# COL351: Analysis and Design of Algorithms

Ragesh Jaiswal, CSE, IITD

# Network Flow

### Algorithm

`Scaling-Max-Flow`

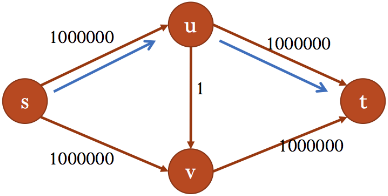
- Start with an $s - t$ flow such that for all $e$, $f(e) = 0$
- $\Delta \leftarrow$ largest power of 2 smaller than $C$
- While $(\Delta \geq 1)$
  - While there is an $s - t$ path $P$ in $G_f(\Delta)$
    - Augment flow along an augmenting path and let $f'$ be the resulting flow
    - Update $f$ to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

- The running time of this algorithm depends on $C$.
- Can we design an algorithm such that the running time depends only on the structure of the graph?
  - *In a model where doing operations on the weights cost $O(1)$ time.*

### Algorithm

`Edmonds-Karp`

- Start with an $s-t$ flow such that for all $e$, $f(e) = 0$
- While there is an $s-t$ path $P$ in $G_f$
  - Find an $s-t$ path in $G_f$ with least hop-length
  - Augment flow along an augmenting path and let $f'$ be the resulting flow
  - Update $f$ to $f'$ and $G_f$ to $G_{f'}$
- return$(f)$



## Algorithm

`Edmonds-Karp`

- Start with an $s-t$ flow such that for all $e$, $f(e) = 0$
- While there is an $s-t$ path $P$ in $G_f$
  - Find an $s-t$ path in $G_f$ with least hop-length
  - Augment flow along an augmenting path and let $f'$ be the resulting flow
  - Update $f$ to $f'$ and $G_f$ to $G_{f'}$
- return($f$)

- How do we bound the running time of the above algorithm?

- Let $d_f(s, v)$ denote the hop-length of the shortest path from $s$ to $v$ in $G_f$.
- Claim 1: For all $v \neq s, t$, $d_f(s, v)$ either remains same or increases with each flow augmentation.

### Proof of claim 1

- Let $f$ be the flow just before the first augmentation that decreases the shortest distance of some vertex. Let $f'$ be the flow after this augmentation.
- Let $v$ be the vertex with minimum value of $d_{f'}(s, v)$ whose shortest distance was reduced.
- Let $u$ be the vertex just before $v$ in the shortest path from $s$ to $v$ in $G_{f'}$.
- Claim 1.1: $d_{f'}(s, u) = d_{f'}(s, v) - 1$.

- Let $d_f(s, v)$ denote the hop-length of the shortest path from $s$ to $v$ in $G_f$.
- Claim 1: For all $v \neq s, t$, $d_f(s, v)$ either remains same or increases with each flow augmentation.

### Proof of claim 1

- Let $f$ be the flow just before the first augmentation that decreases the shortest distance of some vertex. Let $f'$ be the flow after this augmentation.
- Let $v$ be the vertex with minimum value of $d_{f'}(s, v)$ whose shortest distance was reduced.
- Let $u$ be the vertex just before $v$ in the shortest path from $s$ to $v$ in $G_{f'}$.
- Claim 1.1: $d_{f'}(s, u) = d_{f'}(s, v) - 1$.
- Claim 1.2: $d_{f'}(s, u) \geq d_f(s, u)$.

- Let $d_f(s, v)$ denote the hop-length of the shortest path from $s$ to $v$ in $G_f$.
- Claim 1: For all $v \neq s, t$, $d_f(s, v)$ either remains same or increases with each flow augmentation.

**Proof of claim 1**

- Let $f$ be the flow just before the first augmentation that decreases the shortest distance of some vertex. Let $f'$ be the flow after this augmentation.
- Let $v$ be the vertex with minimum value of $d_{f'}(s, v)$ whose shortest distance was reduced.
- Let $u$ be the vertex just before $v$ in the shortest path from $s$ to $v$ in $G_{f'}$.
- Claim 1.1: $d_{f'}(s, u) = d_{f'}(s, v) - 1$.
- Claim 1.2: $d_{f'}(s, u) \geq d_f(s, u)$.
- Claim 1.3: Edge $(u, v)$ is not present in $G_f$.

- Let $d_f(s, v)$ denote the hop-length of the shortest path from $s$ to $v$ in $G_f$.
- Claim 1: For all $v \neq s, t$, $d_f(s, v)$ either remains same or increases with each flow augmentation.

## Proof of claim 1

- Let $f$ be the flow just before the first augmentation that decreases the shortest distance of some vertex. Let $f'$ be the flow after this augmentation.
- Let $v$ be the vertex with minimum value of $d_{f'}(s, v)$ whose shortest distance was reduced.
- Let $u$ be the vertex just before $v$ in the shortest path from $s$ to $v$ in $G_{f'}$.
- Claim 1.1: $d_{f'}(s, u) = d_{f'}(s, v) - 1$.
- Claim 1.2: $d_{f'}(s, u) \geq d_f(s, u)$.
- Claim 1.3: Edge $(u, v)$ is not present in $G_f$.
    - Since otherwise, $d_f(s, v) \leq d_f(s, u) + 1 \leq d_{f'}(s, u) + 1 = d_{f'}(s, v)$.

- Let $d_f(s, v)$ denote the hop-length of the shortest path from $s$ to $v$ in $G_f$.
- Claim 1: For all $v \neq s, t$, $d_f(s, v)$ either remains same or increases with each flow augmentation.

**Proof of claim 1**

- Let $f$ be the flow just before the first augmentation that decreases the shortest distance of some vertex. Let $f'$ be the flow after this augmentation.
- Let $v$ be the vertex with minimum value of $d_{f'}(s, v)$ whose shortest distance was reduced.
- Let $u$ be the vertex just before $v$ in the shortest path from $s$ to $v$ in $G_{f'}$.
- Claim 1.1: $d_{f'}(s, u) = d_{f'}(s, v) - 1$.
- Claim 1.2: $d_{f'}(s, u) \geq d_f(s, u)$.
- Claim 1.3: Edge $(u, v)$ is not present in $G_f$.
- Claim 1.3 implies that $u \neq s$ since otherwise $(u, v)$ cannot be present in $G_{f'}$.
- Claim 1.3 also implies that $(v, u)$ was in the augmenting path implying: $d_f(s, v) = d_f(s, u) - 1 \leq d_{f'}(s, u) - 1 \leq d_{f'}(s, v) - 2$, which is a contradiction. $\square$

- Claim 2: The number of flow augmentations in the `Edmonds-Karp` algorithm is $O(nm)$.

## Proof of claim 2

- An edge is said to be critical while augmentation if it is the *bottleneck* edge.
- Claim 2.1: Any edge can become critical at most $n/2$ times.

- Claim 2: The number of flow augmentations in the `Edmonds-Karp` algorithm is $O(nm)$.

### Proof of claim 2

- An edge is said to be critical while augmentation if it is the *bottleneck* edge.
- Claim 2.1: Any edge can become critical at most $n/2$ times.

### Proof of claim 2.1

- Consider any edge $(u, v)$. Let $f$ be the flow just before $(u, v)$ becomes critical. Then we have $d_f(s, v) = d_f(s, u) + 1$.
- After this, the edge $(u, v)$ disappears. Let $f'$ be the flow just before the augmentation that brings back edge $(u, v)$. Then we have $d_{f'}(s, u) = d_{f'}(s, v) + 1$.

- Claim 2: The number of flow augmentations in the `Edmonds-Karp` algorithm is $O(nm)$.

### Proof of claim 2

- An edge is said to be critical while augmentation if it is the *bottleneck* edge.
- Claim 2.1: Any edge can become critical at most $n/2$ times.

### Proof of claim 2.1

- Consider any edge $(u, v)$. Let $f$ be the flow just before $(u, v)$ becomes critical. Then we have $d_f(s, v) = d_f(s, u) + 1$.
- After this, the edge $(u, v)$ disappears. Let $f'$ be the flow just before the augmentation that brings back edge $(u, v)$. Then we have $d_{f'}(s, u) = d_{f'}(s, v) + 1$.
- Combining the above two we get: $d_{f'}(s, u) = d_{f'}(s, v) + 1 \geq d_f(s, v) + 1 = d_f(s, u) + 2$.
- So, the shortest distance has increased by at least 2 between the instances when $(u, v)$ becomes critical. $\square$

$\square$

### Algorithm

`Edmonds-Karp`

- Start with an $s-t$ flow such that for all $e$, $f(e) = 0$
- While there is an $s-t$ path $P$ in $G_f$
  - Find an $s-t$ path in $G_f$ with least hop-length
  - Augment flow along an augmenting path and let $f'$ be the resulting flow
  - Update $f$ to $f'$ and $G_f$ to $G_{f'}$
- return($f$)

- The running time of `Edmonds-Karp` algorithm is $O(nm^2)$.

# Applications of Network Flow

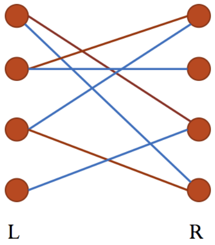
## Definition (Matching in bipartite graphs)

A subset $M$ of edges such that each node appears in at most one edge in $M$.

## Problem

Given a bipartite graph $G = (L, R, E)$, design an algorithm to give a maximum matching in the graph.

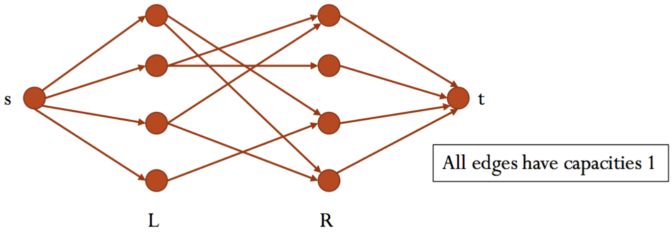
- Example:



**Problem**

Given a bipartite graph $G = (L, R, E)$, design an algorithm to give a maximum matching in the graph.

- Consider the network graph below constructed from the bipartite graph.

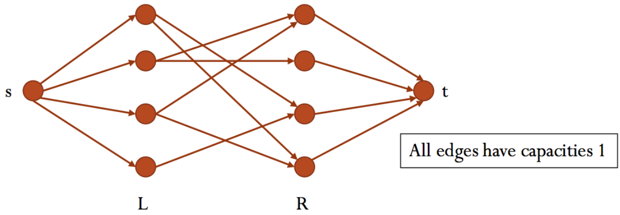



- Claim 1: Suppose there is an integer flow of value $k$ in the network graph. Then the bipartite graph has a matching of size $k$.

**Problem**

Given a bipartite graph $G = (L, R, E)$, design an algorithm to give a maximum matching in the graph.

- Consider the network graph below constructed from the bipartite graph.




- Claim 1: Suppose there is an integer flow of value $k$ in the network graph. Then the bipartite graph has a matching of size $k$.
  Claim 2: Suppose the bipartite graph has a matching of size $k$. Then there is an integer flow of value $k$ in the network graph.

End