Network Flows

Madhavan Mukund

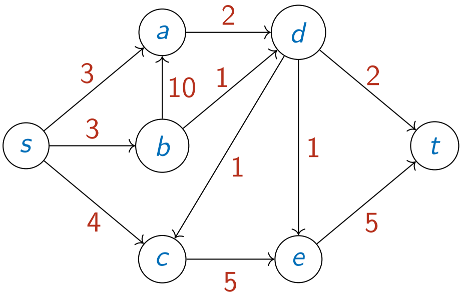
https://www.cmi.ac.in/~madhavan

Programming, Data Structures and Algorithms using Python

Week 11

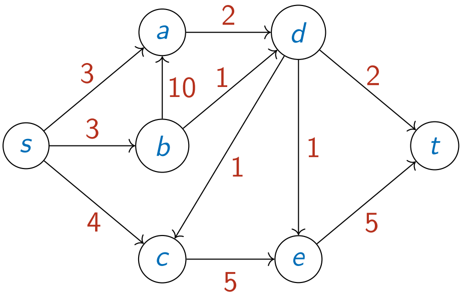
Oil network

- Network of pipelines



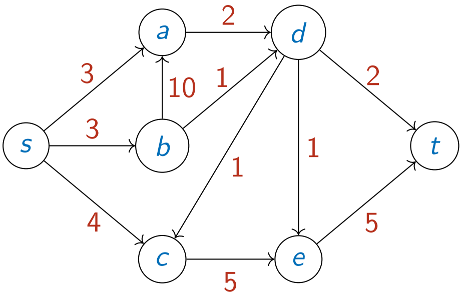
Oil network

- Network of pipelines
- Ship as much oil as possible from s to t



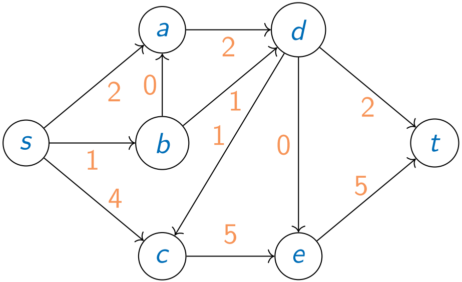
Oil network

- Network of pipelines
- Ship as much oil as possible from s to t
- No storage along the way



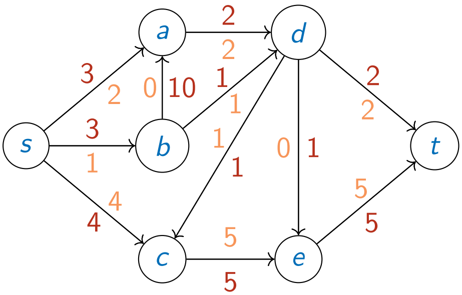
Oil network

- Network of pipelines
- Ship as much oil as possible from s to t
- No storage along the way
- A flow of 7 is possible



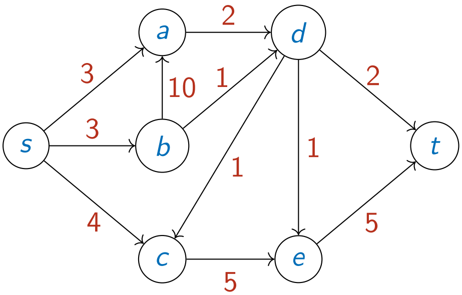
Oil network

- Network of pipelines
- Ship as much oil as possible from s to t
- No storage along the way
- A flow of 7 is possible
- Is this the maximum?



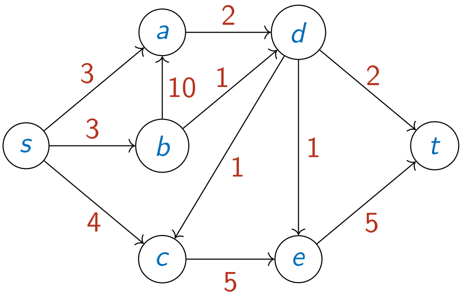
Oil network

- Network: graph $G = (V, E)$
- Special nodes: s (source), t (sink)



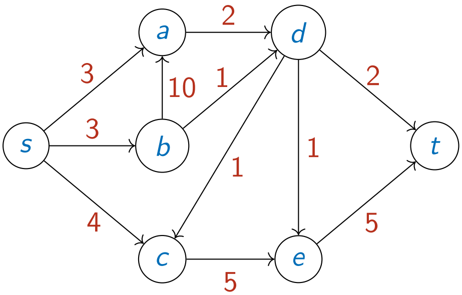
Oil network

- Network: graph $G = (V, E)$
- Special nodes: s (source), t (sink)
- Each edge e has capacity c_e



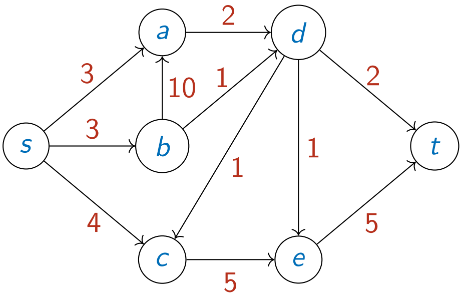
Oil network

- Network: graph $G = (V, E)$
- Special nodes: s (source), t (sink)
- Each edge e has capacity c_e
- Flow: f_e for each edge e
  - $f_e \leq c_e$
  - At each node, except s and t, sum of incoming flows equal sum of outgoing flows



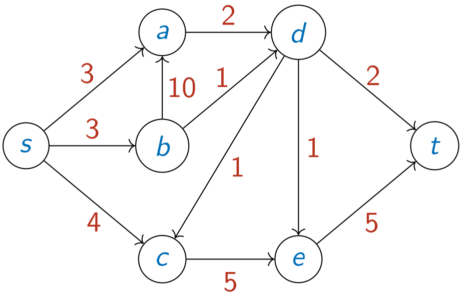
Oil network

- Network: graph $G = (V, E)$
- Special nodes: s (source), t (sink)
- Each edge e has capacity c_e
- Flow: f_e for each edge e
  - $f_e \leq c_e$
  - At each node, except s and t, sum of incoming flows equal sum of outgoing flows
- Total volume of flow is sum of outgoing flow from s



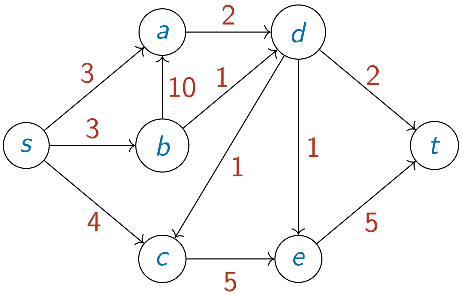
LP formulation

- Variable f_e for each edge e
  - f_{sa}, f_{bd}, f_{ce}, ...



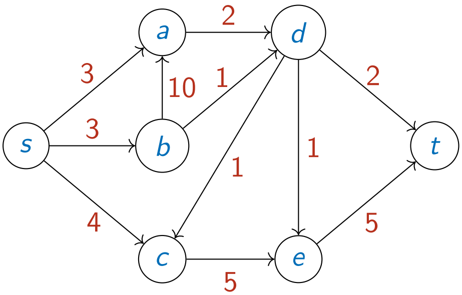
LP formulation

- Variable f_e for each edge e
  - f_{sa}, f_{bd}, f_{ce}, ...
- Capacity constraints per edge
  - $f_{ba} \leq 10$, ...



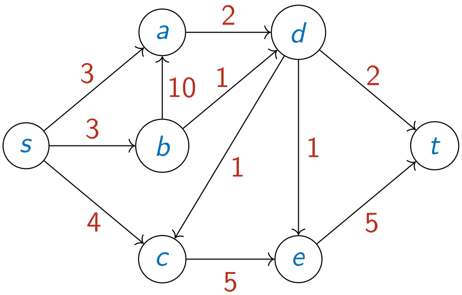
LP formulation

- Variable f_e for each edge e
  - $f_{sa}, f_{bd}, f_{ce}, \ldots$
- Capacity constraints per edge
  - $f_{ba} \leq 10, \ldots$
- Conservation of flow at each internal node
  - $f_{ad} + f_{bd} = f_{dc} + f_{de} + f_{dt}, \ldots$



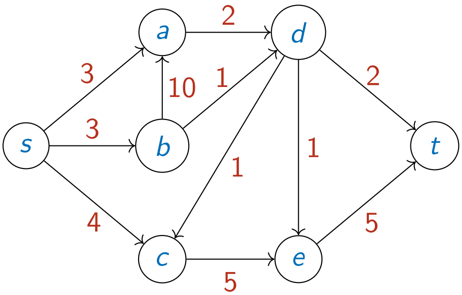
LP formulation

- Variable f_e for each edge e
  - $f_{sa}, f_{bd}, f_{ce}, \ldots$
- Capacity constraints per edge
  - $f_{ba} \leq 10, \ldots$
- Conservation of flow at each internal node
  - $f_{ad} + f_{bd} = f_{dc} + f_{de} + f_{dt}, \ldots$
- Objective: maximize flow volume
  - Maximize $f_{sa} + f_{sb} + f_{sc}$



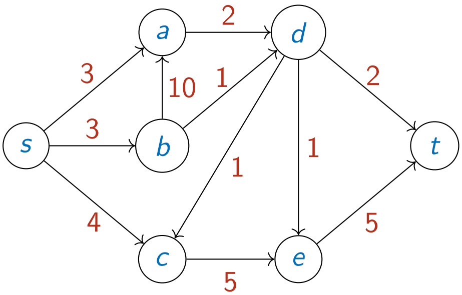
LP formulation

- Variable f_e for each edge e
  - $f_{sa}, f_{bd}, f_{ce}, \ldots$
- Capacity constraints per edge
  - $f_{ba} \leq 10, \ldots$
- Conservation of flow at each internal node
  - $f_{ad} + f_{bd} = f_{dc} + f_{de} + f_{dt}, \ldots$
- Objective: maximize flow volume
  - Maximize $f_{sa} + f_{sb} + f_{sc}$
- Simplex explores vertices of feasible region to solve LP, find maximum flow



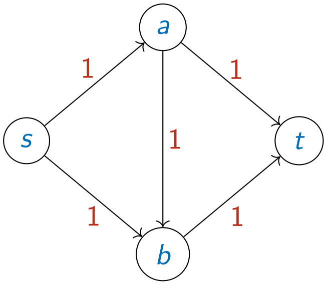
LP formulation

- Variable f_e for each edge e
  - $f_{sa}, f_{bd}, f_{ce}, \ldots$
- Capacity constraints per edge
  - $f_{ba} \leq 10, \ldots$
- Conservation of flow at each internal node
  - $f_{ad} + f_{bd} = f_{dc} + f_{de} + f_{dt}, \ldots$
- Objective: maximize flow volume
  - Maximize $f_{sa} + f_{sb} + f_{sc}$
- Simplex explores vertices of feasible region to solve LP, find maximum flow
- Moving from vertex to vertex gives a more direct algorithm for maximum flow



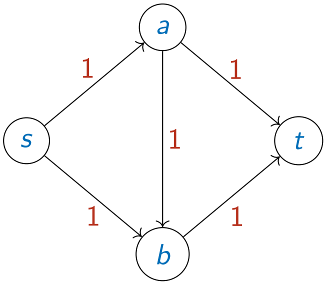
Ford-Fulkerson algorithm

- Start with zero flow



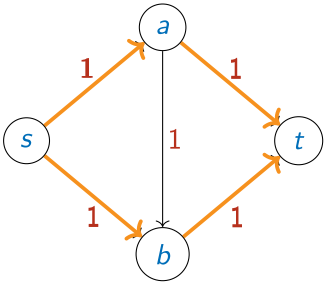
Ford-Fulkerson algorithm

- Start with zero flow
- Choose a path from s to t that is not saturated and augment the flow as much as possible



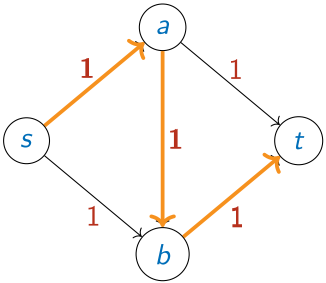
Ford-Fulkerson algorithm

- Start with zero flow
- Choose a path from s to t that is not saturated and augment the flow as much as possible
- Network on the right has max flow 2



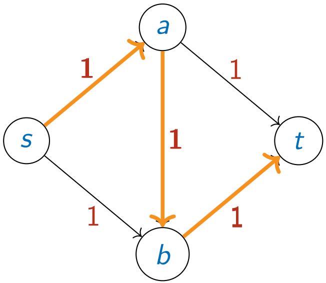
Ford-Fulkerson algorithm

- Start with zero flow
- Choose a path from s to t that is not saturated and augment the flow as much as possible
- Network on the right has max flow 2
- What if one chooses a bad flow to begin with?



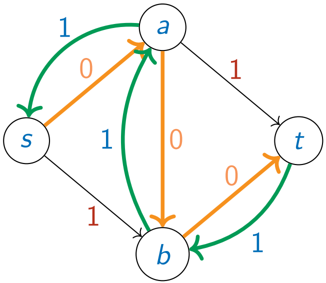
Ford-Fulkerson algorithm

- Start with zero flow
- Choose a path from s to t that is not saturated and augment the flow as much as possible
- Network on the right has max flow 2
- What if one chooses a bad flow to begin with?
- Add reverse edges to undo flow from previous steps



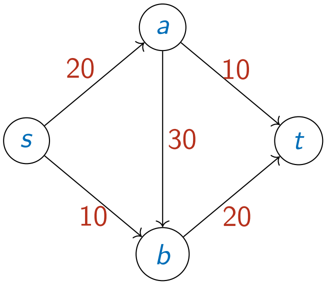
Ford-Fulkerson algorithm

- Start with zero flow
- Choose a path from s to t that is not saturated and augment the flow as much as possible
- Network on the right has max flow 2
- What if one chooses a bad flow to begin with?
- Add reverse edges to undo flow from previous steps
- Residual graph: for each edge e with capacity c_e and current flow f_e
  - Reduce capacity to $c_e - f_e$
  - Add reverse edge with capacity f_e



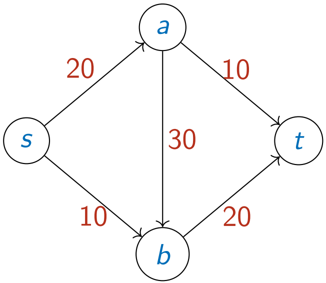
Ford-Fulkerson algorithm

- Start with zero flow



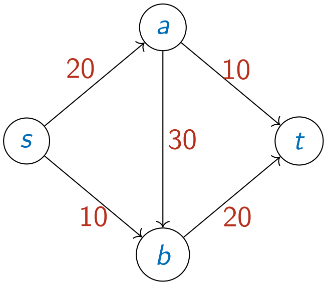
Ford-Fulkerson algorithm

- Start with zero flow
- Choose a path from s to t that is not saturated and augment the flow as much as possible



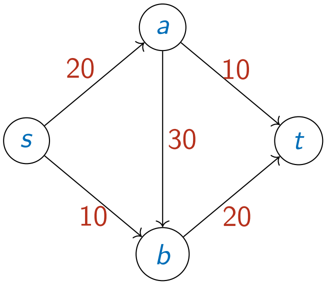
Ford-Fulkerson algorithm

- Start with zero flow
- Choose a path from s to t that is not saturated and augment the flow as much as possible
- Build residual graph



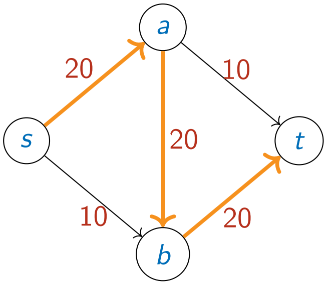
Ford-Fulkerson algorithm

- Start with zero flow
- Choose a path from s to t that is not saturated and augment the flow as much as possible
- Build residual graph
- Repeat the previous two steps till there is no feasible flow from s to t

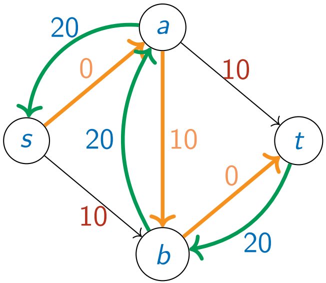


Ford-Fulkerson algorithm

- Start with zero flow
- Choose a path from s to t that is not saturated and augment the flow as much as possible
- Build residual graph
- Repeat the previous two steps till there is no feasible flow from s to t
- Flow 20, $s - a - b - t$,

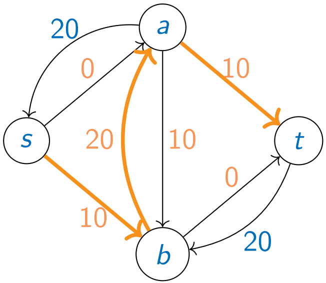


- Start with zero flow
- Choose a path from s to t that is not saturated and augment the flow as much as possible
- Build residual graph
- Repeat the previous two steps till there is no feasible flow from s to t
- Flow 20, $s - a - b - t$, build residual graph



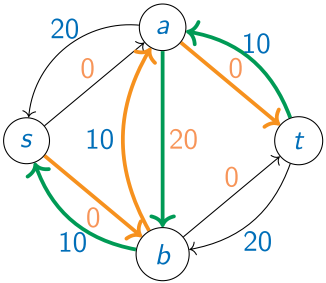
Ford-Fulkerson algorithm

- Start with zero flow
- Choose a path from s to t that is not saturated and augment the flow as much as possible
- Build residual graph
- Repeat the previous two steps till there is no feasible flow from s to t
- Flow 20, $s - a - b - t$, build residual graph
- Add flow 10, $s - b - a - t$,



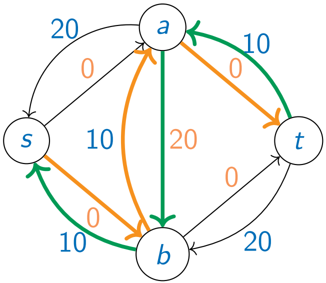
Ford-Fulkerson algorithm

- Start with zero flow
- Choose a path from s to t that is not saturated and augment the flow as much as possible
- Build residual graph
- Repeat the previous two steps till there is no feasible flow from s to t
- Flow 20, $s - a - b - t$, build residual graph
- Add flow 10, $s - b - a - t$, build residual graph



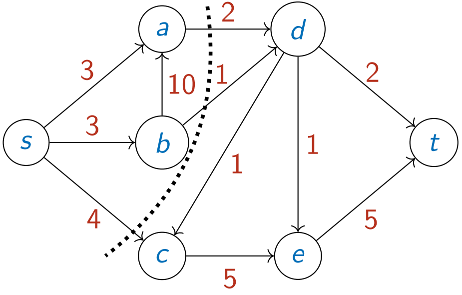
Ford-Fulkerson algorithm

- Start with zero flow
- Choose a path from s to t that is not saturated and augment the flow as much as possible
- Build residual graph
- Repeat the previous two steps till there is no feasible flow from s to t
- Flow 20, $s - a - b - t$, build residual graph
- Add flow 10, $s - b - a - t$, build residual graph
- No more feasible paths from s to t



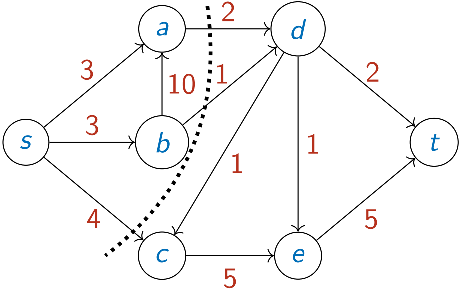
Certificate of optimality

- Edges $\{ad, bd, sc\}$ disconnect s and t
  - (s,t)-cut



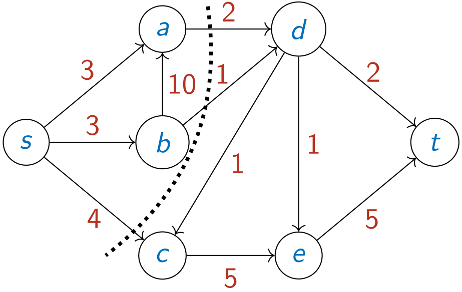
Certificate of optimality

- Edges $\{ad, bd, sc\}$ disconnect s and t
  - (s,t)-cut
- Flow from s to t must go through this cut



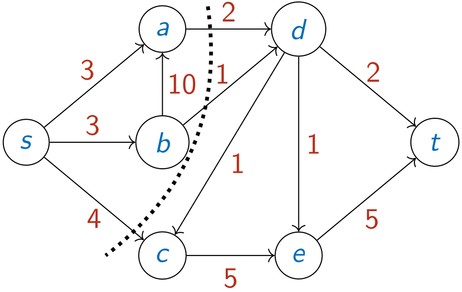
Certificate of optimality

- Edges $\{ad, bd, sc\}$ disconnect s and t
  - (s,t)-cut
- Flow from s to t must go through this cut
- Cannot exceed cut capacity, 7



Certificate of optimality

- Edges $\{ad, bd, sc\}$ disconnect s and t
  - (s, t)-cut
- Flow from s to t must go through this cut
- Cannot exceed cut capacity, 7
- Max flow cannot exceed capacity of min cut

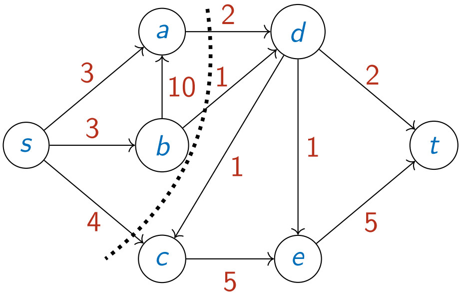


Certificate of optimality

- Edges $\{ad, bd, sc\}$ disconnect s and t
  - (s,t)-cut
- Flow from s to t must go through this cut
- Cannot exceed cut capacity, 7
- Max flow cannot exceed capacity of min cut

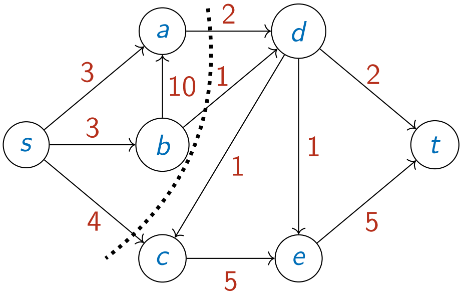
Max flow-min cut theorem

- In fact, max flow is always equal to min cut



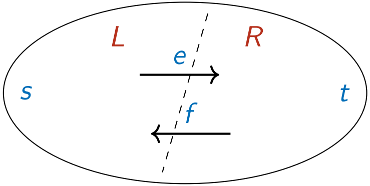
Certificate of optimality

- Edges $\{ad, bd, sc\}$ disconnect s and t
  - (s, t)-cut
- Flow from s to t must go through this cut
- Cannot exceed cut capacity, 7
- Max flow cannot exceed capacity of min cut



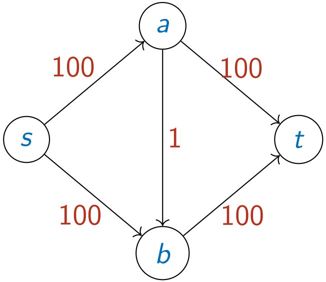
Max flow-min cut theorem

- In fact, max flow is always equal to min cut
- At max flow, no path from s to t in residual graph
  - s can reach L, R can reach t
  - Any edge from L to R must be at full capacity
  - Any edge from R to L must be at zero capacity



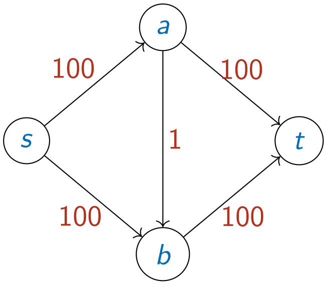
Ford-Fulkerson algorithm

- Choose augmenting paths wisely



Ford-Fulkerson algorithm

- Choose augmenting paths wisely
- If we keep going through the middle edge, 200 iterations to find the max flow
  - Ford-Fulkerson can take time proportional to max capacity



Ford-Fulkerson algorithm

- Choose augmenting paths wisely
- If we keep going through the middle edge, 200 iterations to find the max flow
  - Ford-Fulkerson can take time proportional to max capacity
- Use BFS to find augmenting path with fewest edges
- Iterations bounded by $|V| \times |E|$, regardless of capacities