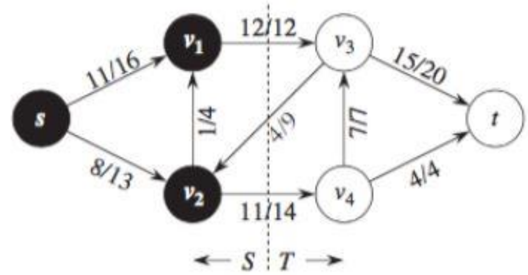
- **Network:**
  - A directed graph where each edge has its weight represented in the form flow/capacity.
  - For any node, Σ incoming flow = Σ outcoming flow.
- **Single-Source Max Flow problem:**
  - Find the maximum amount of flow that can pass through the network at a time.
    - The path goes from a special node that has no incoming edges called the source.
    - To another special node that has no outcoming edges called the sink/target.
  - **Multiple-Source** variant can be transformed to the single source problem:
    - Create a super-source node and add edges with infinite capacity from it to all starting nodes.
    - Create a super-sink node and add edges with infinite capacity all sink nodes to it.
  - **Applications:**
    - Shipping products, Bipartite matching, Image segmentation.
- **Ford-Fulkerson Algorithm:**

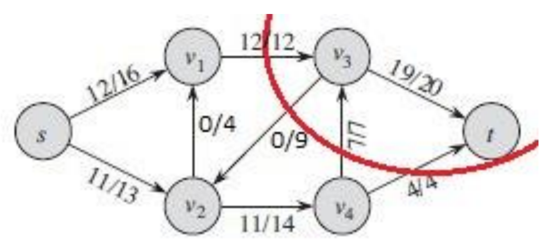
```
1. Begin with flow = 0.
2. While there is an augmenting path in the residual network
3.         Augment the flow through that path.
```

  - **Augmenting path:**
    - Any path (not necessarily directed) from source to sink such that it doesn't pass through a full forward edge ($f = c$), or an empty backward edge ($f = 0$).
    - A directed path through the residual network, that can go through any edge with weight > 0.
  - **Residual network:** the same original network graph, replacing each edge (u, v) with weight f/c with two edges:
    - The edge (u, v) with weight $c - f$.
      - Represents the free space, how much flow can we add here.
    - The edge (v, u) with weight f.
      - Represents how much flow is currently passing that can we decrease.
  - How to **augment the flow** through a path?
    - Calculate the bottleneck value 'm' of the path (minimum-weight edge in the path).
    - For each edge (u, v) in the path
      - Weight on edge (u, v) - = m
      - Weight on edge (v, u) += m
      - If any edge weight became 0, delete it (it can't be used in any path).
  - How to find an augmenting path?
    - **Ford-Fulkerson:** doesn't specify search algorithm
      - Can DFS the residual network.
      - **Time complexity:** $O(|E|.|f^*|)$, f^* is the max flow value.
    - **Edmonds-Karp:** (more efficient implementation): BFS the residual network.
      - **Time complexity:** $O(|V|.|E|^2)$

- **Max-Flow, Min cut theorem:**
    - **Cut (in a network):**
        - Dividing the network into two disjoint sets (S, T), by removing one or more edges.
        - Separate the edges such that the sink is no longer reachable from the source.
    - **Minimum cut:**
        - The cut with minimum **capacity**.
        - Cut capacity: upper bound for the flow that can go through the cut.
            - Σ forward (S → T) flow for all removed edges.
    - **Net flow across a cut:** The total amount of flow that can pass if this cut wasn't there
        - Σ forward (S → T) - Σ backward (S ← T) flow.
    - **Surprisingly,**
        - Net flow across any cut in a network = Value of its flow ≤ Capacity of any cut.
        - Value of max flow = Capacity of min cut.
- **Example:** A flow network with a random cut.
    - No more augmented paths.
    - Net flow = 11 + 8 = 15 + 4 = 12 + 11 – 4 = 19
    - Cut capacity = 12 + 14 = 26



- **However, this situation is not optimal and cannot be reached by the regular FF algorithm.**



- **The correct solution for this graph:**
    - The cut is circled in red.
    - Max flow value = Min cut capacity = 23

- **Visualizer:** https://visualgo.net/en/maxflow