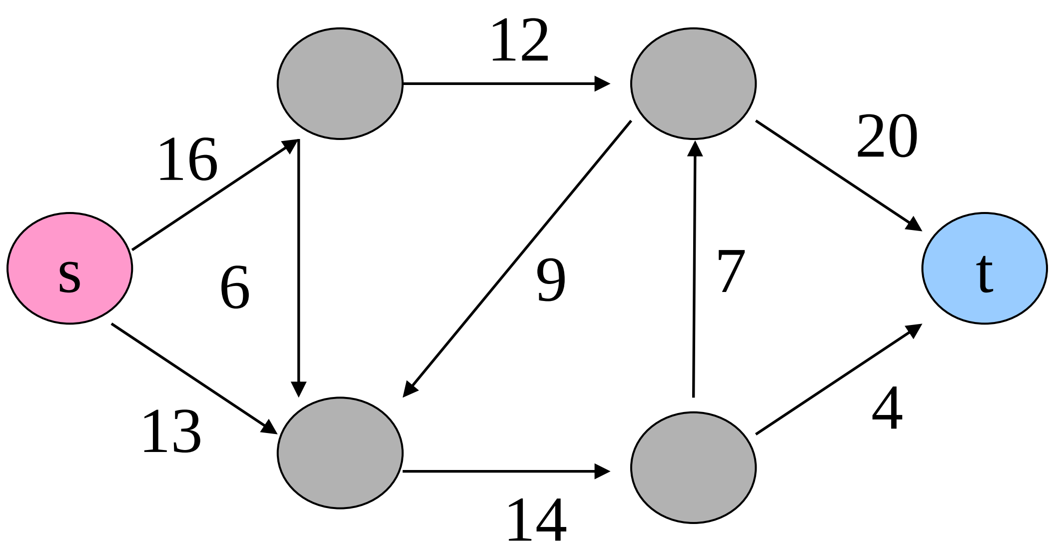
Maximum Flow

Flow networks:

- A flow network G=(V,E): a directed graph, where each edge $(u,v) \in E$ has a nonnegative capacity $c(u,v) >= 0$.
- If $(u,v) \notin E$, we assume that $c(u,v)=0$.
- two distinct vertices :**a source s and a sink t.**

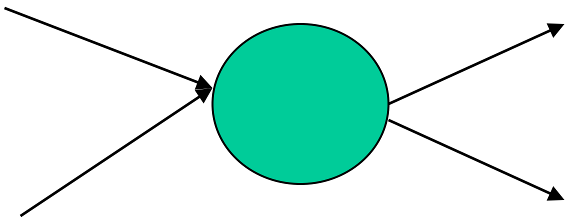


Flow:

- G=(V,E): a flow network with capacity function c.
- s-- the source and t-- the sink.
- A flow in G: a real-valued function $f:V*V \rightarrow R$ satisfying the following two properties:
- Capacity constraint: For all $u,v \in V$,

  we require $0 \le f(u,v) \le c(u,v)$.
- Flow conservation: For all $u \in V-\{s,t\}$, we require

$$\sum_{e.in.v} f(e) = \sum_{e.out.v} f(e)$$

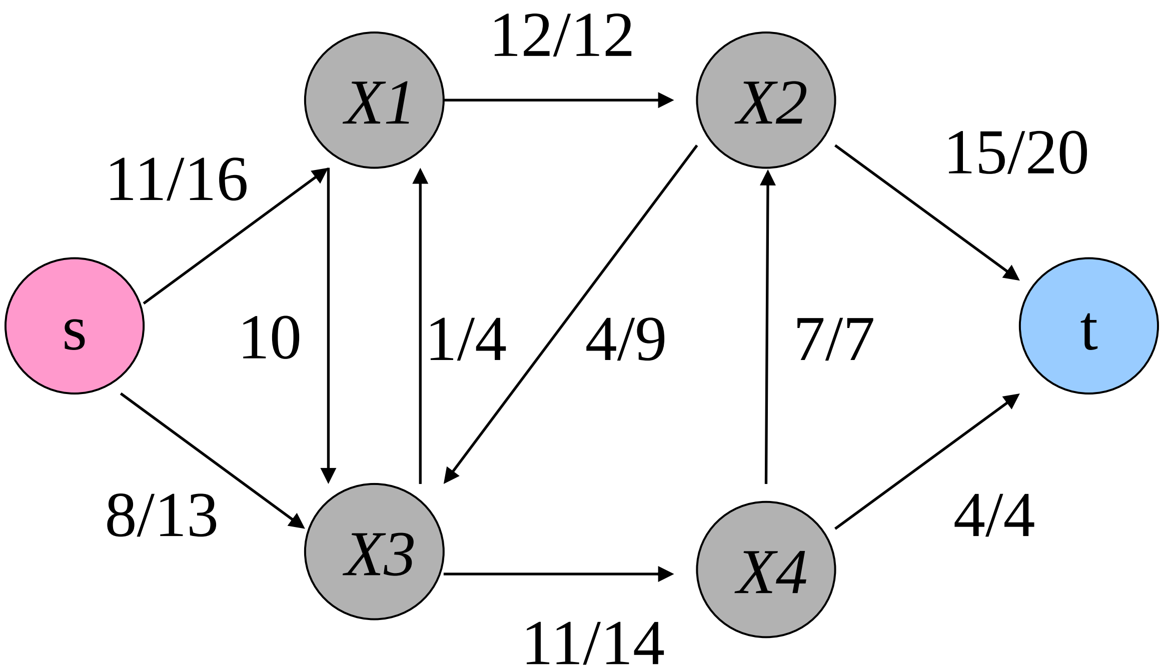


Net flow and value of a flow f:

- The quantity f (u,v) is called the net flow from vertex u to vertex v.
- The value of a flow is defined as

$$|f| = \sum_{v \in V} f(s,v)$$

  - The total flow from source to any other vertices.
  - The same as the total flow from any vertices to **the sink.**



A flow f in G with value $|f| = 19$.

Maximum-flow problem:

- Given a flow network G with source s and sink t
- Find a flow of maximum value from s to t.

- How to solve it efficiently?

The Ford-Fulkerson method:

Continue:

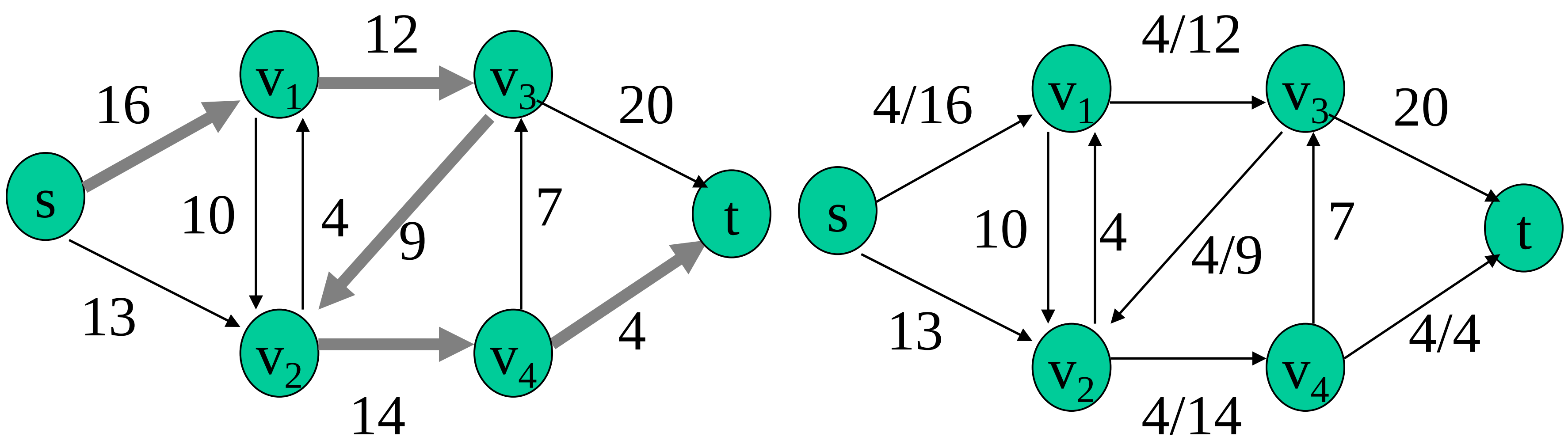
- FORD-FULKERSON-METHOD(G,s,t)
- initialize flow *f* to *0*
- while there exists an *augmenting* path *p*
- do *augment* flow *f* along *p*
- return *f*

Residual networks:

- Given a flow network and a flow, the **residual network** consists of edges that can admit more net flow.
- G=(V,E) --a flow network with source s and sink t
- f: a flow in G.
- The amount of additional net flow from u to v before exceeding the capacity c(u,v) is the residual capacity of (u,v), given by: $c_f(u,v)=c(u,v)-f(u,v)$

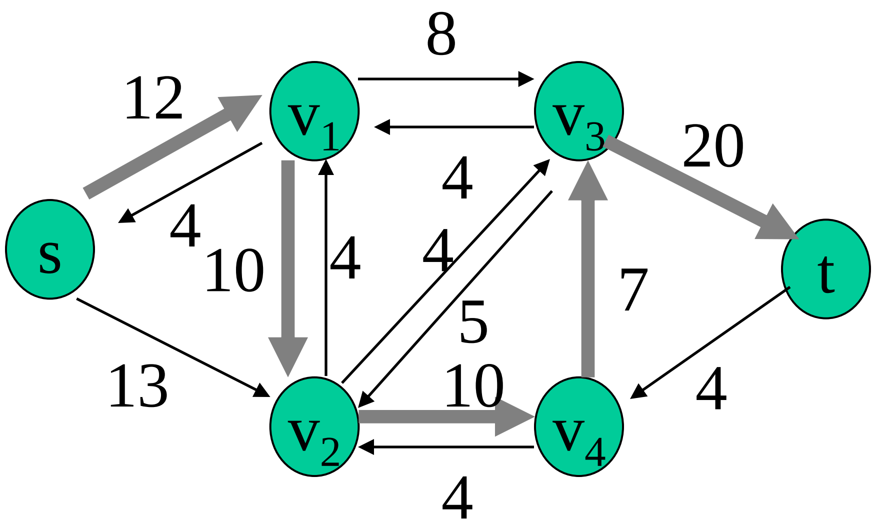
  in the other direction: $c_f(v, u)=c(v, u)+f(u, v)$.

Example of residual network



(a)

Example of Residual network (continued)



(b)

Fact 1:

- Let G=(V,E) be a flow network with source s and sink t, and let f be a flow in G.
- Let G_f be the residual network of G induced by f,and let f' be a flow in G_f.Then, the flow sum f+f' is a flow in G with value $|f + f'| = |f| + |f'|$
- f+f': the flow in the same direction will be added.

  the flow in different directions will be cnacelled.

A flow in a residual network provides a roadmap for adding flow to the original flow network. If f is a flow in G and f' is a flow in the corresponding residual network G_f, we define $f \uparrow f'$, the ***augmentation*** of flow f by f', to be a function from $V \times V$ to $\mathbb{R}$, defined by

$$(f \uparrow f')(u,v) = \begin{cases} f(u,v) + f'(u,v) - f'(v,u) & \text{if } (u,v) \in E \text{ ,} \\ 0 & \text{otherwise .} \end{cases}$$

Let $G = (V, E)$ be a flow network with source s and sink t, and let f be a flow in G. Let G_f be the residual network of G induced by f, and let f' be a flow in G_f. Then the function $f \uparrow f'$ defined in equation (26.4) is a flow in G with value $|f \uparrow f'| = |f| + |f'|$.

For the capacity constraint, first observe that if $(u, v) \in E$, then $c_f(v, u) = f(u, v)$. Therefore, we have $f'(v, u) \le c_f(v, u) = f(u, v)$, and hence

$$
\begin{aligned}
(f \uparrow f')(u, v) &= f(u, v) + f'(u, v) - f'(v, u) && \text{(by equation (26.4))} \\
&\ge f(u, v) + f'(u, v) - f(u, v) && \text{(because } f'(v, u) \le f(u, v)\text{)} \\
&= f'(u, v) \\
&\ge 0 .
\end{aligned}
$$

In addition,

$$
\begin{aligned}
(f \uparrow f')(u, v) \\
&= f(u, v) + f'(u, v) - f'(v, u) \\
&\le f(u, v) + f'(u, v) \\
&\le f(u, v) + c_f(u, v) \\
&= f(u, v) + c(u, v) - f(u, v) \\
&= c(u, v) .
\end{aligned}
$$

For flow conservation, because both f and f' obey flow conservation, we have that for all $u \in V - \{s, t\}$,

$$\begin{aligned}
\sum_{v \in V} (f \uparrow f')(u, v) &= \sum_{v \in V} (f(u, v) + f'(u, v) - f'(v, u)) \\
&= \sum_{v \in V} f(u, v) + \sum_{v \in V} f'(u, v) - \sum_{v \in V} f'(v, u) \\
&= \sum_{v \in V} f(v, u) + \sum_{v \in V} f'(v, u) - \sum_{v \in V} f'(u, v) \\
&= \sum_{v \in V} (f(v, u) + f'(v, u) - f'(u, v)) \\
&= \sum_{v \in V} (f \uparrow f')(v, u) ,
\end{aligned}$$

Finally, we compute the value of $f \uparrow f'$. Recall that we disallow antiparallel edges in G (but not in G_f), and hence for each vertex $v \in V$, we know that there can be an edge (s, v) or (v, s), but never both. We define $V_1 = \{v : (s, v) \in E\}$ to be the set of vertices with edges from s, and $V_2 = \{v : (v, s) \in E\}$ to be the set of vertices with edges to s. We have $V_1 \cup V_2 \subseteq V$ and, because we disallow antiparallel edges, $V_1 \cap V_2 = \emptyset$. We now compute

$$
\begin{aligned}
|f \uparrow f'| &= \sum_{v \in V} (f \uparrow f')(s, v) - \sum_{v \in V} (f \uparrow f')(v, s) \\
&= \sum_{v \in V_1} (f \uparrow f')(s, v) - \sum_{v \in V_2} (f \uparrow f')(v, s) ,
\end{aligned}
$$

$$
\begin{aligned}
&= \sum_{v \in V_1} (f(s, v) + f'(s, v) - f'(v, s)) - \sum_{v \in V_2} (f(v, s) + f'(v, s) - f'(s, v)) \\
&= \sum_{v \in V_1} f(s, v) + \sum_{v \in V_1} f'(s, v) - \sum_{v \in V_1} f'(v, s) \\
&\qquad - \sum_{v \in V_2} f(v, s) - \sum_{v \in V_2} f'(v, s) + \sum_{v \in V_2} f'(s, v)
\end{aligned}
$$

$$= \sum_{v \in V_1} f(s,v) - \sum_{v \in V_2} f(v,s)$$

$$+ \sum_{v \in V_1} f'(s,v) + \sum_{v \in V_2} f'(s,v) - \sum_{v \in V_1} f'(v,s) - \sum_{v \in V_2} f'(v,s)$$

$$= \sum_{v \in V_1} f(s,v) - \sum_{v \in V_2} f(v,s) + \sum_{v \in V_1 \cup V_2} f'(s,v) - \sum_{v \in V_1 \cup V_2} f'(v,s) .$$

$$\begin{aligned} |f \uparrow f'| &= \sum_{v \in V} f(s,v) - \sum_{v \in V} f(v,s) + \sum_{v \in V} f'(s,v) - \sum_{v \in V} f'(v,s) \\ &= |f| + |f'| . \end{aligned}$$

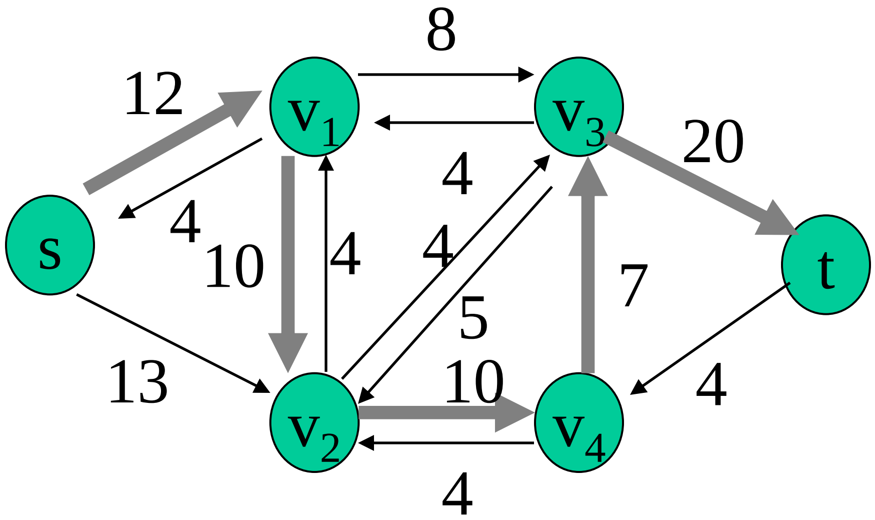
Augmenting paths:

- Given a flow network $G=(V,E)$ and a flow f, an augmenting path is a simple path from s to t in the residual network G_f.

- Residual capacity of p : the maximum amount of net flow that we can ship along the edges of an augmenting path p, i.e., $c_f(p)=\min\{c_f(u,v):(u,v) \text{ is on } p\}$.

The residual capacity is 1.

Example of an augment path (bold edges)



(b)

Cuts of flow networks

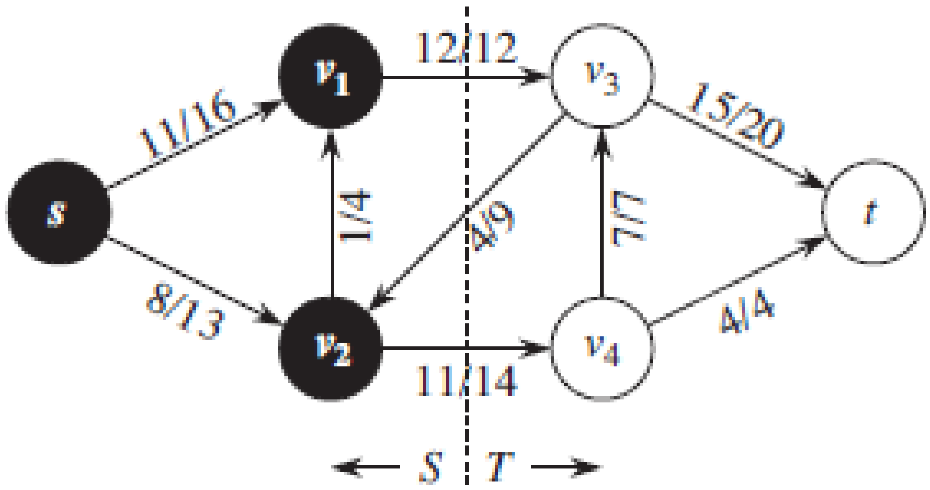
- A ***cut*** (S, T) of flow network G = (V,E) is a partition of V into S and V - S such that s ∈ S and t ∈ T . If f is a flow, then the ***net flow*** f (S,T) across the cut (S, T) is defined to be

$$f(S,T) = \sum_{u \in S} \sum_{v \in T} f(u,v) - \sum_{u \in S} \sum_{v \in T} f(v,u)$$

The *capacity* of the cut (S,T) is

$$c(S,T) = \sum_{u \in S} \sum_{v \in T} c(u,v) .$$

- A ***minimum cut*** of a network is a cut whose capacity is minimum over all cuts of the network.

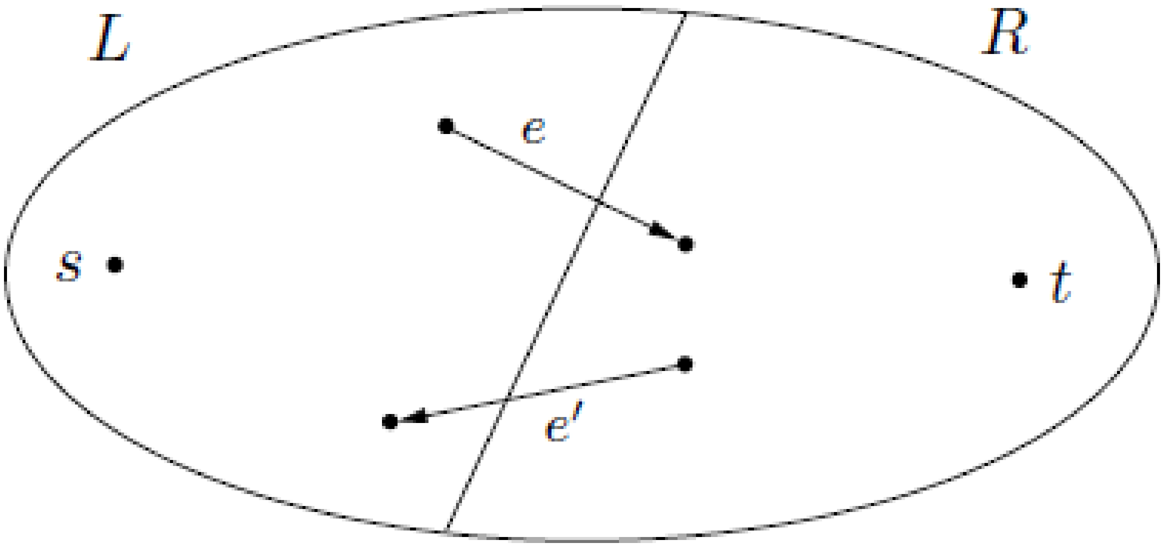


The net flow across this cut is
f (v1, v3) + f (v2, v4) - f (v3, v2) = 12 + 11 - 4 = 19

and the capacity of this cut is
c(v1, v3) + c(v2, v4) = 12 + 14 =26

The size of the maximum flow in a network equals the capacity of the smallest (s, t)-cut.

- Let's see why this is true. Suppose f is the final flow when the algorithm terminates. We know that node t is no longer reachable from s in the residual network G^f . Let L be the nodes that *are* reachable from s in G^f , and let R = V - L be the rest of the nodes. Then (L, R) is a cut in the graph G:



L
R
e
s
t
e'

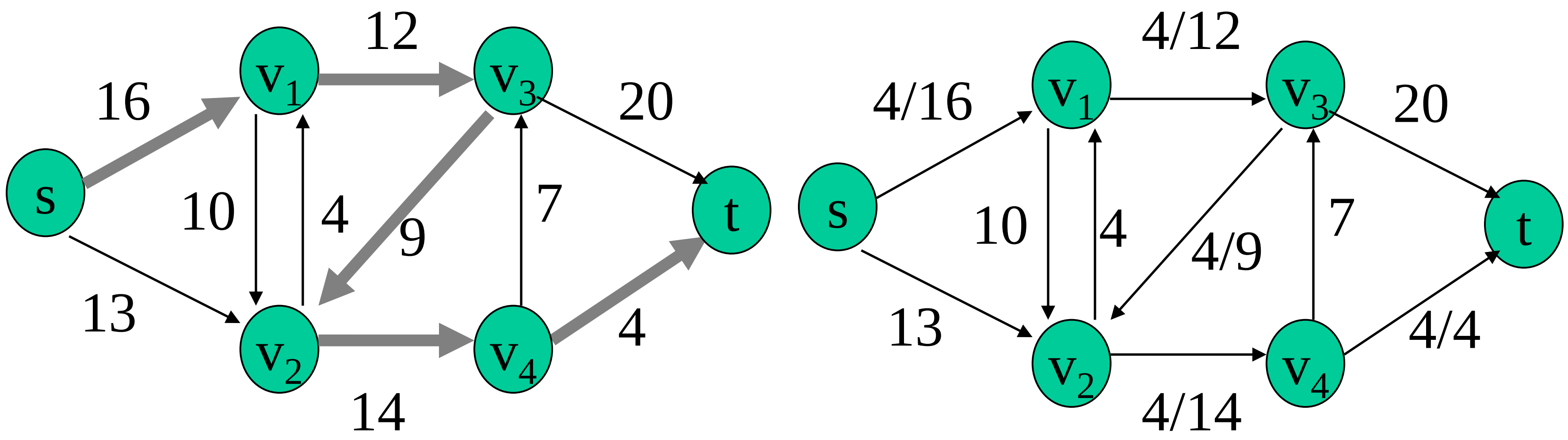
- We claim that size(f) = capacity(L,R): To see this, observe that by the way L is defined, any edge going from L to R must be at full capacity (in the current flow f), and any edge from R to L must have zero flow. (So, in the figure, $f_e = c_e$ and $f_{e0} = 0$.) Therefore the net flow across (L,R) is exactly the capacity of the cut.

The basic Ford-Fulkerson algorithm:

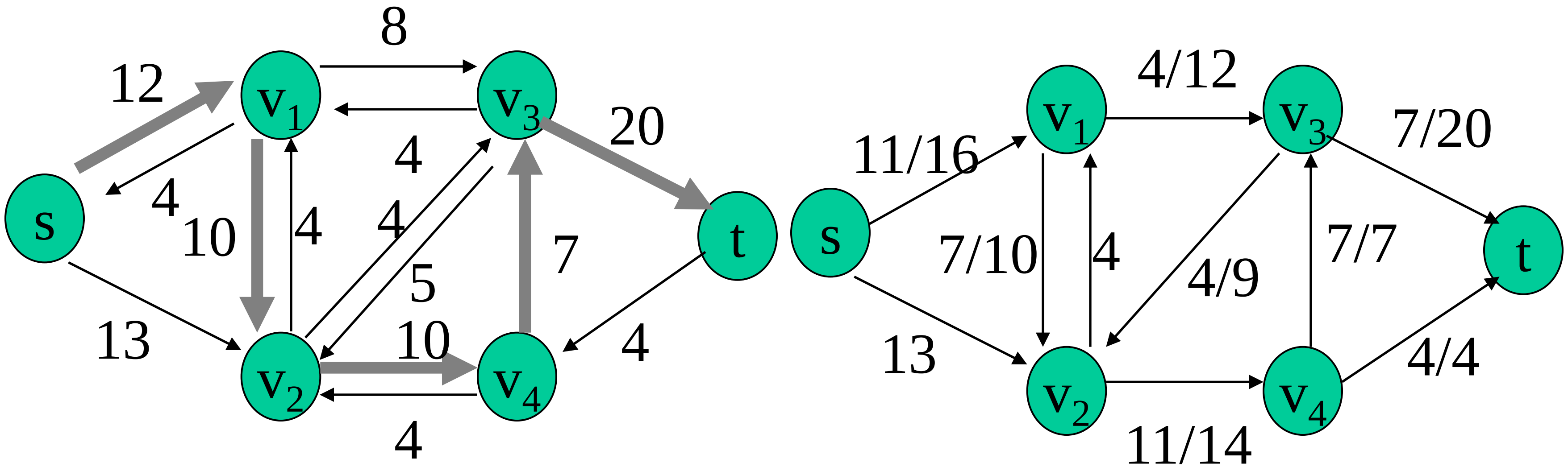
- FORD-FULKERSON(G,s,t)
- for each edge (u,v) $\in$E[G]
-         do f[u,v] $\leftarrow$ 0
-               f[v,u] $\leftarrow$ 0
- while there exists a path p from s to t in the residual network G_f
-         do $c_f(p) \leftarrow \min\{c_f(u,v)$: (u,v) is in p$\}$
-             for each edge (u,v) in p
-                 do f[u,v]$\leftarrow$ f[u,v]+$c_f(p)$
-

Example:

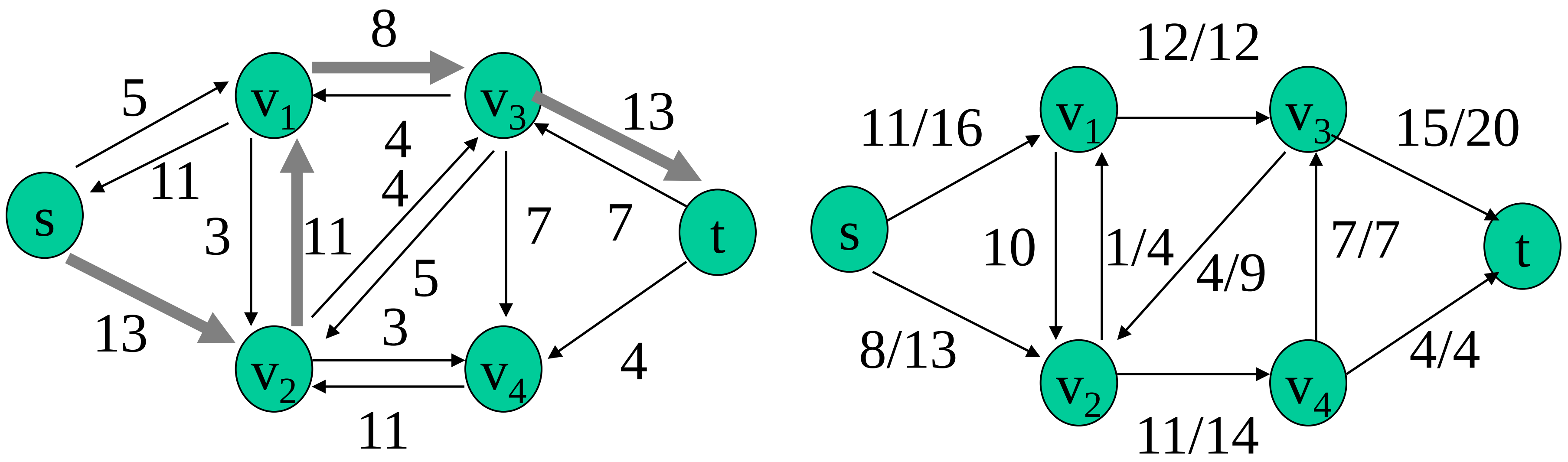
- The execution of the basic Ford-Fulkerson algorithm.
- (a)-(d) Successive iterations of the while loop: The left side of each part shows the residual network G_f from line 4 with a shaded augmenting path p.The right side of each part shows the new flow f that results from adding f_p to f.The residual network in (a) is the input network G.(e) The residual network at the last while loop test.It has no augmenting paths,and the flow f shown in (d) is therefore a maximum flow.



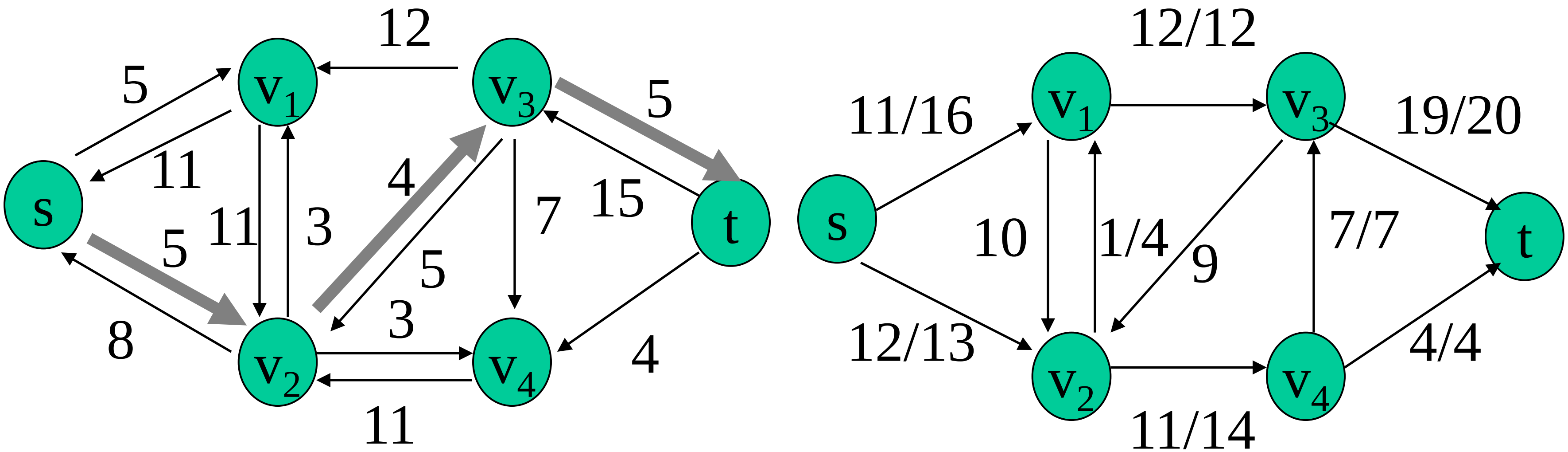
(a)


12
4
10
4
8
4
4
5
10
4
7
20
4
13
s
v1
v2
v3
v4
t
11/16
13
7/10
4
4/12
4/9
7/7
7/20
11/14
4/4
s
v1
v2
v3
v4
t

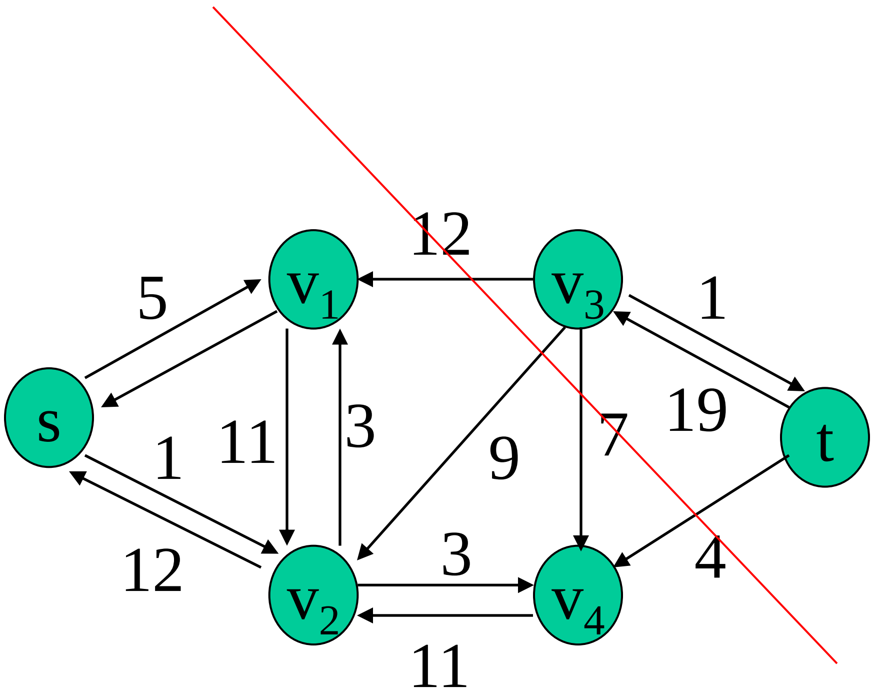
(b)


5
11
13
3
11
8
4
4
5
3
11
7
7
13
4
s
v_1
v_2
v_3
v_4
t
11/16
8/13
10
1/4
12/12
4/9
11/14
7/7
15/20
4/4
s
v_1
v_2
v_3
v_4
t

(c)


12
5
11
s
5
11
3
4
5
7
15
5
8
3
4
11
12/12
11/16
19/20
s
10
1/4
9
7/7
t
12/13
4/4
11/14

(d)



(e)

Time complexity:

- If each c(e) is an *integer*, then time complexity is O(|E|f*), where f* is the maximum flow.
- Reason: each time the flow is increased by at least one.

The Edmonds-Karp algorithm

- Find the augmenting path using breadth-first search.
- Breadth-first search gives the shortest path for graphs (Assuming the length of each edge is 1.)
- Time complexity of Edmonds-Karp algorithm is $O(VE^2)$.

If the Edmonds-Karp algorithm is run on a flow network $G = (V, E)$ with source s and sink t, then for all vertices $v \in V - \{s, t\}$, the shortest-path distance $\delta_f(s, v)$ in the residual network G_f increases monotonically with each flow augmentation.

Proof We will suppose that for some vertex $v \in V - \{s, t\}$, there is a flow augmentation that causes the shortest-path distance from s to v to decrease, and then we will derive a contradiction. Let f be the flow just before the first augmentation that decreases some shortest-path distance, and let f' be the flow just afterward. Let v be the vertex with the minimum $\delta_{f'}(s, v)$ whose distance was decreased by the augmentation, so that $\delta_{f'}(s, v) < \delta_f(s, v)$. Let $p = s \leadsto u \to v$ be a shortest path from s to v in $G_{f'}$, so that $(u, v) \in E_{f'}$ and

$$\delta_{f'}(s, u) = \delta_{f'}(s, v) - 1 .$$

Because of how we chose v, we know that the distance of vertex u from the source s did not decrease, i.e.,

$$\delta_{f'}(s, u) \geq \delta_f(s, u) .$$

We claim that $(u,v) \notin E_f$. Why? If we had $(u,v) \in E_f$, then we would also have

$$\begin{aligned}
\delta_f(s,v) &\le \delta_f(s,u)+1 \\
&\le \delta_{f'}(s,u)+1 \\
&= \delta_{f'}(s,v)
\end{aligned}$$

How can we have $(u,v) \notin E_f$ and $(u,v) \in E_{f'}$? The augmentation must have increased the flow from v to u. The Edmonds-Karp algorithm always augments flow along shortest paths, and therefore the shortest path from s to u in G_f has (v,u) as its last edge. Therefore,

$$\begin{aligned}
\delta_f(s,v) &= \delta_f(s,u)-1 \\
&\le \delta_{f'}(s,u)-1 \\
&= \delta_{f'}(s,v)-2
\end{aligned}$$

which contradicts our assumption that $\delta_{f'}(s,v) < \delta_f(s,v)$.

If the Edmonds-Karp algorithm is run on a flow network $G = (V, E)$ with source s and sink t, then the total number of flow augmentations performed by the algorithm is $O(VE)$.

Let u and v be vertices in V that are connected by an edge in E. Since augmenting paths are shortest paths, when (u, v) is critical for the first time, we have

$$\delta_f(s, v) = \delta_f(s, u) + 1 .$$

Once the flow is augmented, the edge (u, v) disappears from the residual network. It cannot reappear later on another augmenting path until after the flow from u to v is decreased, which occurs only if (v, u) appears on an augmenting path. If f' is the flow in G when this event occurs, then we have

$$\delta_{f'}(s, u) = \delta_{f'}(s, v) + 1 .$$

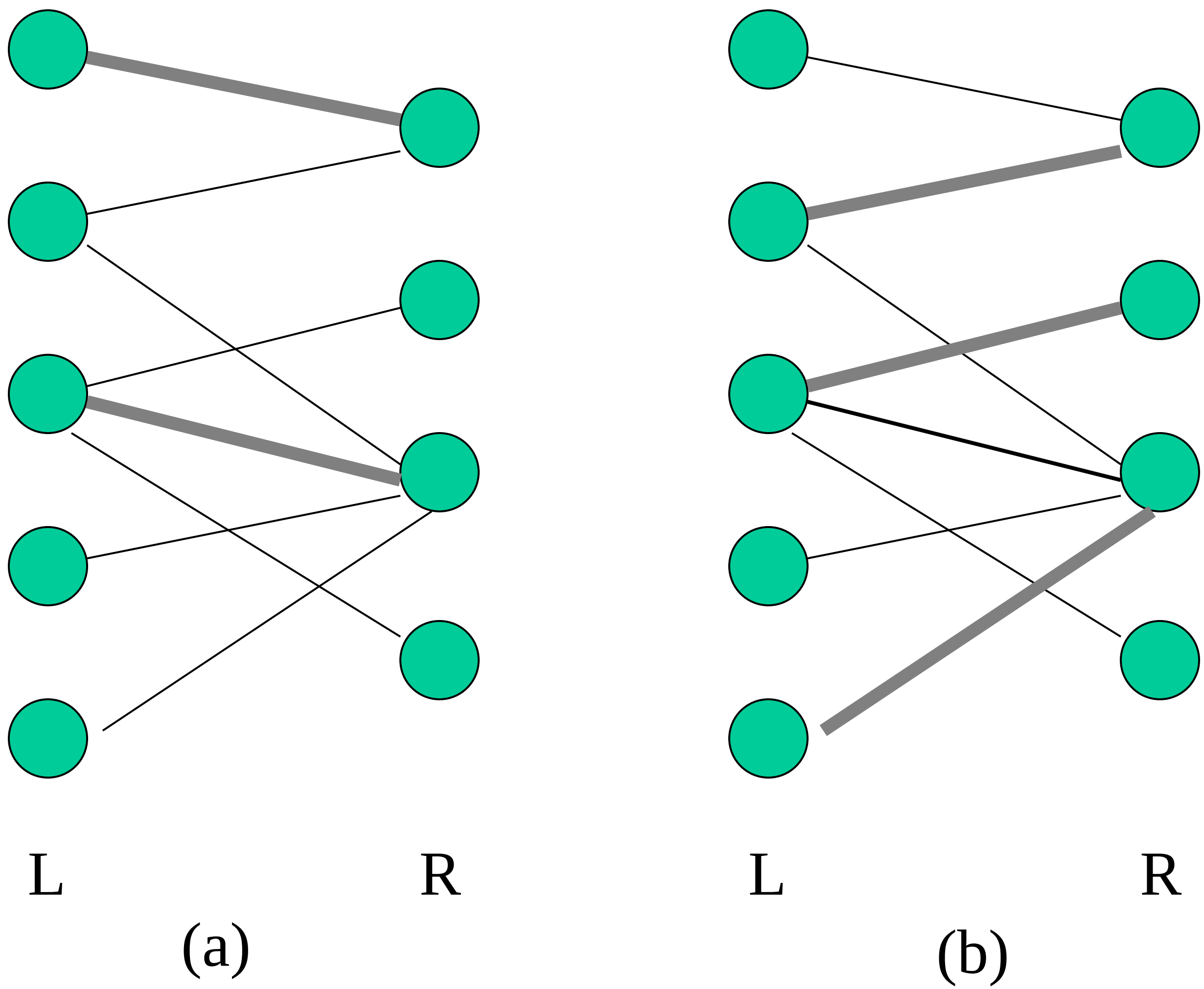
Since $\delta_f(s, v) \le \delta_{f'}(s, v)$

$$\begin{aligned} \delta_{f'}(s, u) &= \delta_{f'}(s, v) + 1 \\ &\ge \delta_f(s, v) + 1 \\ &= \delta_f(s, u) + 2 . \end{aligned}$$

Maximum bipartite matching:

- Bipartite graph: a graph (V, E), where $V=L\cup R$, $L\cap R=$empty, and for every $(u, v)\in E$, $u \in L$ and $v \in R$.
- Given an undirected graph $G=(V,E)$, a matching is a subset of edges $M\subseteq E$ such that for all vertices $v\in V$,at most one edge of M is incident on v.We say that a vertex $v \in V$ is matched by matching M if some edge in M is incident on v;otherwise, v is unmatched. A maximum matching is a matching of maximum cardinality,that is, a matching M such that for any matching M', we have .

$$|M| \geq |M'|$$

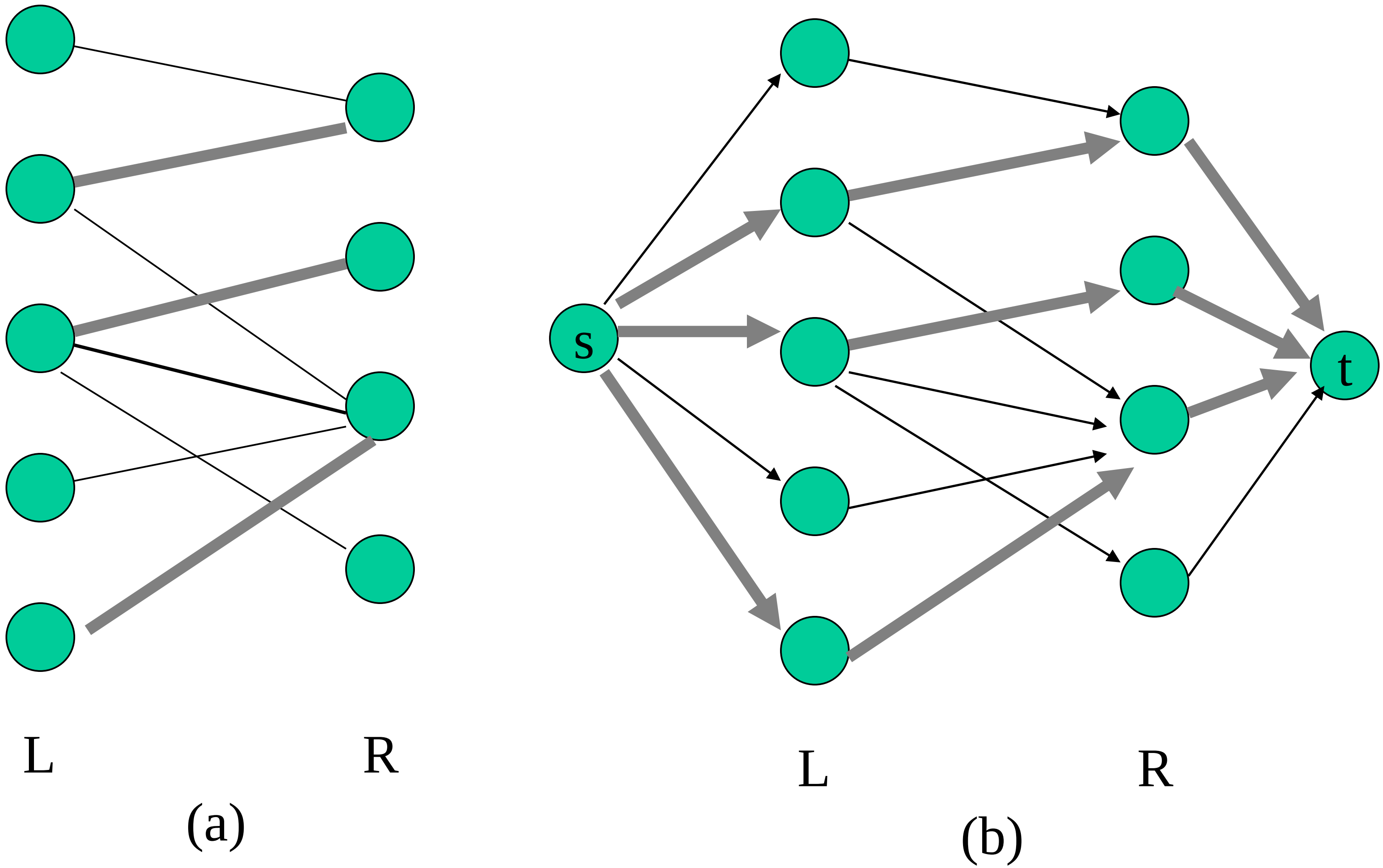


A bipartite graph G=(V,E) with vertex partition V=L∪R.(a)A matching with cardinality 2.(b) A maximum matching with cardinality 3.

Finding a maximum bipartite matching:

- We define the corresponding flow network $G'=(V',E')$ for the bipartite graph G as follows. Let the source s and sink t be new vertices not in V, and let $V'=V\cup\{s,t\}$.If the vertex partition of G is $V=L \cup R$, the directed edges of G' are given by $E'=\{(s,u):u\in L\} \cup\{(u,v):u \in L,v \in R,\text{and } (u,v) \in E\} \cup\{(v,t):v \in R\}$.Finally, we assign unit capacity to each edge in E'.

- We will show that a matching in G corresponds directly to a flow in G's corresponding flow network G'. We say that a flow f on a flow network $G=(V,E)$ is integer-valued if $f(u,v)$ is an integer for all $(u,v) \in V*V$.

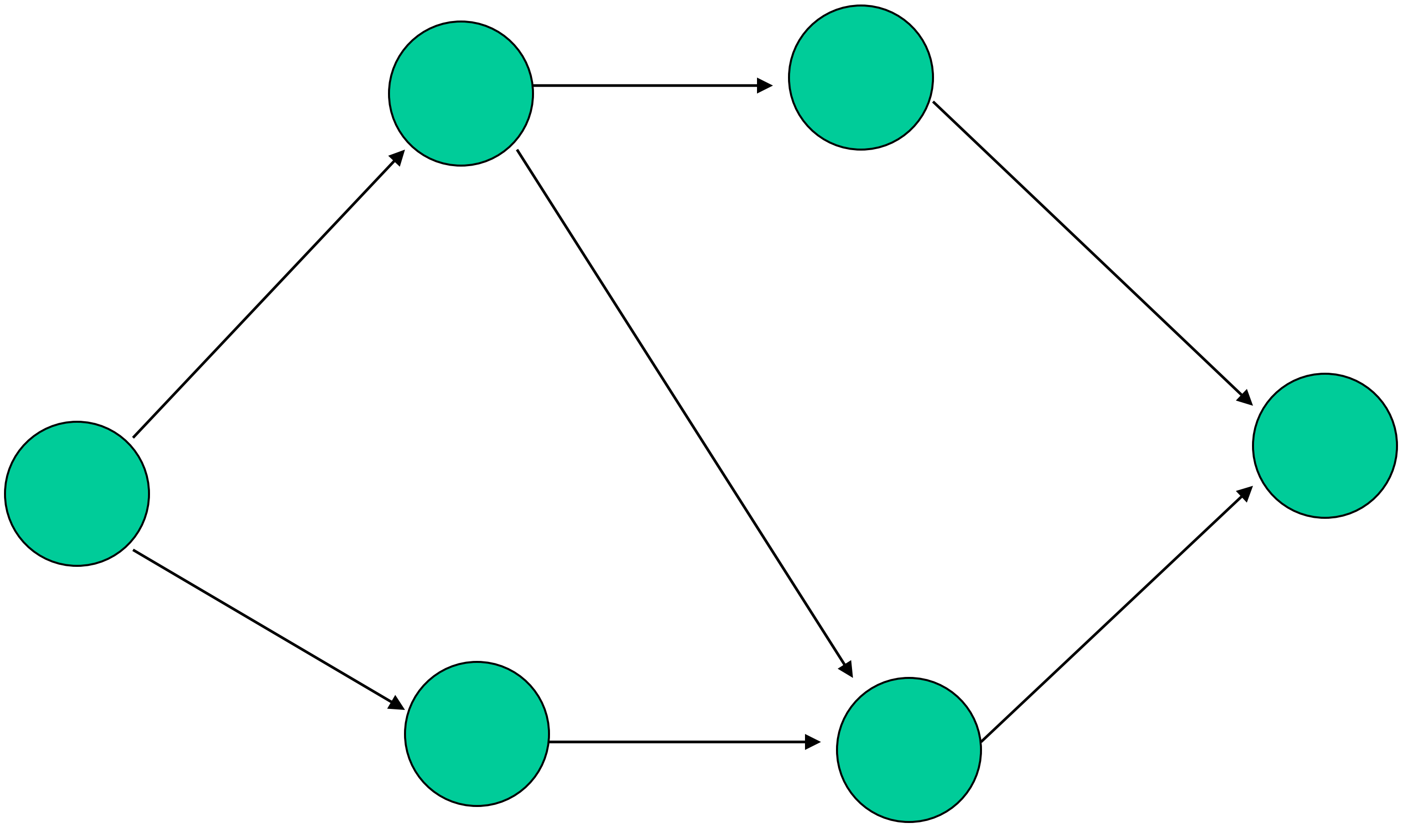


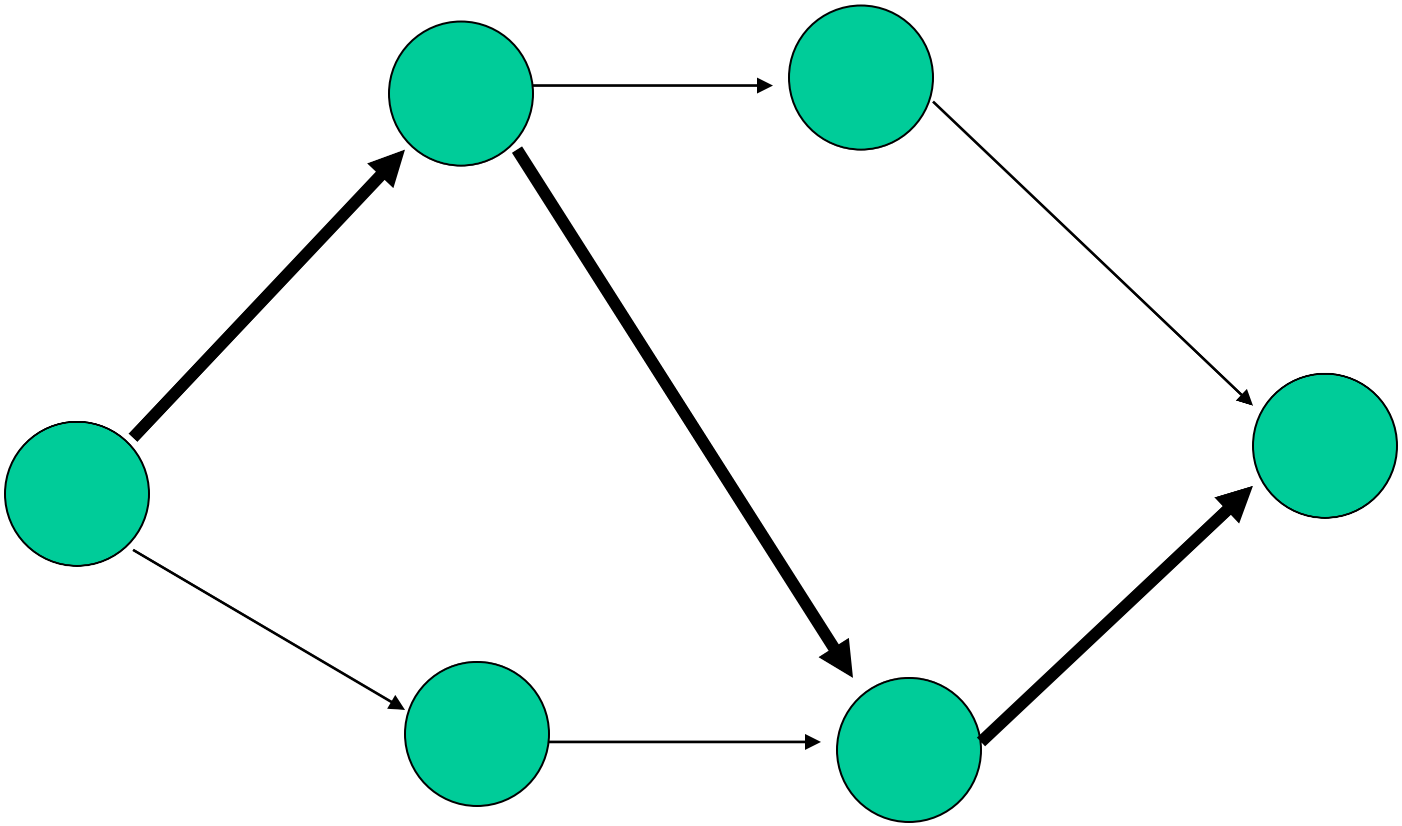
(a)The bipartite graph G=(V,E) with vertex partition V=L∪R. A maximum matching is shown by shaded edges.(b) The corresponding flow network.Each edge has unit capacity.Shaded edges have a flow of 1,and all other edges carry no flow.

Continue:

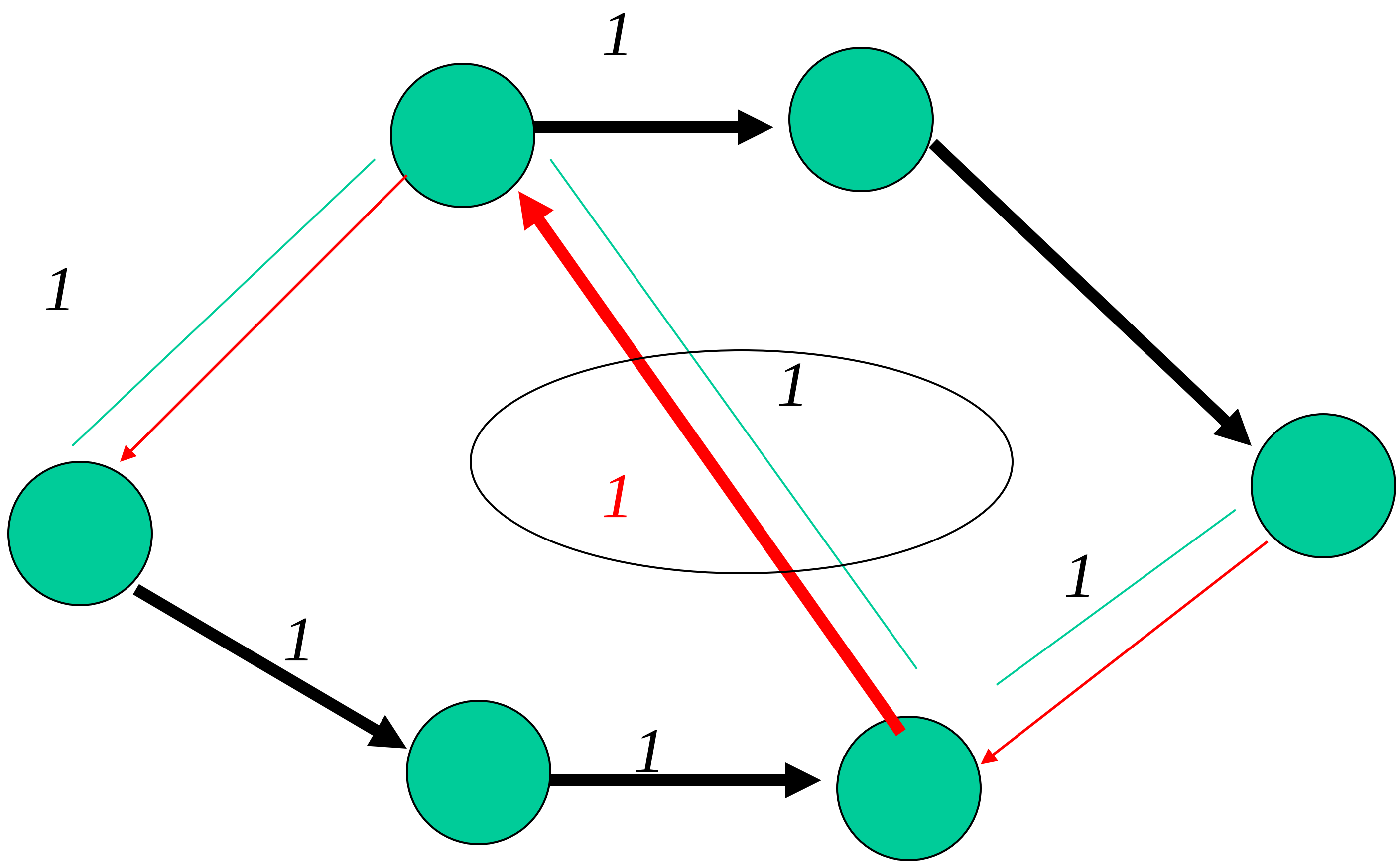
- Lemma .
- Let G=(V,E) be a bipartite graph with vertex partition V=L∪R, and let G'=(V',E') be its corresponding flow network. If M is a matching in G, then there is an integer-valued flow f in G' with value $|f| = |M|$. Conversely, if f is an integer-valued flow in G', then there is a matching M in G with cardinality $|M| = |f|$.
- Reason: The edges incident to s and t ensures this.
  - Each node in the L has in-degree 1
  - Each node in the R has out-degree 1.
  - So each node in the bipartite graph can be involved once in the flow.

Example:





Aug. path:



Residual network. Red edges are new edges in the residual network. The new aug. path is bold. Green edges are old aug. path. old flow=1.

Maximum Flow
(Push-relabel algorithms)

- Lemma：

  Let $G=(V,E)$ be a flow network, f be a preflow in G, and let *h* be a

  Height function on V.

  For any two vertices $u, v \in V$, if $h(u)>h(v)+1$,

  Then (u,v) is not an edge in the residual graph.

– The basic operation PUSH(u,v) can be applied if u is an overflowing vertex, $c_f(u,v)>0$, and $u.h=v.h+1$.
– u.e： the execess flow stored at u.
– u.h： the height of u.
– $\Delta_f(u,v)$： the amount of flow can be pushed from u to v.

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.*

$$\Delta_f(u,v) = \min(u.e, c_f(u,v))$$

$$\text{if } (u,v) \in E$$

$$(u,v).f = (u,v).f + \Delta_f(u,v)$$

$$\text{else } (v,u).f = (v,u).f - \Delta_f(u,v)$$

$$u.e = u.e - \Delta_f(u,v)$$

$$v.e = v.e + \Delta_f(u,v)$$ *}*

– Saturating push：

if (u,v) becomes saturated($c_f(u,v)=0$ afterward); otherwise, it is a nonsaturating push.

- lemma：

  After a non-saturating push from u to v, the vertex u is no longer overflowing.

  Proof:

  Since the push was non-saturating, the amount of flow $\Delta_f(u, v)$ actually pushed is u.e.

- The basic operation Relabel(u) applies if u is overflowing and if $c_f(u,v)>0$ implies $u.h \le v.h$ for all v.

- ***Relabel(u)***

  { * *Applies when : u is overflowing and for all* $v \in V, (u,v) \in E_f$ *implies* $u.h \le v.h.$

  * *Action : Increase the height of u.*

  $$u.h = 1 + \min\{v.h : (u,v) \in E_f\}$$

  }

- When u is relabeled, E_f must contain at least one edge that leaves u.

- $u.e = \sum_{v \in V} f(v,u) - \sum_{v \in V} f(u,v) > 0 \Rightarrow$ there must be at least one vertex v s.t. (v,u).f>0.

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

$$\Rightarrow (u,v) \in E_f$$

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 $v \in s.Adj$

$(s,v).f = c(s,v)$

$v.e = c(s,v)$

$s.e = s.e - c(s,v)$

}

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

and

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

Generic-Push-Relabel(G)

{ *Initialize-Preflow(G,s) ;*

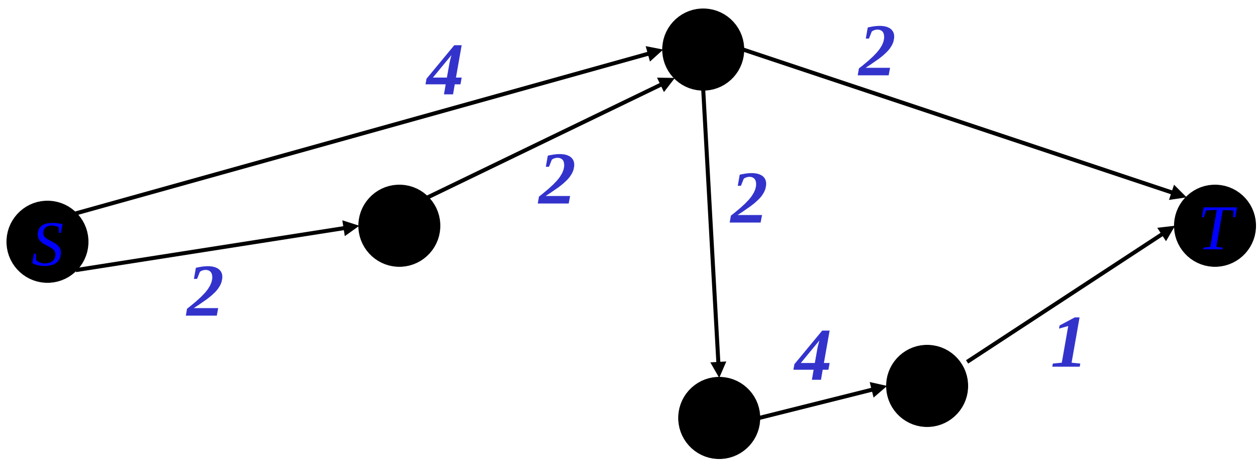
While there exists an applicable push or relabel operation.

do select an applicable push or relabel operation

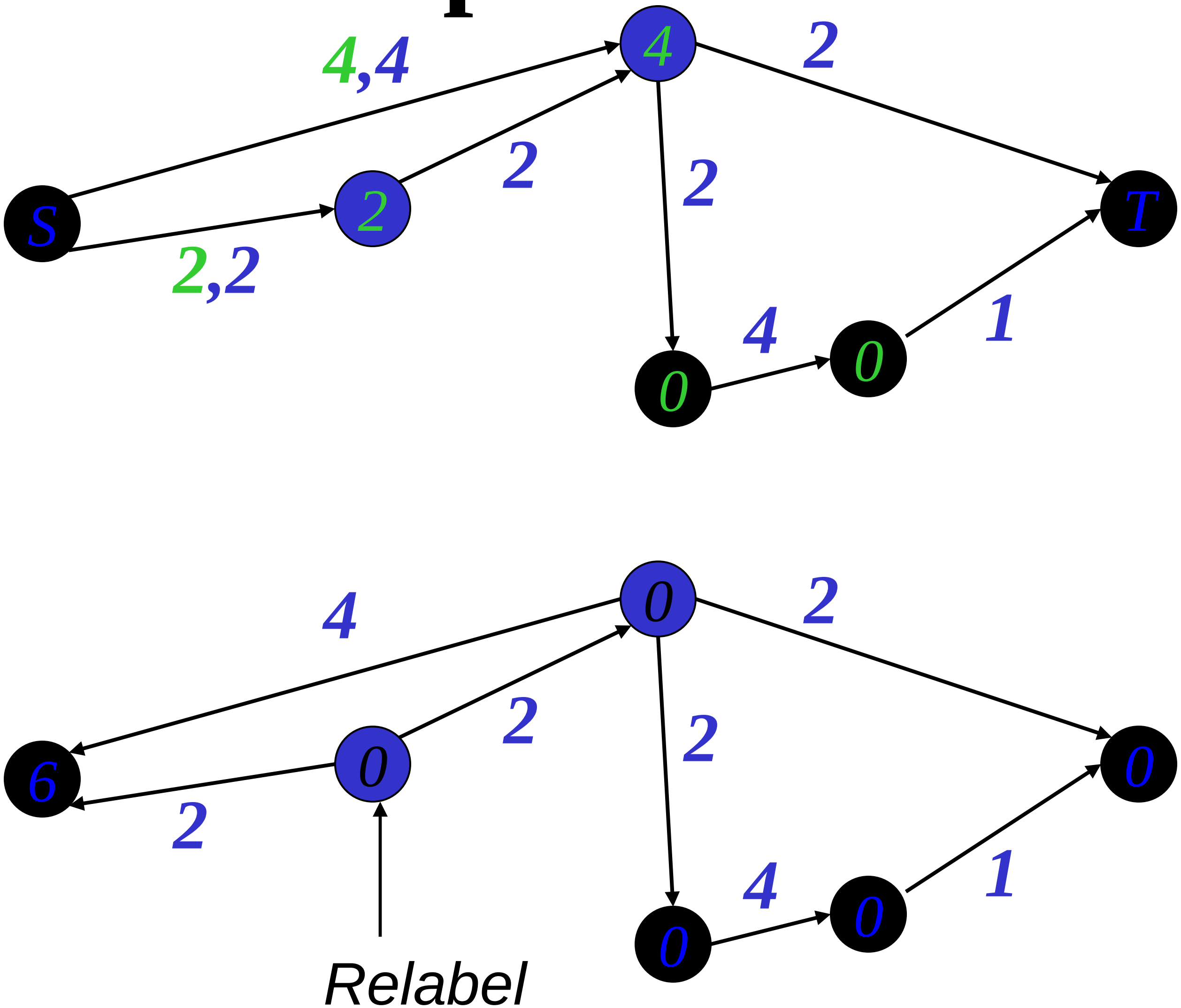
and perform it;

}

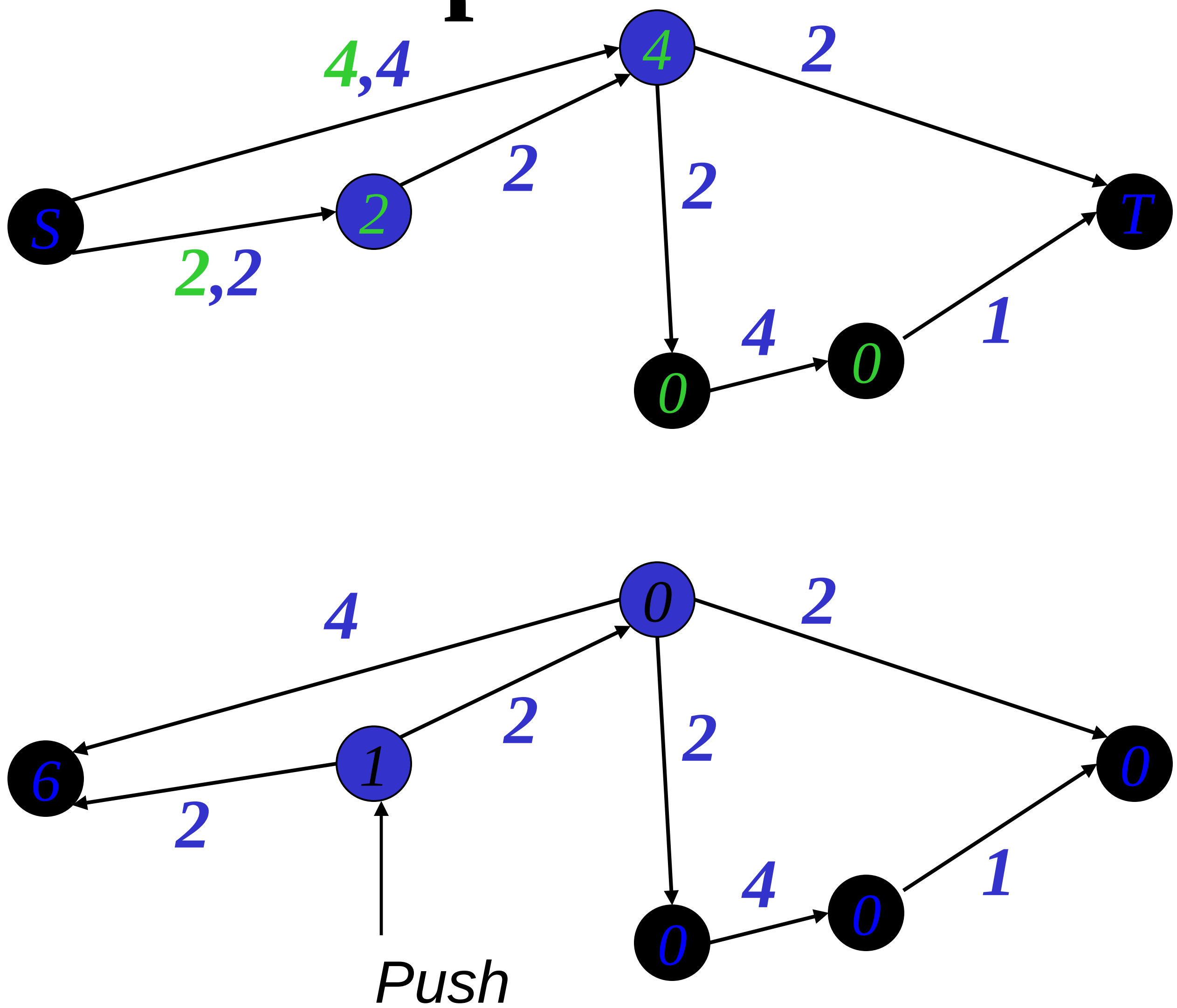
Example - *Saturate all source edges*



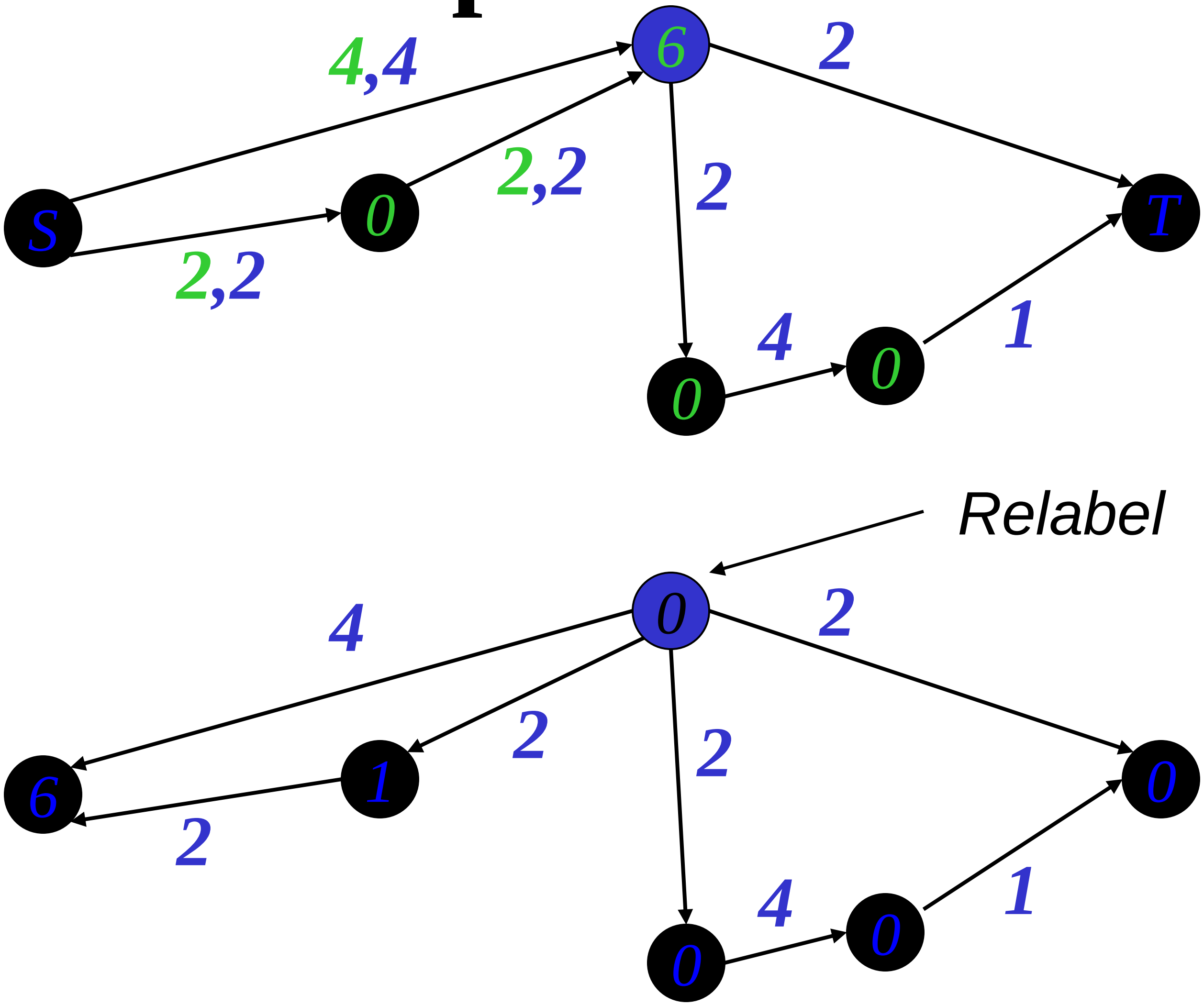
Example – contd.



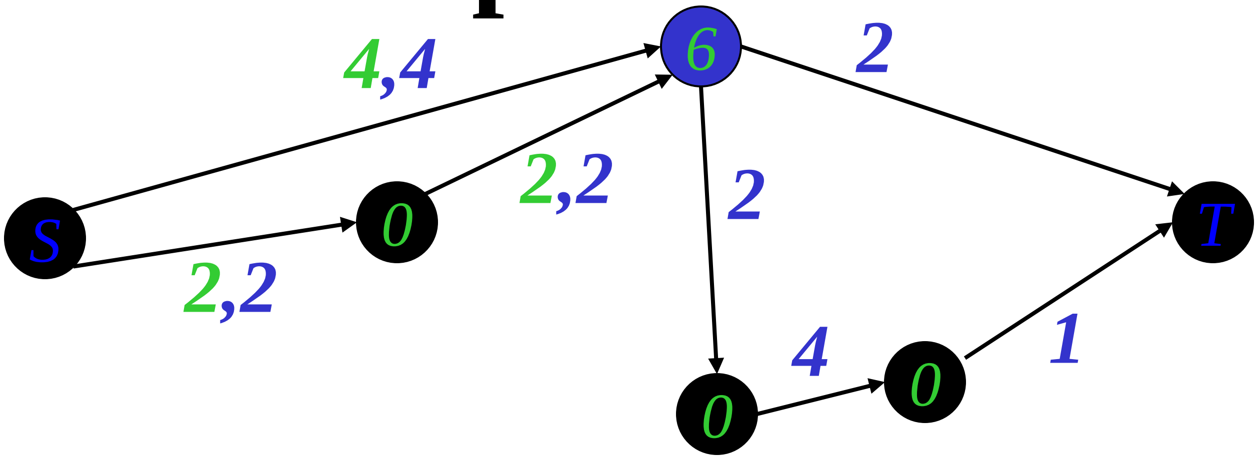
Example – contd.

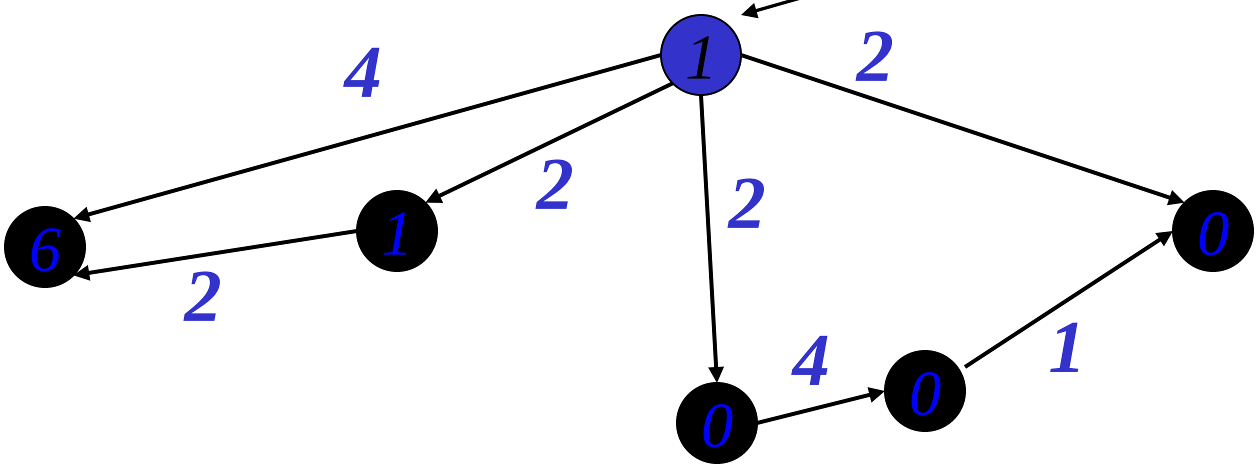


Example – contd.

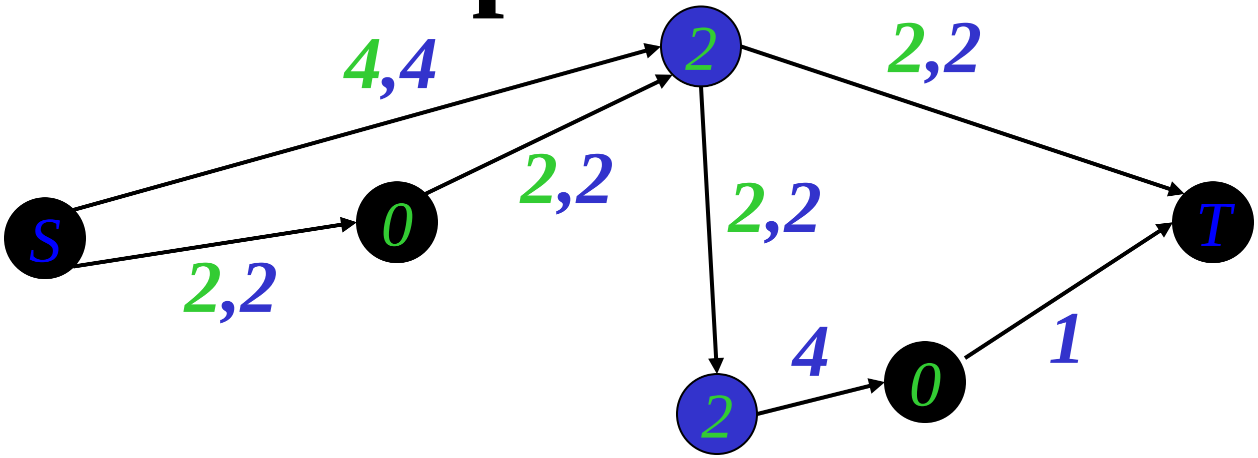


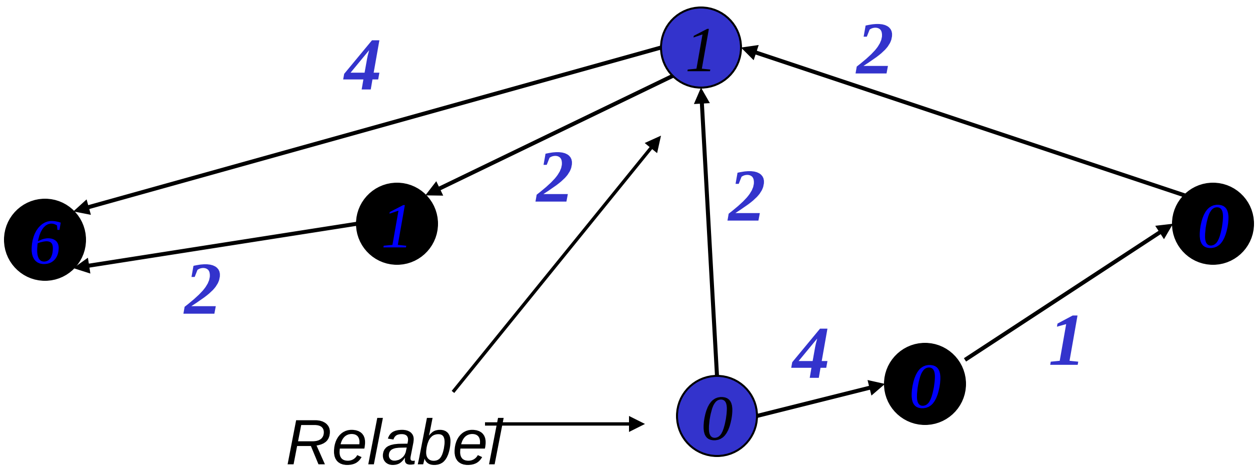
Example – contd.



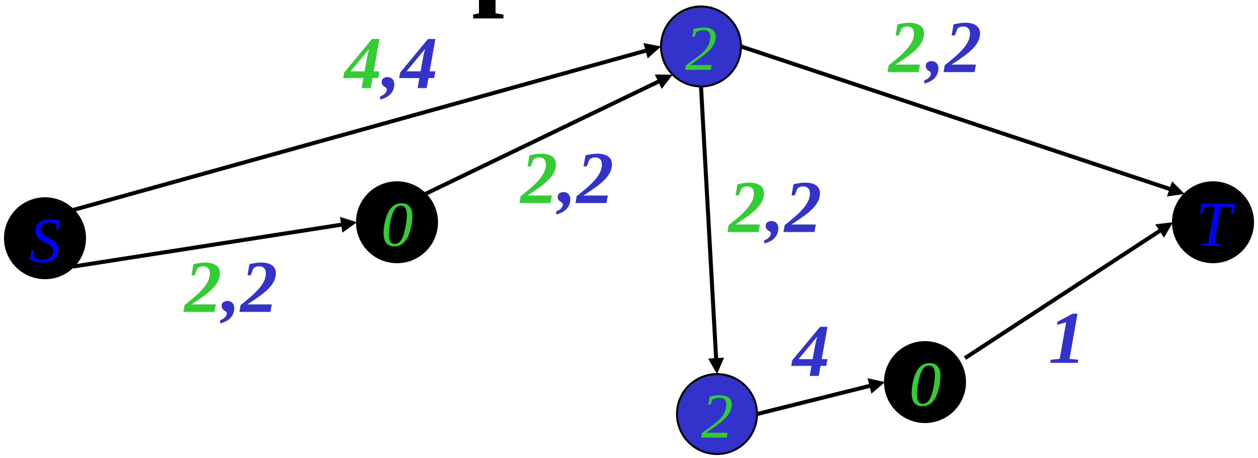


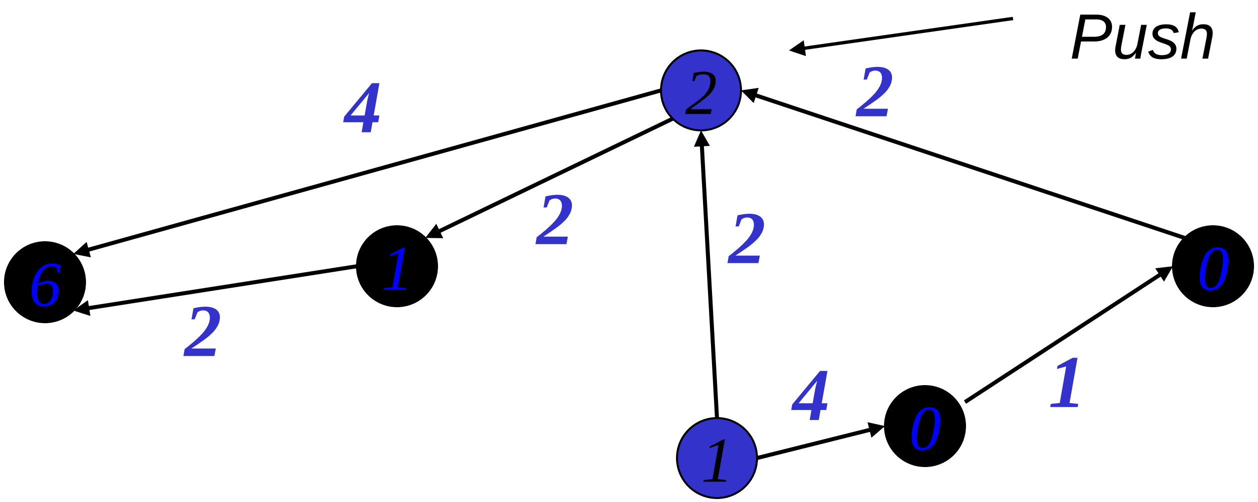
Example – contd.



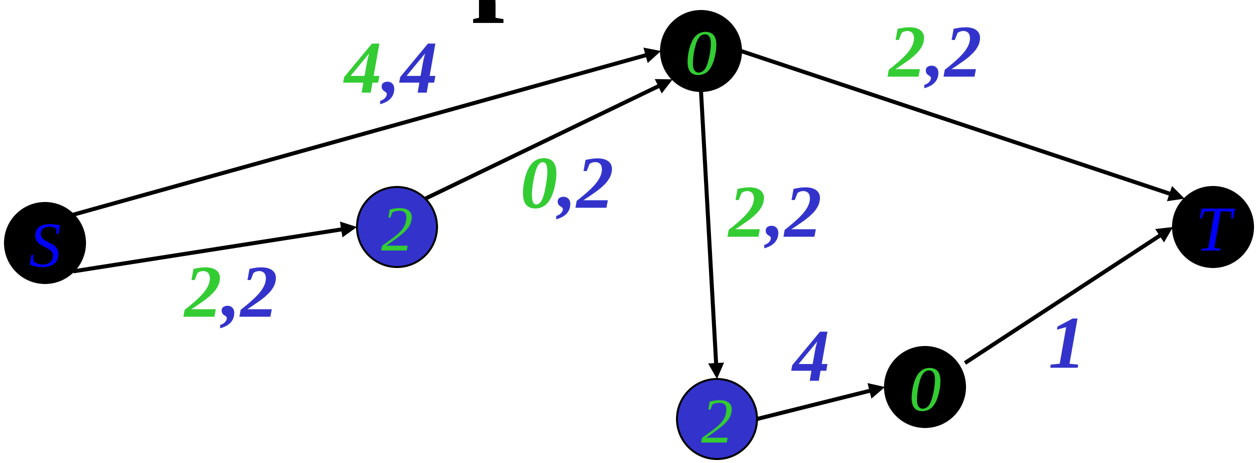


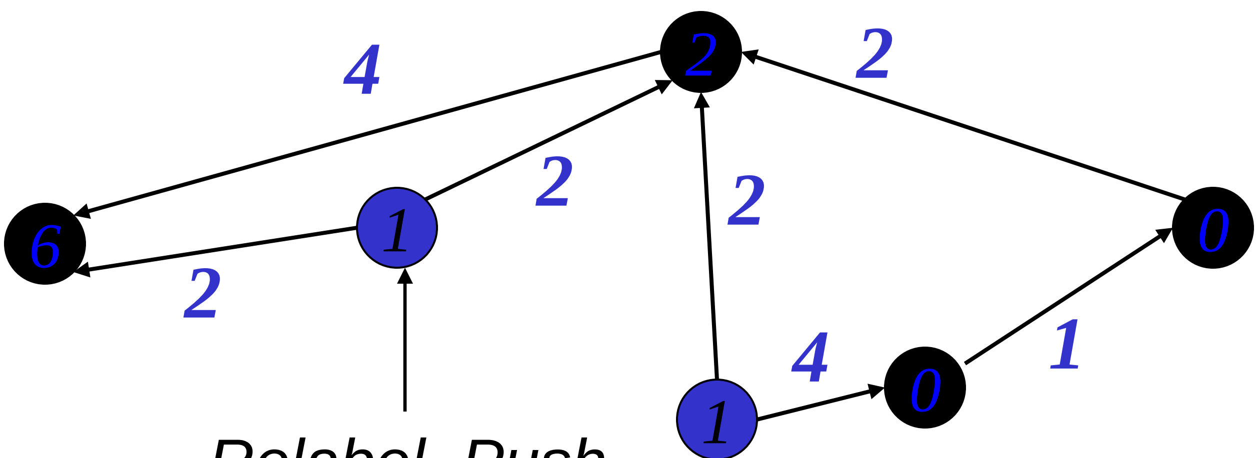
Example – contd.





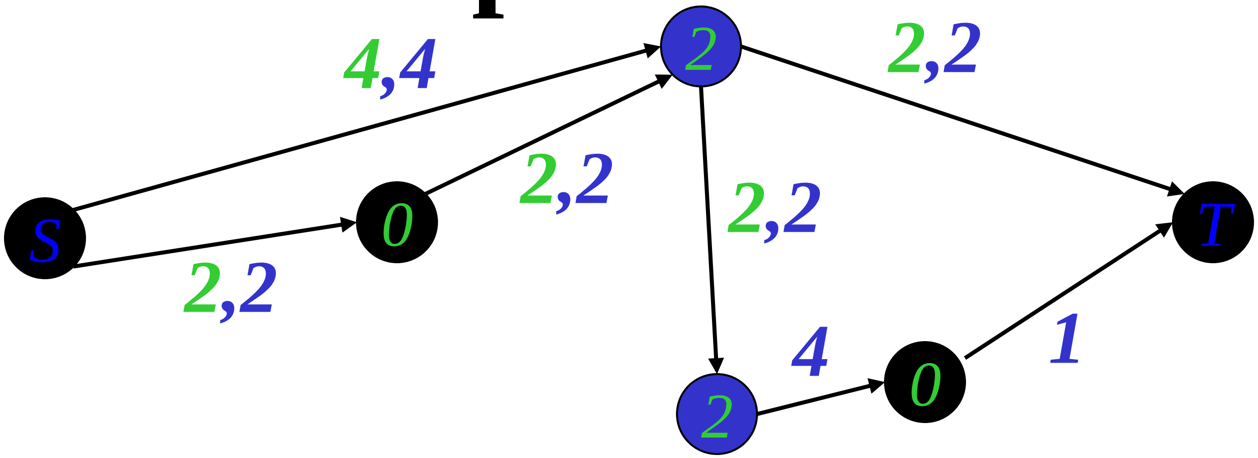
Example – contd.

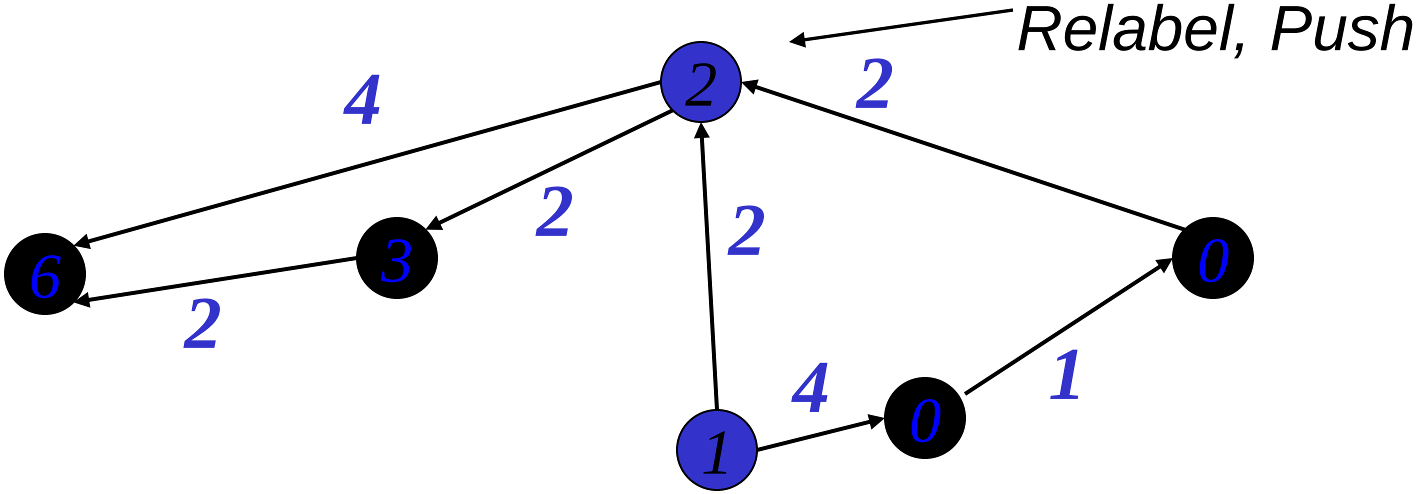




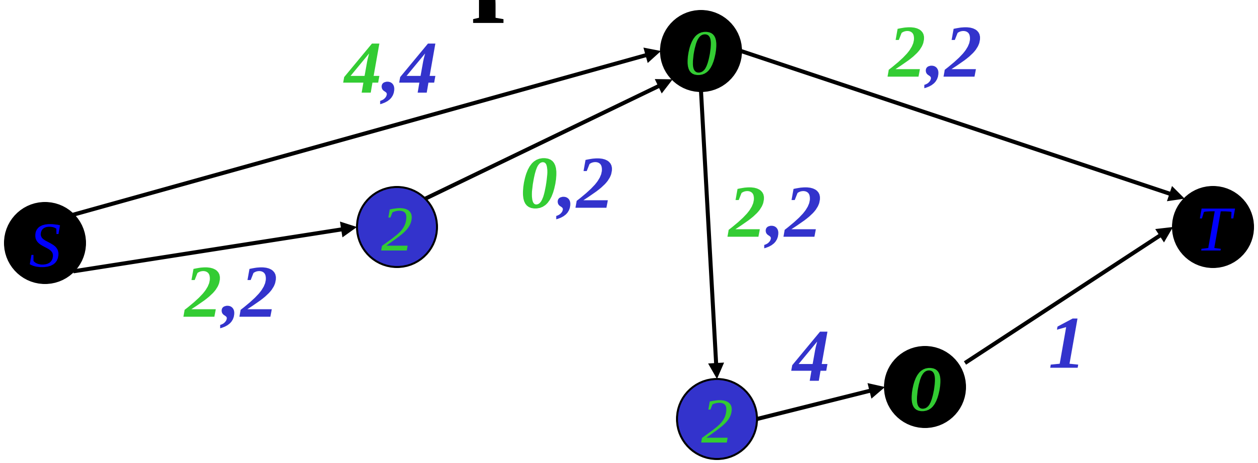
Relabel, Push

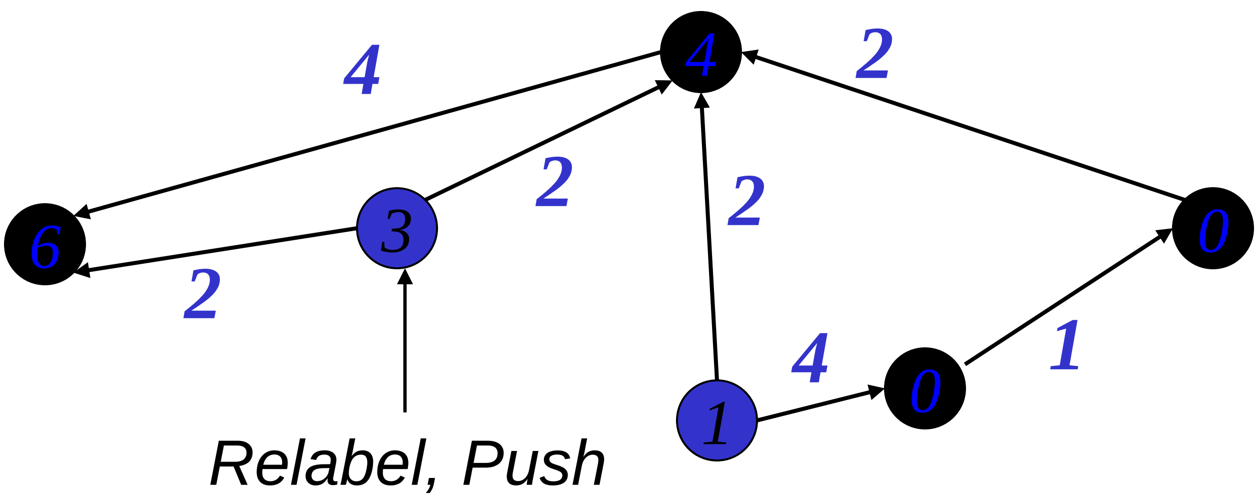
Example – contd.



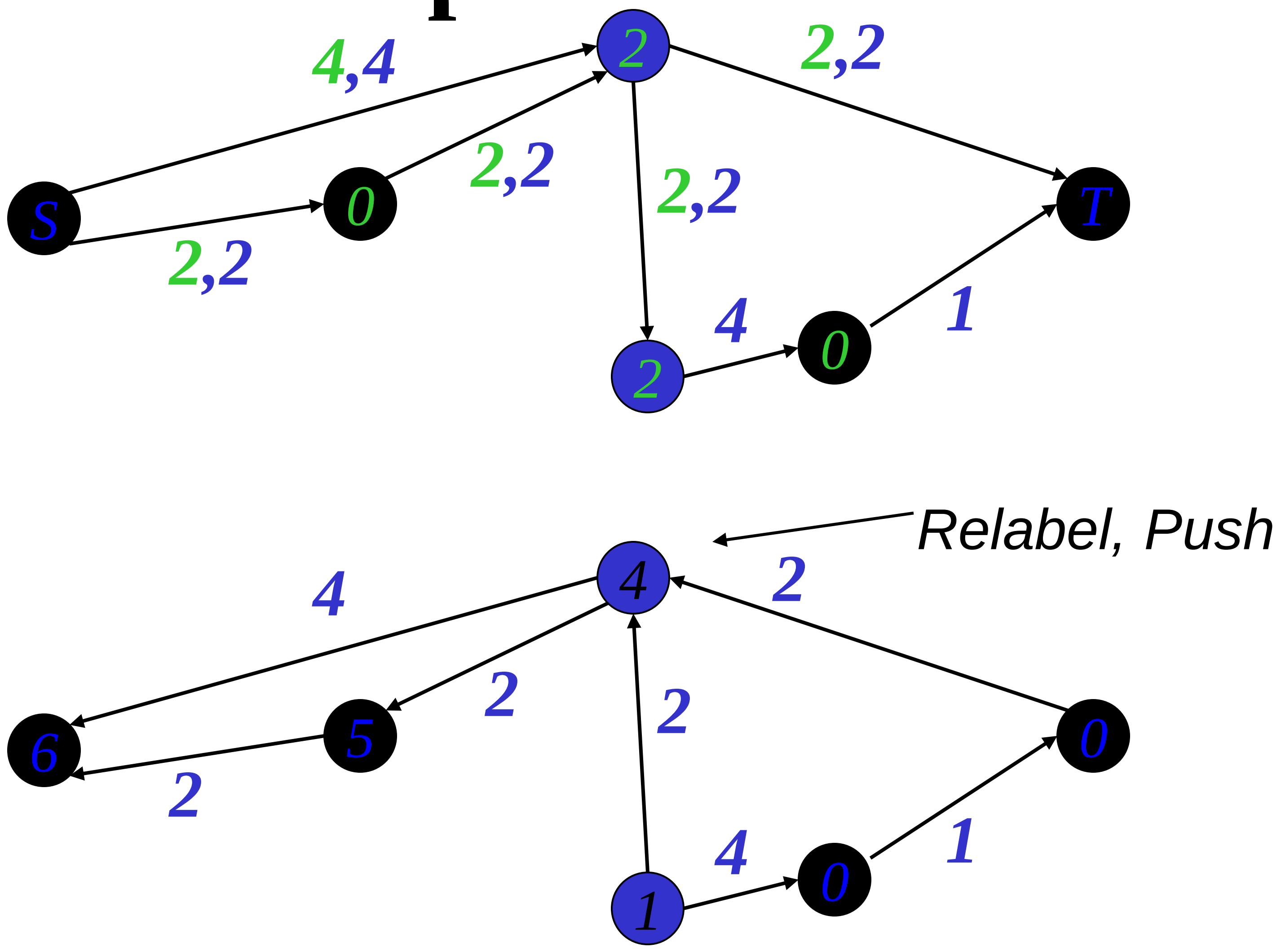


Example – contd.

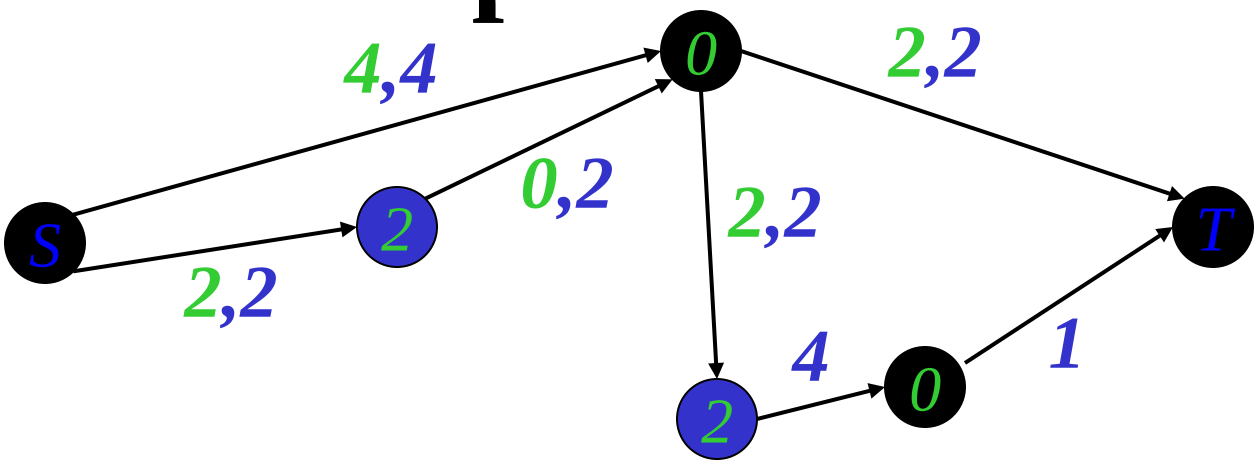


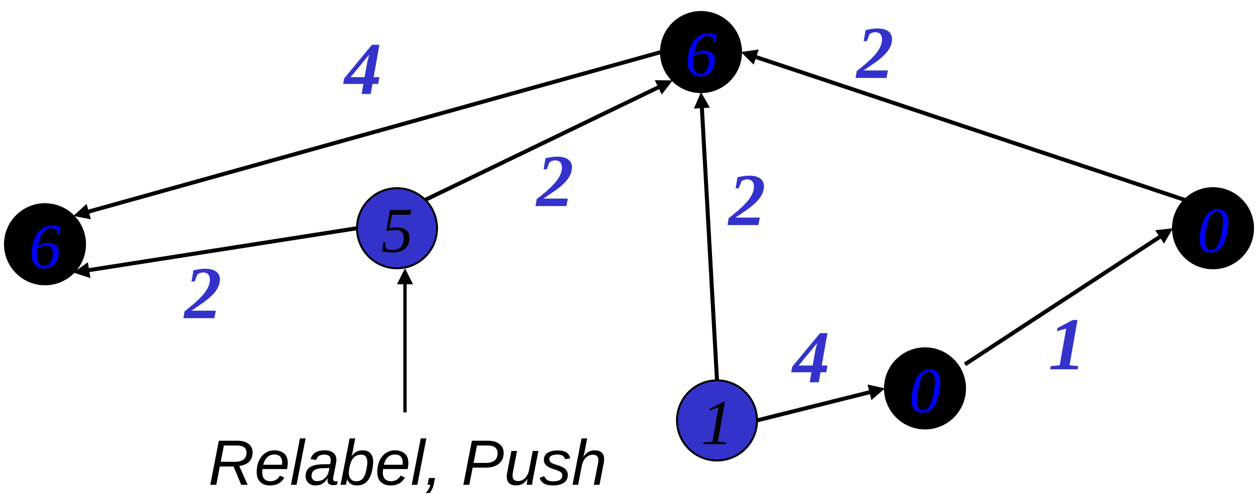


Example – contd.

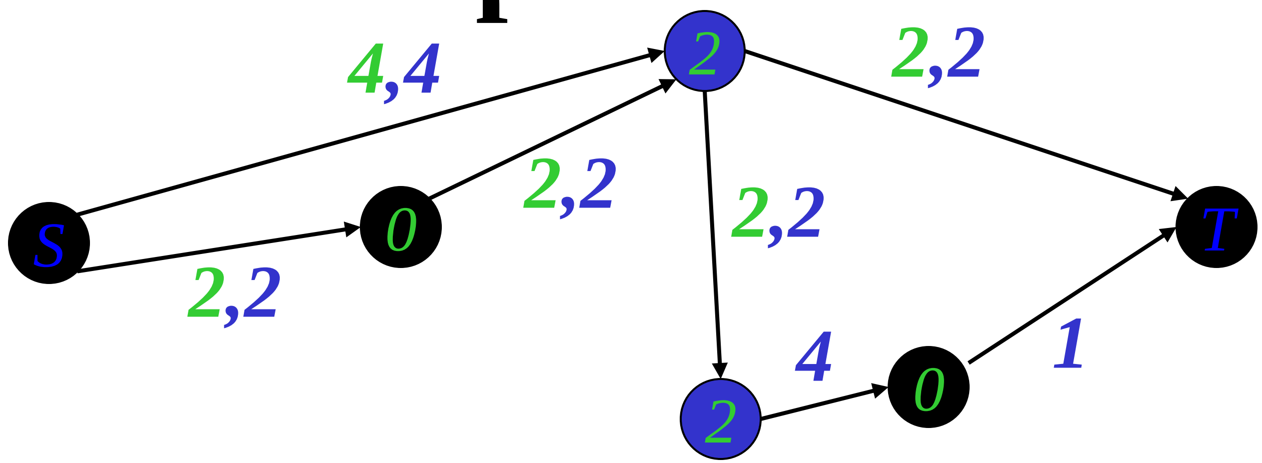


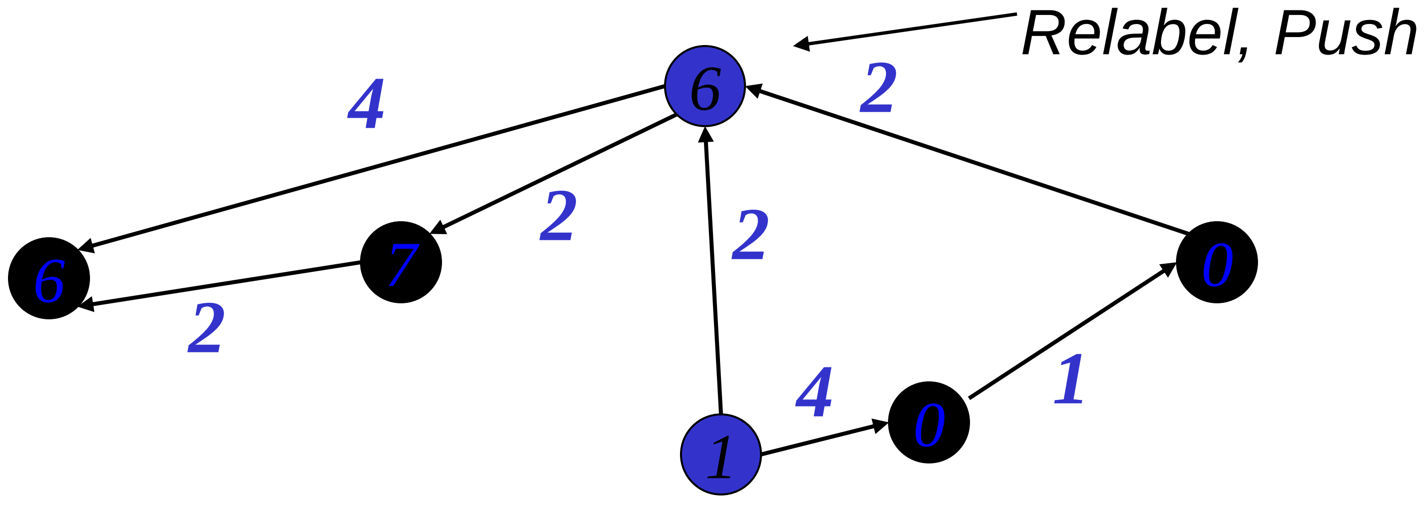
Example – contd.



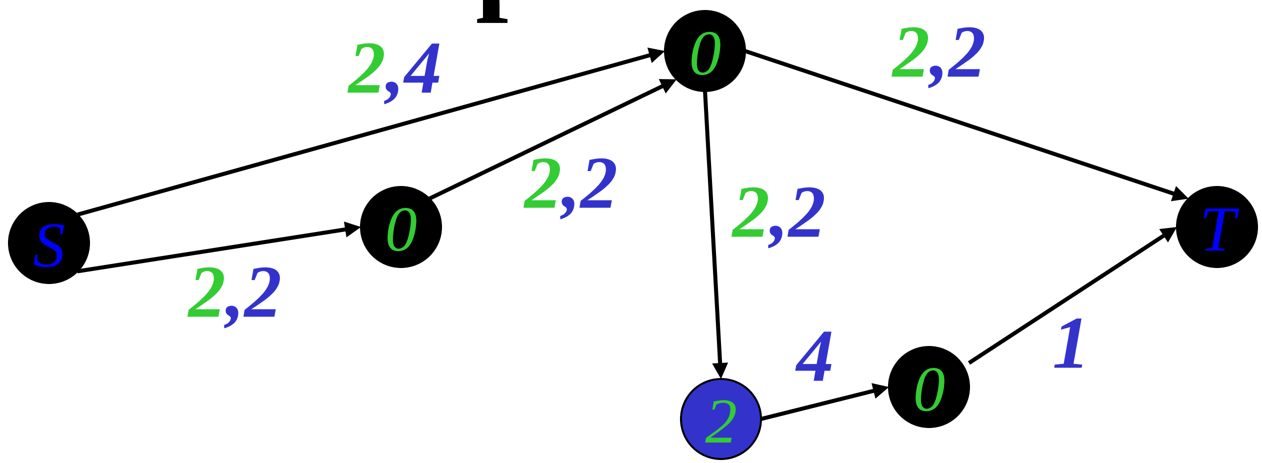


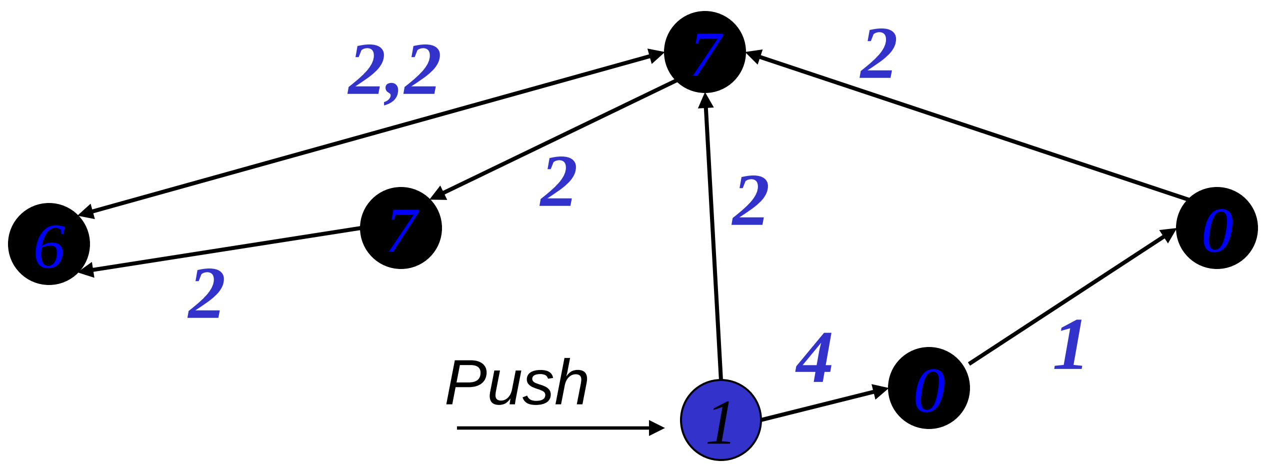
Example – contd.



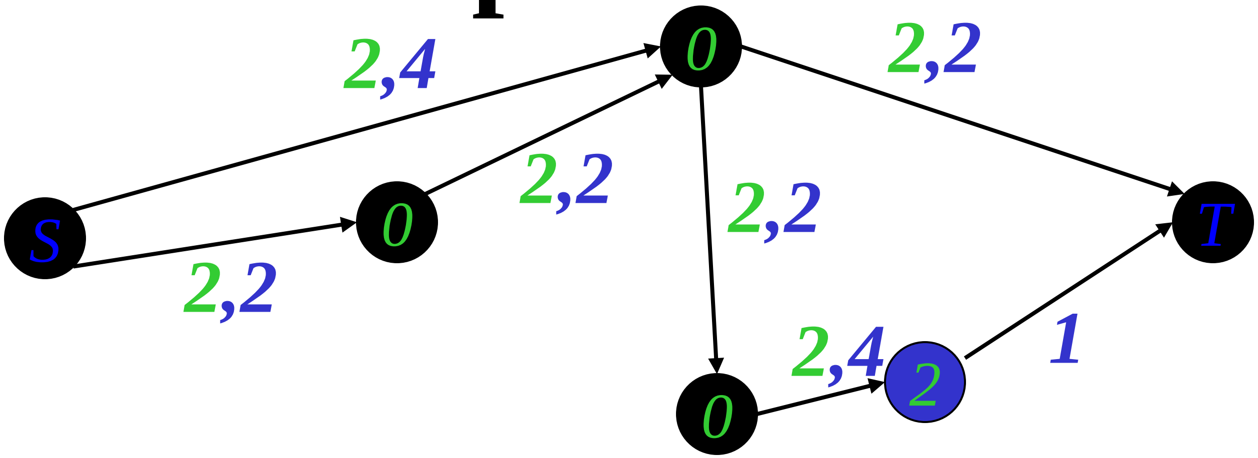


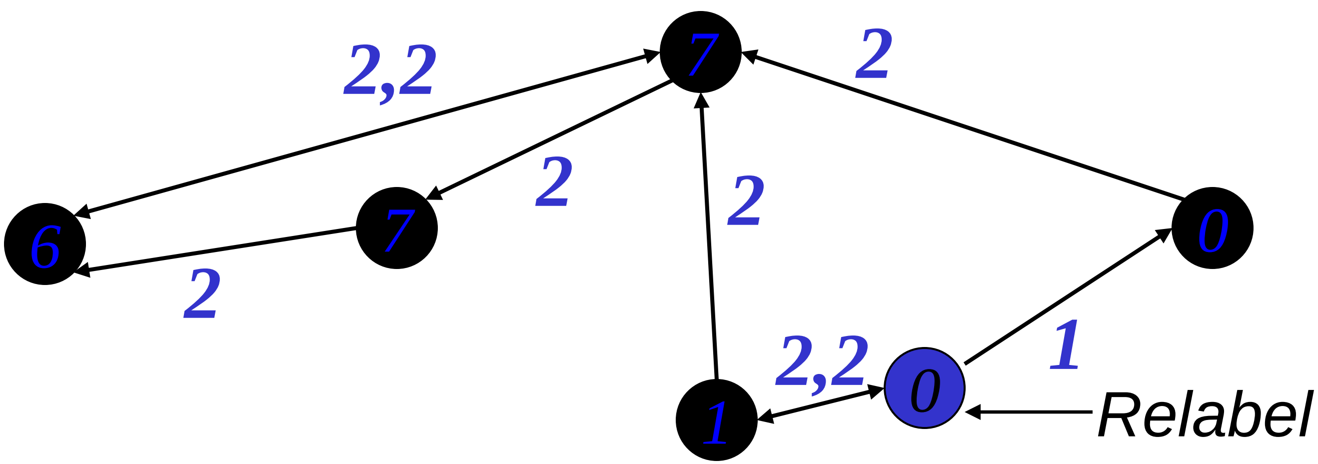
Example – contd.



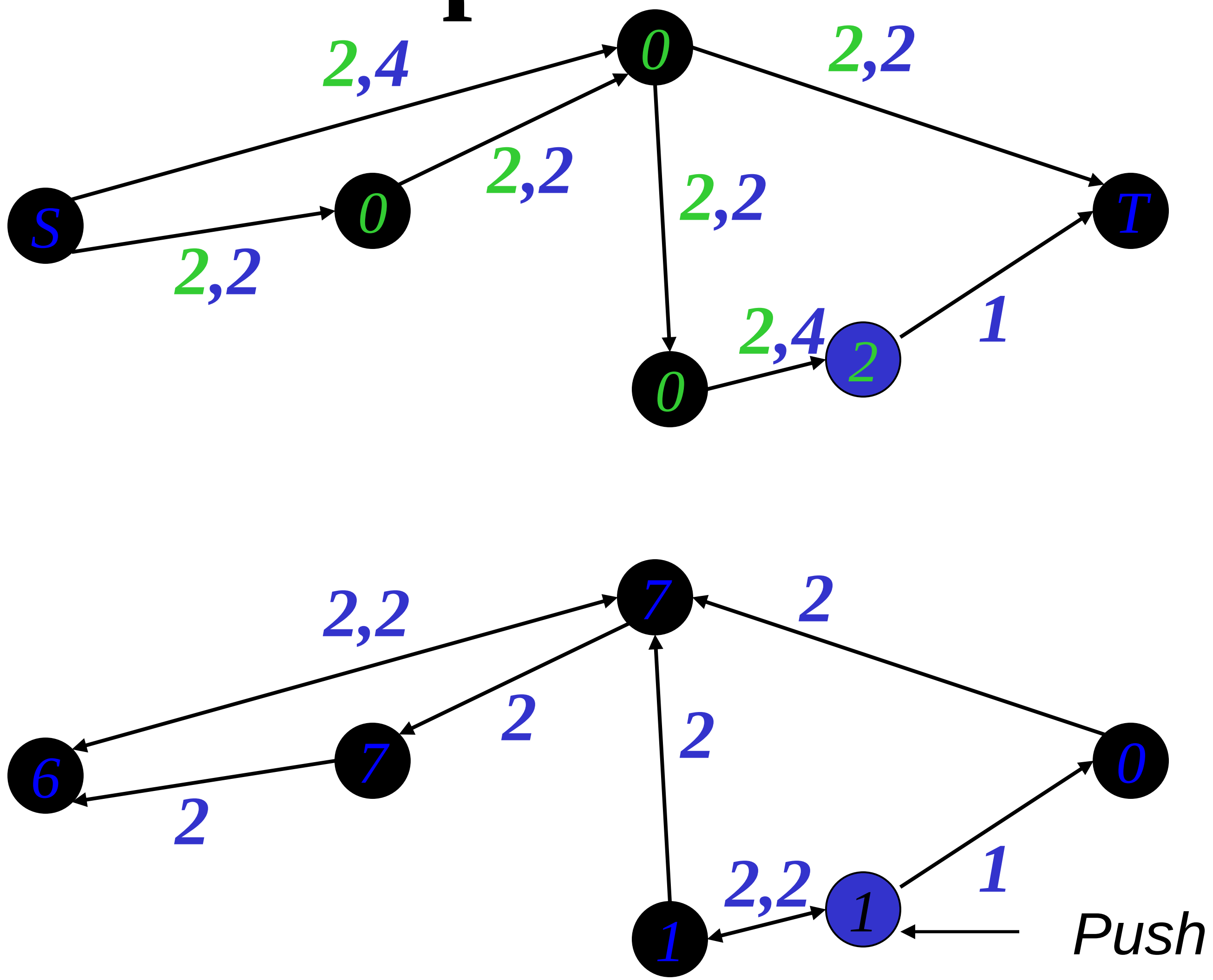


Example – contd.

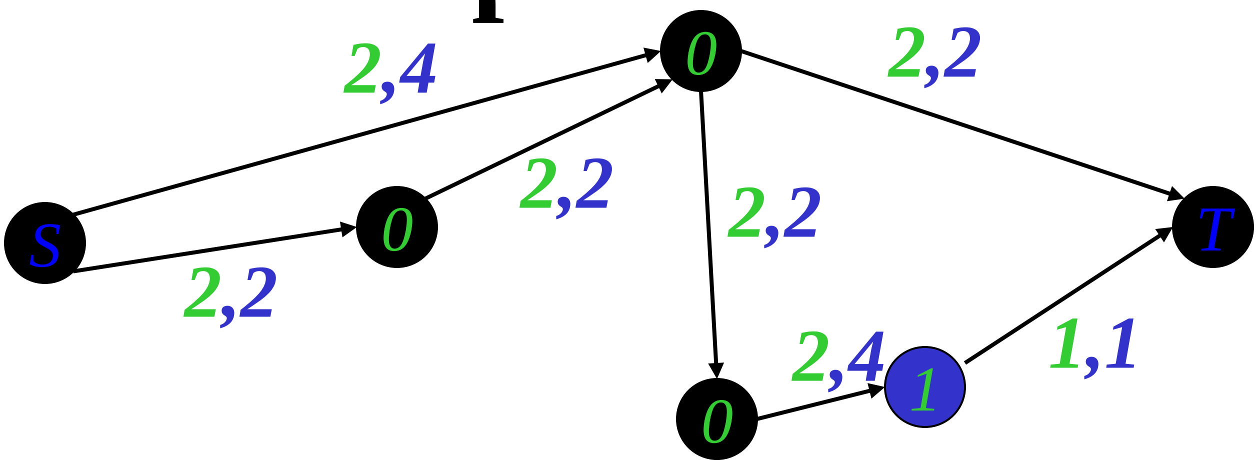


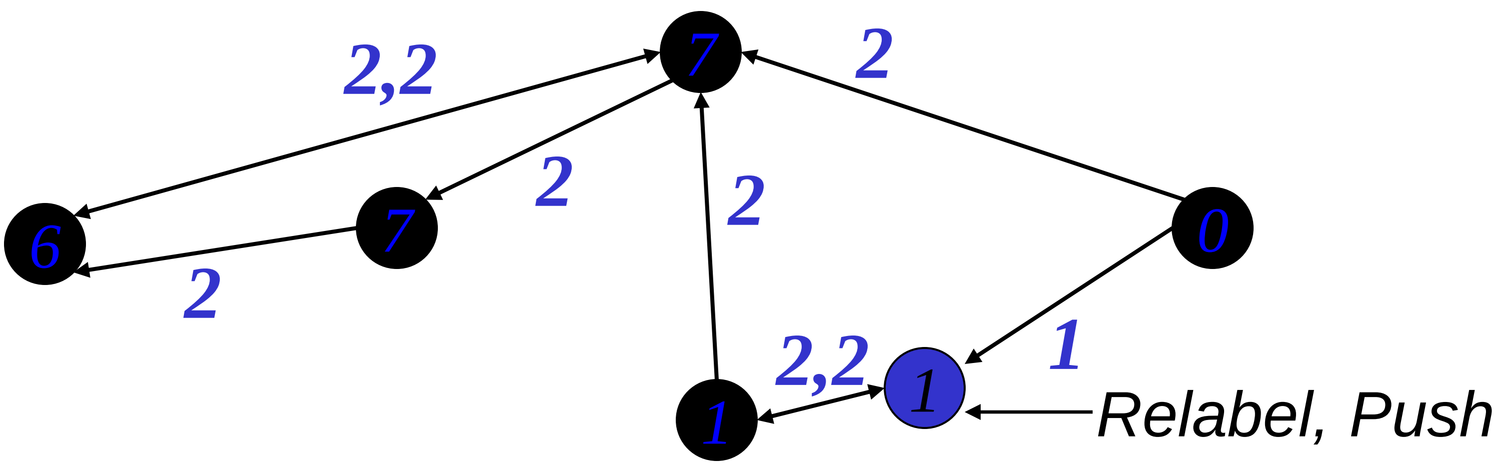


Example – contd.

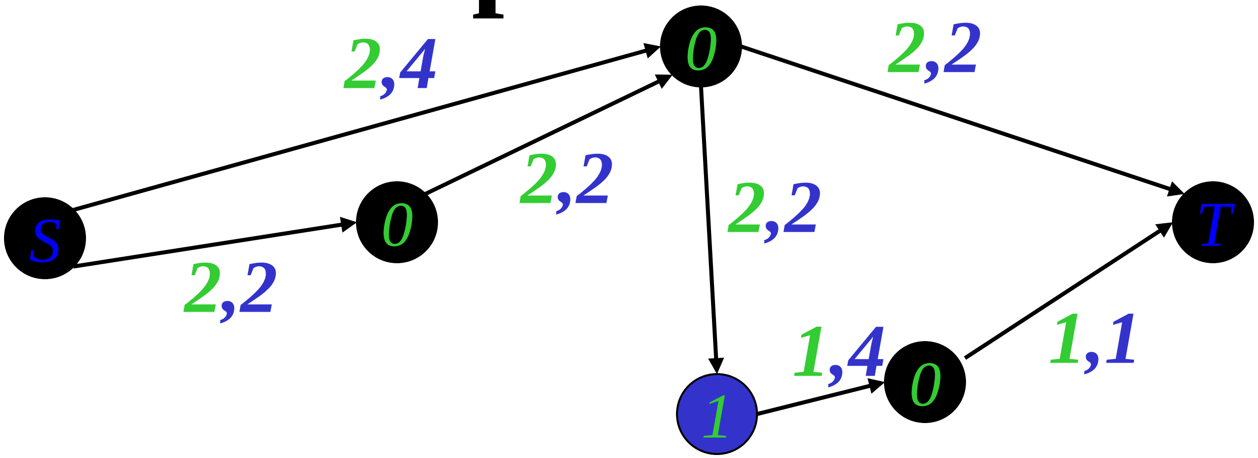


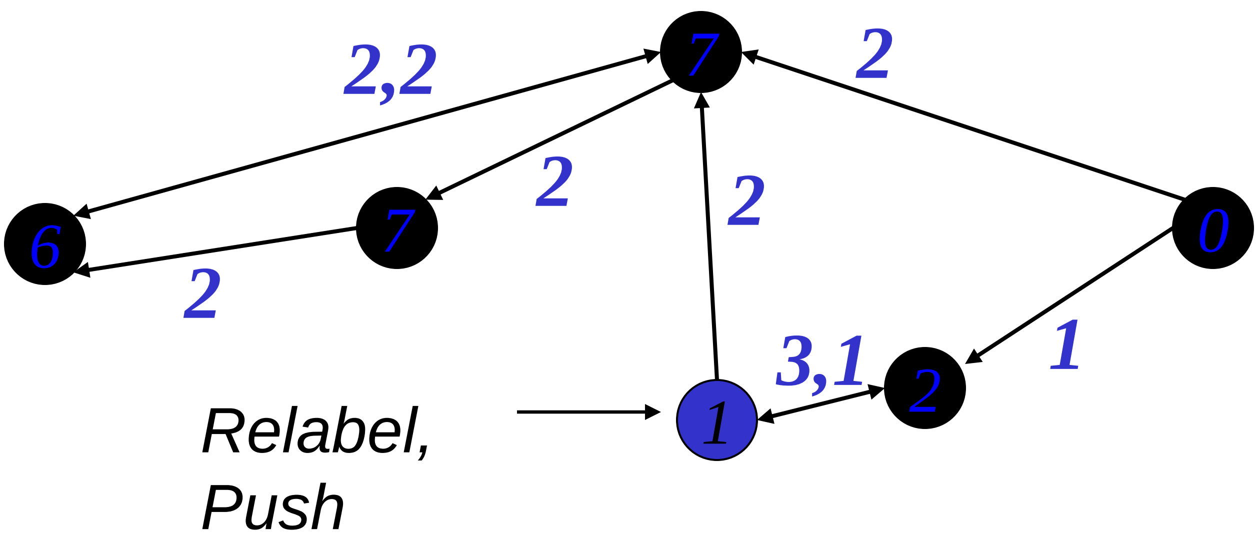
Example – contd.



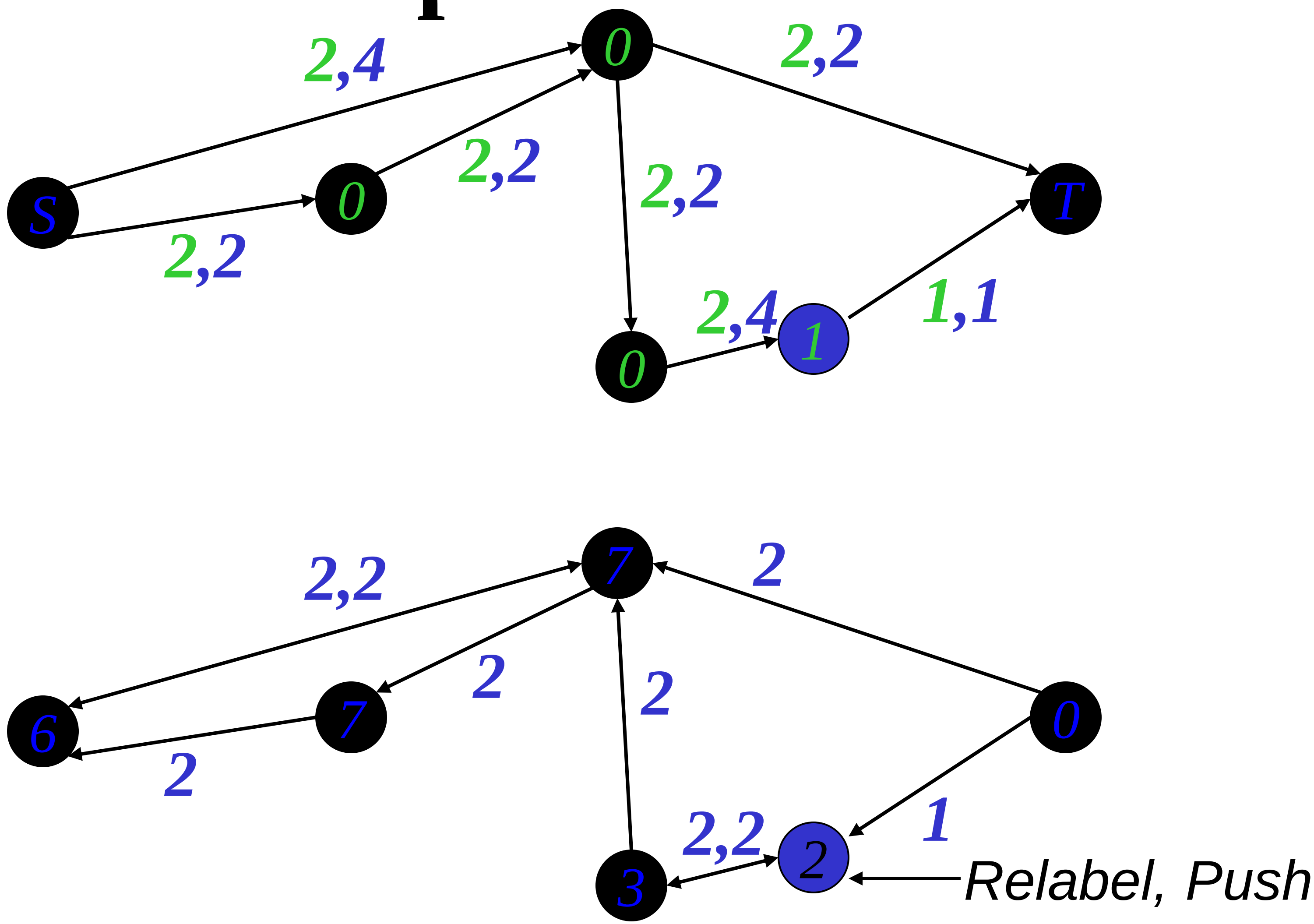


Example – contd.

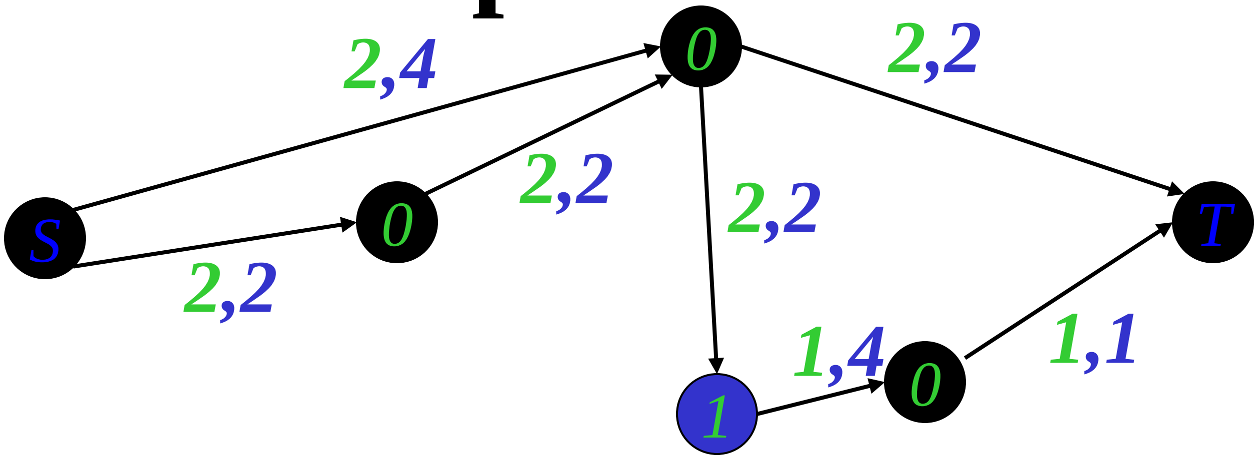


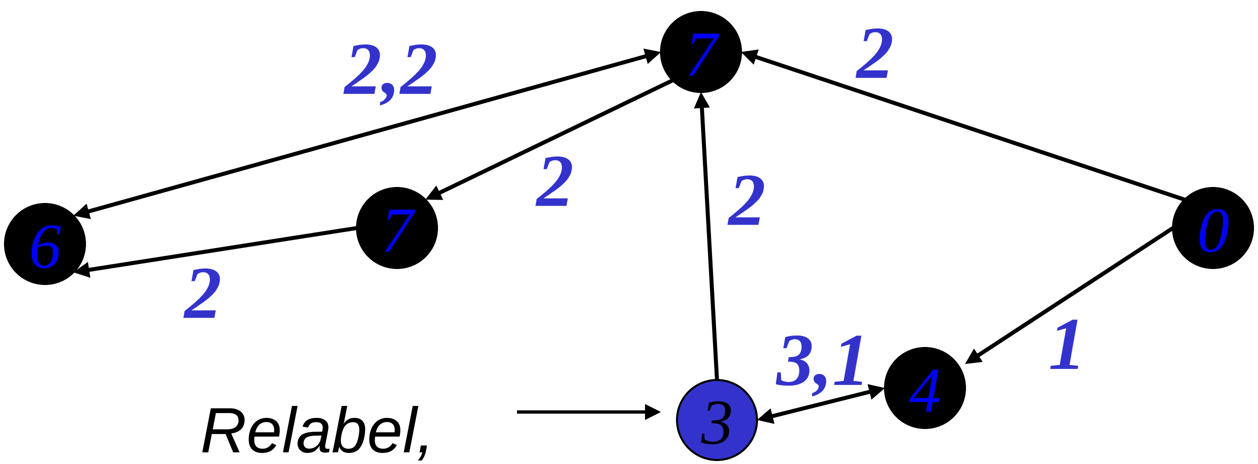


Example – contd.



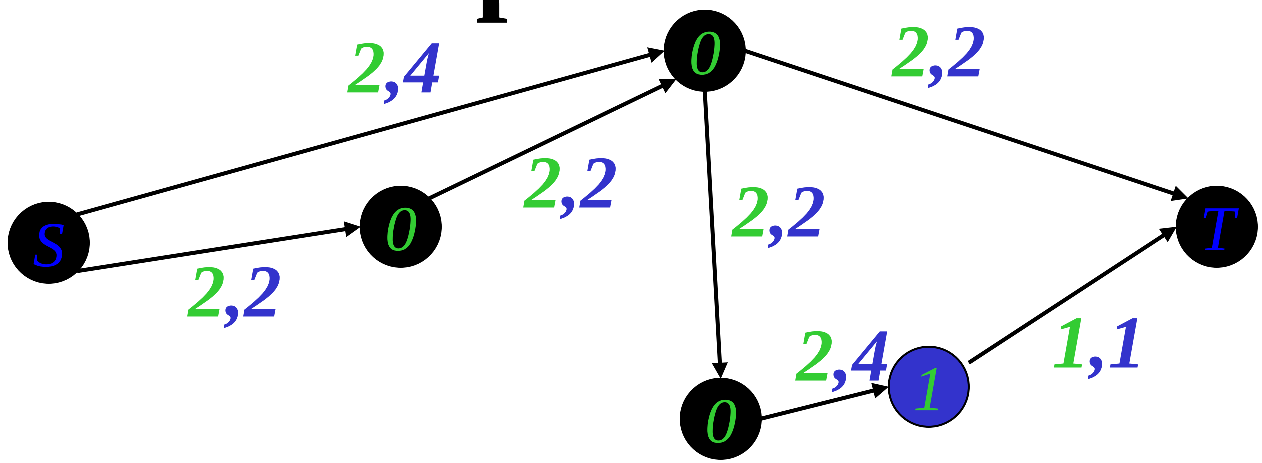
Example – contd.

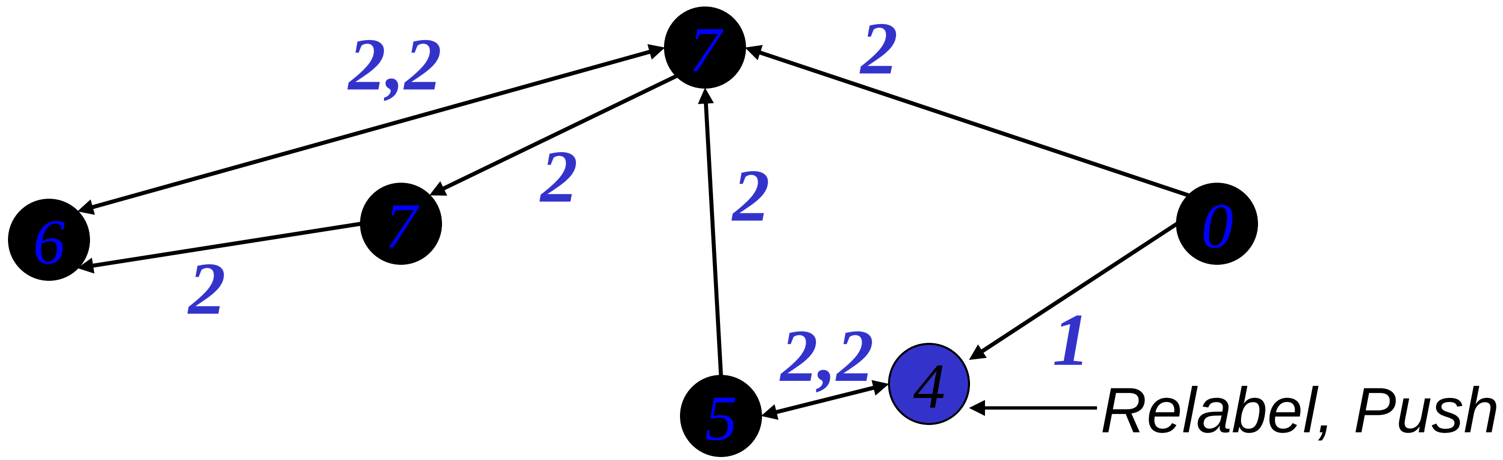




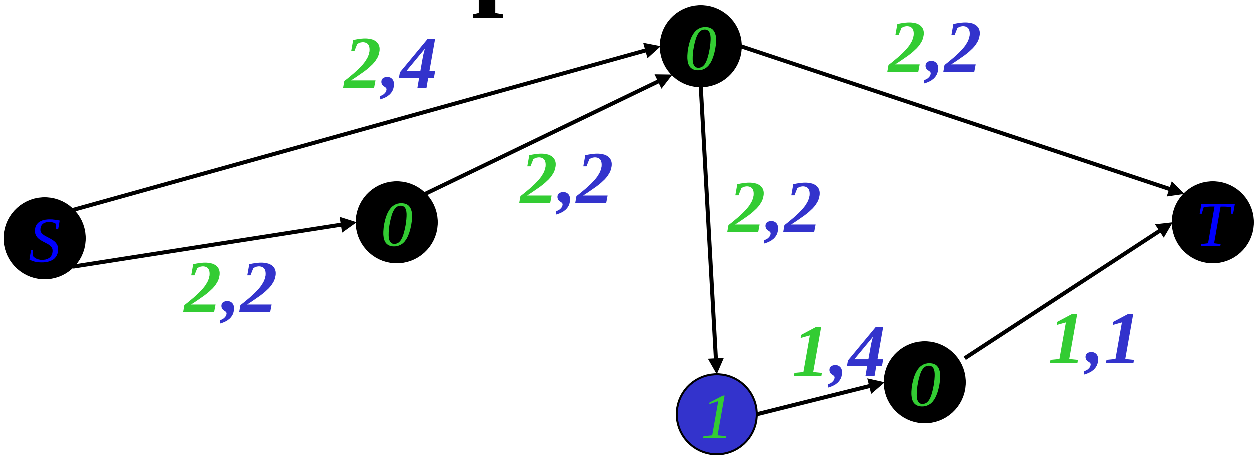
Relabel, Push

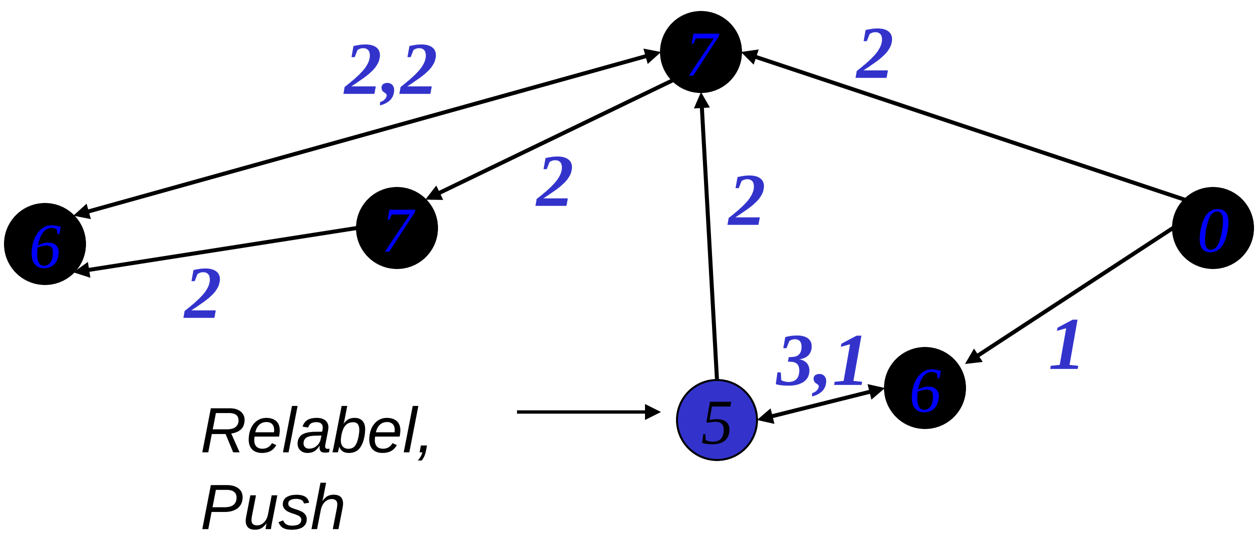
Example – contd.



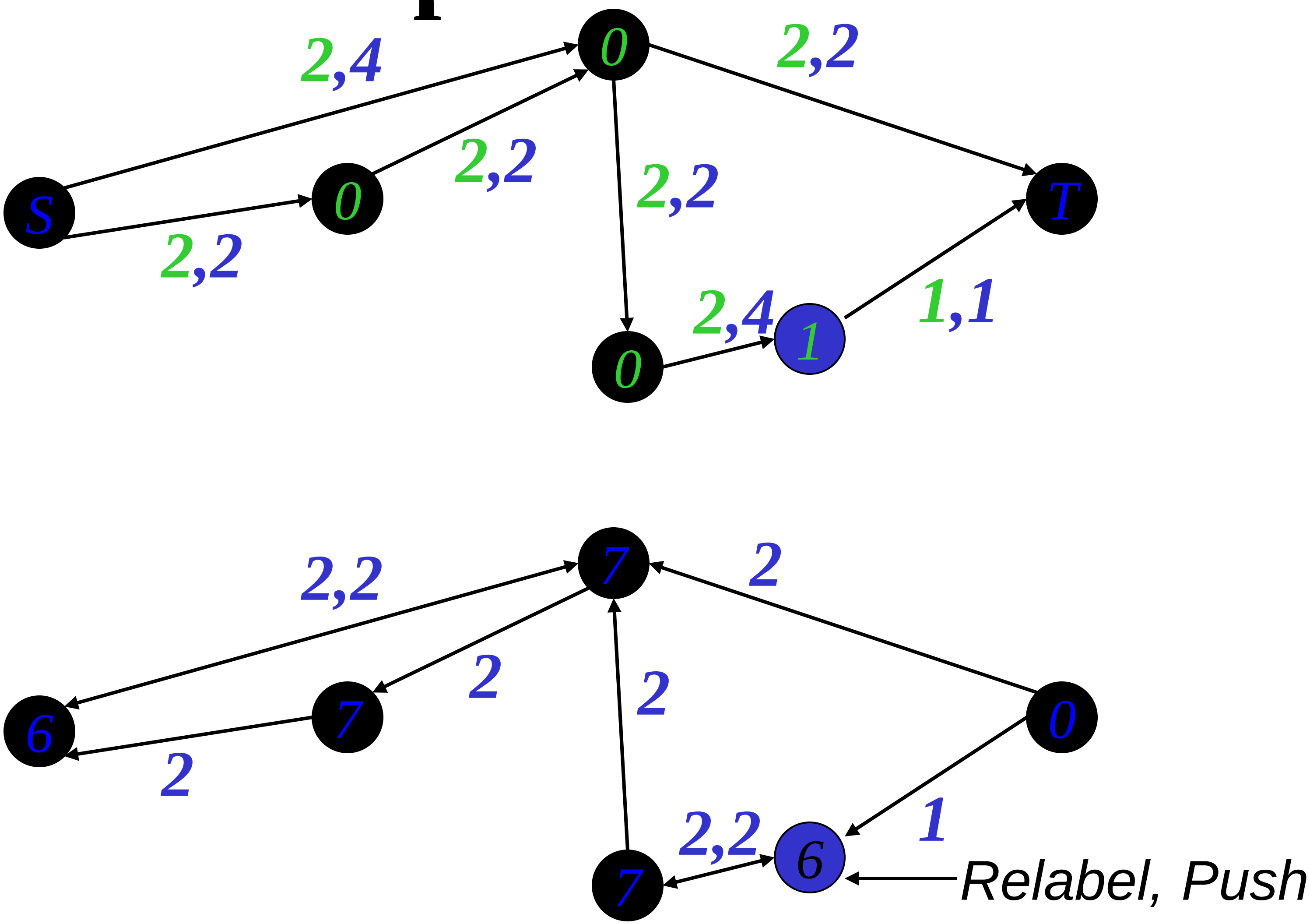


Example – contd.

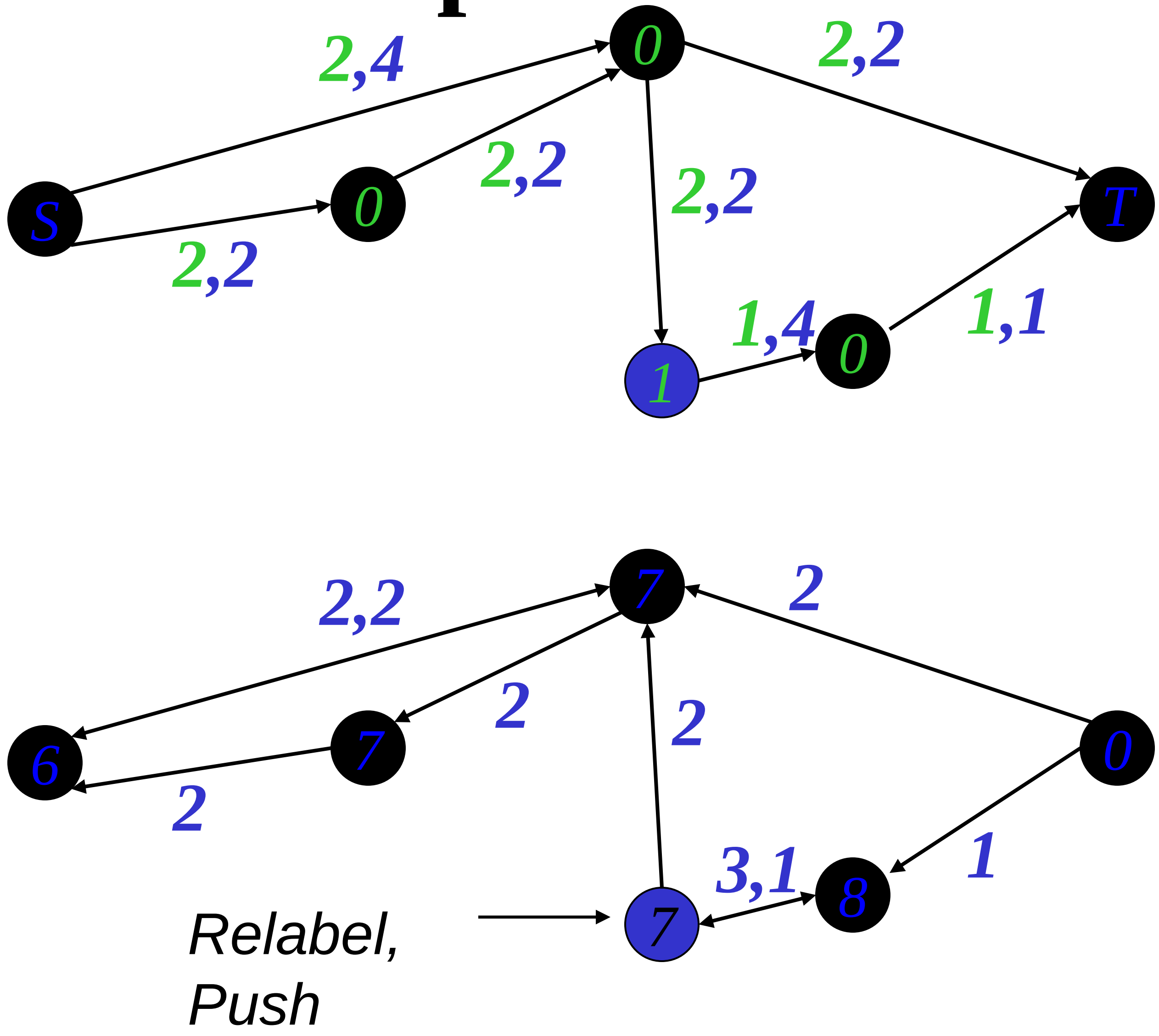




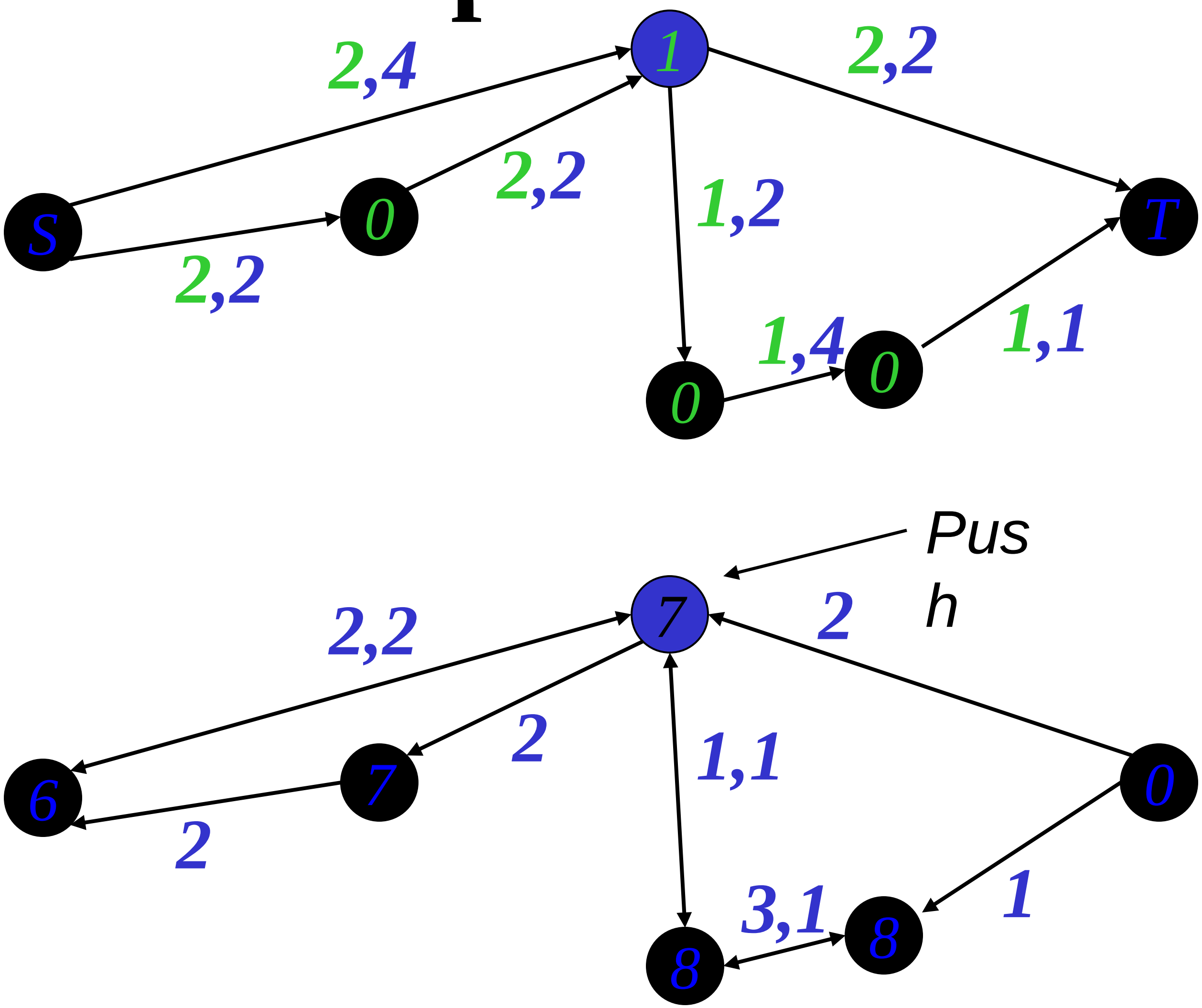
Example – contd.



Example – contd.



Example – contd.



Example – contd.