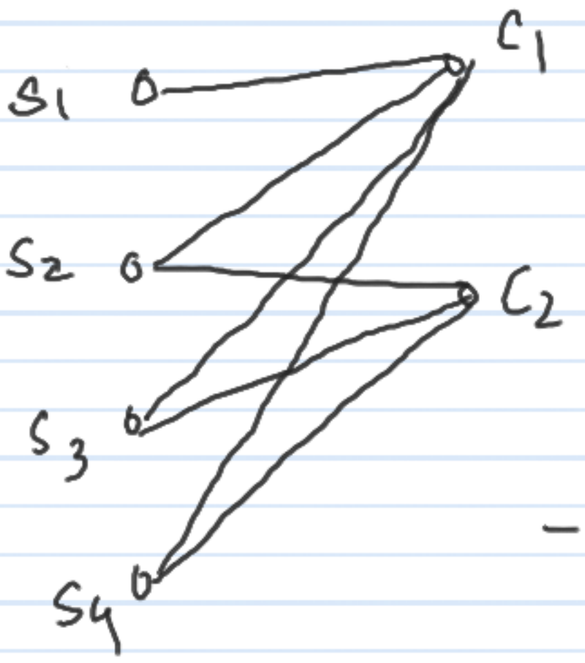
adjacent students are partitioned and each partition has its individual quota

$q(c, P_i)$: max # of students c can accomodate from P_i

Goal:
Compute a max matching respecting constraints

An Example :

$q_r(c_1) = 2$, $q_r(c_2) = 2$



$P_1 = \{s_1, s_2\}$ $q_r = 1$

$P_2 = \{s_3, s_4\}$ $q_r = 1$

$P_1' = \{s_1, s_2, s_3\}$ $q_r = 1$

$P_2' = \{s_4\}$ $q_r = 1$

- in this example we cannot assign s_1 and s_2 to c_1 because of P_1 and its quota 1
- similarly, we cannot assign s_1 and s_3 to c_2 because of P_1' and its quota 1
- A possible assignment is $M = \{(s_1, c_1)(s_2, c_2)(s_4, c_2)\}$ → is this valid / maximum?