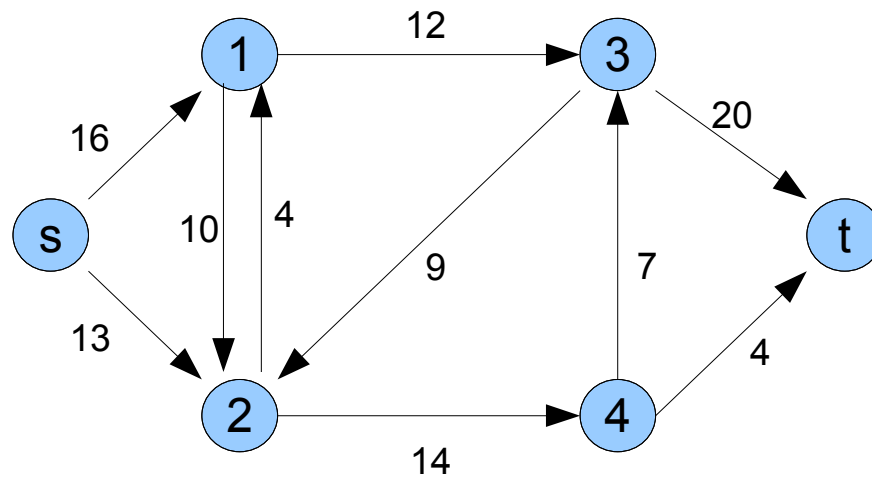


Edmonds-Karp Algorithm

- Use breadth-first search!!!
- This variant of Ford-Fulkerson algorithm runs in $O(nm^2)$.



Lemma 1

- $\Delta_f(v)$ = minimum number of edges that have to be traversed from s to a vertex v in G_f
- Claim: $\Delta_f(v)$ increases monotonically with each flow augmentation for every v in G_f

Proof of Lemma 1

- By contradiction.
- Let f' denote the flow after the first Δ -decreasing flow augmentation. Let v denote the vertex with the smallest decreased $\Delta_{f'}$ value and let (u, v) be the edge on the edge-minimal path to v in $G_{f'}$. Let f denote the flow just before f' . We know that $\Delta_{f'}(v) < \Delta_f(v)$, and
 - $\Delta_{f'}(u) = \Delta_f(u) - 1$
 - $\Delta_{f'}(u) \geq \Delta_f(u)$
 - Assume that (u, v) is in G_f
 - $\Delta_f(v) \leq \Delta_{f'}(u) + 1 \leq \Delta_f(u) + 1 = \Delta_{f'}(v)$

Proof of Lemma 1 - Continued

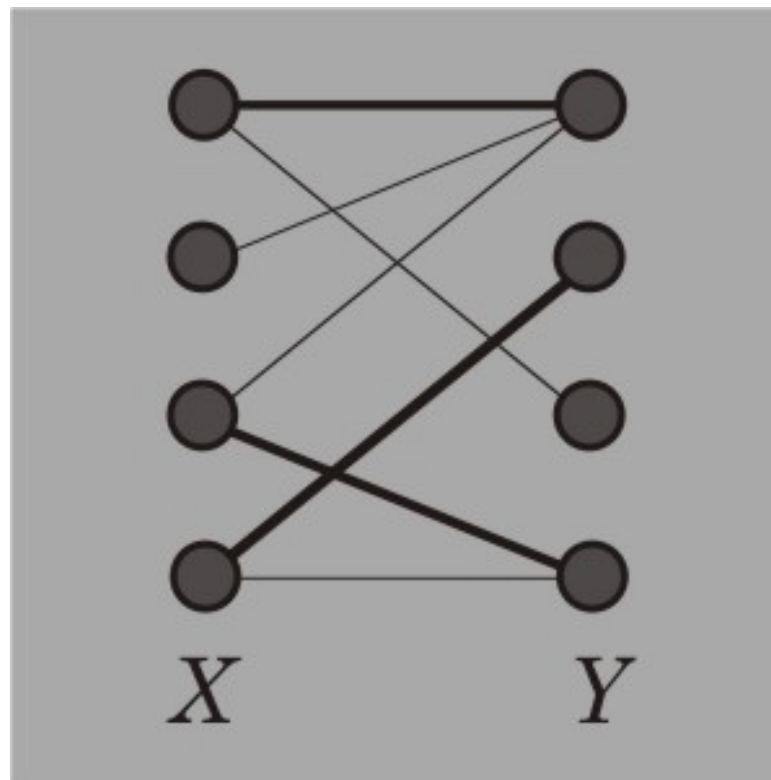
- Hence (u, v) is in $G_{f'}$ but not in G_f
- This is only possible if the augmentation of f increased the flow from v to u .
- Edmonds-Karp algorithm augments along shortest paths.
Therefore
- $\Delta_f(v) = \Delta_f(u) - 1 \leq \Delta_{f'}(u) - 1 = \Delta_{f'}(v) - 2$
- This contradicts our assumption that $\Delta_{f'}(v) < \Delta_f(v)$

Lemma 2

- An edge (u,v) on the augmenting path P in G_f is **critical** if the residual capacity of P is equal to the residual capacity of (u,v) .
- Claim: An edge (u,v) can be critical at most $n/2 - 1$ times.
- Proof: When (u,v) is critical on an augmenting path P , we must have $\Delta_f(v) = \Delta_f(u) + 1$.
- When the flow is augmented along P , (u,v) disappears from the residual network.
- It reappears when (v,u) is on the augmenting path for some flow f' and $\Delta_{f'}(u) = \Delta_{f'}(v) + 1$
- $\Delta_{f'}(u) = \Delta_{f'}(v) + 1 \geq \Delta_f(v) + 1 = \Delta_f(u) + 2$
- $\Delta_f(u)$ is at most $n-2$ ((u,v) being critical implies that $u \neq t$)
- (u,v) can be critical at most $(n-2)/2$ times

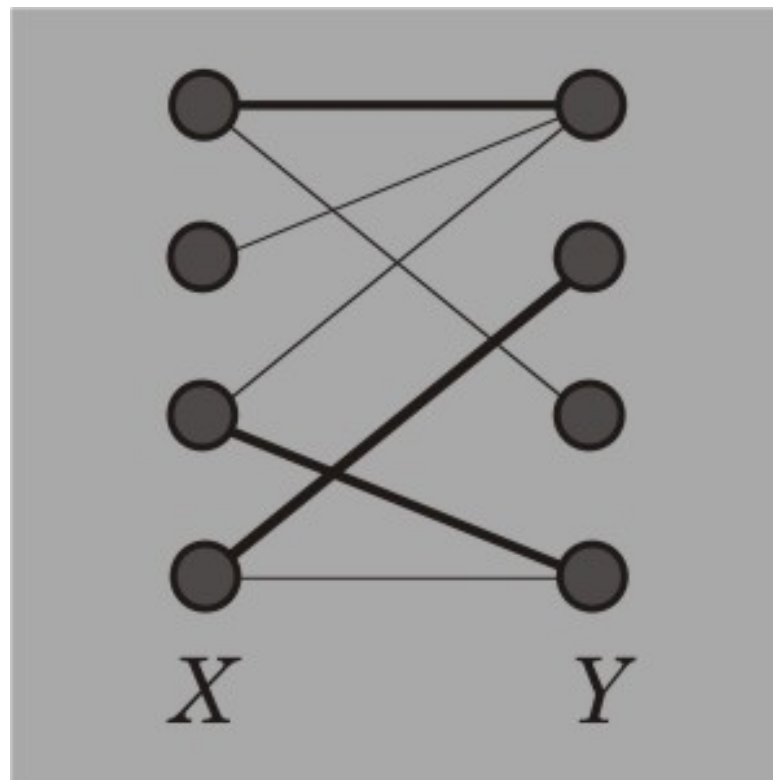
Bipartite Graphs

- A graph $G = (V, E)$ is **bipartite** if its vertices can be partitioned into two subsets X and Y such that every edge connects a vertex in X with a vertex in Y .



Maximum Matching in Graphs

- A **matching** is a subset of edges M in E such that each vertex v in V is incident with at most one edge of M . A **maximum matching** is a matching with the maximum number of edges.



Relating Flow to Matching in Bipartite Graphs

- Add source vertex and connect it to all vertices in X
- Add sink vertex and connect all vertices in Y to it.
- Unit capacities for all edges.

Matching Defines Integral Flow

- Bipartite graph $G = (V, E)$.
- Flow network $G' = (V', E')$.
- If M is a matching in G then there is an integral flow f in G' of value $|f| = |M|$.
- Proof: For every edge (u, v) in M , let $f(s, u) = f(u, v) = f(v, t) = 1$ and $f(u, s) = f(v, u) = f(t, v) = -1$. For all other edges (u, v) in E' , let $f(u, v) = 0$.
- Check that f satisfies skew symmetry, capacity constraints and flow conservation.
- The paths through the edges of matching are vertex disjoint (apart from s and t). It is obvious that $|f| = |M|$ and there is integer flow through each edge.

Flow Defines Matching

- Integral flow network $G' = (V', E')$.
- Bipartite graph $G = (V, E)$.
- If f is a flow in G' of value $|f|$ then M is a matching in G , $|M| = |f|$.
- Proof. Unit capacities and integrality of flow ensures that only one unit of flow can enter a vertex of X . Hence this unit of flow must leave such a vertex through exactly one edge.
- Similarly only one unit of flow can leave a vertex of Y . Hence this unit of flow can enter such a vertex through exactly one edge.
- Let M be the edges from X to Y with unit flow.
- M is a matching.

$$|M| = f(X, Y) = f(X, V') - f(X, X) - f(X, s) - f(X, t) = 0 - 0 + f(s, X) - 0 = f(s, V') = |f|$$

Max Matching Defines Max Flow

Max Flow Defines Max Matching

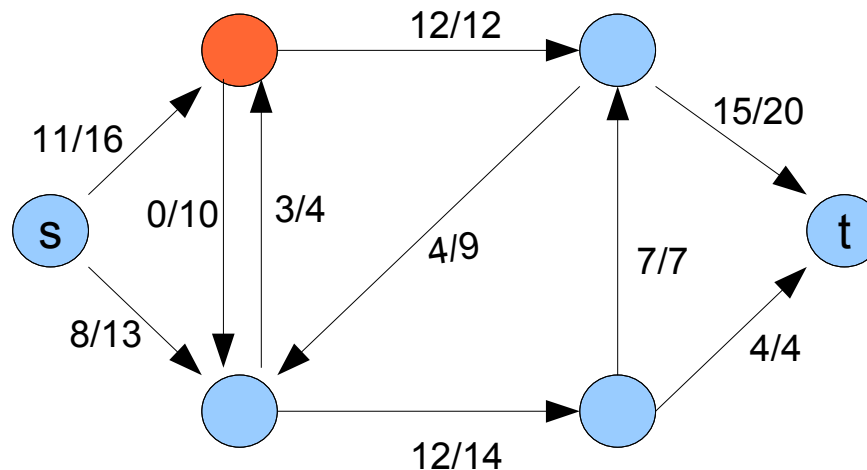
- Follows immediately if we can show that max flow algorithm returns integral flow when capacities are integer.
- Easy induction proof, see exercise 26.3-2

Push-Relabel Methods

- Work on one vertex at a time rather than entire residual network.
- Do not maintain flow conservation property until at the very end.
- Run in $O(n^2m)$. Can be improved to $O(n^3)$. Edmonds-Karp algorithm runs in $O(nm^2)$.

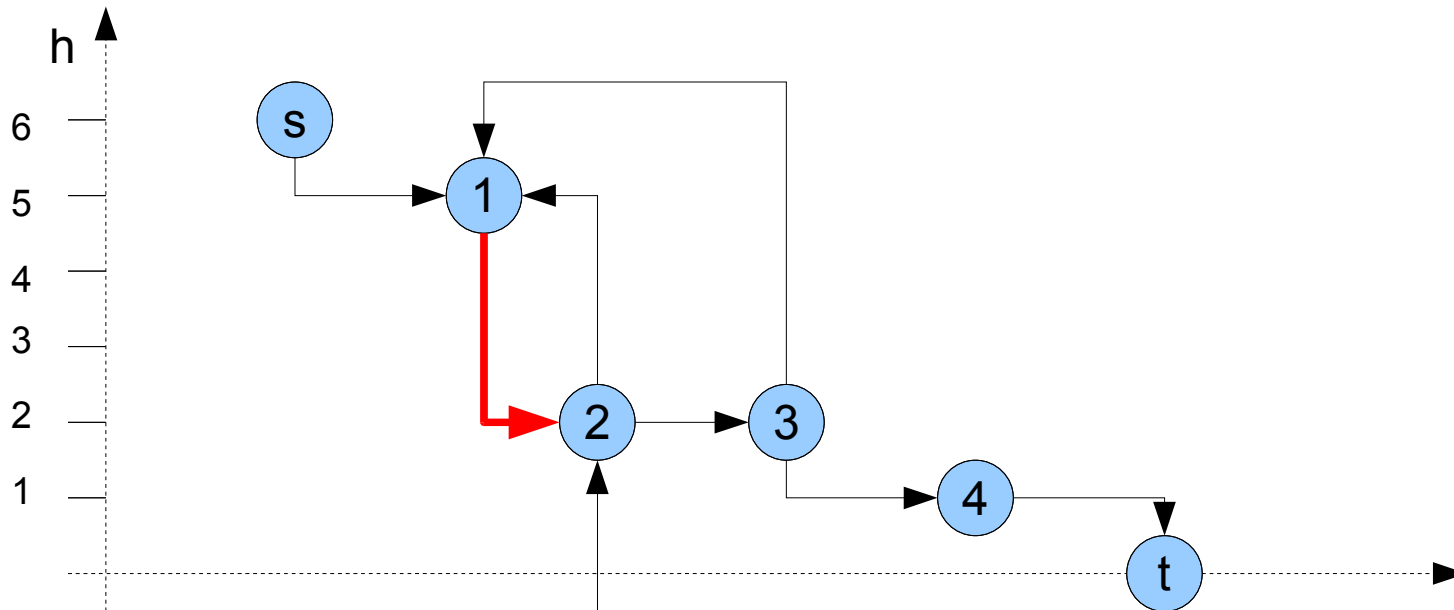
Preflow

- **Preflow**: Real-valued function $f : V \times V \rightarrow R$ satisfying:
 - Capacity constraint: $f(u,v) \leq c(u,v)$ for all $u, v \in V$
 - Skew symmetry: $f(u,v) = -f(v,u)$ for all $u, v \in V$
 - Flow conservation: $\sum_{v \in V} f(v, u) \geq 0$ for all $u \in V - \{s, t\}$
- **Excess flow** into vertex u : $e_f(u) = \sum_{v \in V} f(v, u)$
- Vertex u is **overflowing** if $e_f(u) > 0$



Height Function

- Let $G = (V, E)$ be a flow network with source s and sink t . Let f be a preflow in G .
- **Height function:** Integer-valued function $h : V \rightarrow N$ satisfying:
 - $h(s) = |V|$, $h(t) = 0$
 - $h(u) \leq h(v) + 1$ for every residual edge (u, v) .



Initialization

- Initial preflow:

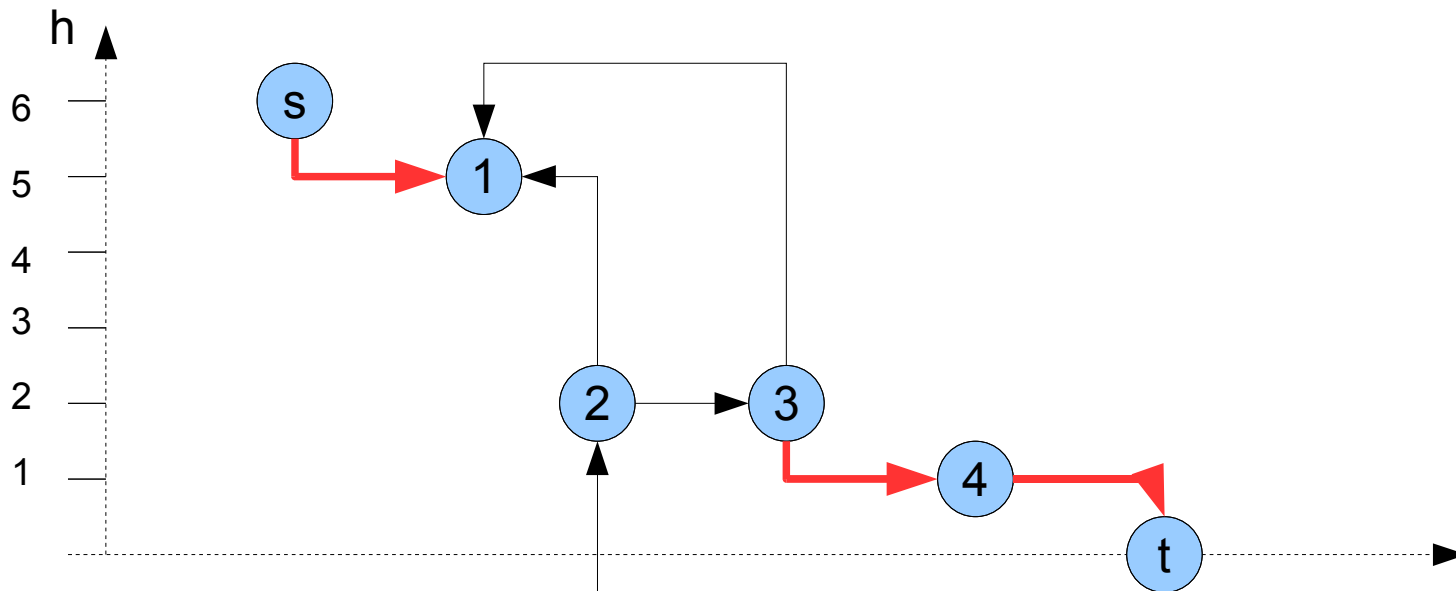
$$f[u, v] = \begin{cases} c(u, v) & \text{if } u = s \\ -c(v, u) & \text{if } v = s \\ 0 & \text{otherwise} \end{cases}$$

- Initial height function:

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

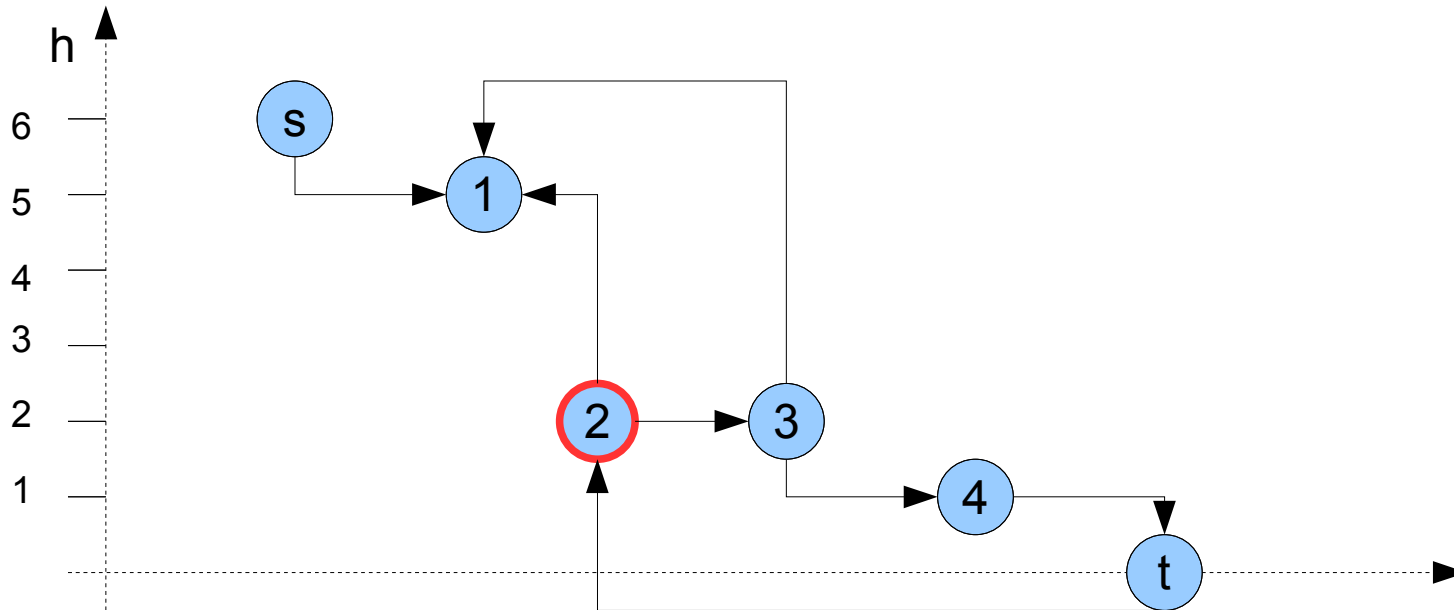
Push

- Consider an edge (u,v) such that
 - $e_f(u) > 0$, $c_f(u,v) > 0$, $h(u) = h(v) + 1$
- Push $\min \{ e_f(u), c_f(u,v) \}$ units of flow from u to v .
 - Update $f(u,v)$ and $f(v,u)$, $e_f(u)$ and $e_f(v)$

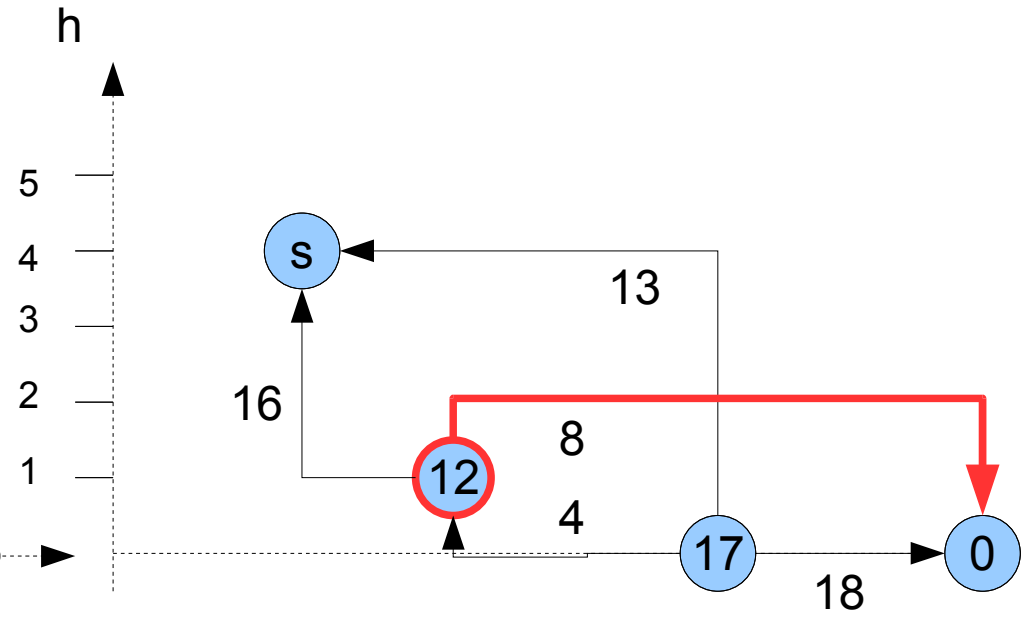
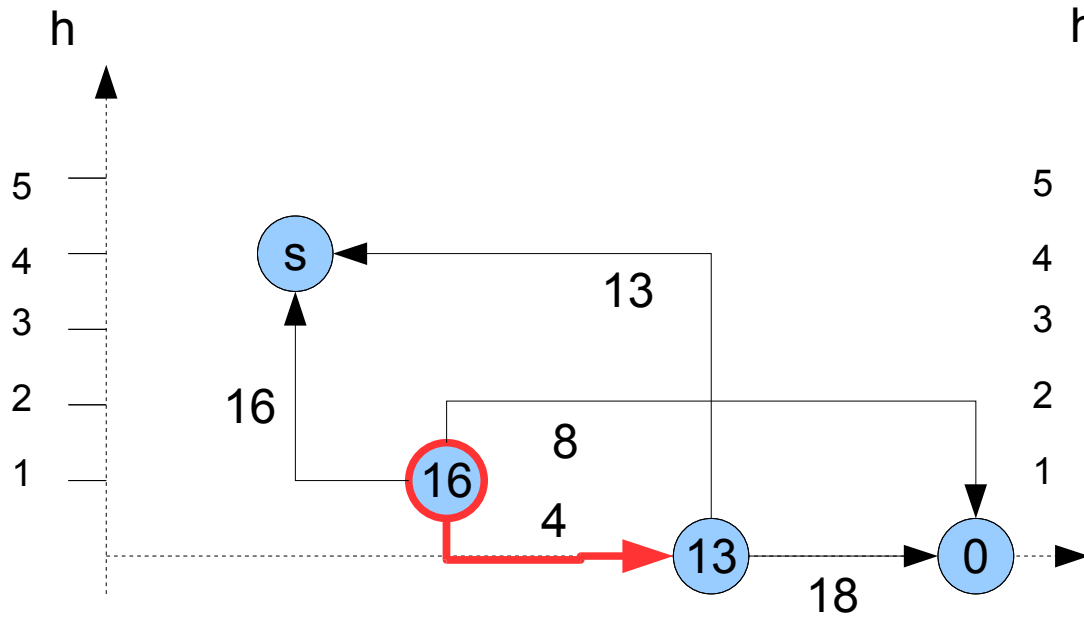
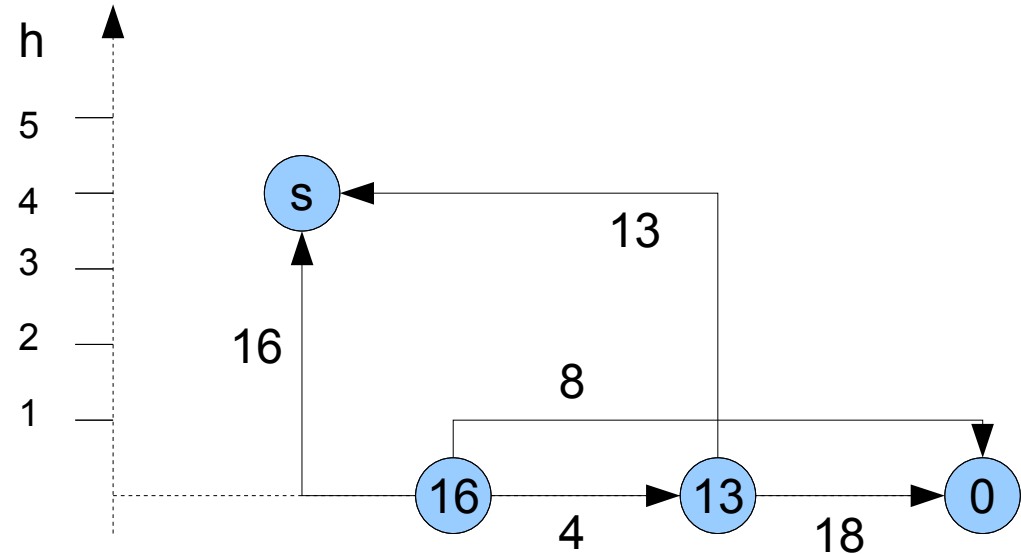
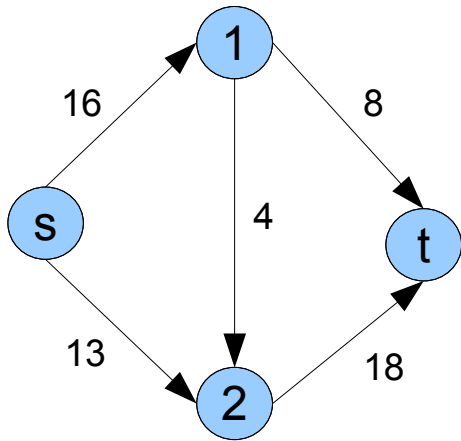


Relabel

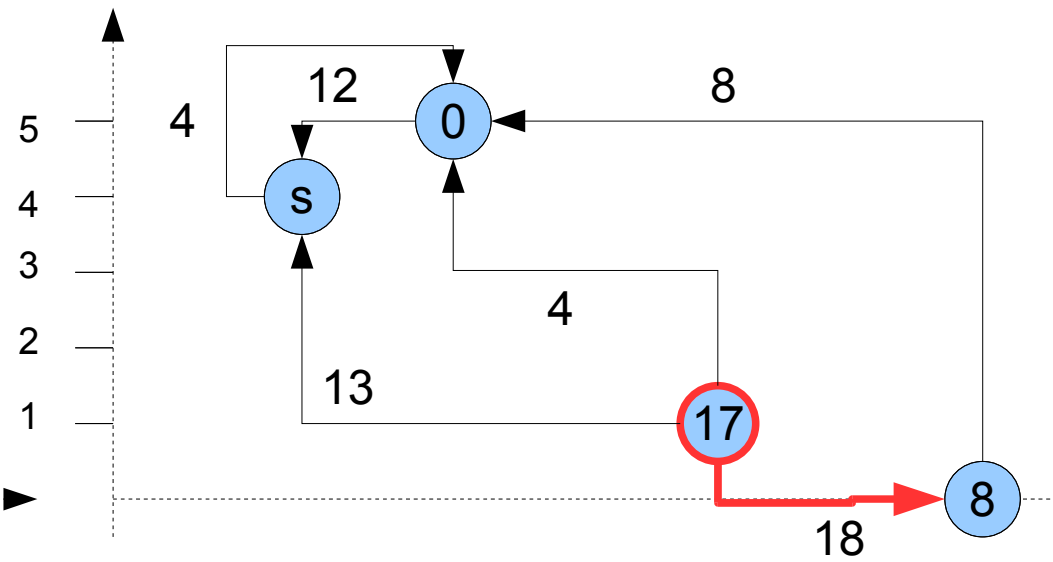
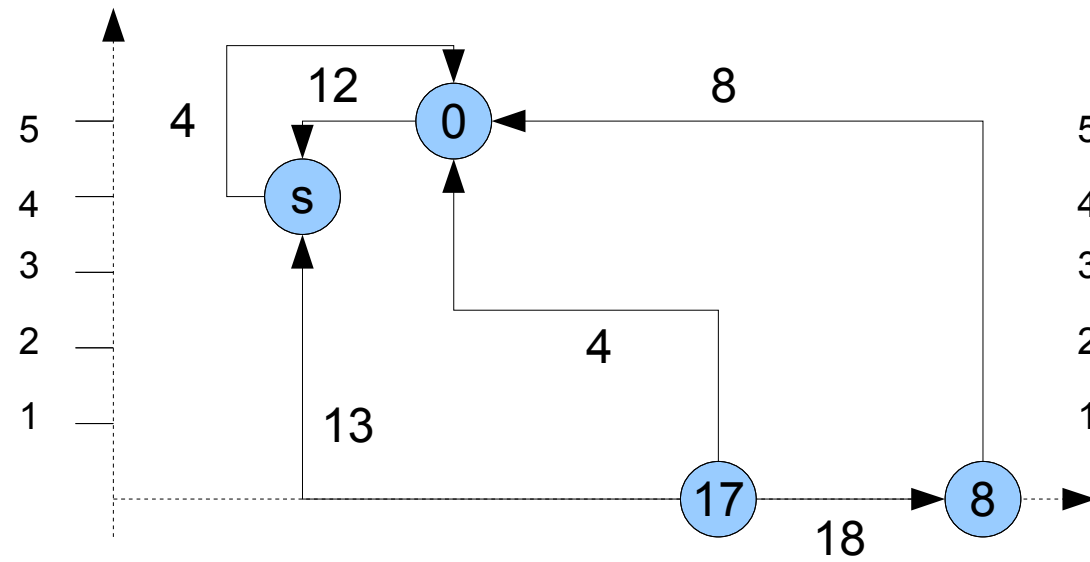
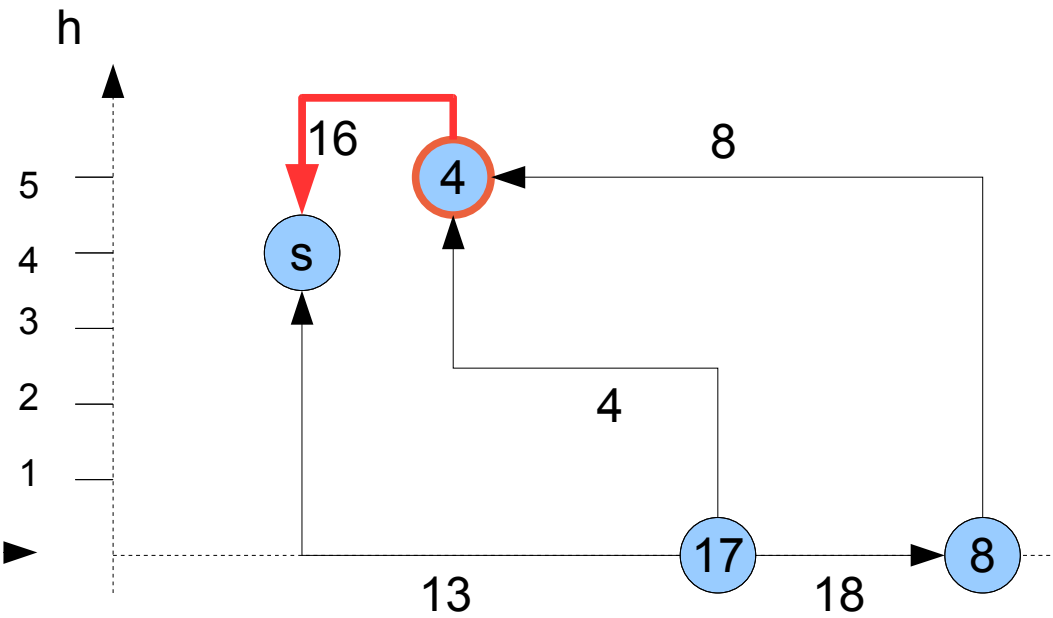
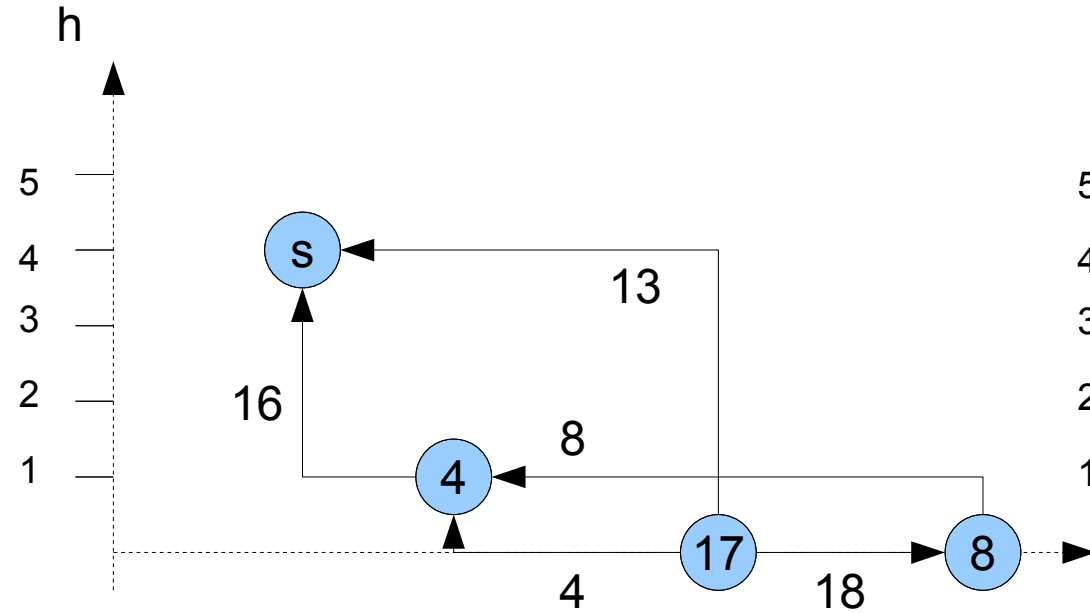
- Consider a vertex u such that
 - $e_f(u) > 0$
 - $h(u) \leq h(v)$ for all edges from $u \in E_f$
- Let $h(u) = 1 + \min \{ h(v) : (u,v) \in E_f \}$



Push Relabel - Example



Push Relabel - Example



Push Relabel - Example

