

COL351: Analysis and Design of Algorithms

Ragesh Jaiswal, CSE, IITD

Network Flow

Maximum flow

Algorithm

Ford-Fulkerson

- Start with a flow f such that $f(e) = 0$
- While there is an $s - t$ path P in G_f
 - Augment flow along an $s - t$ path and let f' be resulting flow
 - Update f to f' and G_f to $G_{f'}$
- return(f)

- How do we prove that the flow returned by the Ford-Fulkerson algorithm is the maximum flow?

Network Flow

Maximum flow

- Theorem 1: Let f be the flow returned by the Ford-Fulkerson algorithm. Then f maximizes $v(f) = \sum_{e \text{ out of } s} f(e)$.

Definition (f^{in} and f^{out})

Let S be a subset of vertices and f be a flow. Then

$$f^{in}(S) = \sum_{e \text{ into } S} f(e) \quad \text{and} \quad f^{out}(S) = \sum_{e \text{ out of } S} f(e)$$

Network Flow

Maximum flow

- Theorem 1: Let f be the flow returned by the Ford-Fulkerson algorithm. Then f maximizes $v(f) = \sum_{e \text{ out of } s} f(e)$.

Definition (f^{in} and f^{out})

Let S be a subset of vertices and f be a flow. Then

$$f^{in}(S) = \sum_{e \text{ into } S} f(e) \quad \text{and} \quad f^{out}(S) = \sum_{e \text{ out of } S} f(e)$$

Definition ($s - t$ cut)

A partition of vertices (A, B) is called an $s - t$ cut iff A contains s and B contains t .

Definition (Capacity of $s - t$ cut)

The capacity of an $s - t$ cut (A, B) is defined as

$$C(A, B) = \sum_{e \text{ out of } A} c(e).$$

Network Flow

Maximum flow

- Theorem 1: Let f be the flow returned by the Ford-Fulkerson algorithm. Then f maximizes $v(f) = \sum_{e \text{ out of } s} f(e)$.

Proof

- Claim 1.1: For any $s - t$ cut (A, B) and any $s - t$ flow f , $v(f) = f^{out}(A) - f^{in}(A)$.

Network Flow

Maximum flow

- Theorem 1: Let f be the flow returned by the Ford-Fulkerson algorithm. Then f maximizes $v(f) = \sum_{e \text{ out of } s} f(e)$.

Proof

- Claim 1.1: For any $s - t$ cut (A, B) and any $s - t$ flow f , $v(f) = f^{out}(A) - f^{in}(A)$.

Proof of claim 1.1.

$v(f) = f^{out}(\{s\}) - f^{in}(\{s\})$ and for all other nodes $v \in A$, $f^{out}(\{v\}) - f^{in}(\{v\}) = 0$. So,

$$v(f) = \sum_{v \in A} (f^{out}(\{v\}) - f^{in}(\{v\})) = f^{out}(A) - f^{in}(A).$$



Network Flow

Maximum flow

- Theorem 1: Let f be the flow returned by the Ford-Fulkerson algorithm. Then f maximizes $v(f) = \sum_{e \text{ out of } s} f(e)$.

Proof

- Claim 1.1: For any s - t cut (A, B) and any s - t flow f , $v(f) = f^{out}(A) - f^{in}(A)$.
- Claim 1.2: Let f be any s - t flow and (A, B) be any s - t cut. Then $v(f) \leq C(A, B)$.

Network Flow

Maximum flow

- Theorem 1: Let f be the flow returned by the Ford-Fulkerson algorithm. Then f maximizes $v(f) = \sum_{e \text{ out of } s} f(e)$.

Proof

- Claim 1.1: For any s - t cut (A, B) and any s - t flow f ,
 $v(f) = f^{out}(A) - f^{in}(A)$.
- Claim 1.2: Let f be any s - t flow and (A, B) be any s - t cut. Then
 $v(f) \leq C(A, B)$.

Proof of claim 1.2.

$$v(f) = f^{out}(A) - f^{in}(A) \leq f^{out}(A) \leq C(A, B). \quad \square$$

Network Flow

Maximum flow

- Theorem 1: Let f be the flow returned by the Ford-Fulkerson algorithm. Then f maximizes $v(f) = \sum_{e \text{ out of } s} f(e)$.

Proof

- Claim 1.1: For any s - t cut (A, B) and any s - t flow f ,
 $v(f) = f^{out}(A) - f^{in}(A)$.
- Claim 1.2: Let f be any s - t flow and (A, B) be any s - t cut. Then
 $v(f) \leq C(A, B)$.
- Claim 1.3: Let f be an s - t flow such that there is no s - t path in G_f . Then there is an s - t cut (A^*, B^*) such that
 $v(f) = C(A^*, B^*)$. Furthermore, f is a flow with maximum value and (A^*, B^*) is an s - t cut with minimum capacity.



Network Flow

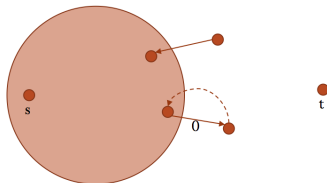
Maximum flow

- Claim 1.3: Let f be an s - t flow such that there is no s - t path in G_f . Then there is an s - t cut (A^*, B^*) such that $v(f) = C(A^*, B^*)$. Furthermore, f is a flow with maximum value and (A^*, B^*) is an s - t cut with minimum capacity.

Proof of claim 1.3

- Let A^* be all vertices reachable from s in the graph G_f (see figure below). Then we have:

$$\begin{aligned}v(f) &= f^{out}(A^*) - f^{in}(A^*) \\ &= f^{out}(A^*) - 0 \\ &= C(A^*, B^*)\end{aligned}$$



A^* (all vertices reachable from s in G_f)

Network Flow

Maximum flow

Theorem (Max-flow-min-cut theorem)

In every flow network, the maximum value of s - t flow is equal to the minimum capacity of s - t cut.

- Summary:
 - Ford-Fulkerson Algorithm:
 - Given network with integer capacities, find a source-to-sink path and push as much flow along the path as possible.
 - Update the residual capacity of edges in the residual graph.
 - Repeat.
 - Proof of correctness:
 - The algorithm terminates (since the capacities are integers).
 - Max-flow-min-cut theorem: In every flow network, the maximum value of s - t flow is equal to the minimum capacity of s - t cut.

Network Flow

Maximum flow

- Summary:
 - Ford-Fulkerson Algorithm:
 - Given network with integer capacities, find a source-to-sink path and push as much flow along the path as possible.
 - Update the residual capacity of edges in the residual graph.
 - Repeat.
 - Proof of correctness:
 - **The algorithm terminates (since the capacities are integers).**
 - Max-flow-min-cut theorem: In every flow network, the maximum value of s - t flow is equal to the minimum capacity of s - t cut.
- What if the capacities are not integers? Does the algorithm terminate?

Network Flow

Maximum flow

- Summary:
 - Ford-Fulkerson Algorithm:
 - Given network with integer capacities, find a source-to-sink path and push as much flow along the path as possible.
 - Update the residual capacity of edges in the residual graph.
 - Repeat.
 - Proof of correctness:
 - **The algorithm terminates (since the capacities are integers).**
 - Max-flow-min-cut theorem: In every flow network, the maximum value of s - t flow is equal to the minimum capacity of s - t cut.
- What if the capacities are not integers? Does the algorithm terminate?
 - There is a network where the edges have non-integer capacities where the Ford-Fulkerson algorithm does not terminate.

Applications of Network Flow

Network Flow

Bipartite Matching

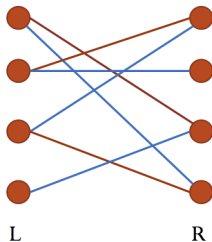
Definition (Matching in bipartite graphs)

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.

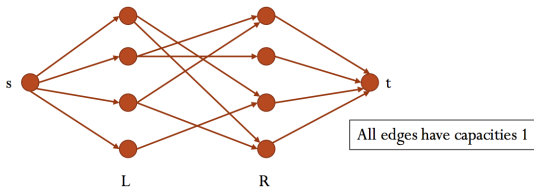
- Example:



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.

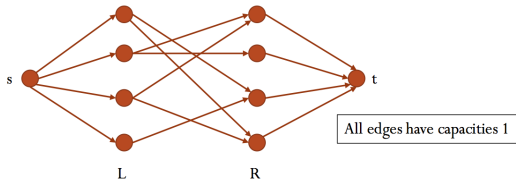


- Claim 1: Suppose there is an integer flow of value k in the network graph. Then the bipartite graph has a matching of size k .

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 .
 - Consider those bipartite edges along which the flow is 1. Argue that due to flow conservation these edges form a matching.

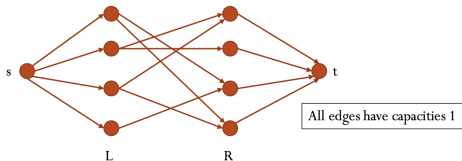
Network Flow

Bipartite Matching

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 .
 - Consider those bipartite edges along which the flow is 1. Argue that due to flow conservation these edges form a matching.
- 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.

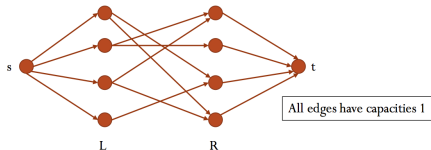
Network Flow

Bipartite Matching

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 .
 - Consider those bipartite edges along which the flow is 1. Argue that due to flow conservation these edges form a matching.
- 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.
 - Consider the flow where the flow along the edges in the matching is 1.

Network Flow

Bipartite Matching

Problem

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

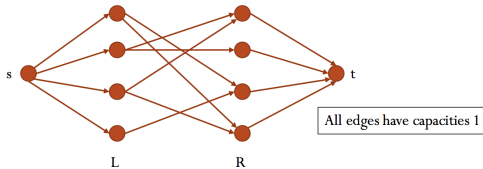


Figure: Network construction from Bipartite graph

Algorithm

Max-Matching(G)

- Construct the network G' using G as shown in Figure
- Execute the Ford-Fulkerson algorithm on G' to obtain flow f
- Let M be all bipartite edges with flow value 1
- return(M)

Network Flow

Bipartite Matching

Problem

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

Algorithm

Max-Matching(G)

- Construct the network G' using G as shown in Figure
- Execute the Ford-Fulkerson algorithm on G' to obtain flow f
- Let M be all bipartite edges with flow value 1
- return(M)

- What is the running time of the above algorithm?

Problem

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

Algorithm

Max-Matching(G)

- Construct the network G' using G as shown in Figure
- Execute the Ford-Fulkerson algorithm on G' to obtain flow f
- Let M be all bipartite edges with flow value 1
- return(M)

- What is the running time of the above algorithm? $O((m+n) \cdot n)$
 - Creating G' takes $O(m+n)$ time.
 - Running the Ford-Fulkerson algorithm on G' takes $O(n \cdot (m+n))$ time since the capacity of the cut (A, B) in G' where $A = \{s\}$, is n .

End