Push–relabel algorithms

Introduction

- Many of the asymptotically fastest maximum–flow algorithms are push–relabel algorithms
- Fastest actual implementations of maximum–flow algorithms are based on the push–relabel method.
- Push–relabel methods also efficiently solve other flow problems, such as the minimum–cost flow problem

Introduction

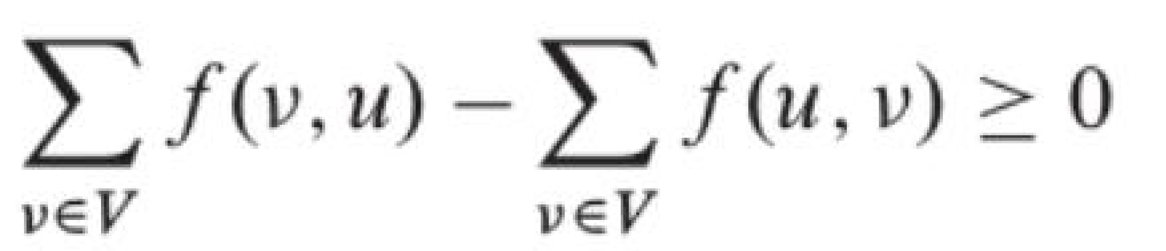
- Goldberg’s “generic” maximum–flow algorithm, which has a simple implementation that runs in $O(V^2E)$ time, thereby improving upon the $O(VE^2)$ bound of the Edmonds–Karp algorithm
- Work in a more localized manner than the Ford–Fulkerson method
- Doesn't examine entire residual network to find an augmenting path, work on one vertex at a time, looking only at the vertex’s neighbors in the residual network

Ford–Fulkerson vs Push–Relabel

- Work in a more localized manner
- Doesn't examine entire residual network to find an augmenting path, work on one vertex at a time, looking only at the vertex’s neighbors in the residual network
- do not maintain the flow–conservation property throughout their execution
- In–flow to a vertex can be temporarily more than the out–flow of the vertex

Ford–Fulkerson vs Push–Relabel

- They do, however, maintain a preflow, which is a function f : V x V → R that satisfies the capacity constraint and the following relaxation of flow conservation:



$$\sum_{v \in V} f(v,u) - \sum_{v \in V} f(u,v) \geq 0$$

- for all vertices u Є V – {s}. That is, the flow into a vertex may exceed the flow out

Ford–Fulkerson vs Push–Relabel

- We call the quantity

$$e(u) = \sum_{v \in V} f(v,u) - \sum_{v \in V} f(u,v)$$

- the excess flow into vertex u.
- Excess at a vertex is the amount by which the flow in exceeds the flow out
- We say that a vertex u ∈ V – {s, t} is overflowing if e(u) > 0.

Intitution of Push–Relabel

- Directed edges correspond to pipes
- Vertices, which are pipe junctions, have two interesting properties
  - (i) to accommodate excess flow, each vertex has an out–flow pipe leading to an arbitrarily large reservoir that can accumulate fluid.
  - (ii) each vertex, its reservoir, and all its pipe connections sit on a platform whose height increases as the algorithm progresses
- We push flow only downhill, that is, from a higher vertex to a lower vertex

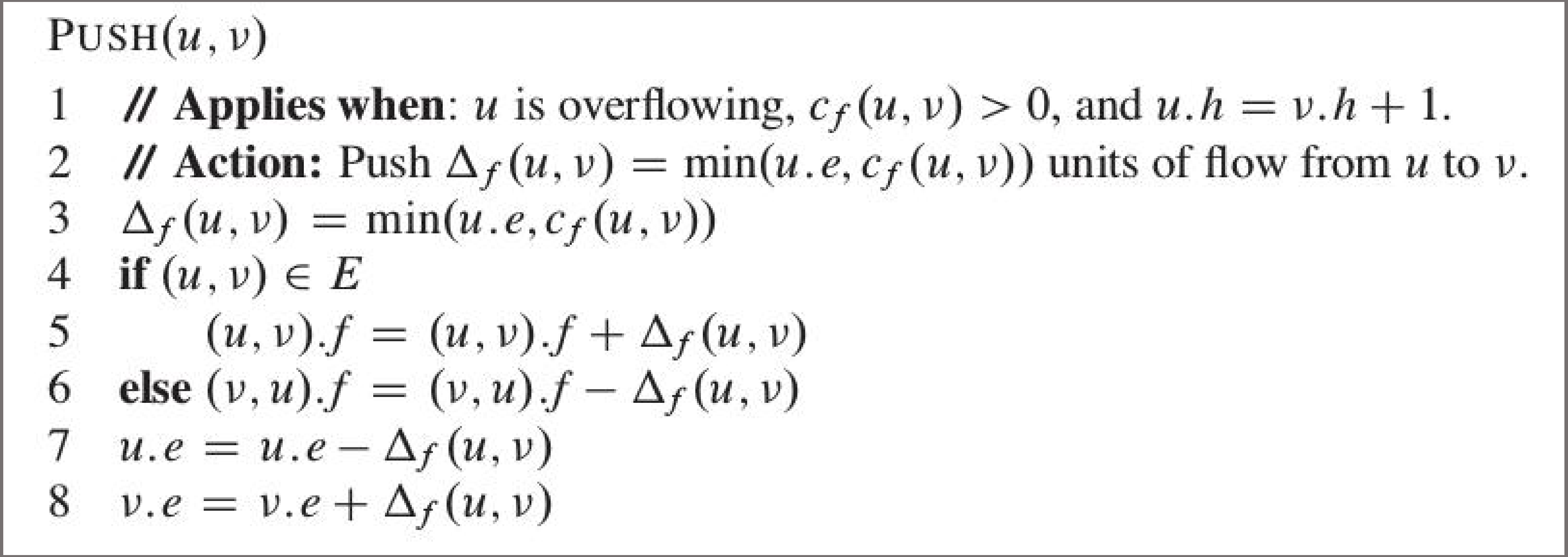
Ideaof Push–Relabel

- Initialize height of source at |V| and the height of the sink at 0
- All other vertex heights start at 0 and increase with time
- algorithm first sends as much flow as possible downhill from the source toward the sink
- to rid an overflowing vertex u of its excess flow, we must increase its height—an operation called “relabeling” vertex u
- We increase its height to one unit more than the height of the lowest of its neighbors to which it has an unsaturated pipe

Make Pre–flow to legal flow

- algorithm sends the excess collected in the reservoirs of overflowing vertices back to the source by continuing to relabel vertices to above the fixed height |V| of the source
- once we have emptied all the reservoirs, the preflow is not only a “legal” flow, it is also a maximum flow

Make Pre–flow to legal flow

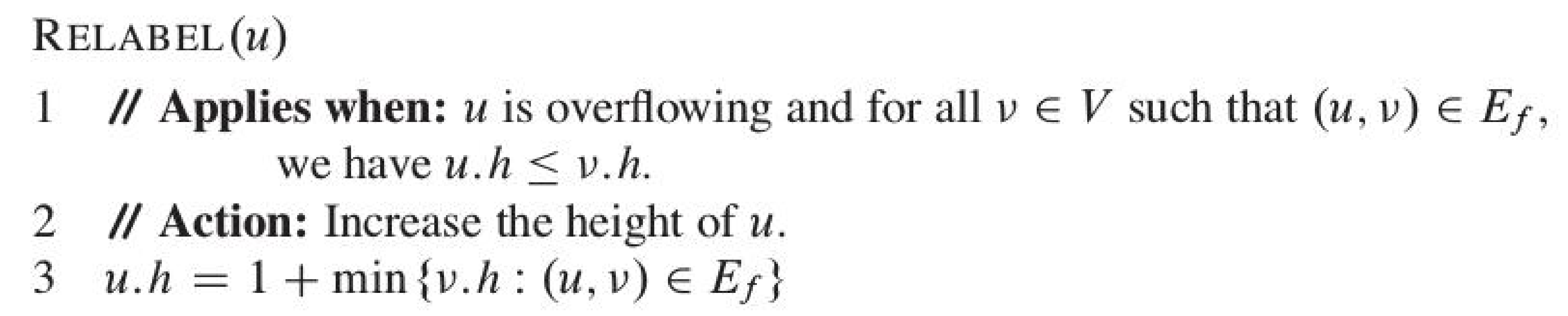


```
PUSH$(u, v)$
// Applies when: $u$ is overflowing, $c_f(u, v) > 0$, and $u.h = v.h + 1$.
// Action: Push $\Delta_f(u, v) = \min(u.e, c_f(u, v))$ units of flow from $u$ to $v$.
$\Delta_f(u, v) = \min(u.e, c_f(u, v))$
if $(u, v) \in E$
    $(u, v).f = (u, v).f + \Delta_f(u, v)$
else $(v, u).f = (v, u).f - \Delta_f(u, v)$
$u.e = u.e - \Delta_f(u, v)$
$v.e = v.e + \Delta_f(u, v)$
```

Make Pre–flow to legal flow

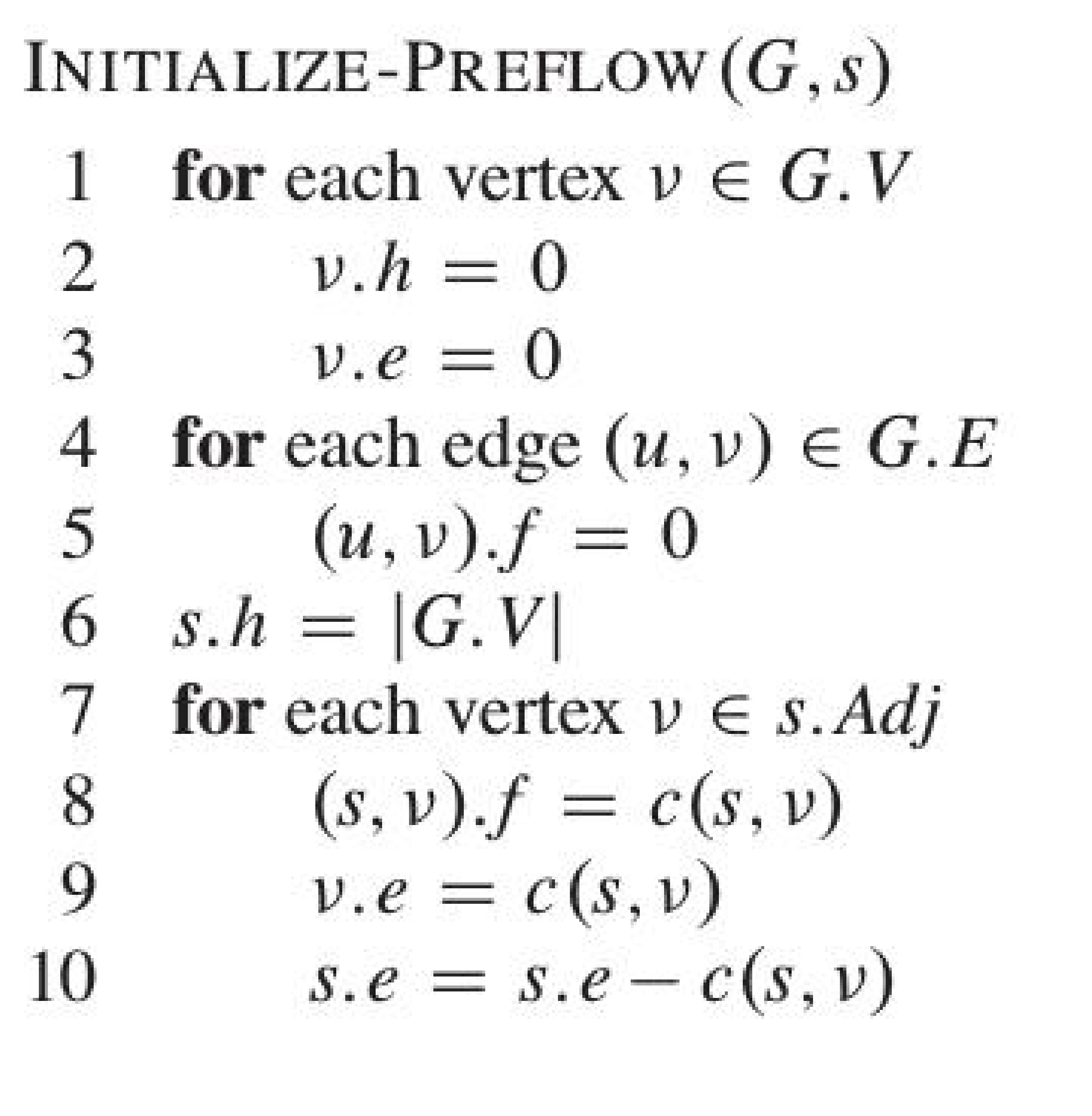
- We call the operation PUSH (u, v) a push from u to v
- If a push operation applies to some edge (u, v) leaving a vertex u, we also say that the push operation applies to u.
- It is a saturating push if edge (u,v) in the residual network becomes saturated (c_f (u, v) = 0 afterward); otherwise, it is a nonsaturating push.
- If an edge becomes saturated, it disappears from the residual network

Make Pre–flow to legal flow



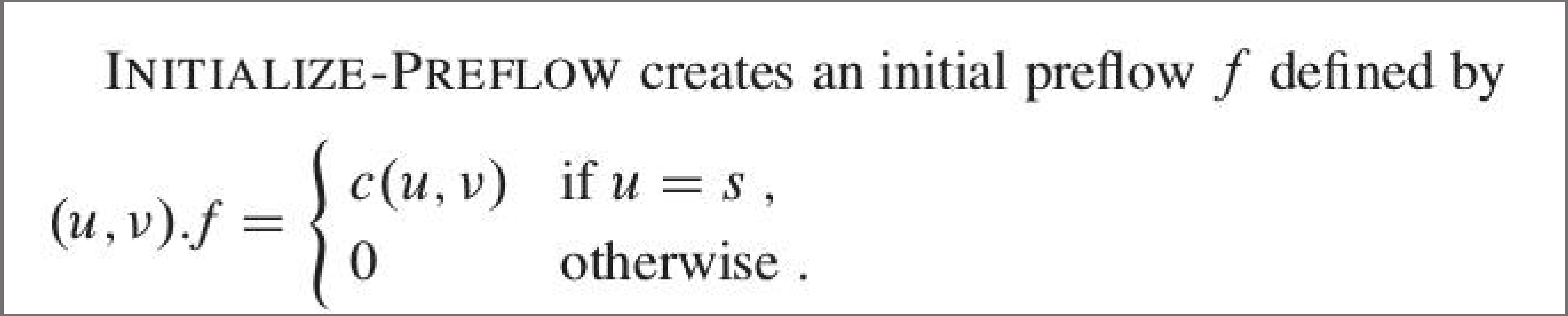
```
RELABEL($u$)
  // Applies when: $u$ is overflowing and for all $v \in V$ such that $(u, v) \in E_f$,
          we have $u.h \le v.h$.
  // Action: Increase the height of $u$.
  $u.h = 1 + \min\{v.h : (u, v) \in E_f\}$
```

Make Pre–flow to legal flow



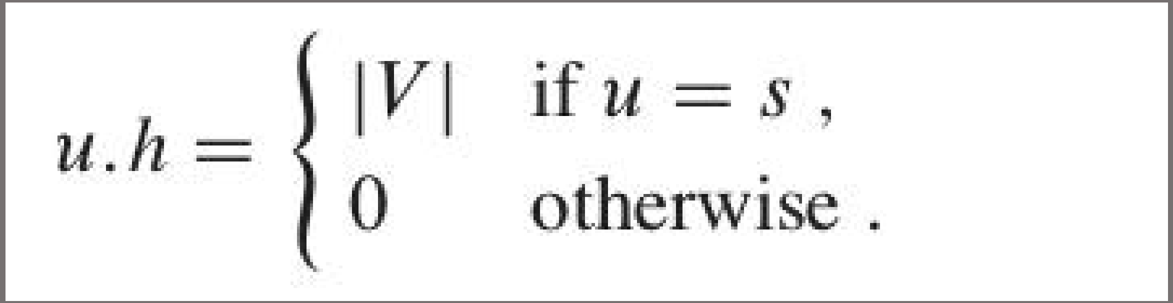
```
INITIALIZE-PREFLOW(G, s)
for each vertex $v \in G.V$
    $v.h = 0$
    $v.e = 0$
for each edge $(u, v) \in G.E$
    $(u, v).f = 0$
$s.h = |G.V|$
for each vertex $v \in s.Adj$
    $(s, v).f = c(s, v)$
    $v.e = c(s, v)$
    $s.e = s.e - c(s, v)$
```

Make Pre–flow to legal flow

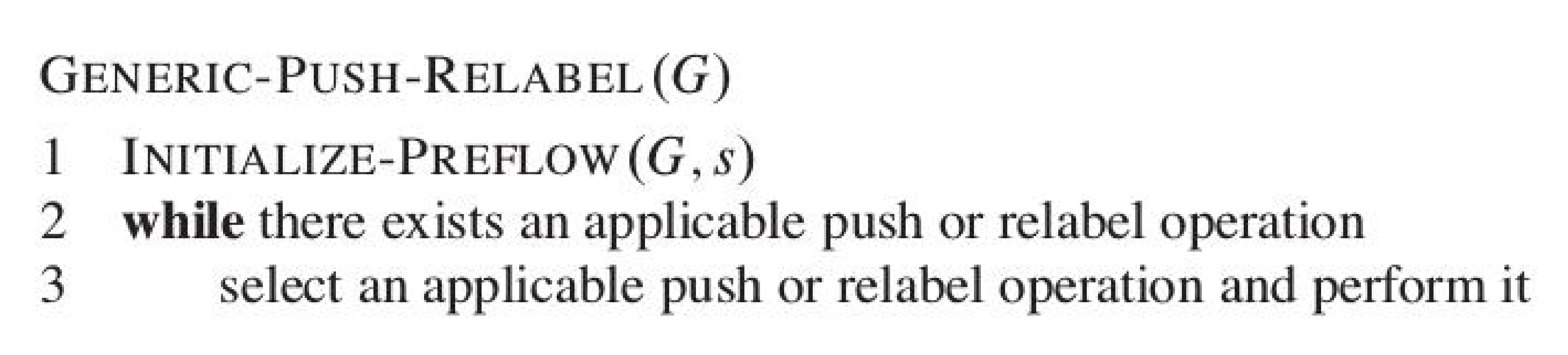


INITIALIZE-PREFLOW creates an initial preflow f defined by

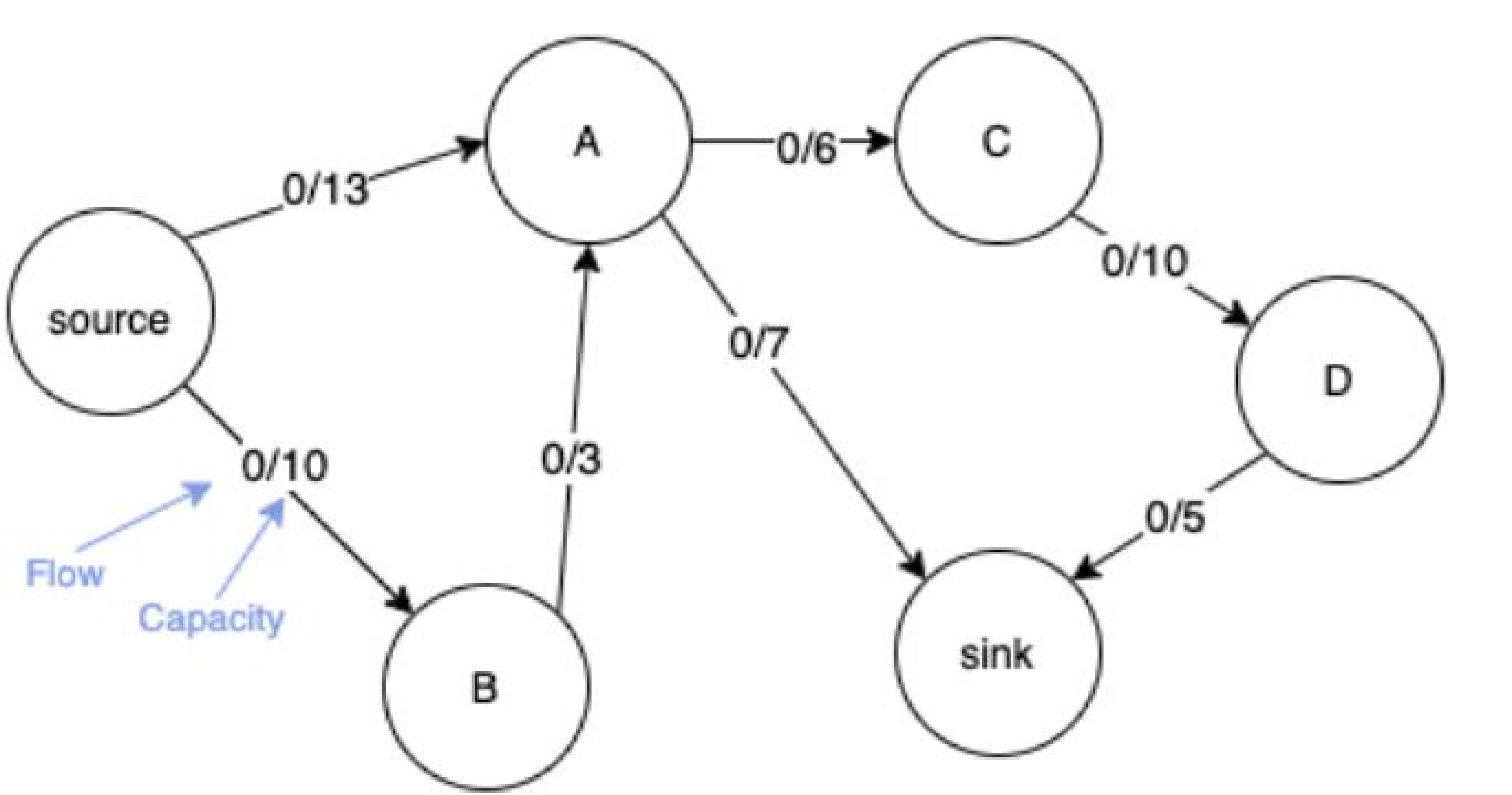
$$(u,v).f = \begin{cases} c(u,v) & \text{if } u = s \,, \\ 0 & \text{otherwise} \,. \end{cases}$$



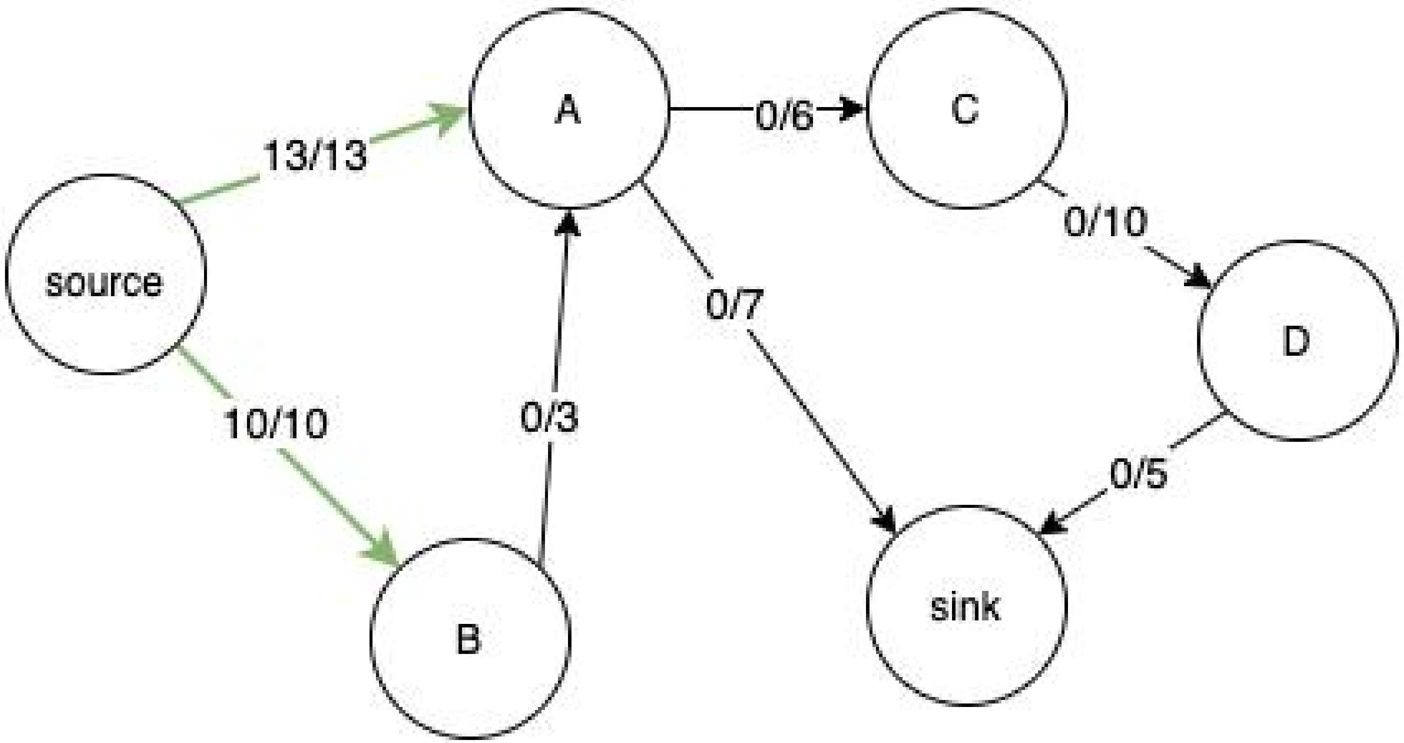
$$u.h = \begin{cases} |V| & \text{if } u = s \; , \\ 0 & \text{otherwise} \; . \end{cases}$$



```
GENERIC-PUSH-RELABEL$(G)$
INITIALIZE-PREFLOW$(G, s)$
while there exists an applicable push or relabel operation
    select an applicable push or relabel operation and perform it
```

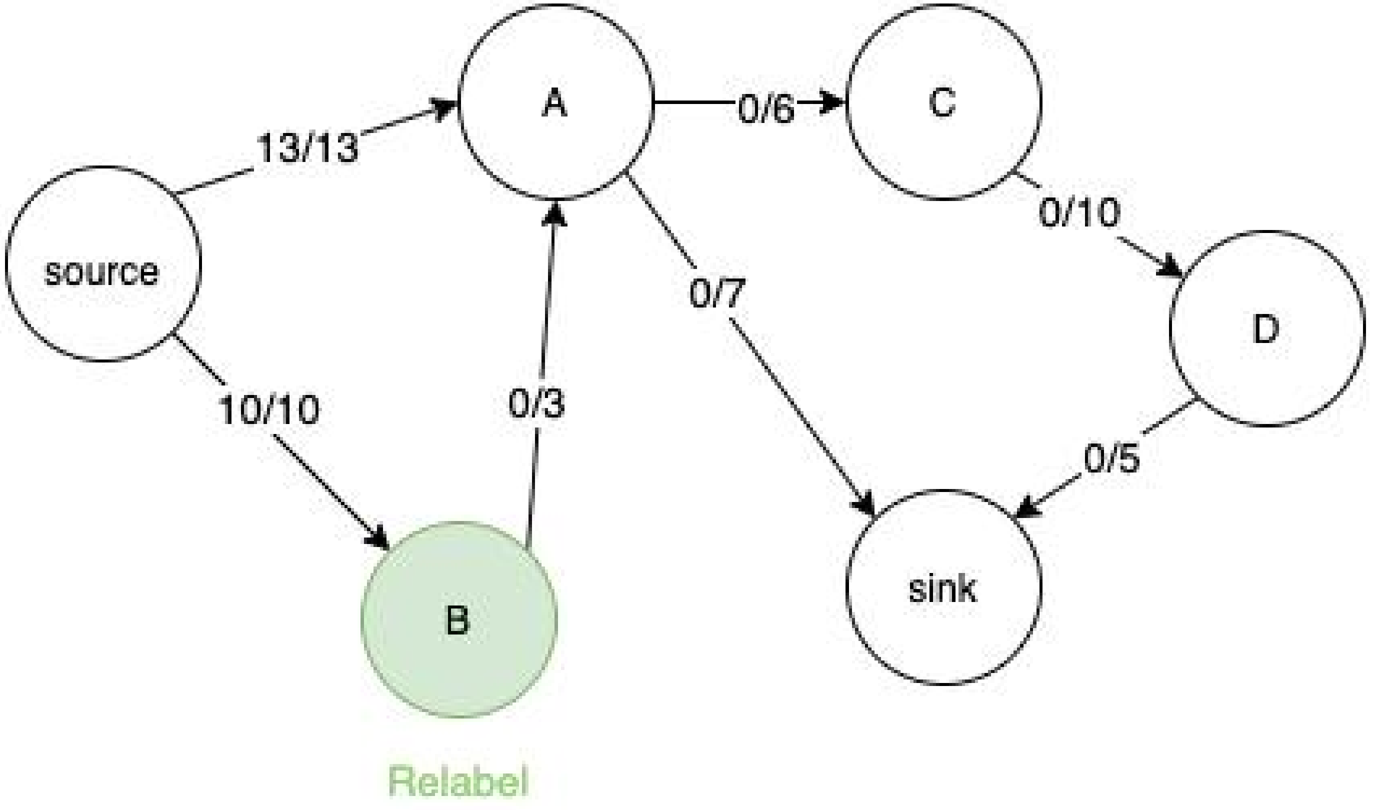

A
C
0/13
0/6
source
0/10
0/7
D
0/10
0/3
0/5
Flow
Capacity
B
sink

Initialise the graph by setting heights and excess flow



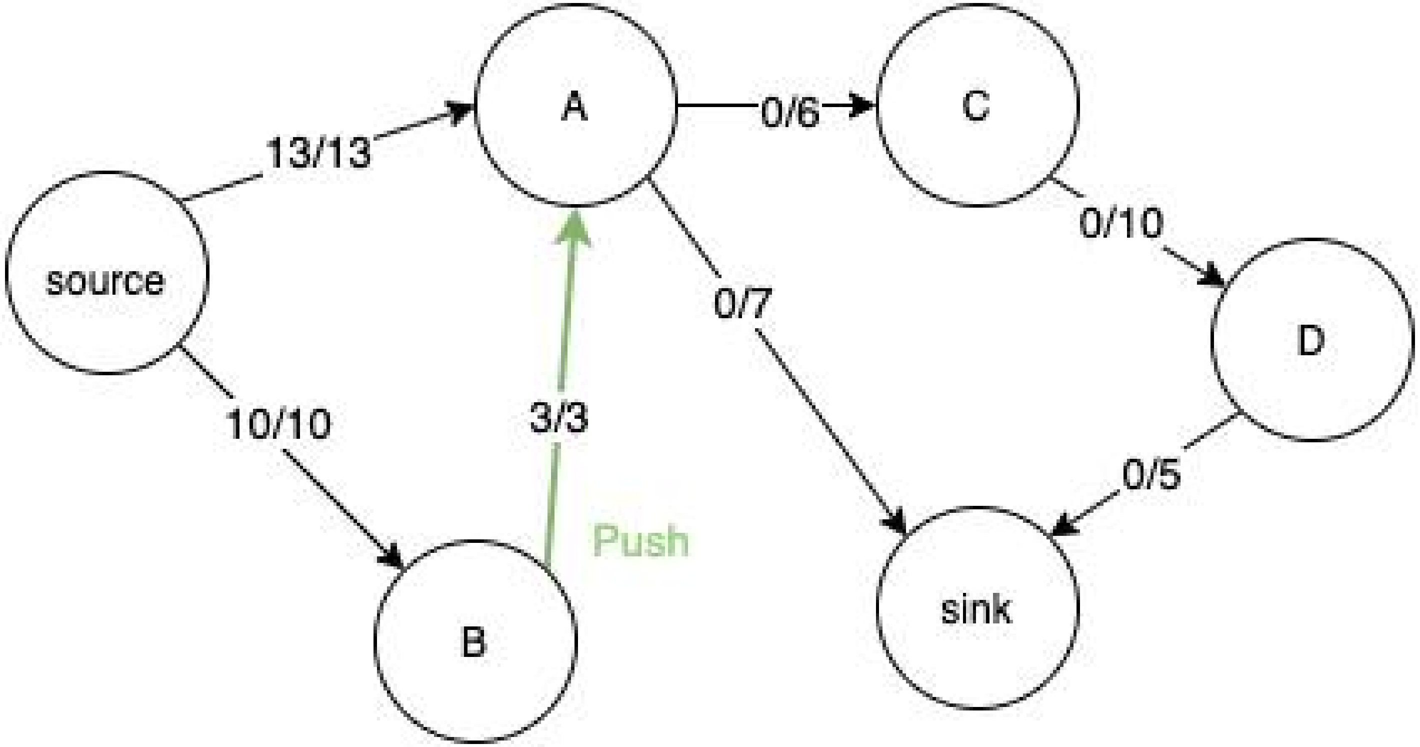
Node	Height	Excess Flow
source	6	
A	0	13
B	0	10
C	0	
D	0	
sink	0	

Consider vertex B. It cannot transfer its excess flow as its adjacent node A has the same height. So we relabel it.



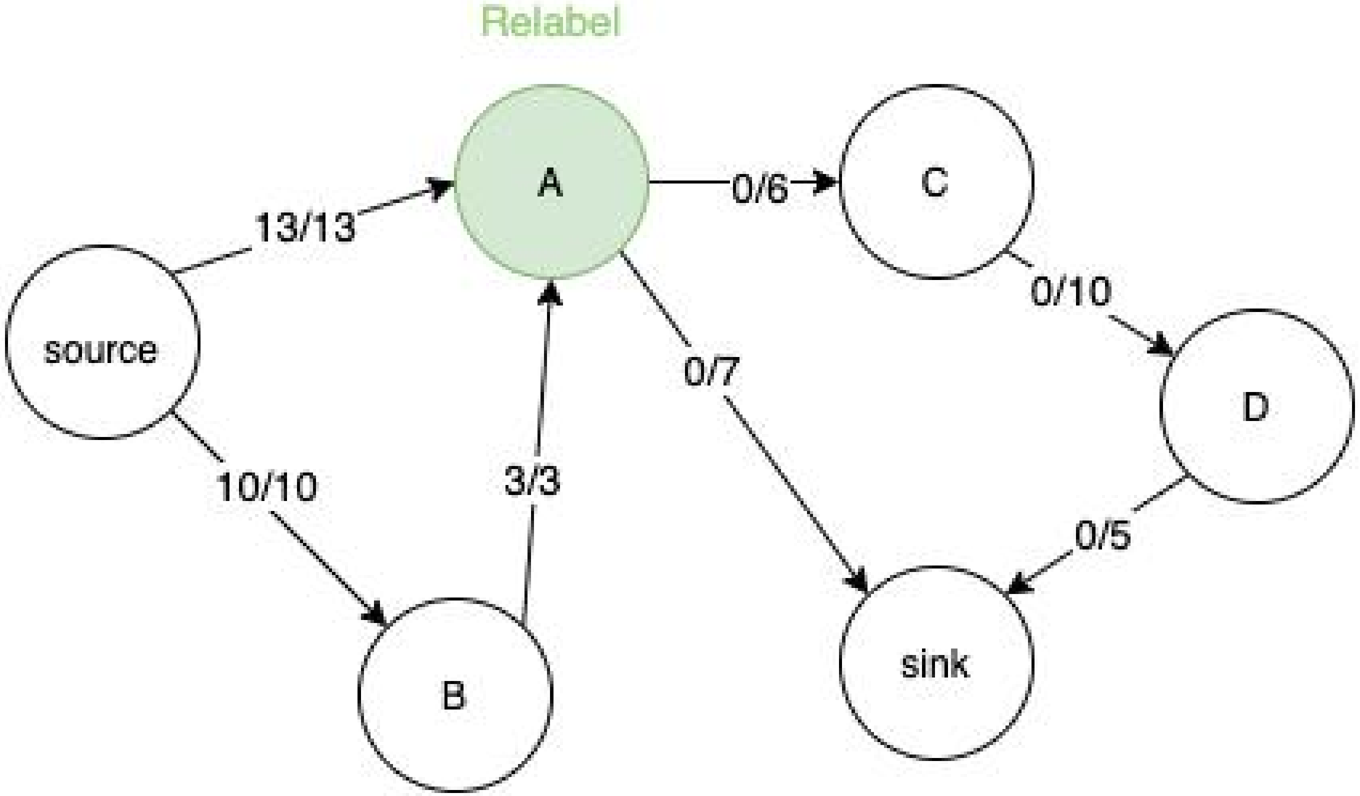
Node	Height	Excess Flow
source	6	-
A	0	13
B	1	10
C	0	
D	0	
sink	0	-

Now B can push its excess flow to A

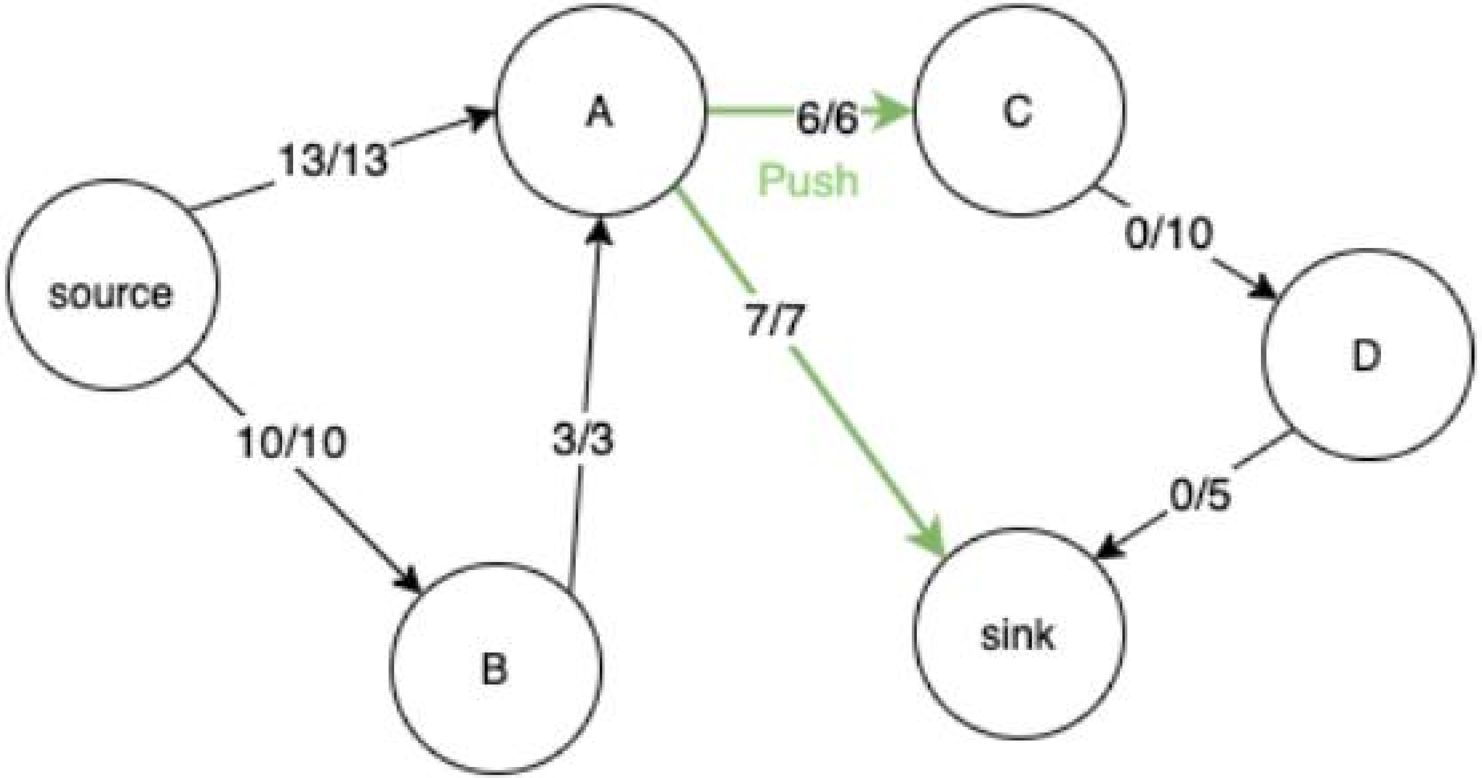


Node	Height	Excess Flow
source	6	-
A	0	16
B	1	7
C	0	
D	0	
sink	0	-

Similarly consider node A. We relabel its height to 1 so that it can transfer flow to its adjacent nodes C and sink.

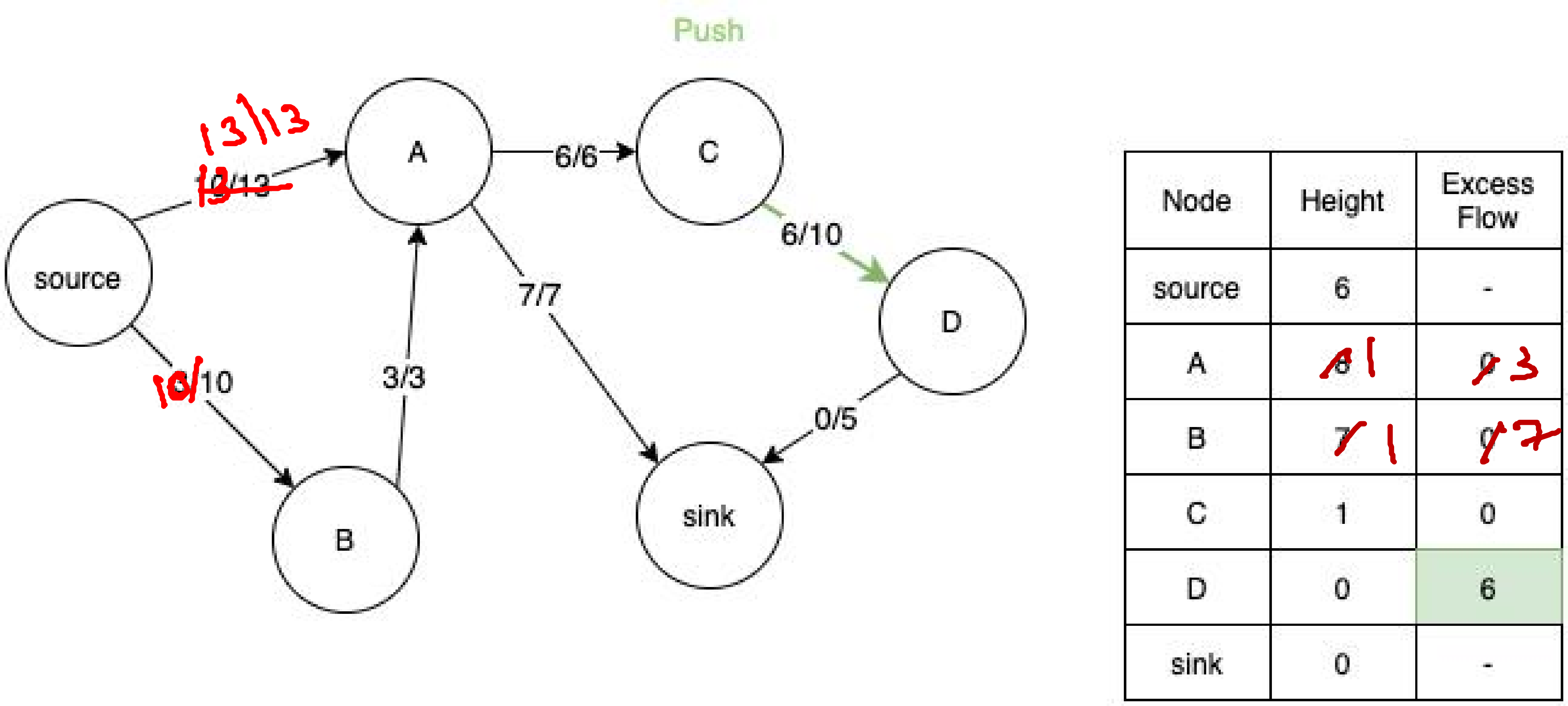


Node	Height	Excess Flow
source	6	-
A	1	16
B	1	7
C	0	
D	0	
sink	0	-



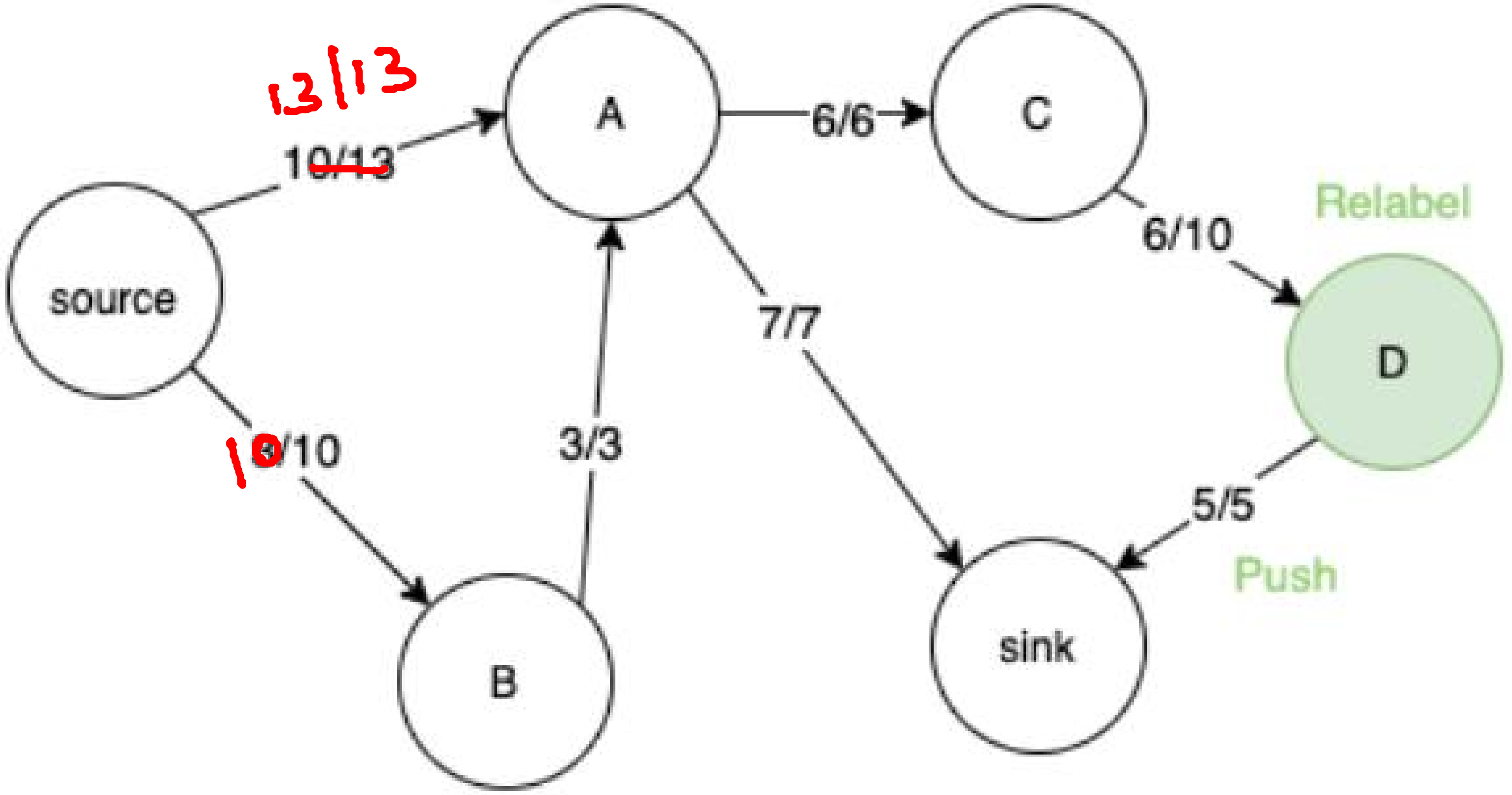
Node	Height	Excess Flow
source	6	-
A	1	3
B	1	7
C	0	6
D	0	
sink	0	-

Now we consider node C and push its extra flow to node D



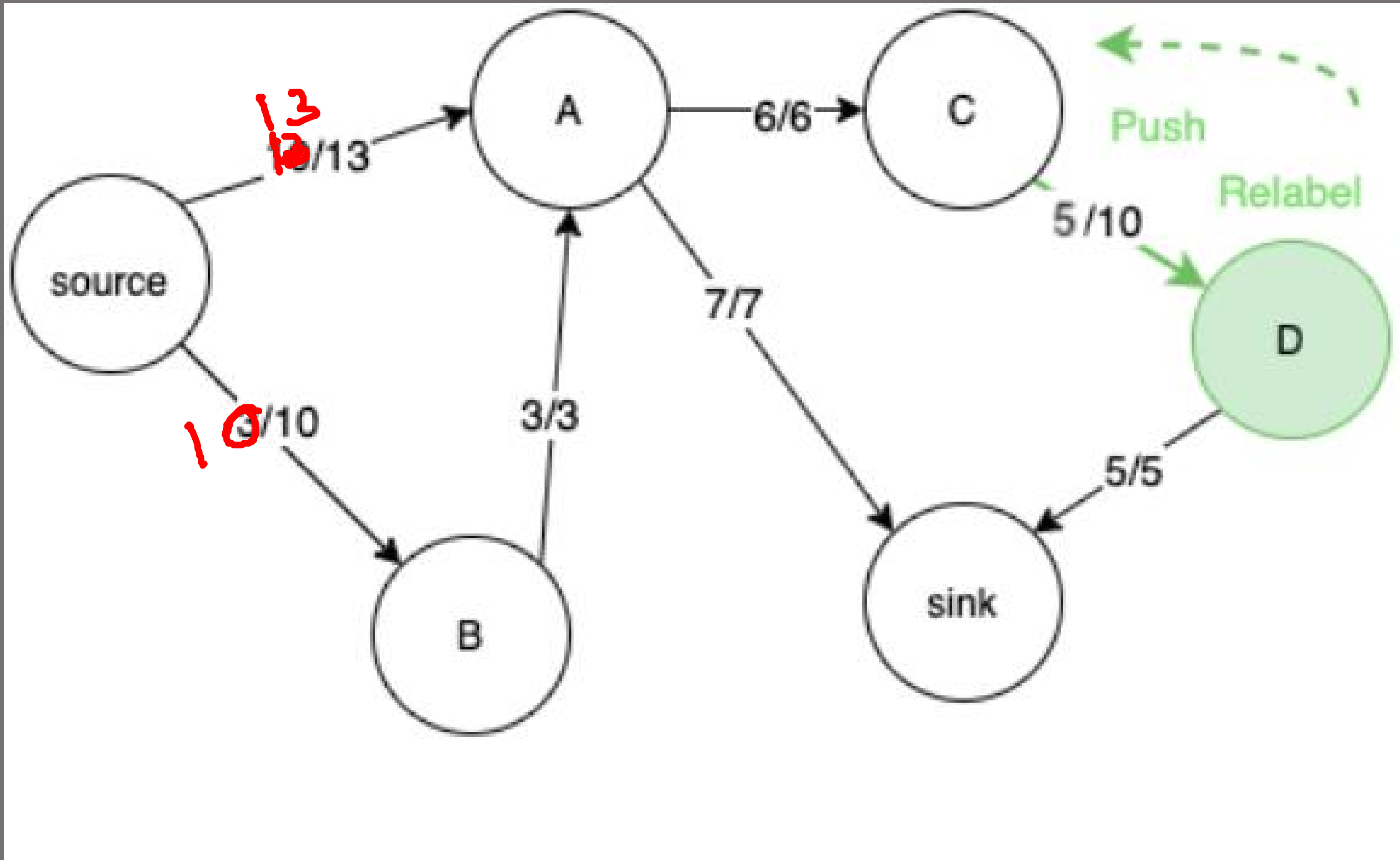
Node	Height	Excess Flow
source	6	-
A	~~8~~ 1	~~0~~ 3
B	~~7~~ 1	~~0~~ 7
C	1	0
D	0	6
sink	0	-

We relabel D and push its extra flow to sink. However we are still left with 1 unit of extra flow in D.



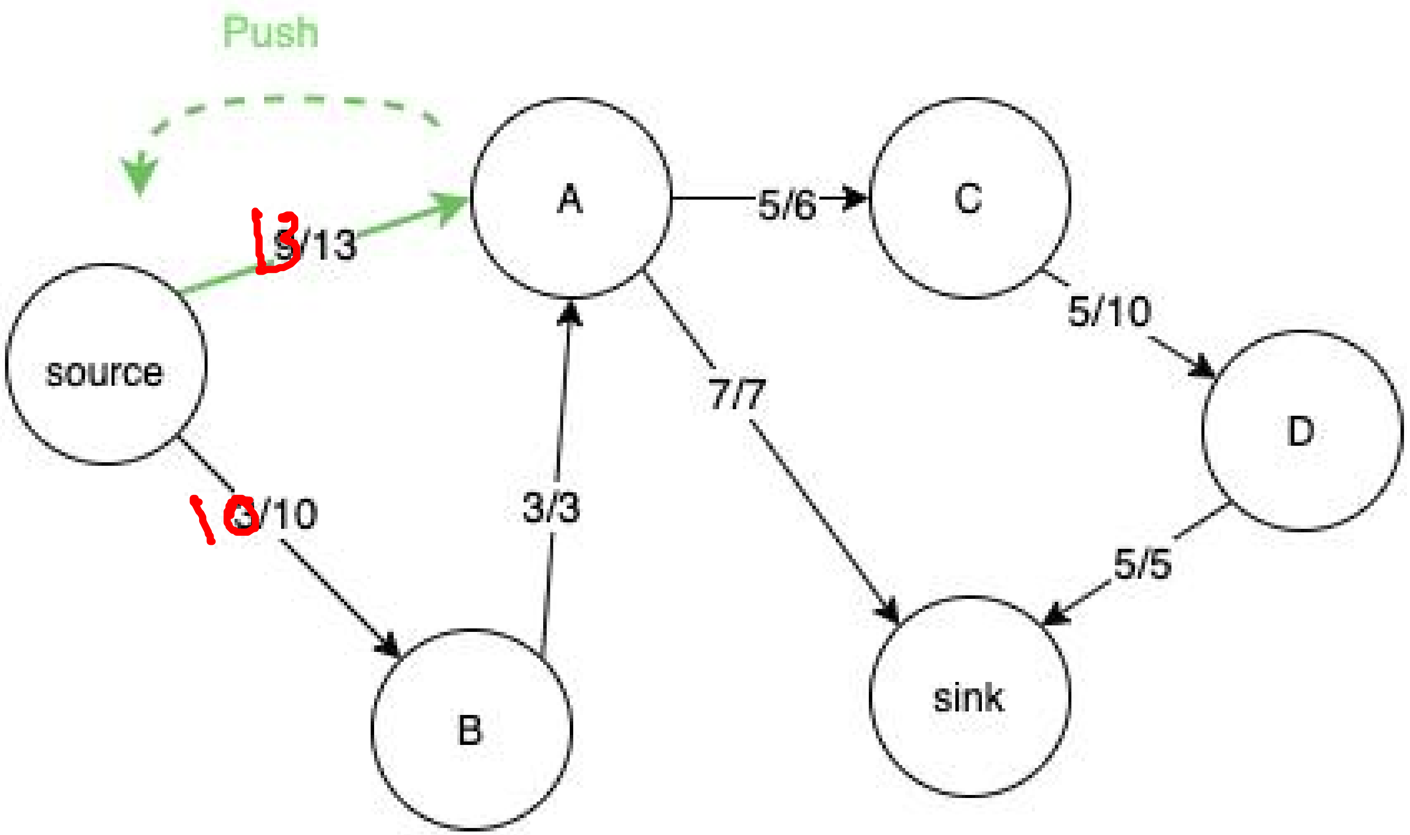
Node	Height	Excess Flow
source	6	-
A	~~8~~ \	~~0~~ 3
B	~~7~~ \	~~0~~ 7
C	1	0
D	1	1
sink	0	-

So, we relabel D again and push back the extra flow to C.



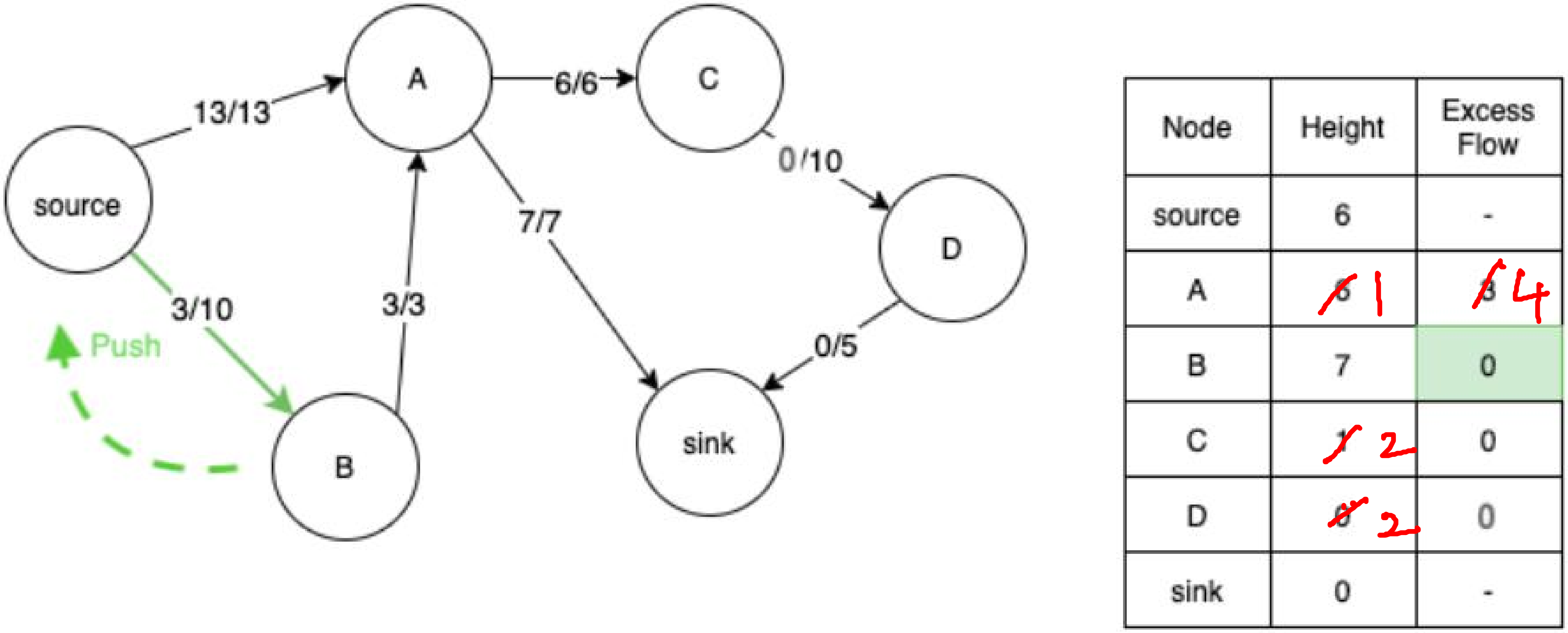
Node	Height	Excess Flow
source	6	-
A	~~8~~ 1	~~0~~ 3
B	~~7~~ 1	~~0~~ 7
C	1	1
D	2	0
sink	0	-

Push excess flow in C to A



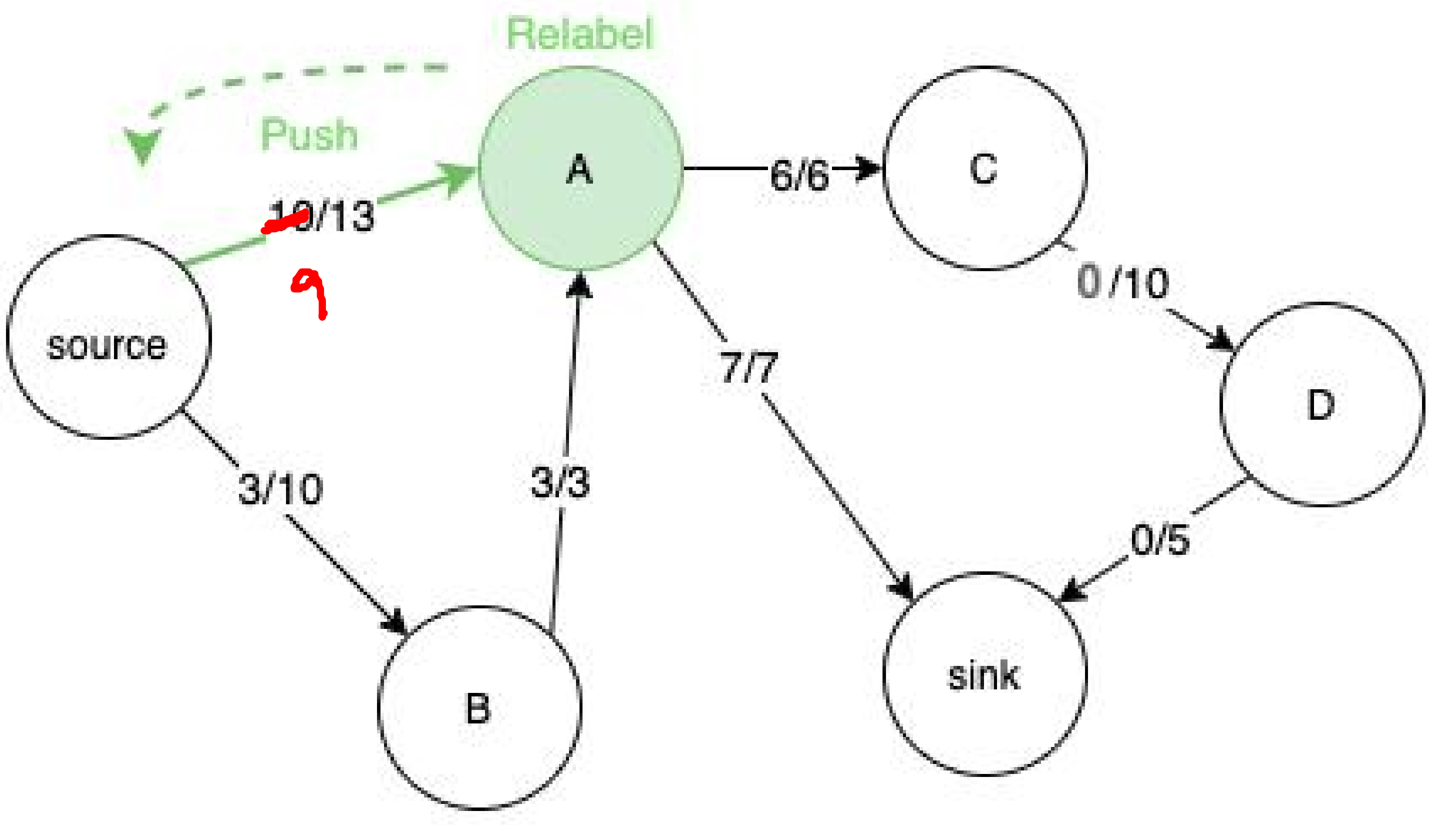
Node	Height	Excess Flow
source	6	-
A	~~8~~ 1	4= ~~0~~ 13+1
B	~~7~~ 1	~~0~~ 7
C	~~6~~ 2	0
D	~~5~~ 2	0
sink	0	-

A to and fro operation between A and B happens till height of B is greater than source node. Now we can push back the extra flow back to source node.



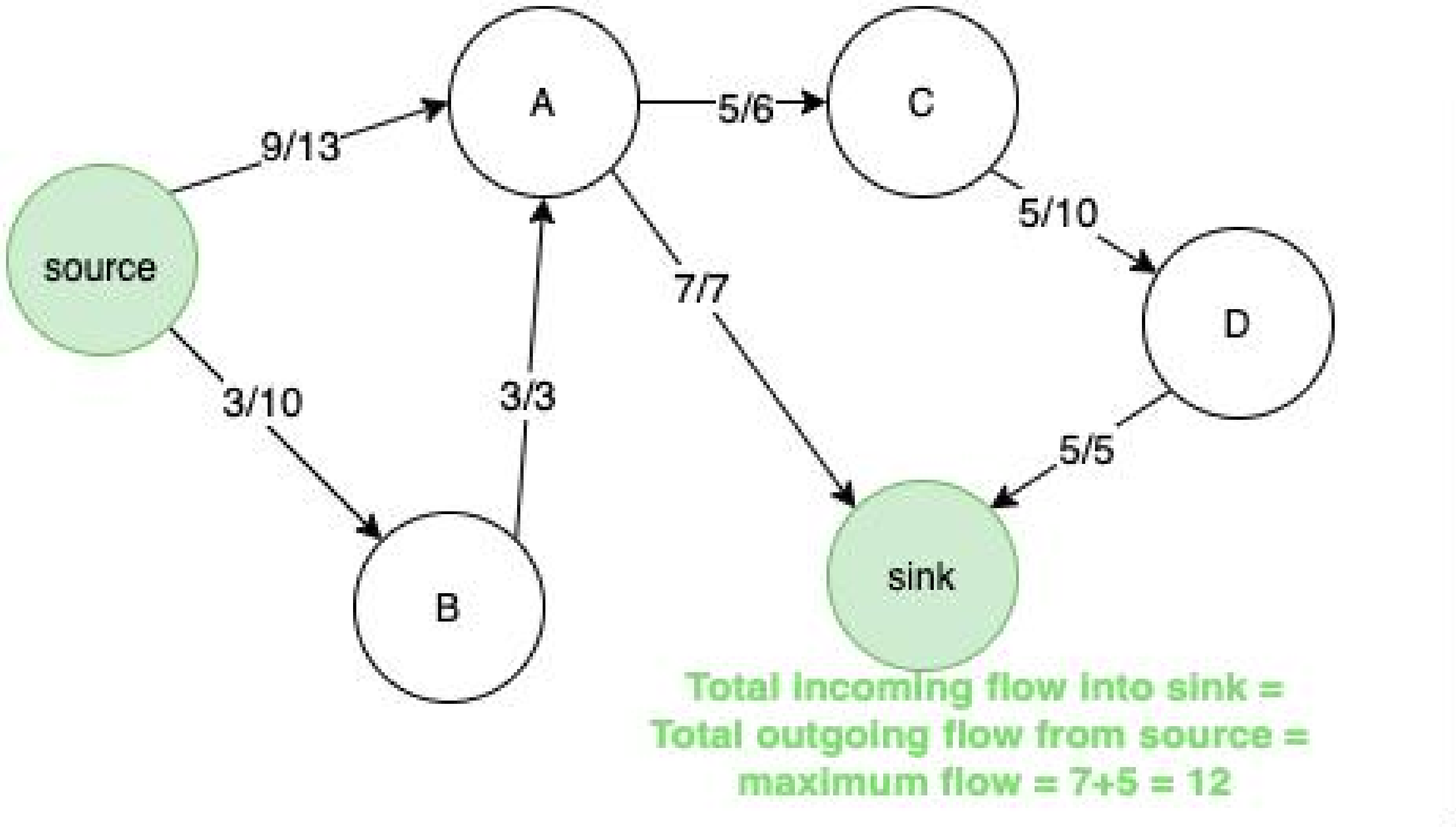
Node	Height	Excess Flow
source	6	-
A	~~6~~ 1	~~3~~ 4
B	7	0
C	~~1~~ 2	0
D	~~0~~ 2	0
sink	0	-

Similarly, when we relabel A, its height also becomes greater than source and we can push back extra flow to the source. Now both A and B nodes have 0 extra flow



Node	Height	Excess Flow
source	6	-
A	8	0
B	7	0
C	~~1~~ 2	0
D	~~0~~ 2	0
sink	0	-

Maximum flow can be calculated by summing the total outgoing flow from the source or total incoming flow in the sink. In this case it is equal to 12.



Total incoming flow into sink =
Total outgoing flow from source =
maximum flow = 7+5 = 12

Node	Height	Excess Flow
source	6	-
A	~~8~~ 11	~~0~~ 4
B	7	0
C	~~8~~ 12	0
D	~~8~~ 10	0
sink	0	-