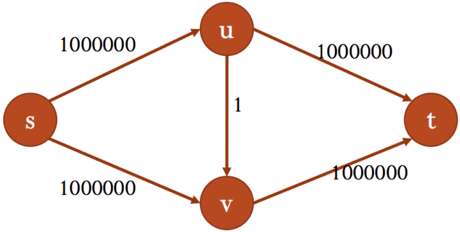
COL351: Analysis and Design of Algorithms

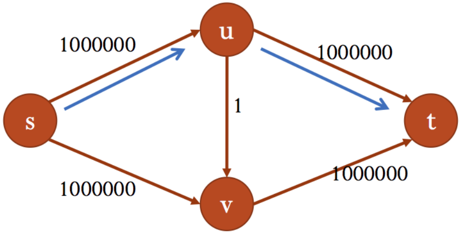
Ragesh Jaiswal, CSE, IITD

Network Flow

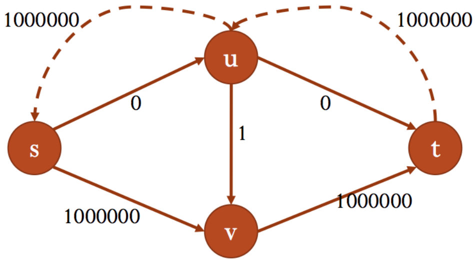
- Let $C = \sum_{e \text{ out of } s} c(e)$.
- The running time of the `Ford-Fulkerson` algorithm is $O(m \cdot C)$.
- C could be very large compared to the size of the graph. Consider an example below.



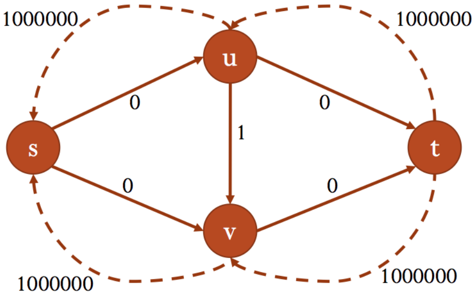
- Consider the favorable case where the augmenting paths s, u, t and s, v, t are chosen.



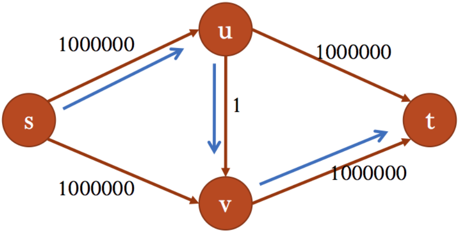
- Consider the favorable case where the augmenting paths s, u, t and s, v, t are chosen.



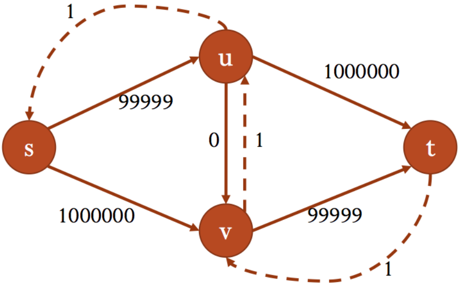
- Consider the favorable case where the augmenting paths s, u, t and s, v, t are chosen.
- Max flow is found in 2 augmentations.



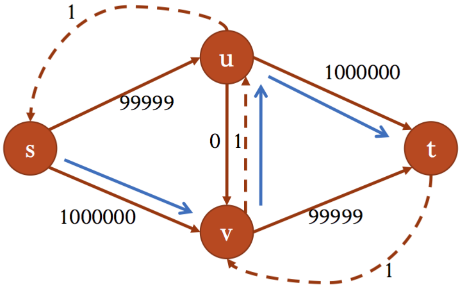
- Now consider the case when augmenting paths s, u, v, t and s, v, u, t are chosen repeatedly.



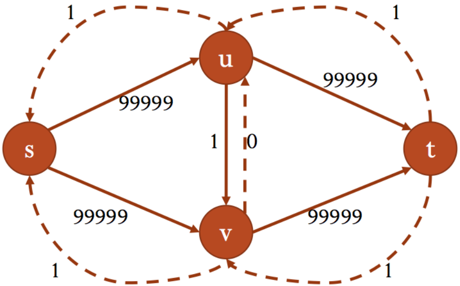
- Now consider the case when augmenting paths s, u, v, t and s, v, u, t are chosen repeatedly.



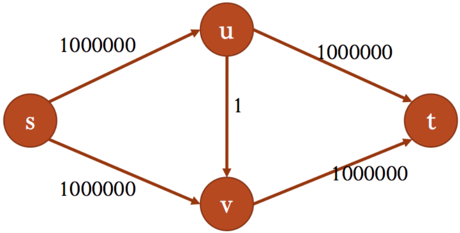
- Now consider the case when augmenting paths s, u, v, t and s, v, u, t are chosen repeatedly.



- Now consider the case when augmenting paths s, u, v, t and s, v, u, t are chosen repeatedly.



- Let $C = \sum_{e \text{ out of } s} c(e)$.
- The running time of the `Ford-Fulkerson` algorithm is $O(m \cdot C)$.
- C could be very large compared to the size of the graph.
  - For the example below, we might get a better running time if we could hide the edge with small capacity when looking for an augmenting path.
- General idea: Use all edges with large capacities before considering edges with smaller capacity.



- For an s-t flow and a positive integer Δ, let $G_f(\Delta)$ denote the subgraph of the residual graph G_f that consists of all vertices but only edges with residual capacity of at least Δ.
- Idea: Instead of finding augmenting paths in G_f, we will find augmenting paths in $G_f(\Delta)$ for smaller and smaller values of Δ.

Algorithm

`Scaling-Max-Flow`

- Start with an s-t flow such that for all e, $f(e) = 0$
- $\Delta \leftarrow$ largest power of 2 smaller than C
- While $(\Delta \geq 1)$
  - While there is an s-t path P in $G_f(\Delta)$
    - Augment flow along an augmenting path and let f' be the resulting flow
    - Update f to f' and $G_f(\Delta)$ to $G_{f'}(\Delta)$
  - $\Delta \leftarrow \Delta/2$
- return(f)

Algorithm

`Scaling-Max-Flow`

- Start with an $s-t$ flow such that for all e, $f(e)=0$
- $\Delta \leftarrow$ largest power of 2 smaller than C
- While $(\Delta \geq 1)$
  - While there is an $s-t$ path P in $G_f(\Delta)$
    - Augment flow along an augmenting path and let f' be the resulting flow
    - Update f to f' and $G_f(\Delta)$ to $G_{f'}(\Delta)$
  - $\Delta \leftarrow \Delta/2$
- return(f)

- Claim 1: The algorithm returns max. flow on termination.

Algorithm

`Scaling-Max-Flow`

- Start with an $s-t$ flow such that for all e, $f(e) = 0$
- $\Delta \leftarrow$ largest power of 2 smaller than C
- While $(\Delta \geq 1)$
  - While there is an $s-t$ path P in $G_f(\Delta)$
    - Augment flow along an augmenting path and let f' be the resulting flow
    - Update f to f' and $G_f(\Delta)$ to $G_{f'}(\Delta)$
  - $\Delta \leftarrow \Delta/2$
- return(f)

- Claim 1: The algorithm returns max. flow on termination.
- Claim 2: The outer while loop runs for at most $(1 + \lceil \log C \rceil)$ steps.

Algorithm

`Scaling-Max-Flow`

- Start with an $s-t$ flow such that for all e, $f(e) = 0$
- $\Delta \leftarrow$ largest power of 2 smaller than C
- While $(\Delta \geq 1)$
  - While there is an $s-t$ path P in $G_f(\Delta)$
    - Augment flow along an augmenting path and let f' be the resulting flow
    - Update f to f' and $G_f(\Delta)$ to $G_{f'}(\Delta)$
  - $\Delta \leftarrow \Delta/2$
- return(f)

- Claim 1: The algorithm returns max. flow on termination.
- Claim 2: The outer while loop runs for at most $(1 + \lceil \log C \rceil)$ steps.
- Claim 3: Each augmentation increases the flow by at least Δ (whatever the current value of Δ is).

Algorithm

`Scaling-Max-Flow`

- Start with an $s-t$ flow such that for all e, $f(e) = 0$
- $\Delta \leftarrow$ largest power of 2 smaller than C
- While $(\Delta \geq 1)$
  - While there is an $s-t$ path P in $G_f(\Delta)$
    - Augment flow along an augmenting path and let f' be the resulting flow
    - Update f to f' and $G_f(\Delta)$ to $G_{f'}(\Delta)$
  - $\Delta \leftarrow \Delta/2$
- return(f)

- Claim 1: The algorithm returns max. flow on termination.
- Claim 2: The outer while loop runs for at most $(1 + \lceil \log C \rceil)$ steps.
- Claim 3: Each augmentation increases the flow by at least Δ (whatever the current value of Δ is).
- Claim 4: Let f be the flow at the end of a Δ-*scaling* phase. Then there is an $s-t$ cut (A, B) such that $c(A, B) \leq v(f) + m \cdot \Delta$.
  - Corollary: The max flow in the graph has value at most $v(f) + m \cdot \Delta$.

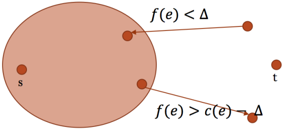
- Claim 4: Let f be the flow at the end of a Δ-*scaling* phase. Then there is an $s-t$ cut (A, B) such that $c(A, B) \leq v(f) + m \cdot \Delta$.
  - Corollary: The max flow in the graph has value at most $v(f) + m \cdot \Delta$.

Proof of Claim 4.

Let A be the set of vertices that are reachable from s in $G_f(\Delta)$ (see figure below). Then we have

$$\begin{aligned} v(f) &= \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ into } A} f(e) \\ &\geq \sum_{e \text{ out of } A} (c(e) - \Delta) - \sum_{e \text{ into } A} \Delta \\ &\geq c(A, B) - m \cdot \Delta. \end{aligned}$$

□



A (all vertices reachable from s in $G_f(\Delta)$.

Algorithm

`Scaling-Max-Flow`

- Start with an $s-t$ flow such that for all e, $f(e) = 0$
- $\Delta \leftarrow$ largest power of 2 smaller than C
- While ($\Delta \geq 1$)
  - While there is an $s-t$ path P in $G_f(\Delta)$
    - Augment flow along an augmenting path and let f' be the resulting flow
    - Update f to f' and $G_f(\Delta)$ to $G_{f'}(\Delta)$
  - $\Delta \leftarrow \Delta/2$
- return(f)

- Claim 1: The algorithm returns max. flow on termination.
- Claim 2: The outer while loop runs for at most $(1 + \lceil \log C \rceil)$ steps.
- Claim 3: Each augmentation increases the flow by at least Δ (whatever the current value of Δ is).
- Claim 4: Let f be the flow at the end of a Δ-*scaling* phase. Then there is an $s-t$ cut (A, B) such that $c(A, B) \leq v(f) + m \cdot \Delta$.
  - Corollary: The max flow in the graph has value at most $v(f) + m \cdot \Delta$.
- Claim 5: The total number of iterations of the inner while loop is at most $2m$.

Algorithm

`Scaling-Max-Flow`
- Start with an $s-t$ flow such that for all e, $f(e) = 0$
- $\Delta \leftarrow$ largest power of 2 smaller than C
- While $(\Delta \geq 1)$
  - While there is an $s-t$ path P in $G_f(\Delta)$
    - Augment flow along an augmenting path and let f' be the resulting flow
    - Update f to f' and $G_f(\Delta)$ to $G_{f'}(\Delta)$
  - $\Delta \leftarrow \Delta/2$
- return(f)

- Claim 1: The algorithm returns max. flow on termination.
- Claim 2: The outer while loop runs for at most $(1 + \lceil \log C \rceil)$ steps.
- Claim 3: Each augmentation increases the flow by at least Δ (whatever the current value of Δ is).
- Claim 4: Let f be the flow at the end of a Δ-*scaling* phase. Then there is an $s-t$ cut (A, B) such that $c(A, B) \leq v(f) + m \cdot \Delta$.
  - Corollary: The max flow in the graph has value at most $v(f) + m \cdot \Delta$.
- Claim 5: The total number of iterations of the inner while loop is at most $2m$.
- Claim 6: The running time of `Scaling-Max-Flow` algorithm is $O(m^2 \cdot \log C)$.

End