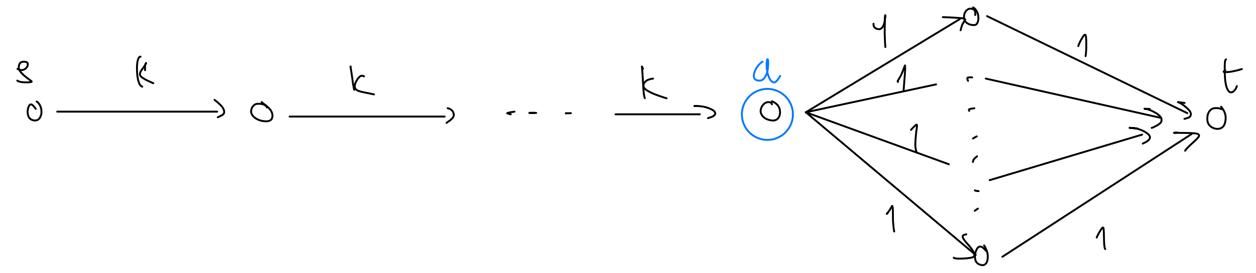
PUSH - RELABEL ALGORITHM

Motivation:

Given this flow network



- Ford-Fulkerson and Edmonds-Karp will take $\sim O(k^2)$ to find maximum flow - minimum capacity.
- If we can somehow push flow up to the a vertex, store it there, then continue push flow from $a \to t$, then we would have a polynomial time algorithm since we don't have to re-push flow from $s \to a$ for every iteration
  $\Rightarrow$ Main idea of Push-Relabel algorithm

Concept: Preflow

Main-idea of Push-Relabel algorithm suggests violating the Flow Conservation theorem (i.e at vertex a, flow going in > flow going out). So we introduce the concept of Preflow

Preflow

Preflow $\{f_e\}: e \in E$ satisfies 2 things:
- Nonnegative capacity: $0 \leqslant f_e \leqslant c_e \quad \forall e \in E$
- Flow in $\geqslant$ Flow out (except at source s)

Does not affect residual graph G_f

The existence of preflow doesn't change the way we construct residual graph G_f

Concept: Excess flow $\alpha_f(v)$

- Recall we are trying to find maximum flow, not preflow. So by the time the algorithm terminate, Flow Conservation should be respected $\Leftrightarrow$ Output Flow, not Preflow
- We can frame the problem as "Minimizing the difference between Flow in and Flow out", called "Excess"

Excess

$\boxed{\begin{array}{l}\text{The difference between Flow in and Flow out} \\ e(v) := \text{inflow}(v) - \text{outflow}(v)\end{array}}$

The algorithm:

- Not trying to find an augmenting path
- At each iteration, at some vertex v, the goal is to get rid of the Preflow and restore Flow Conservation
- So, we can think of it as:

Pre flow G_f $\xrightarrow{\text{some iterations of transformations}}$ Flow-restored G_f

Invariants

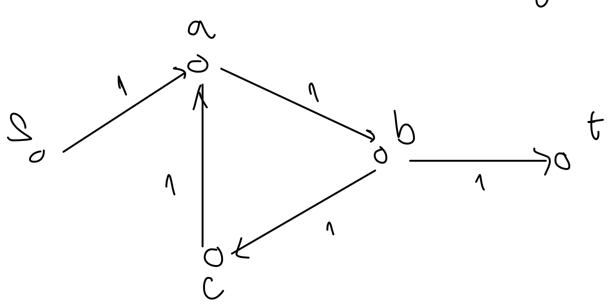
The "transformation" need to preserve these invariants

Let $h(v) : v \in V$ be the "height" of the vertex s.t.:

1. $h(s) = n \quad (n = |V|)$
2. $h(t) = 0$
3. $\forall\ (u, v) \in E_f \in G_f$:
   $h(u) \leq h(v) + 1$

Why?

- Let say we dont respect these invariant and just simply push flow like how we did with Ford-Fulkerson, then its possible we will push flow indefinitely in circle



- Recall at each iteration, the current vertex try to "get rid of preflow".
  So at vertex b, there are 2 choices $\begin{cases}\text{push unit flow to t} \\ \text{push unit flow to c}\end{cases}$

$\Rightarrow$ If push to c is chosen everytime, then we end up pushing in circle and never restore Flow Conservation.

Proof: Invariants hold $\Rightarrow$ Residual graph G_f does not have a path from (s) $\longrightarrow$ (t)

- Proof by contradiction: Assume $\exists$ path from $s \to t$

$\Rightarrow$ # edges such path $\leq n-1$ (1)

- Invariants state that, there $\exists$ path from $s \to t$:

$$\left.\begin{array}{l} \text{(1) starts from } s \text{ of } h(s) = n, \\ \text{(2) we can reach } t \text{ of } h(t) = 0, \\ \text{(3) one step at a time } h(u) \leq h(v)+1 \ \forall (u,v) \end{array}\right\}$$

$\Rightarrow$ # edges of such path $\geq n$ (2)

(1) contradicts (2) $\Rightarrow$ There is no path from $s \to t$ in G_f

Recall from Ford-Fulkerson and Edmonds-Karp:

No path from $s \to t$ in G_f $\Leftrightarrow$ Maximum flow

Differences between Ford-Fulkerson and Push-Relabel

- Ford-Fulkerson:
  - Invariants: "feasibility", meaning Flow Conservation is respected at all time. Start from 0 flow.
  - Goal: disconnect s and t, meaning no more augmenting path can be found. Works toward maximum flow

- Push-Relabel:
  - Invariants: disconnect s and t, meaning no path from $s \to t$ can be found
  - Goal: works toward restoring "feasibility", Flow Conservation is respected at the end.

Pseudo: Push-Relabel algorithm

- **Initialization:**

  Set initial height:

  $h(s) := n \qquad (n = |V|)$

  $h(v) := 0 \qquad \forall\ v \neq s$

  Set initial preflow:

  $f(s,u) = c(s,u) \quad \forall\ (s,u)$ be edges going out of s

  $f(u,v) = 0$ all other edges

  $\Rightarrow$ { Flow in > flow out at vertex ⓤ neighbor of s
  Invariants

- **Main loop:**

  While $\exists\ u \neq s,t$ with $e(u) > 0$

  (choose u with largest $h(u)$)

  If "can push", $\exists$ edge (u,v) s.t. $\begin{cases} c_f(u,v) > 0 \\ h(u) \leq h(v) + 1 \end{cases}$

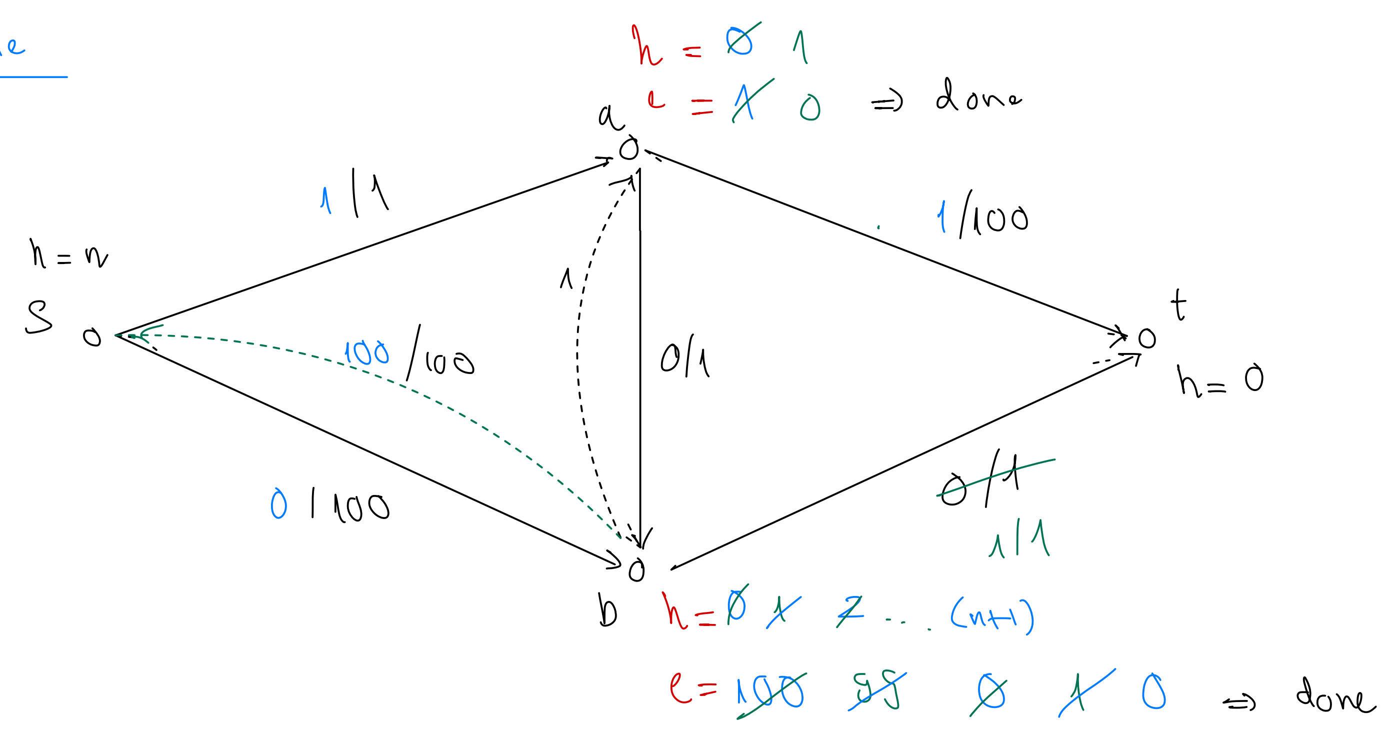
  $\Delta = \min\{e(u),\ c_f(u,v)\}$

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

  Else: (Relabel)

  $h(u) := h(u) + 1$

Example



The max flow is 2.

Complexity:

Terminate after $O(n^2)$ relabel

$O(n^3)$ pushes

Proof Complexity:

We use this key lemma to prove the bound of relabels

$$\boxed{\begin{array}{l}\textbf{Key Lemma}\\ \text{If } e(v) > 0, \text{ then } \exists \text{ a path } v \to s \text{ in } G_f\\ \text{Intuition: } e(v) > 0 \Rightarrow \text{Theres flow going from } s \to v\\ \qquad\qquad\qquad\quad \Rightarrow \text{Theres is "a way back" } v \to s\end{array}}$$

$$\boxed{\begin{array}{l}\textbf{Collary: Max height of vertex}\\ h(v) \leq 2n-1\end{array}}$$

Proof:

- Key lemma: If $e(v) > 0$, $\exists$ a path $v \to s$
- length path $v \to s \leq n-1$
- $h(s) = n$

$\Rightarrow h(v) - h(s) \leq n-1$

$\Leftrightarrow h(v) - n \leq n-1$

$\Leftrightarrow h(v) \leq 2n-1$

Proof 1: # Relabels $\leq O(n^2)$

We know:
- Relabel only when $e(v) > 0$ (key lemma)
- New relabeled height $\leq 2n-1$ (collary)
- There are n vertices to be relabeled

$\Rightarrow O(n(2n-1)) \sim O(n^2)$

Proof 2: # Pushes $\leq O(n^3)$

We know: There are 2 kind of pushes: "Saturating push", $c_f(v,w) > 0$ and "Non-saturating push", $e(v) > 0$

Case 1: "Saturating" push, $\Delta = c_f(v,w)$

Claim: "Between 2 saturing pushes on the same edge, (v,w) each vertices v and w relabeled at least 2 times"

Saturating push at time t: $\quad v \longrightarrow w \quad h(v) > h(w)$

⋮ some relabel has to happen

Saturating push at time $t+n$: $\quad v \longleftarrow w \quad h(v) < h(w)$

Since $h(v), h(w) \leq O(n)$

$\Rightarrow$ # relabels between saturating pushes $\leq O(n)$

$\Rightarrow$ Claim proven

There are m edges $\Rightarrow$ $O(m \cdot n)$

<u>Case 2</u>: "Non-saturating" pushes, $\Delta = e(v)$

Claim: "Between 2 relabels on the same vertex v, there are at most n non-saturating pushes"

If claim proven, easily see that case 2 complexity is $O(n^3)$ since # relabels is $O(n^2)$

Proof: Recall we pick the highest vertex among $\forall v$ s.t $e(v) > 0$

$\Leftrightarrow$ $h(v)$ larger than all other vertex with excess $e(w) > 0$

$\Rightarrow$ $h(v)$ stays highest until the next relabel

$\Rightarrow$ $e(v) = 0$ until the next relabel (flow only goes down-hill)

$\Rightarrow$ implies at most n non-saturating pushes until the next relabel (if all n non-saturating pushes performed, algorithm stops, because no more excess left)

$\Rightarrow$ Claim Proven