CSL 356: Analysis and Design of Algorithms

Ragesh Jaiswal

CSE, IIT Delhi

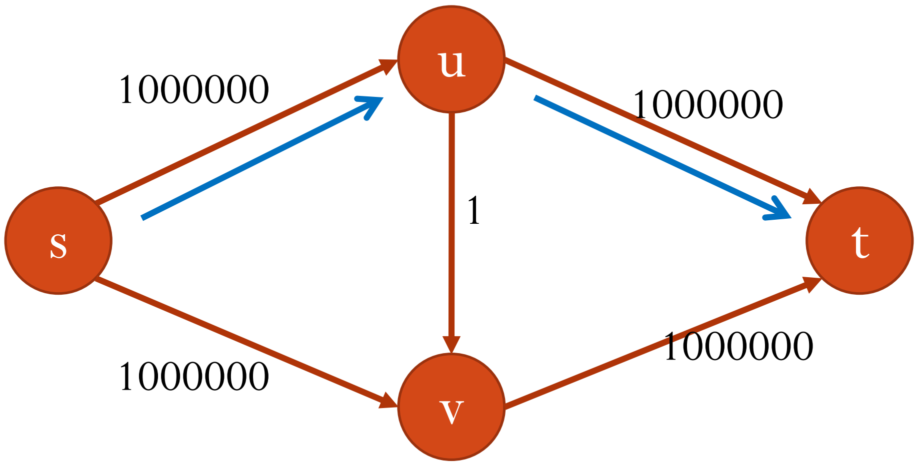
Network Flow

Strongly polynomial time algorithm for max-flow

Network Flow: Edmonds-Karp

Max-Flow *//Edmonds-Karp algorithm*

- Start with a flow f such that $f(e) = 0$
- while there is an $s - t$ path P in G_f
  - Find an $s - t$ path with **least hop-length**
    - Execute the augmenting path algorithm to obtain f'
    - Update f to f' and G_f to $G_{f'}$
- return f



Network Flow: Edmonds-Karp

- 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.

Network Flow: Edmonds-Karp

- 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:
  - 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 path length was reduced.
  - Let u be the vertex just before **v** in the shortest path from **s** to v in $G_{f'}$.

Network Flow: Edmonds-Karp

- Claim 1: For all $v \neq s, t$, $d_f(s, v)$ either remains same or increases with each flow augmentation.
- Proof:
  - 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 path length was reduced.
  - Let u be the vertex just before v in the shortest path from s to v in $G_{f'}$.
  - We have:
    - $d_{f'}(s, u) = d_{f'}(s, v) - 1$
    - $d_{f'}(s, u) \geq d_f(s, u)$
  - Claim: (u, v) is not present in G_f.
  - Proof: Since otherwise,

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

Network Flow: Edmonds-Karp

- Claim 1: For all $v \neq s, t$, $d_f(s, v)$ either remains same or increases with each flow augmentation.
- Proof:
  - 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 path length was reduced.
  - Let u be the vertex just before v in the shortest path from s to v in $G_{f'}$.
  - We have:
    - $d_{f'}(s, u) \;=\; d_{f'}(s, v) - 1$
    - $d_{f'}(s, u) \;\geq\; d_f(s, u)$
  - Claim: (u, v) is not present in G_f.
  - Proof: Since otherwise,

    $$d_f(s, v) \;\leq\; d_f(s, u) + 1 \;\leq\; d_{f'}(s, u) + 1 \;=\; d_{f'}(s, v).$$
  - This means that (v, u) was in the augmenting path. This means: $d_f(s, v) \;= d_f(s, u) \;-\; 1 \;\leq\; d_{f'}(s, u) - 1 \;\leq\; d_{f'}(s, v) - 2$

Network Flow: Edmonds-Karp

- Claim 2: The total number of flow augmentations in the Edmonds–Karp algorithm is $O(nm)$.
- Proof:
  - An edge is said to be critical while augmentation if it is the bottleneck edge.
  - Claim: Any edge can become critical at most $(n/2)$ times.
  - Proof:
    - Consider any edge (u, v). Let f be the flow just before (u, v) becomes critical. The we have

$$d_f(s, v) = d_f(s, u) + 1 \qquad (1)$$

    - 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

$$d_{f'}(s, u) = d_{f'}(s, v) + 1 \qquad (2)$$

Network Flow: Edmonds-Karp

- Claim 2: The total number of flow augmentations in the Edmonds–Karp algorithm is $O(nm)$.
- Proof:
  - An edge is said to be critical while augmentation if it is the bottleneck edge.
  - Claim: Any edge can become critical at most $(n/2)$ times.
  - Proof:
    - Consider any edge (u, v). Let f be the flow just before (u, v) becomes critical. The we have
      $$d_f(s, v) = d_f(s, u) + 1 \qquad (1)$$
    - 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
      $$d_{f'}(s, u) = d_{f'}(s, v) + 1 \qquad (2)$$
    - Using (1) and (2) we get:
      $$d_{f'}(s, u) = d_{f'}(s, v) + 1 \geq d_f(s, v) + 1 = d_f(s, u) + 2$$
    - The shortest distance has increased by 2 between the instances when (u, v) becomes critical.

Network Flow: Edmonds-Karp

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

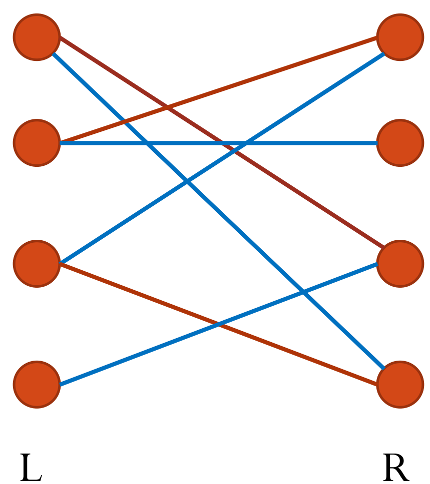
- Theorem: The running time of Edmonds-Karp algorithm is $O(nm^2)$.

Applications of Network Flow

Bipartite Matching

Bipartite Matching

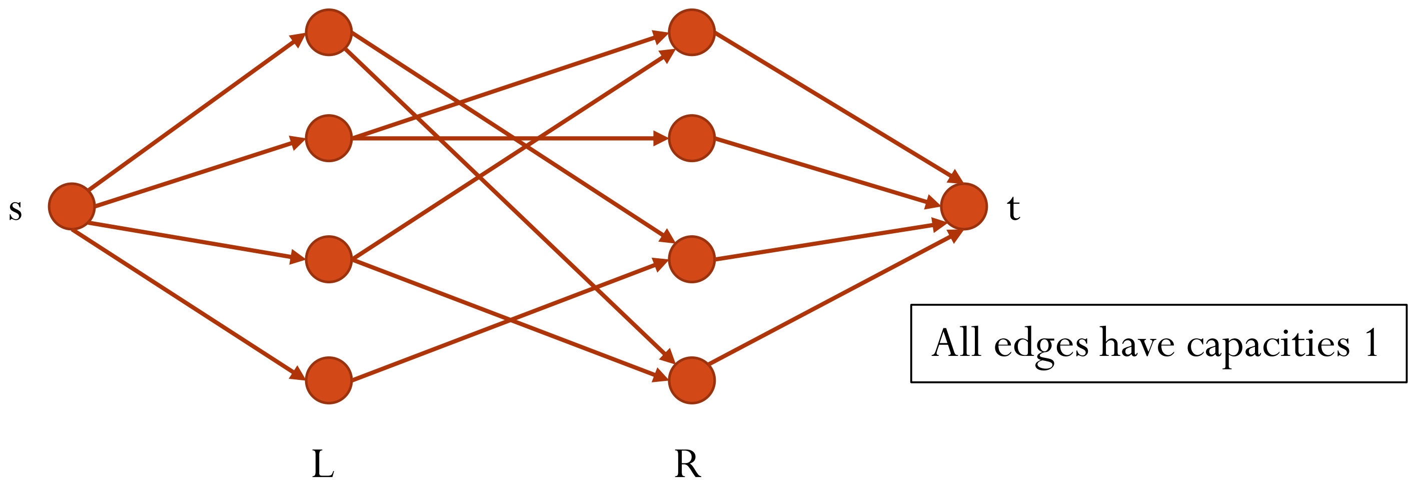
- <u>Problem(Bipartite matching)</u>: Given a bipartite graph $G = (L, R, E)$ give a maximum *matching* in the graph.
  - <u>Example</u>:



- <u>*Matching*</u>: A subset M of edges such that each node appears in at most one edge in M.

Bipartite Matching

- <u>Problem(Bipartite matching)</u>: Given a bipartite graph $G = (L, R, E)$ give a maximum *matching* in the graph.
  - <u>Example</u>:



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

Network Flow: Applications

Hall's Theorem

Network Flow: Applications

- Given a bipartite graph $G = (X, Y, E)$, consider the matching problem. We say that G has *perfect matching* if all vertices are present in a matching.

Network Flow: Applications

- Given a bipartite graph $G = (X, Y, E)$, consider the matching problem. We say that G has *perfect matching* if all vertices are present in a matching.
  - $|X| = |Y|$

Network Flow: Applications

- Given a bipartite graph $G = (X, Y, E)$, consider the matching problem. We say that G has *perfect matching* if all vertices are present in a matching.
  - $|X| = |Y|$
  - For a subset A of X let $N(A)$ denote the neighboring vertices of A in G. There is no perfect matching if there is an A, $|A| > |N(A)|$.
    - Is the converse also true?

Network Flow: Applications

- Given a bipartite graph $G = (X, Y, E)$, consider the matching problem. We say that G has *perfect matching* if all vertices are present in a matching.
  - $|X| = |Y|$
  - For a subset A of X let $N(A)$ denote the neighboring vertices of A in G. There is no perfect matching if there is an A, $|A| > |N(A)|$.
    - Is the converse also true?

- <u>Hall's Theorem</u>: Given any bipartite graph $G = (X, Y, E)$, there is a perfect matching in G if and only if for every subset A of vertices of X, we have $|A| \leq |N(A)|$.

End