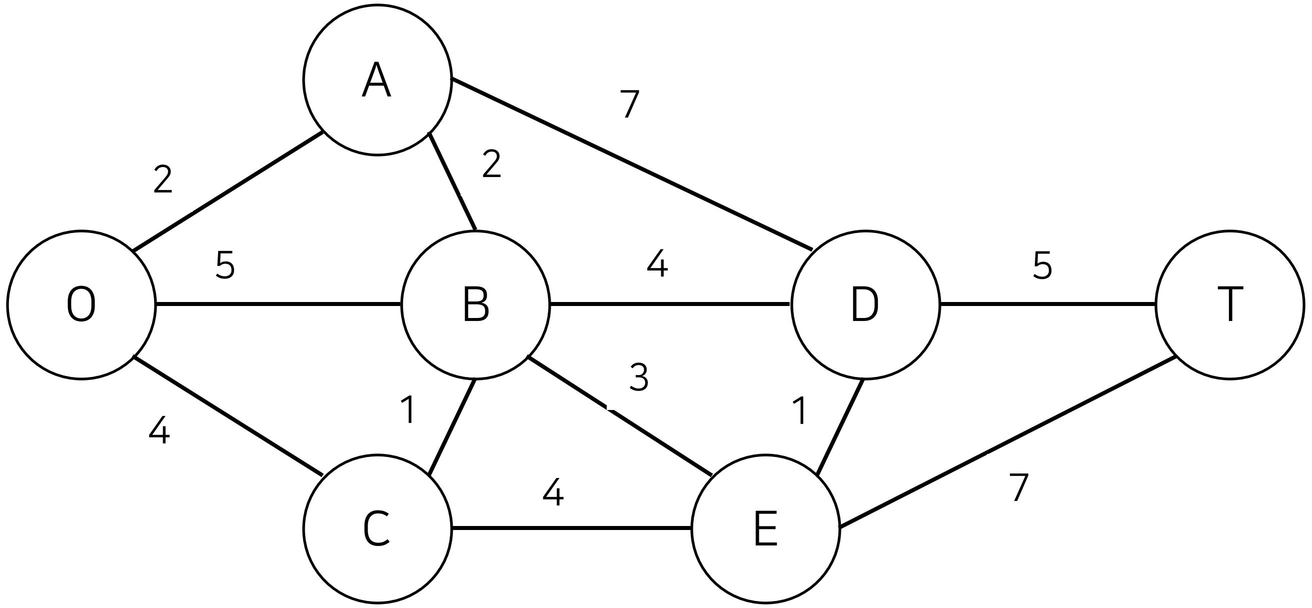
Network Optimization Models

Taek-Ho Lee
Department of Industrial Engineering, SeoulTech
Mail: taekho.lee@seoultech.ac.kr

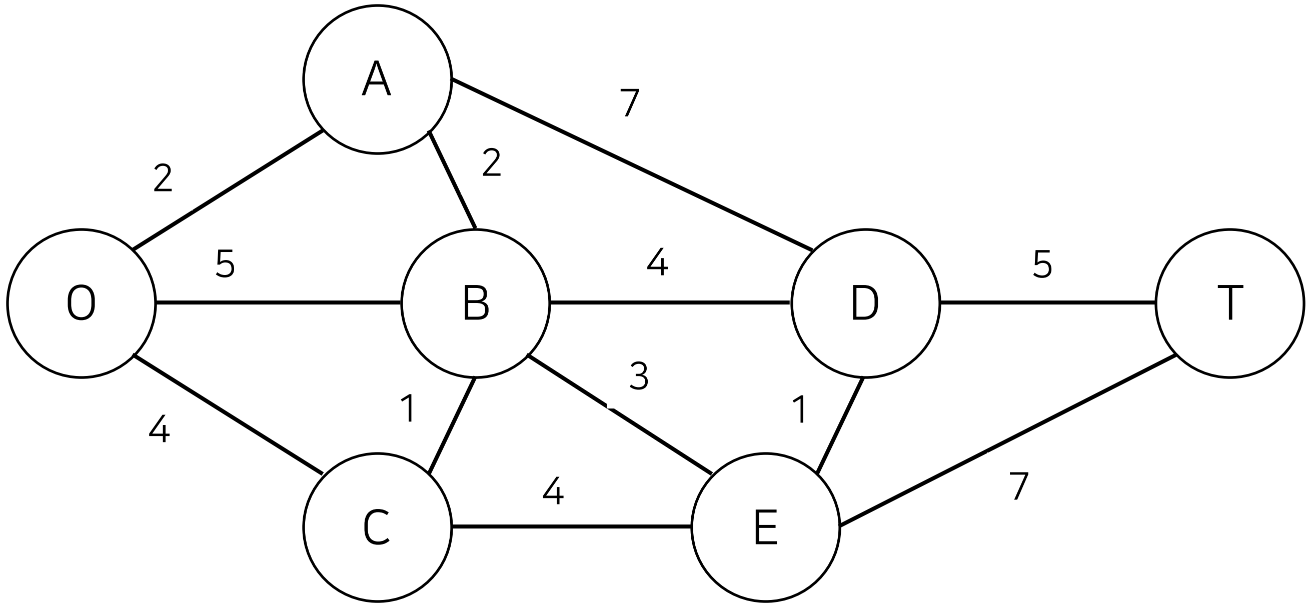
Prototype Example

- Configuration of the Park Road Network
  - Each number represents the distance of the corresponding road.



Prototype Example

- Configuration of the Park Road Network
  - (Q1) What is the shortest path from the entrance (O) to T? → Shortest path Problem
  - (Q2) When configuring a telecommunication network among nodes, what is the minimum required total length of telecommunication lines? → Minimum Spanning Tree Problem
  - (Q3) If there are limits on the number of times each segment can be used, what is the maximum number of trips possible from O to T? → Maximum Flow Problem



The Terminology of Networks

- Terminologies
  - Nodes (or Vertices)
  - Arcs (or Links, Edges, Branches)
    - ✓ Directed arc
    - ✓ Undirected arc
  - Network
    - ✓ Directed network: when the network consists only of directed arcs.
    - ✓ Undirected network: when the network consists only of undirected arcs.
  - Path: An ordered sequence of nodes and edges connecting two nodes
    - ✓ Directed path: For $i \rightarrow j$, all connecting arcs point towards node j.
    - ✓ Undirected path: A sequence of connected edges regardless of direction.

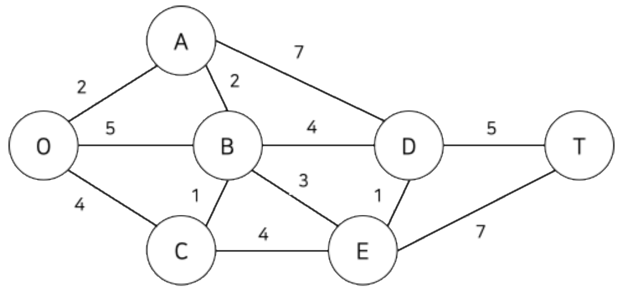
The Terminology of Networks

- Terminologies
  - Cycle: a path where the starting and ending node are the same
  - Connected: indicates that there exists a path between two nodes
    - ✓ Connected network: a network in which every pair of nodes is connected
  - Tree: a connected network without any cycle; it may not include all nodes.
    - ✓ Spanning tree: a connected network of n nodes linked by $(n-1)$ edges
  - Arc capacity: the maximum flow allowed on an arc in a flow network
  - Supply node: a node where outflow exceeds inflow
  - Demand node: a node where inflow exceeds outflow
  - Transshipment or intermediate nodes: nodes satisfying flow conservation, where inflow equals outflow

The Shortest-path Problem

- The Shortest-path Problem Algorithm
  - Objective of the n-th iteration: Find the node that is the n-th closest from the starting point.
  - Input of the n-th iteration: Paths and distances from the starting point to the $(n-1)$-th closest nodes (solved nodes)
  - Candidates for the n-th closest node: Among the nodes directly connected to solved nodes but not yet solved, choose the one with the shortest distance (ties are possible)
  - Computation of the n-th closest node: For each candidate-solved node pair, calculate the shortest distance as: Distance(start → candidate) = Distance(start → solved node) + Distance (solved node → candidate)

The Shortest-path Problem



- The Shortest-path Problem Algorithm – Example

n	Solved nodes (among them, the nodes connected to unsolved nodes)	The closest unsolved node connected	Total distance	The n-th closest node	Shortest distance	Last connecting arc
1	O	A	2	A	2	OA
2 3	O A	C B	4 2+2	C B	4 4	OC AB
4	A B C	D E E	2+7=9 4+3=7 4+4=8	E	7	BE
5	A B E	D D D	2+7=9 4+4=8 7+1=8	D D	8 8	BD ED
6	D E	T T	8+5=13 7+7=14	T	13	DT

The Shortest-path Problem

- Other Applications of the Shortest-path Problem
  - Minimizing total travel distance
  - Minimizing the total cost or time of a sequence of consecutive tasks

Minimum Spanning Tree Problem

- Overview of the Minimum Spanning Tree (MST) Problem
  - A network consisting only of nodes is given, along with edges of positive length that could be added to the network.
  - The requirement is to design the network in which every pair of nodes is connected by adding edges.
  - The objective is to minimize the total length of edges inserted while satisfying the connectivity requirement.

Minimum Spanning Tree Problem

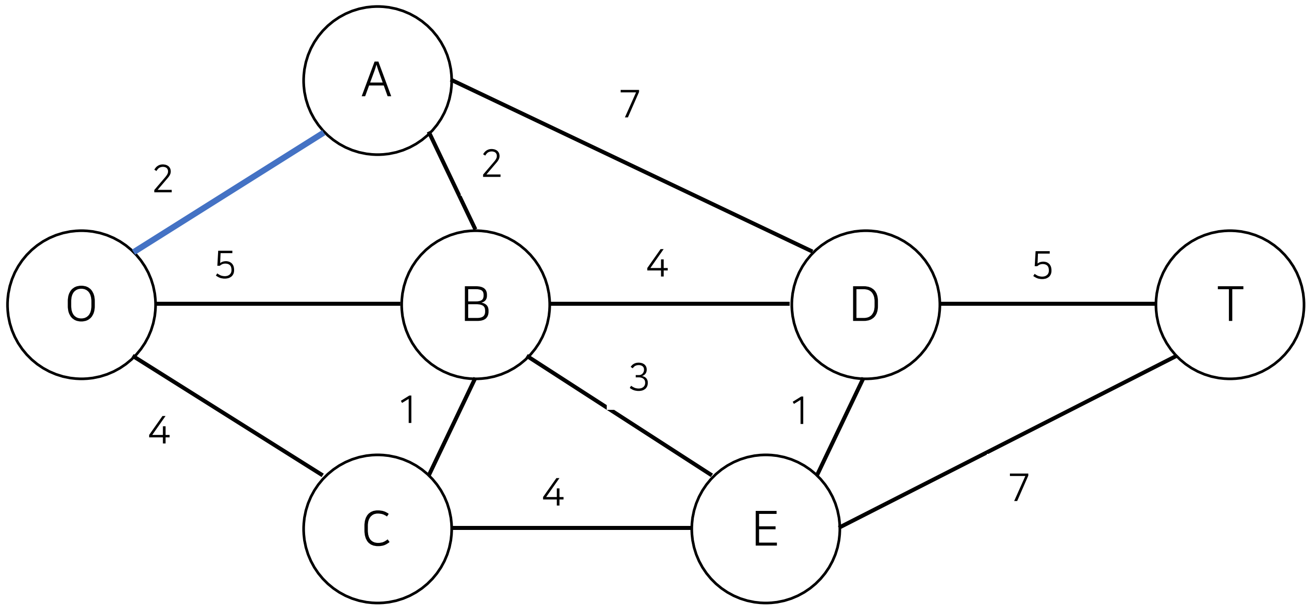
- The Minimum Spanning Tree Algorithm: Idea
  - The MST Problem is one of the few problems where a greedy decision at each step still leads to the optimal solution.
  - Therefore, it can be solved using a very simple method.
  - Regardless of which node is chosen first, the initial step is to select the shortest edge that connects to another node, with no need to consider future decisions.
  - The second step is to find the closest node (among the unconnected ones) to the current connected network and add the corresponding edge.
  - Repeat this process until all nodes are connected, then terminate.

Minimum Spanning Tree Problem

- The Minimum Spanning Tree Algorithm
  - 1. Select an arbitrary node and connect it to the closest (other) node.
  - 2. Among the unconnected nodes, find the one closest to the connected network and connect it. Repeat this step until all nodes are connected.
  - 3. Tie-breaking: if there are two or more equally closest nodes in Step 1 or Step 2, choosing any of them will yield an optimal solution. However, in this case, there may be multiple optimal solutions. To obtain all optimal solutions, each tied case must be explored separately.

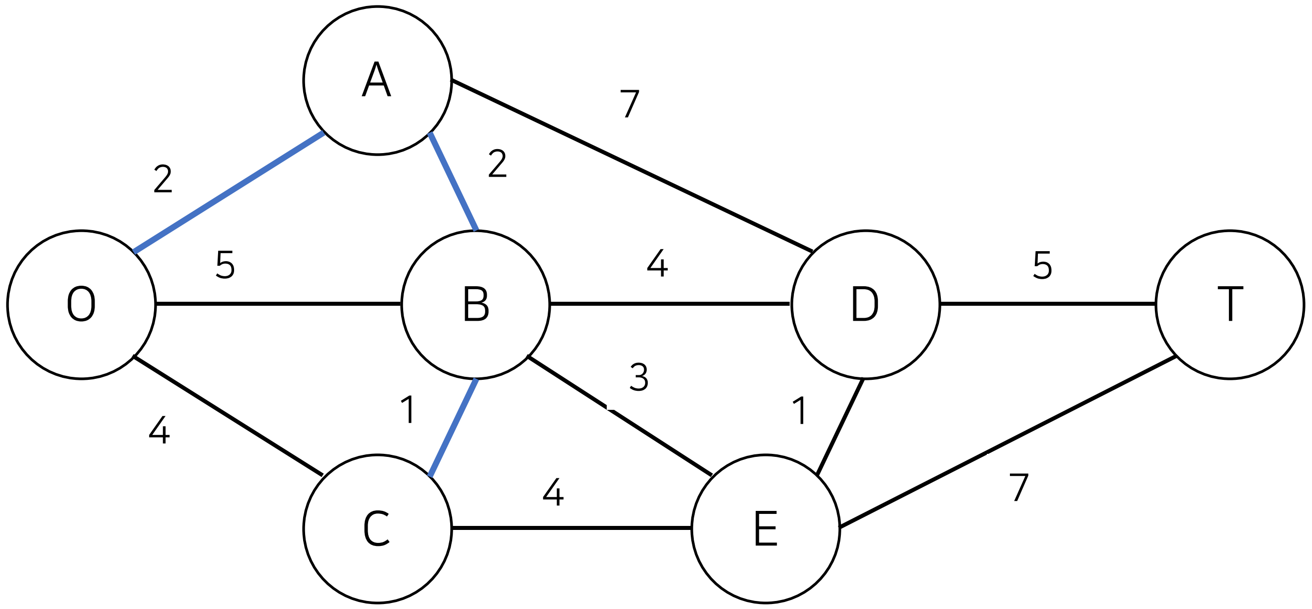
Minimum Spanning Tree Problem

- The Minimum Spanning Tree Algorithm - Example
  - Suppose we choose node O as the initial node.
  - 1. The closest node not yet connected to O is A. Therefore, connect node A to O.



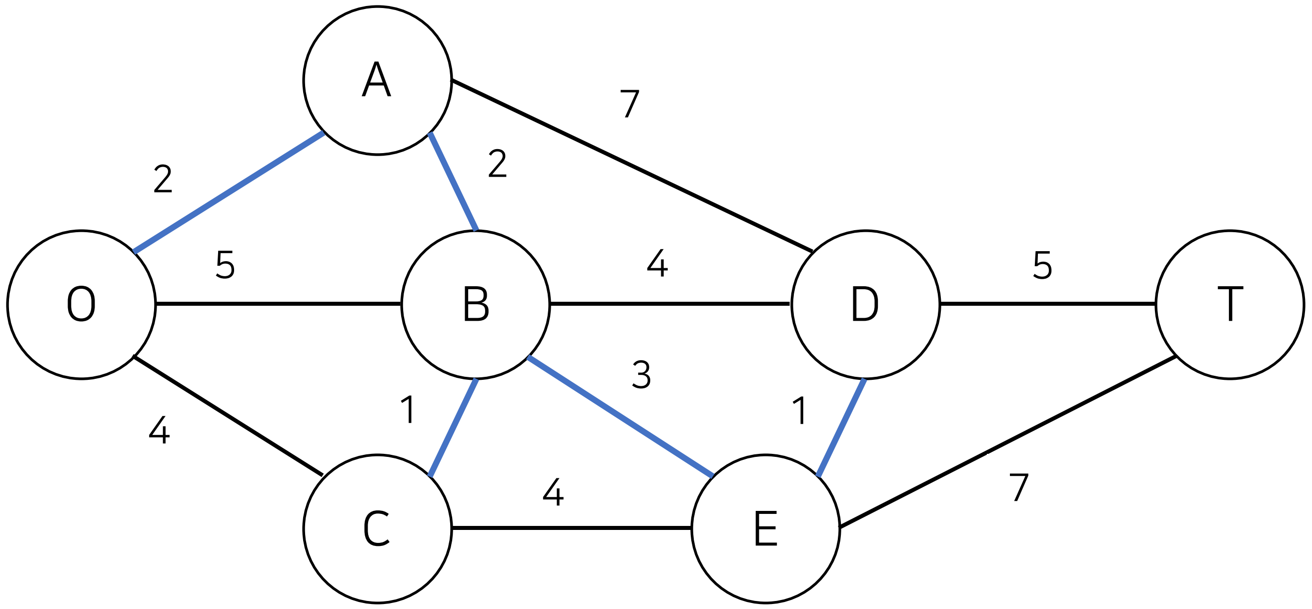
Minimum Spanning Tree Problem

- The Minimum Spanning Tree Algorithm - Example
  - 2. The closest unconnected node to O to O or A is B. Connect node B to A.
  - 3. The closest unconnected node to O, A, B is C. Connect node C to B.



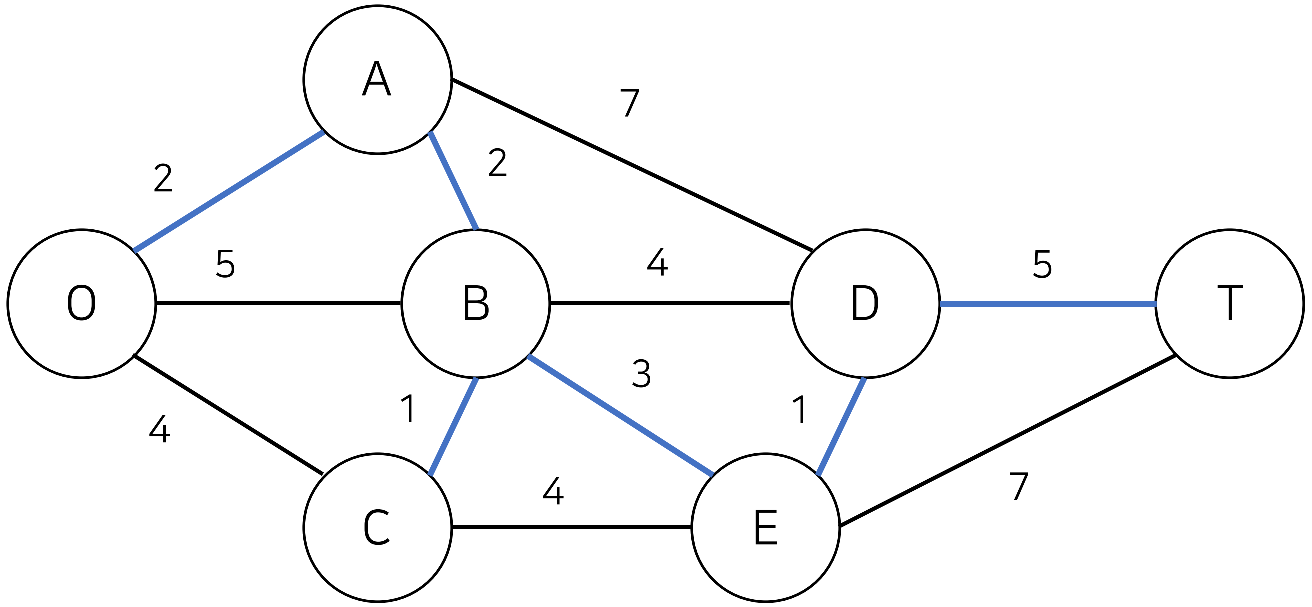
Minimum Spanning Tree Problem

- The Minimum Spanning Tree Algorithm - Example
  - 4. The closest unconnected node to O, A, B, C is E. Connect node E to B..
  - 5. The closest unconnected node to O, A, B, C, E is D. Connect node D to E.



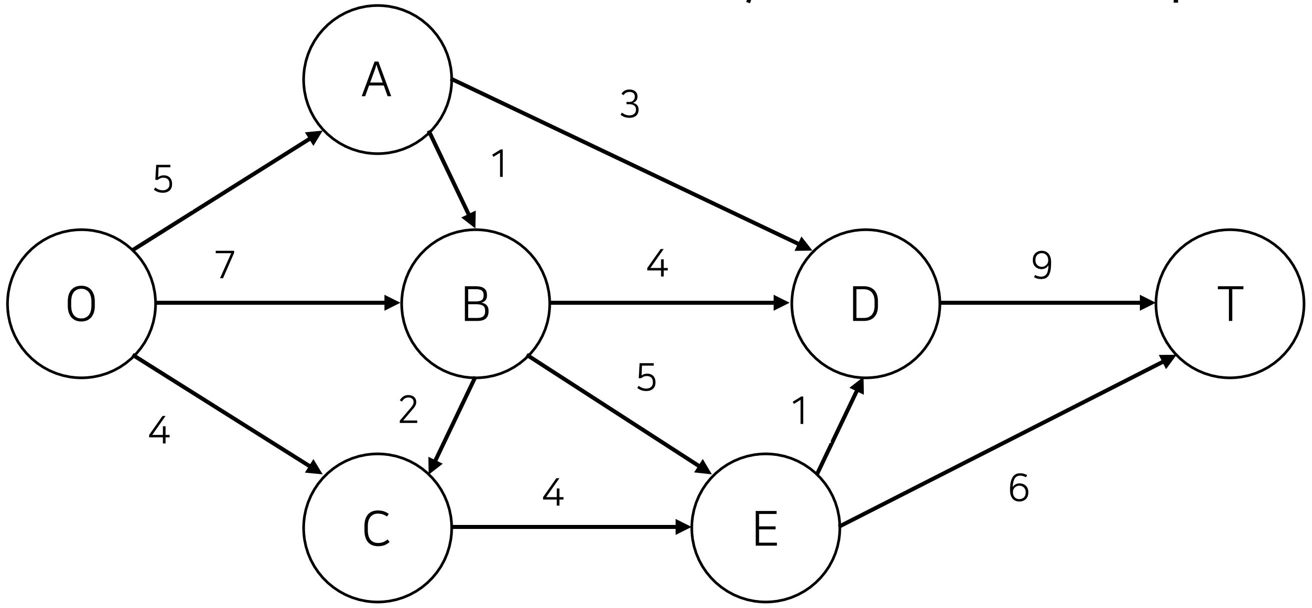
Minimum Spanning Tree Problem

- The Minimum Spanning Tree Algorithm - Example
  - 6. The only node not yet connected is T, and its closest node is D. Therefore, connect node T to node D.

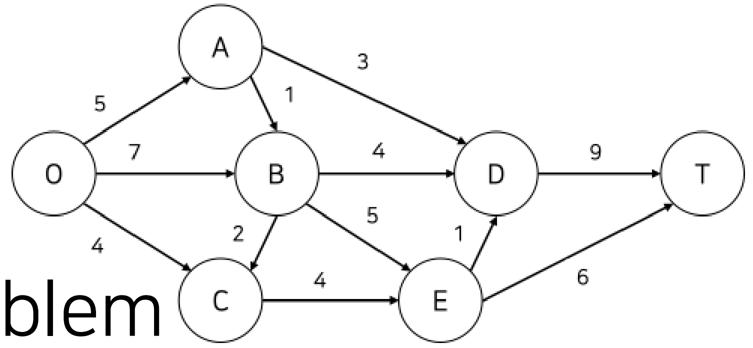


The Maximum Flow Problem

- The Maximum Flow Problem
  - In the given park road network example, suppose that the number of vehicle trips per road per day is limited for environmental protection.
  - For each road, the allowed travel direction (from the entrance to the destination) and the maximum one-way number of trips are shown in the figure.



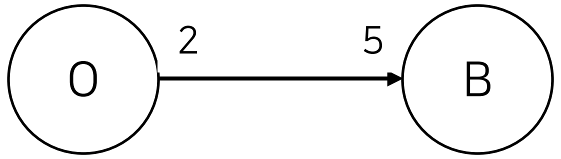
The Maximum Flow Problem



- General Description of The Maximum Flow Problem
  - In a directed and connected network, flow starts from a Source node (in the example, O) and ends at a Sink node (in the example, T).
  - All other nodes are called intermediate nodes.
  - Flow along arcs can only move in the direction of the arrow, and the maximum amount of flow on each arc is limited by its capacity. All arcs connected to the source node point outward, and all arcs connected to the sink node point inward.
  - The objective of to maximize the total flow from the source to the sink. The maximum flow can be measured either as the total outflow from the source or equivalently as the total inflow into the sink.

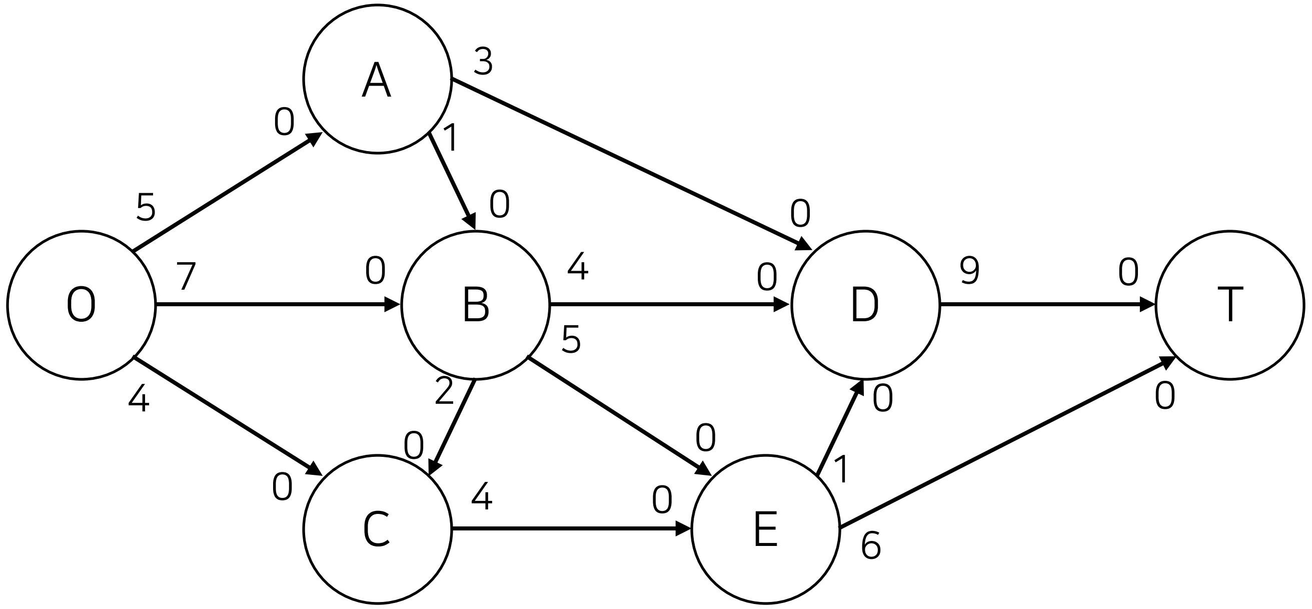
The Maximum Flow Problem

- Key Concepts for the Maximum Flow Problem Algorithm
  - Residual network: After a certain amount of flow has been assigned to arcs, the residual network shows the remaining capacity (residual capacity) that can still be assigned on each arc.
    - ✓ For example, if the arc O → B has a total capacity of 7 and 5 units are already used, then the residual capacity available is 2.
    - ✓ This can be illustrated in a diagram showing the updated residual capacities on the arcs.



The Maximum Flow Problem

- Key Concepts for the Maximum Flow Problem Algorithm
  - Residual network: After a certain amount of flow has been assigned to arcs, the residual network shows the remaining capacity (residual capacity) that can still be assigned on each arc.
    - ✓ Before any flow is assigned, the initial residual network of the park example is exactly the same as the original network, since all arc capacities are fully available.



The Maximum Flow Problem

- Key Concepts for the Maximum Flow Problem Algorithm
  - Augmenting path: In the residual network, an augmenting path is a directed path from the source to the sink in which all arcs have positive residual capacity.
  - The minimum residual capacity among the arcs on this path determines how much additional flow can be sent along the path.
  - This minimum value is called the residual capacity of the augmenting path.
  - By repeatedly finding augmenting paths, the total flow in the original network can be increased.

The Maximum Flow Problem

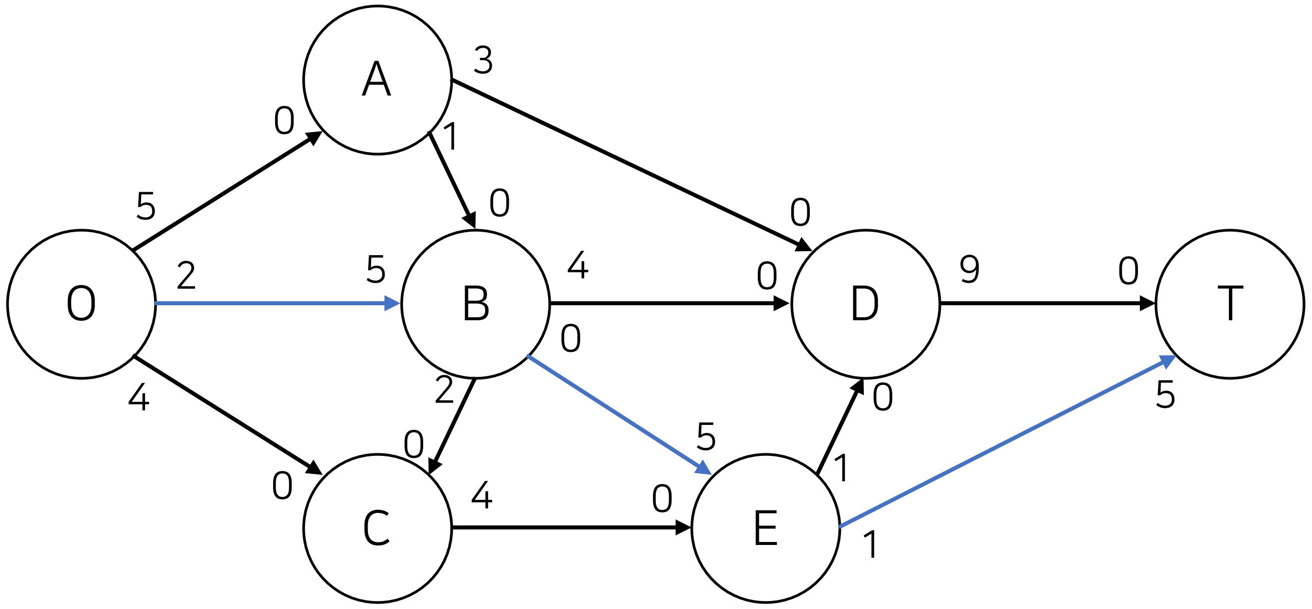
- The Augmenting Path Algorithm
  - The augmenting path algorithm repeatedly searches for an augmenting path and adds flow to the original network equals to the residual capacity of that path.
  - This process continues until no augmenting path remains and no additional flow from the source to the sink can be added.

The Maximum Flow Problem

- The Augmenting Path Algorithm
  - 1. In the residual network, find an augmenting path from the source to the sink such that all arcs on the path have positive residual capacity.
    - ✓ If no augmenting path exists, the currently assigned flow is already the optimal flow.
  - 2. Identify the minimum residual capacity among the arcs on this path, denoted c*. Increase the flow along this path by c*.
  - 3. Reduce the residual capacity of each arc on this path by c*. Increase the residual capacity of each arc in the reverse direction by c*. Return to Step
- Note
  - If multiple augmenting paths exist in Step 1, the choice of path becomes an important issue in solving large-scale problems, but this detail is not covered here.
  - The focus is only on the systematic procedure for finding augmenting paths.

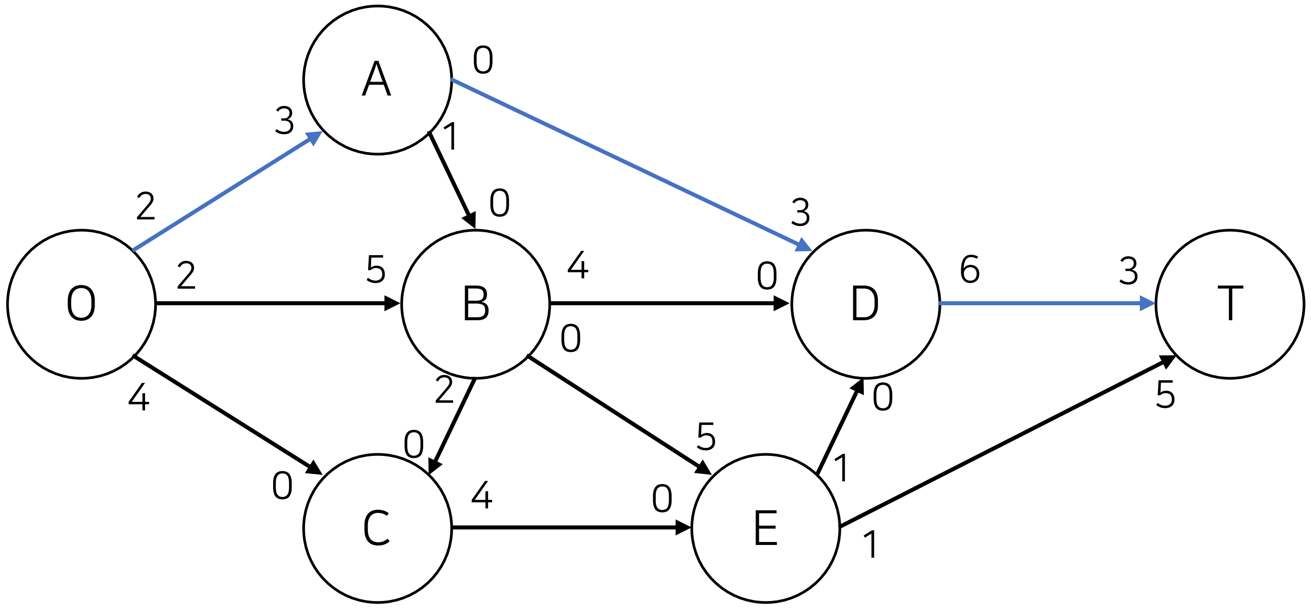
The Maximum Flow Problem

- The Augmenting Path Algorithm - Example
  - Iteration 1. One augmenting path is O-B-E-T, and its residual capacity is min{7, 5, 6} = 5. After assigning a flow of 5 along this path, the residual network is updated as shown.



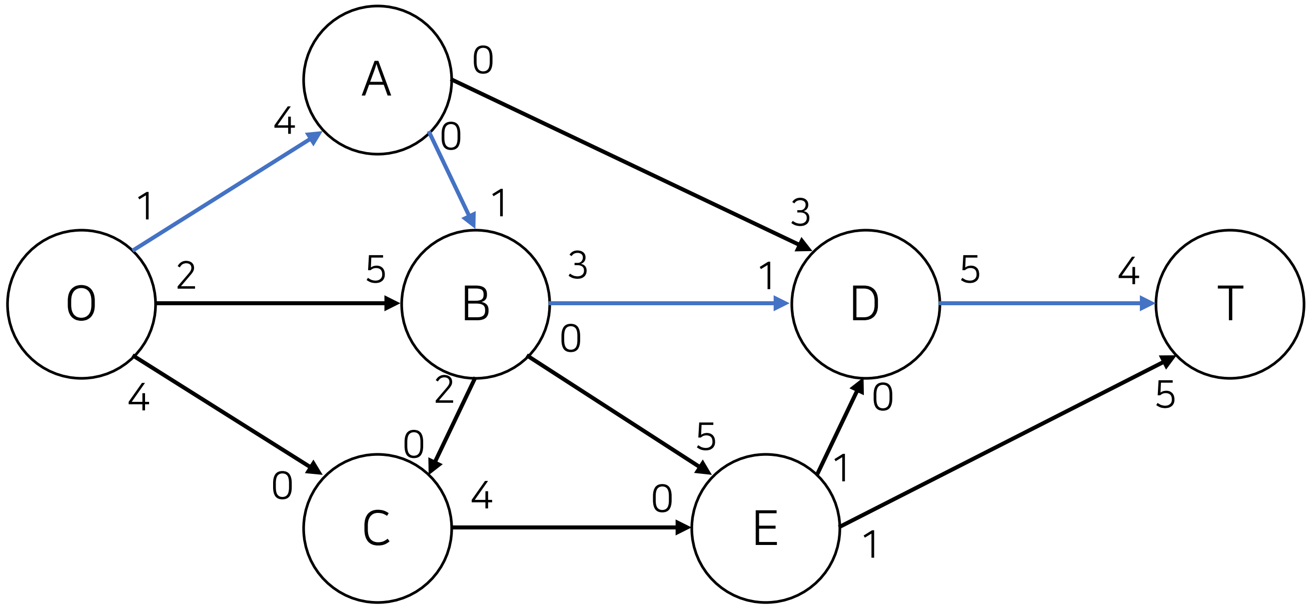
The Maximum Flow Problem

- The Augmenting Path Algorithm - Example
  - Iteration 2. One augmenting path is O-A-D-T, and its residual capacity is min{5, 3, 9} = 3. After assigning a flow of 3 along this path, the residual network is updated as shown.



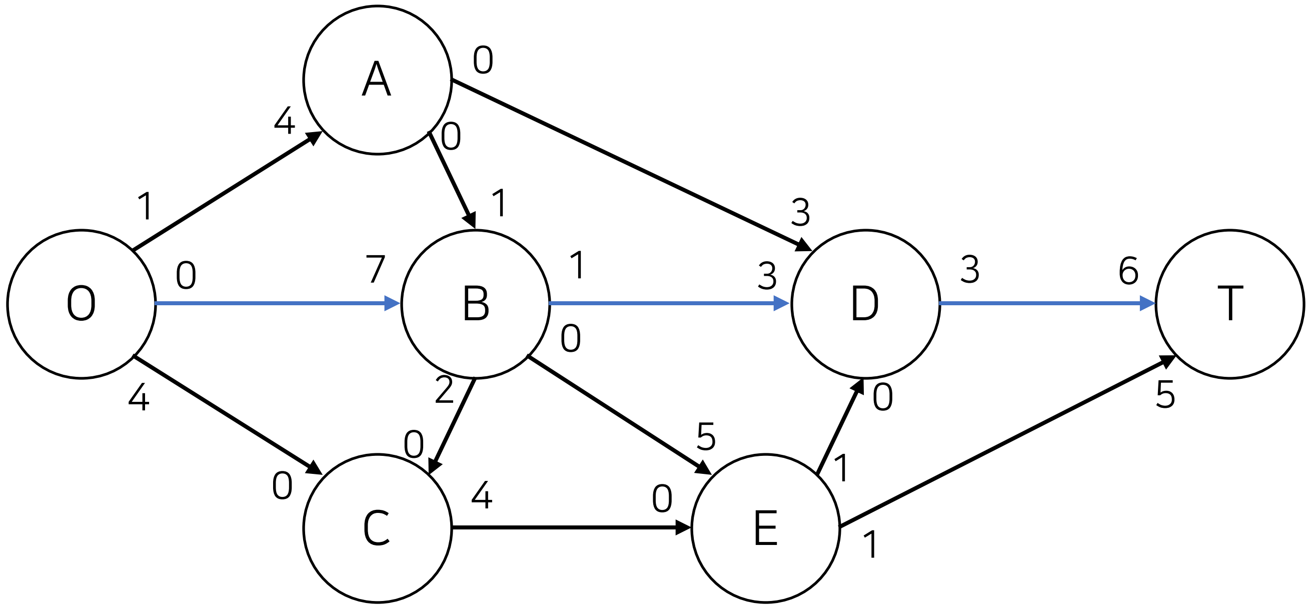
The Maximum Flow Problem

- The Augmenting Path Algorithm - Example
  - Iteration 3. One augmenting path is O-A-B-D-T, and its residual capacity is min {2, 1, 4, 6} = 1. After assigning a flow of 1 along this path, the residual network is updated as shown.



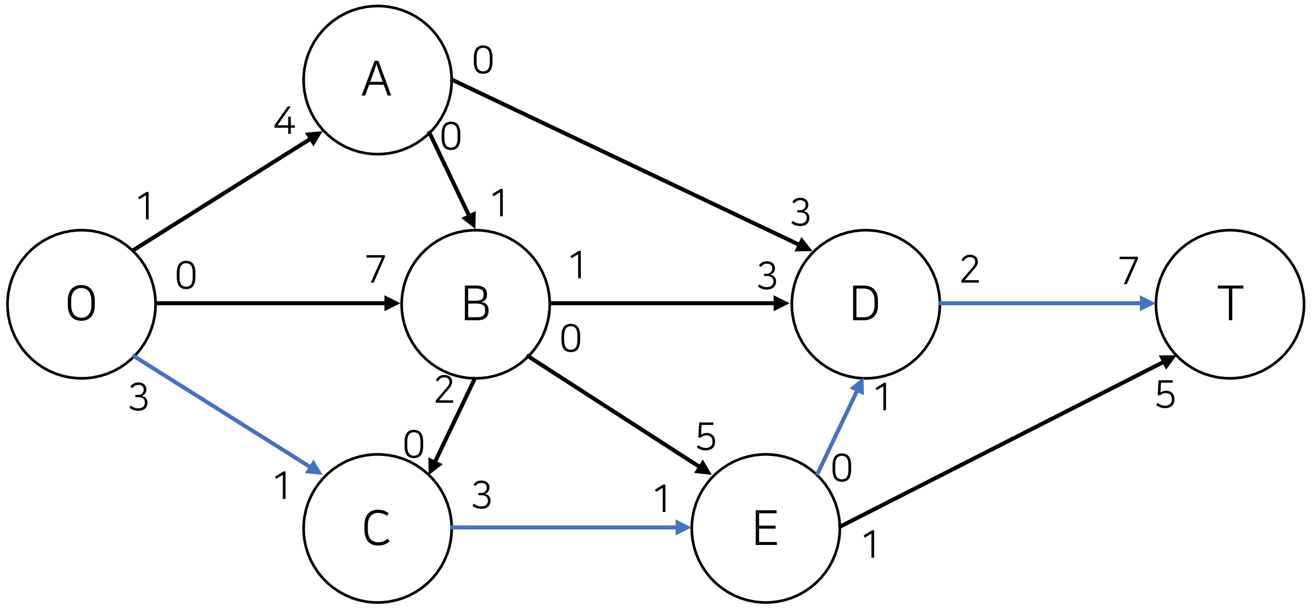
The Maximum Flow Problem

- The Augmenting Path Algorithm - Example
  - Iteration 4. One augmenting path is O-B-D-T, and its residual capacity is min{2, 3, 5} = 2. After assigning a flow of 2 along this path, the residual network is updated as shown.



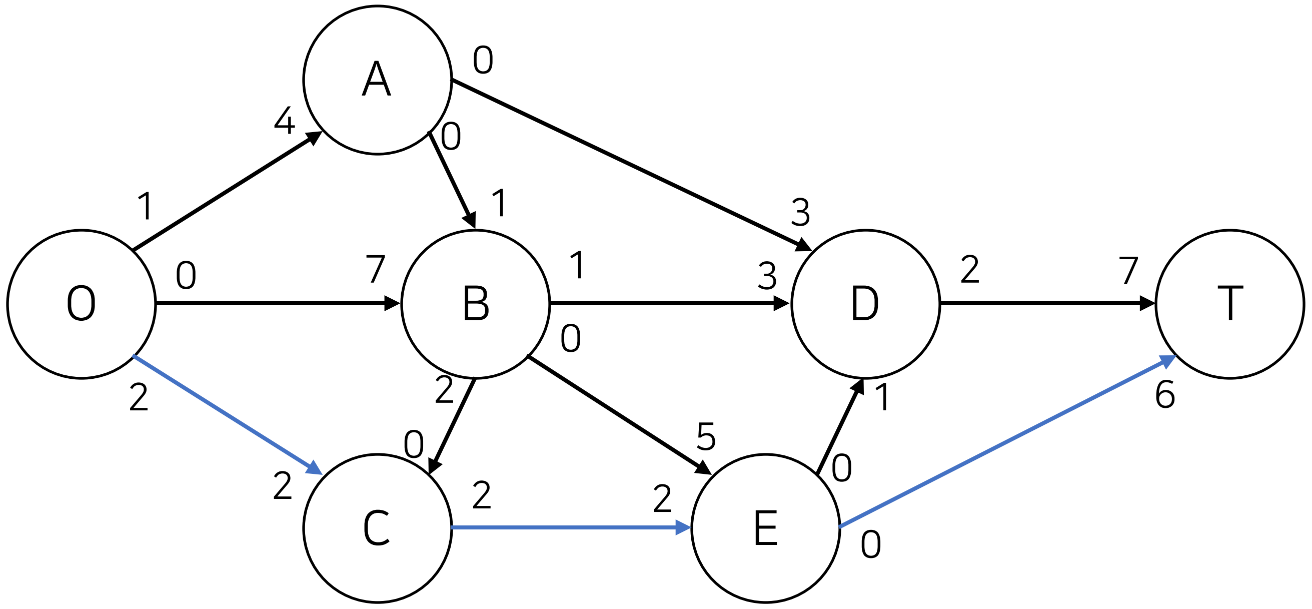
The Maximum Flow Problem

- The Augmenting Path Algorithm - Example
  - Iteration 5. One augmenting path is O-C-E-D-T, and its residual capacity is min{4, 4, 1, 3} = 1. After assigning a flow of 1 along this path, the residual network is updated as shown.



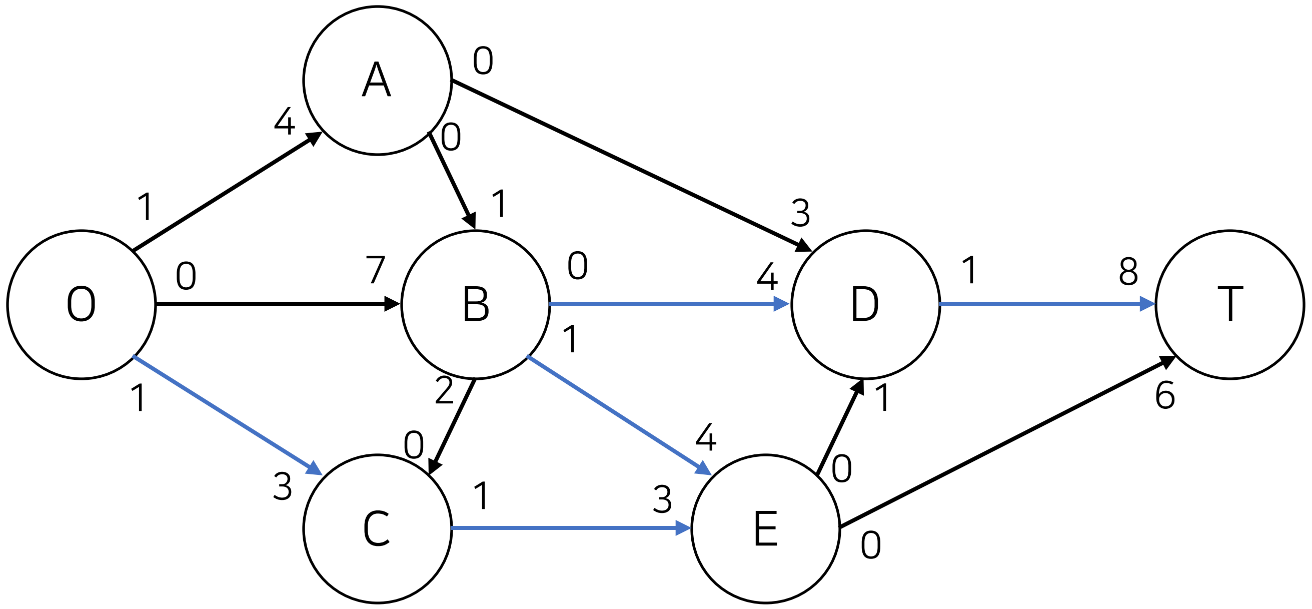
The Maximum Flow Problem

- The Augmenting Path Algorithm - Example
  - Iteration 6. One augmenting path is O-C-E-T, and its residual capacity is min{3, 3, 1} = 1. After assigning a flow of 1 along this path, the residual network is updated as shown.



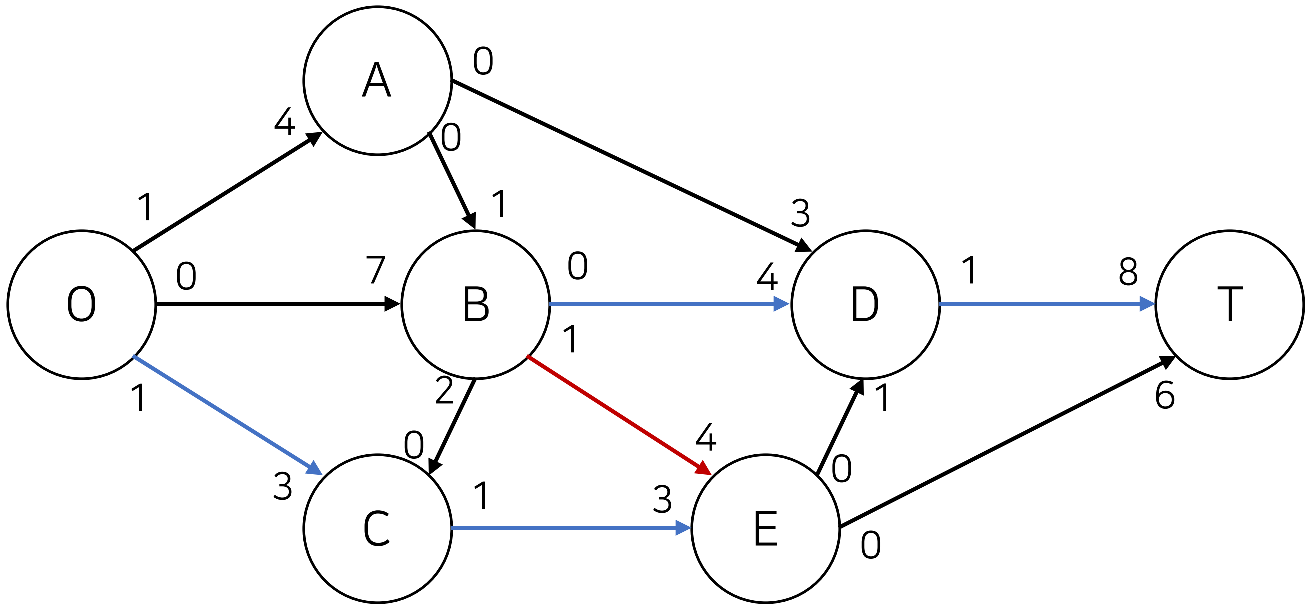
The Maximum Flow Problem

- The Augmenting Path Algorithm - Example
  - Iteration 7. One augmenting path is O-C-E-B-D-T, and its residual capacity is min{2, 2, 5, 1, 2} = 1. After assigning a flow of 1 along this path, the residual network is updated as shown.
  - Since no further augmenting paths exist, the current flow is optimal.



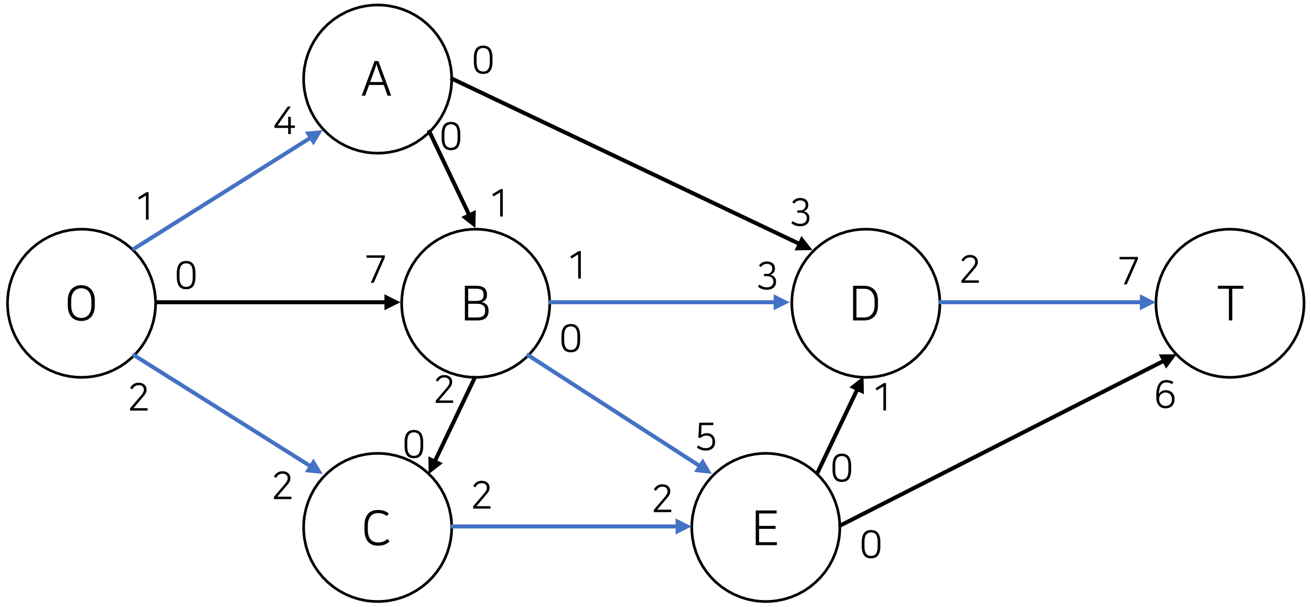
The Maximum Flow Problem

- The Augmenting Path Algorithm - Example
  - In the 7th iteration, canceling 1 unit of flow from B to E can be treated as if sending 1 unit of flow along the reverse direction E to B.
  - This approach ensures that no additional feasible flow is overlooked, allowing the algorithm to correctly identify further augmenting paths if they exist.



The Maximum Flow Problem

- Searching for an Augmenting Path
  - A systematic procedure is required to find augmenting paths.
    - ✓ Start with arcs from the source with positive residual capacity → identify reachable nodes.
    - ✓ From each reachable node, extend to other not-yet-reached nodes along arcs with positive residual capacity → expand the set of reachable nodes (repeat this process).
  - Example. In the 7th iteration, the procedure is applied.



The Minimum Cost Flow Problem

- The Minimum Cost Flow Problem
  - Plays a central role in network optimization models because it both encompasses a wide variety of application problems and allows solutions to be obtained very efficiently.
  - The Maximum Flow Problem, Shortest Path Problem, and Transportation and Assignment Problems are all special cases of the minimum-cost flow problem.

The Minimum Cost Flow Problem

- The Characteristics of the Minimum-Cost Flow Problem
  - 1. The network is directed and connected.
  - 2. At least one node is a supply node.
  - 3. Among the other nodes, at least one is a demand node.
  - 4. All remaining nodes are intermediate nodes.
  - 5. Flow along an arc can only move in its given direction and cannot exceed the arc's capacity (If bidirectional flow is possible, a pair of arcs in opposite direction is required.)
  - 6. The network is structured so that the arcs have sufficient capacity for the flow generated at supply nodes to reach all demand nodes.
  - 7. The cost of flow through an arc is proportional to the amount of flow, and the unit flow cost is known.
  - 8. The objective is to minimize the total cost required to send the available supploy through the network to satisfy demand. (Alternatively, the goal may be to maximize total profit.)

The Minimum Cost Flow Problem

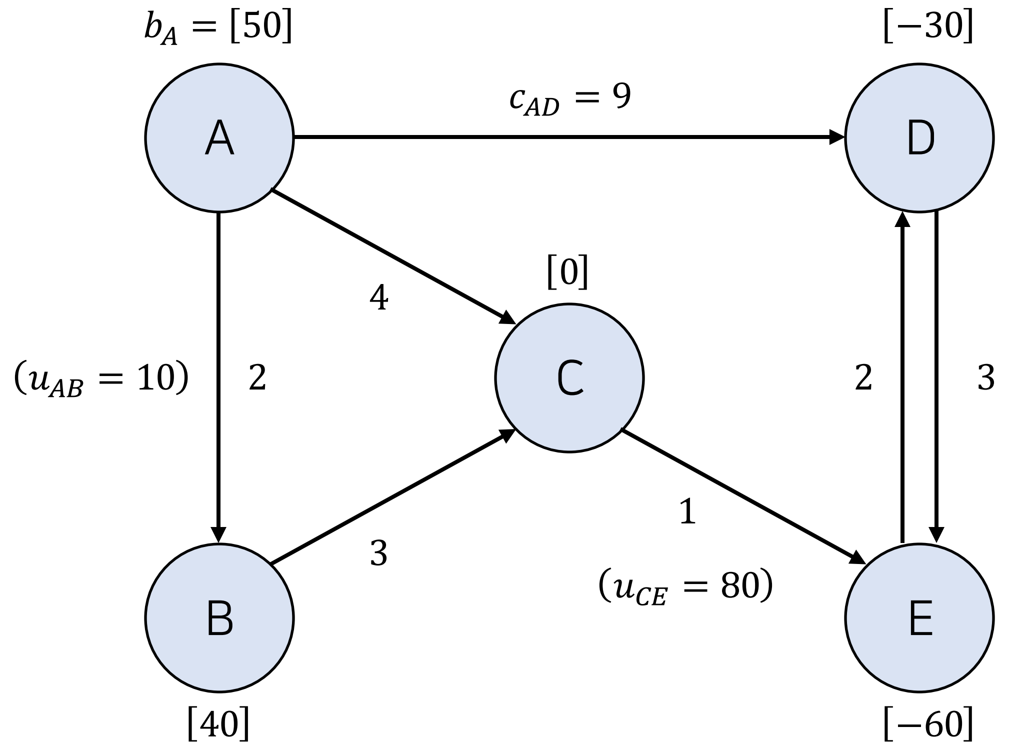
- Formulation of the Minimum-Cost Flow Problem
  - Suppose there are n nodes in the network.
    - ✓ Define x_{ij}: the amount of flow along arc $i \to j$
  - The given information is as follows:
    - ✓ c_{ij}: the unit cost of flow on arc $i \to j$
    - ✓ u_{ij}: the capacity of arc $i \to j$
    - ✓ b_i: the net flow generated at node i
      - $b_i > 0$: supply node
      - $b_i = 0$: intermediate node
      - $b_i < 0$: demand node

The Minimum Cost Flow Problem

- Formulation of the Minimum-Cost Flow Problem
  - Objective: Generate flow through the network to satisfy the given demand at minimum cost within available supply limits.
    - ✓ Minimize $Z = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$
      - (Flow balance for each node i) $\sum_{j=1}^{n} x_{ij} - \sum_{j=1}^{n} x_{ji} = b_i$
      - (Capacity bounds for each arc $i \to j$) $0 \leq x_{ij} \leq u_{ij}$
    - ✓ Application (lower bounds on arcs): If a minimum flow $L_{ij} > 0$ is required, substitute $x'_{ij} = x_{ij} - L_{ij} \to x_{ij} = x'_{ij} + L_{ij}$
    - ✓ Feasibility property: A necessary condition for feasibility in the minimum-cost flow problem is $\sum_{i=1}^{n} b_i = 0$.
      - If this does not hold, introduce dummy supply/demand to balance the network.
    - ✓ Integrality property: If all b_i and u_{ij} are integers, then in every BFS (including the optimum), all BVs are integers.

The Minimum Cost Flow Problem

- The Minimum-Cost Flow Problem – Example
  - Minimize $Z = 2x_{AB} + 4x_{AC} + 9x_{AD} + 3x_{BC} + x_{CE} + 3x_{DE} + 2x_{ED}$
    - ✓ $x_{AB} + x_{AC} + x_{AD} = 50$
    - ✓ $-x_{AB} + x_{BC} = 40$
    - ✓ $-x_{AC} - x_{BC} + x_{CE} = 0$
    - ✓ $-x_{AD} + x_{DE} - x_{ED} = -30$
    - ✓ $-x_{CE} - x_{DE} + x_{ED} = -60$
    - ✓ $x_{AB} \leq 10, \;\; x_{CE} \leq 80, \;\; x_{ij} \geq 0$
  - The constraint matrix consists of coefficients (+1, -1), and one of the node constraints is redundant (because $\sum_i b_i = 0$) → Therefore, a BFS has a total of n-1 BVs.



The Minimum Cost Flow Problem

- The Minimum-Cost Flow Problem – Special Cases
  - Transportation Problem
    - ✓ Treat each supply point as a supply node and each destination as a demand node.
    - ✓ There are no intermediate nodes; all arcs are directed from supply nodes to demand nodes. Let x_{ij} be the quantity from supply i to demand j, and c_{ij} the unit shipping cost.
    - ✓ Typically, there are no upper bounds on arcs, so $u_{ij} = \infty$.
  - Assignment Problem
    - ✓ As a special case of the transportation problem, it can be modeled as a minimum-cost flow problem as above.
    - ✓ Additionally, (1) the number of supply nodes equals the number of demand nodes; (2) for each supply node, $b_i = 1$; and (3) for each demand node, $b_i = -1$.
  - Transportation with Intermediate Transshipment
    - ✓ In a more general setting, transportation problems with intermediate transshipment stages between sources and destinations can be represented as minimum-cost flow problems.

The Minimum Cost Flow Problem

- The Minimum-Cost Flow Problem – Special Cases
  - Shortest Path Problem
    - ✓ The starting node becomes a supply node with a supply 1($b_i = 1$), and the final destination becomes a demand node with a demand 1($b_i = -1$). All other nodes become transshipment nodes.
    - ✓ Since the shortest path problem is undirected, each edge is replaced with a pair of arcs in opposite directions.
    - ✓ The distance between nodes i and j becomes the unit cost c_{ij}.
  - Maximum Flow Problem
    - ✓ The network already consists of one supply node, one demand node, and the remaining intermediate nodes.
    - ✓ In the maximum flow problem, arcs have no unit costs, so all arc costs c_{ij} are set to 0.
    - ✓ Choose a sufficiently large value $\bar{F}$ as the upper bound on the feasible flow through the network, and set the supply and demand at the source and sink nodes to $\bar{F}$.
    - ✓ Add an auxiliary arc directly connecting the supply node and the demand node, assigning it a very large unit cost $c_{ij} = M$ and infinite capacity $u_{ij} = \infty$.

The Network Simplex Method

- The Network Simplex Method
  - A simplified version of the Simplex Method used to solve the minimum cost flow problem.
  - The same basic procedures determining the entering BV, determining the leaving BV, and computing the new BFS are carried out based on the network structure without using a Simplex tableau.

The Network Simplex Method

- The Network Simplex Method
  - The use of the upper bound technique
    - ✓ To efficiently handle arc capacity upper-bound constraints $x_{ij} \leq u_{ij}$, the upper bound technique is used.
    - ✓ This approach allows capacity constraints to be treated similarly to non-negativity constraints, so that they are considered when deciding the leaving BV.
    - ✓ Summary: when $x_{ij} = u_{ij}$, replace it with $x_{ij} = u_{ij} - y_{ij}$, making $y_{ij} = 0$ a NBV.
      - Interpretation of y_{ij}: $y_{ij} > 0$ becomes a BV, it can be viewed as a flow from $j \rightarrow i$ (in effect, canceling out flow from $i \rightarrow j$)
      - When $x_{ij} = u_{ij}$ and is replaced by $x_{ij} = u_{ij} - y_{ij}$, the arc $i \rightarrow j$ is effectively replaced with $j \rightarrow i$, with the maximum capacity u_{ij}, and since it cancels the original flow x_{ij}, its unit cost becomes $-c_{ij}$.
      - To reflect $x_{ij} = u_{ij}$, decrease b_i by u_{ij} and increase b_j by u_{ij}, thereby shifting the net flow.
      - Later, if $y_{ij} = u_{ij}$ reaches its upper bound, replace it back with $y_{ij} = u_{ij} - x_{ij}$, making $x_{ij} = 0$ a NBV again.

The Network Simplex Method

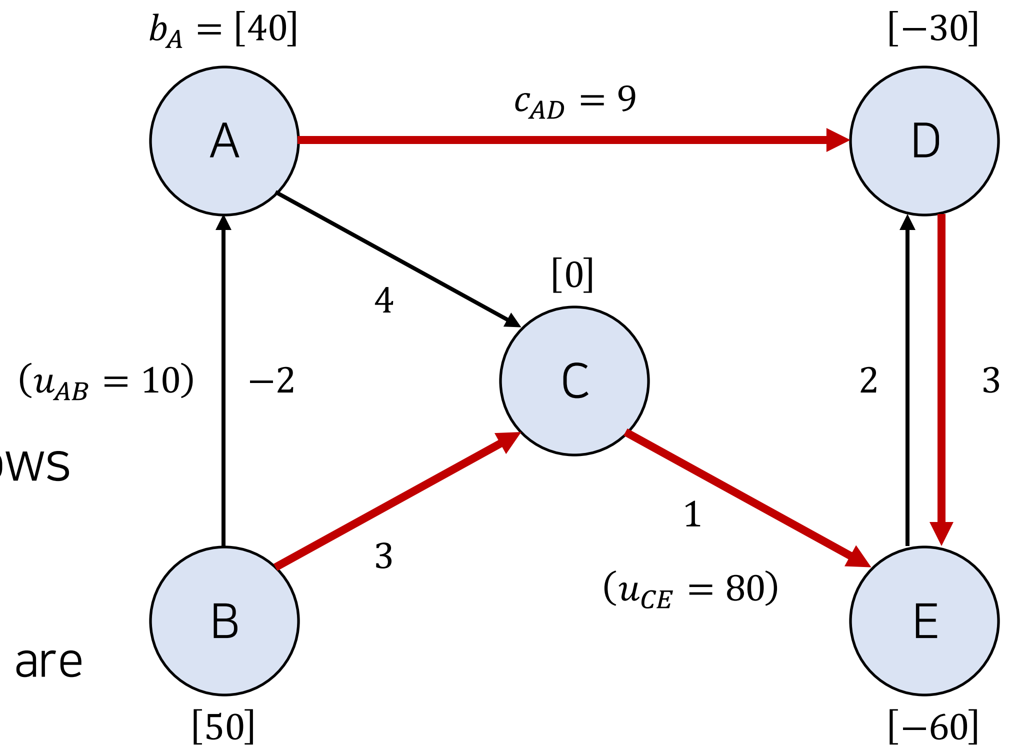
- Relationship between BFS and Feasible Spanning Tree
  - In the minimum cost flow problem with n nodes, there are n constraints, but one is redundant. Hence, a BFS contains $n-1$ BVs(=basic arcs).
    - ✓ The remaining arcs are NBVs (non-basic arcs).
  - A key property of basic arcs is that they do not form an undirected cycle.
    - ✓ A set of n-1 arcs without undirected cycles is called a spanning tree.
    - ✓ Therefore, a BFS can be obtained by finding a spanning tree. (BFS $\leftrightarrow$ Spanning Tree)

The Network Simplex Method

- Relationship between BFS and Feasible Spanning Tree
  - Spanning tree solution
    - ✓ Set the variables corresponding to arcs not included in the spanning tree (x_{ij} or y_{ij}) to 0.
    - ✓ Solve the system of node-balance equations to obtain the variables corresponding to arcs included in the spanning tree.
    - ✓ A general spanning tree solution is obtained without considering non-negativity or capacity constraints on basic arcs, so feasibility must be checked separately.
  - Feasible spanning tree
    - ✓ A spanning tree whose solution from the node constraints also satisfies all other constraints ($0 \le x_{ij} \le u_{ij}$ or $0 \le y_{ij} \le u_{ij}$).
  - Fundamental correspondence in the Network Simplex Method
    - ✓ Basic solution $\leftrightarrow$ Spanning tree
    - ✓ Basic feasible solution $\leftrightarrow$ Feasible spanning tree

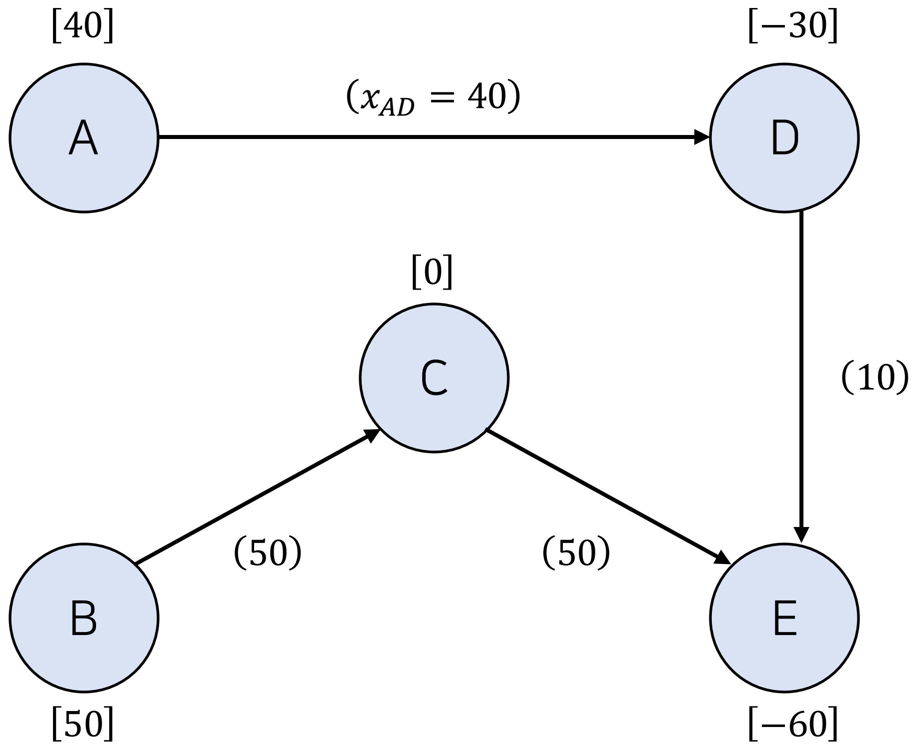
The Network Simplex Method

- Relationship between BFS and Feasible Spanning Tree - Example
  - Example. State after reversing due to satisfying the upper bound of 10 on the flow $A \rightarrow B$
    - ✓ Node-balance constraints
      - $-y_{AB} + x_{AC} + x_{AD} = 40$
      - $y_{AB} + x_{BC} = 50$
      - $-x_{AC} - x_{BC} + x_{CE} = 0$
      - $-x_{AD} + x_{DE} - x_{ED} = -30$
      - $-x_{CE} - x_{DE} + x_{ED} = -60$
    - ✓ Solution of the spanning tree composed of red arrows
      - $y_{AB} = 0, x_{AC} = 0, x_{ED} = 0$
      - $x_{AD} = 40, \;\; x_{BC} = 50, \;\; x_{CE} = 50, \;\; x_{DE} = 10$
      - Since the non-negativity and upper-bound constraints are satisfied, this is a BFS.



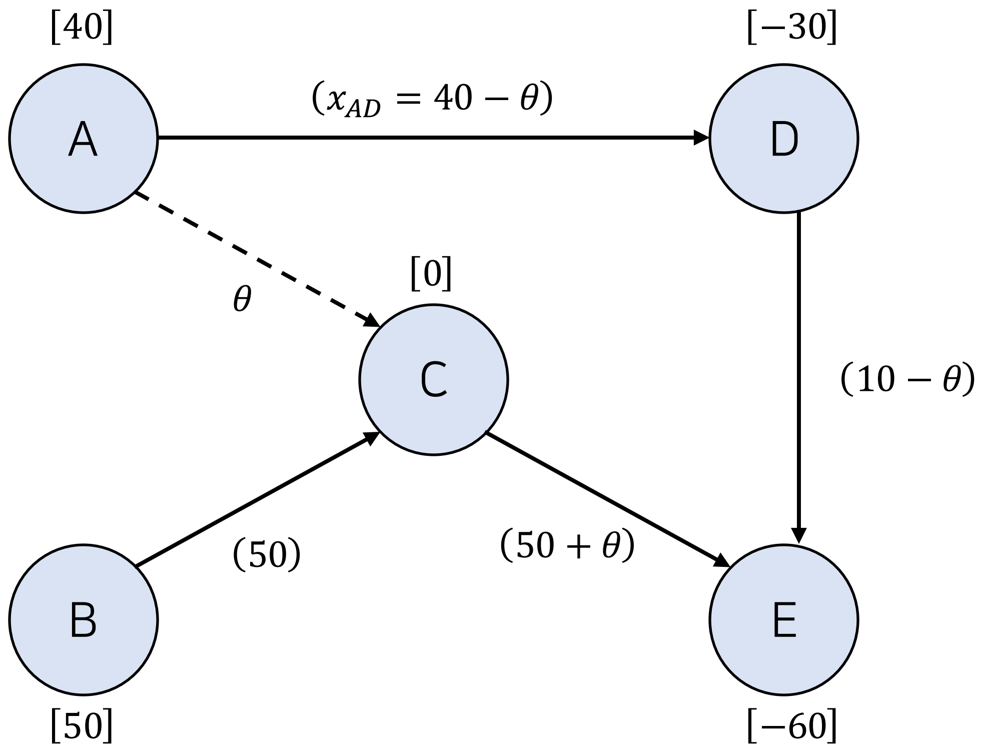
The Network Simplex Method

- Initial Feasible Spanning Tree



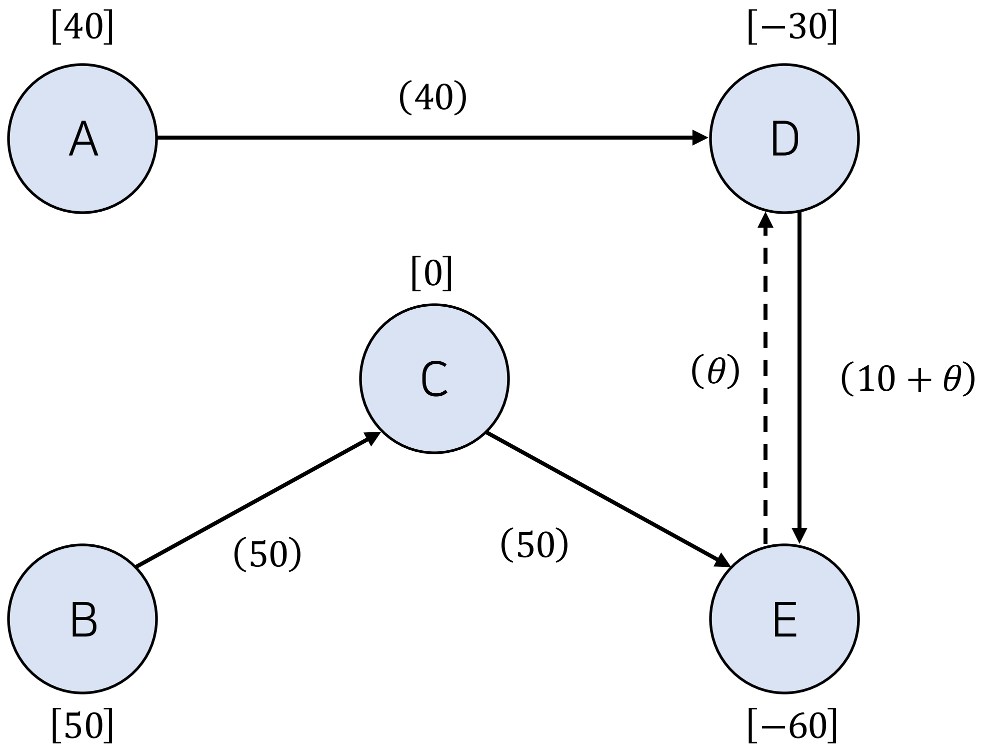
The Network Simplex Method

- Determining the Entering BV
  - Among the NBVs, select the one that improves Z the most.
    - ✓ (ex) If we set $x_{AC} = \theta$ (i.e., increase the flow along A→C by θ), then an undirected cycle is formed. In this case, the cycle is AC-CE-DE-AD.
    - ✓ Now, examining the effect of increasing $x_{AC} = \theta$ on Z, $\Delta Z = c_{AC} \cdot \theta + c_{CE} \cdot \theta + c_{DE} \cdot (-\theta) + c_{AD} \cdot (-\theta) = 4\theta + \theta - 3\theta - 9\theta = -7\theta$
    - ✓ Thus, introducing x_{AC} as the entering BV decreases the objective by 7θ.



The Network Simplex Method

- Determining the Entering BV
  - Among the NBVs, select the one that improves Z the most.
    - ✓ (ex) If we set $x_{AC} = \theta$ (i.e., increase the flow along A→C by θ), then an undirected cycle is formed. In this case, the cycle is AC-CE-DE-AD.
    - ✓ Now, examining the effect of increasing $x_{AC} = \theta$ on Z, $\Delta Z = c_{AC} \cdot \theta + c_{CE} \cdot \theta + c_{DE} \cdot (-\theta) + c_{AD} \cdot (-\theta) = 4\theta + \theta - 3\theta - 9\theta = -7\theta$.
    - ✓ If we set $y_{AB} = \theta$, then the cycle BA-AD-DE-EC-CB is formed. The effect on Z is $\Delta Z = (-c_{AB}) \cdot \theta + c_{AD} \cdot \theta + c_{DE} \cdot \theta + c_{CE} \cdot (-\theta) + c_{BC} \cdot (-\theta) = -2\theta + 9\theta + 3\theta - \theta - 3\theta = 6\theta$.
    - ✓ If we set $x_{ED} = \theta$, then the cycle ED-DE is formed, the effect on Z is $\Delta Z = c_{ED} \cdot \theta + c_{DE} \cdot \theta = 2\theta + 3\theta = 5\theta$.

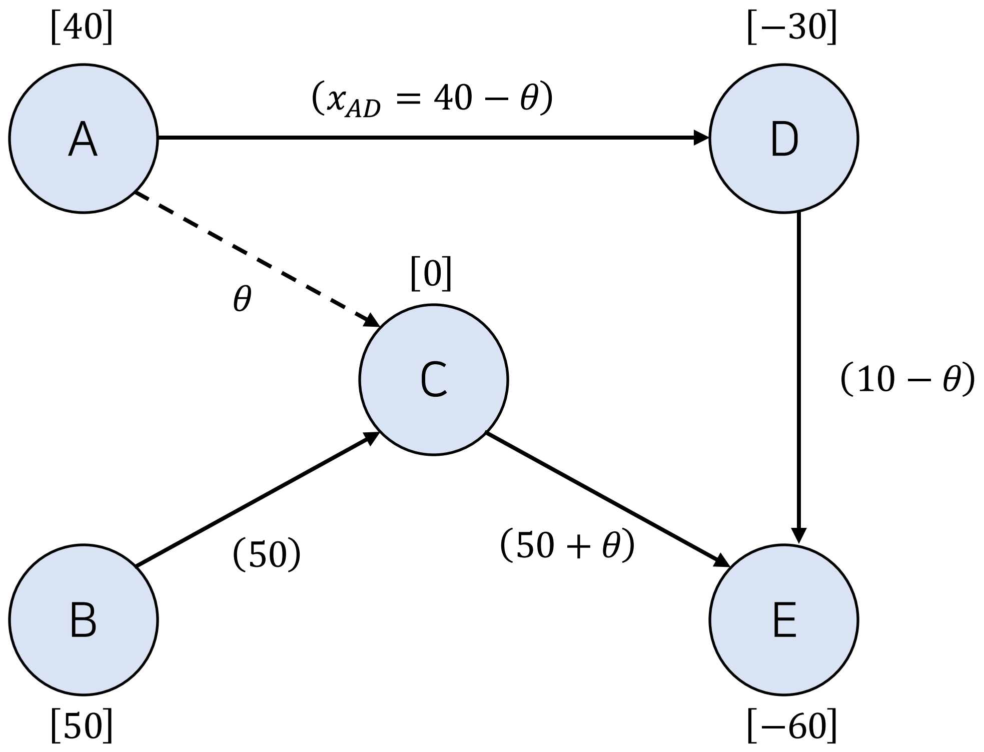


The Network Simplex Method

- Determining the Entering BV
  - Among the NBVs, we calculate the effect on the objective value Z when each variable increases by θ.
    - ✓ $x_{AC} = \theta$:
      $\Delta Z = c_{AC} \cdot \theta + c_{CE} \cdot \theta + c_{DE} \cdot (-\theta) + c_{AD} \cdot (-\theta) = 4\theta + \theta - 3\theta - 9\theta = -7\theta$
    - ✓ $y_{AB} = \theta$: the cycle BA-AD-DE-EC-CB is formed
      → $\Delta Z = (-c_{AB}) \cdot \theta + c_{AD} \cdot \theta + c_{DE} \cdot \theta + c_{CE} \cdot (-\theta) + c_{BC} \cdot (-\theta) = -2\theta + 9\theta + 3\theta - \theta - 3\theta = 6\theta$
    - ✓ $x_{ED} = \theta$: the cycle ED-DE is formed
      → $\Delta Z = c_{ED} \cdot \theta + c_{DE} \cdot \theta = 2\theta + 3\theta = 5\theta$
    - ✓ Summary of change rates per unit increase $\Delta Z = \begin{cases} -7 \\ 6 \\ 5 \end{cases}$
    - ✓ Since the goal is minimizing Z, the most desirable entering BV is x_{AC}.

The Network Simplex Method

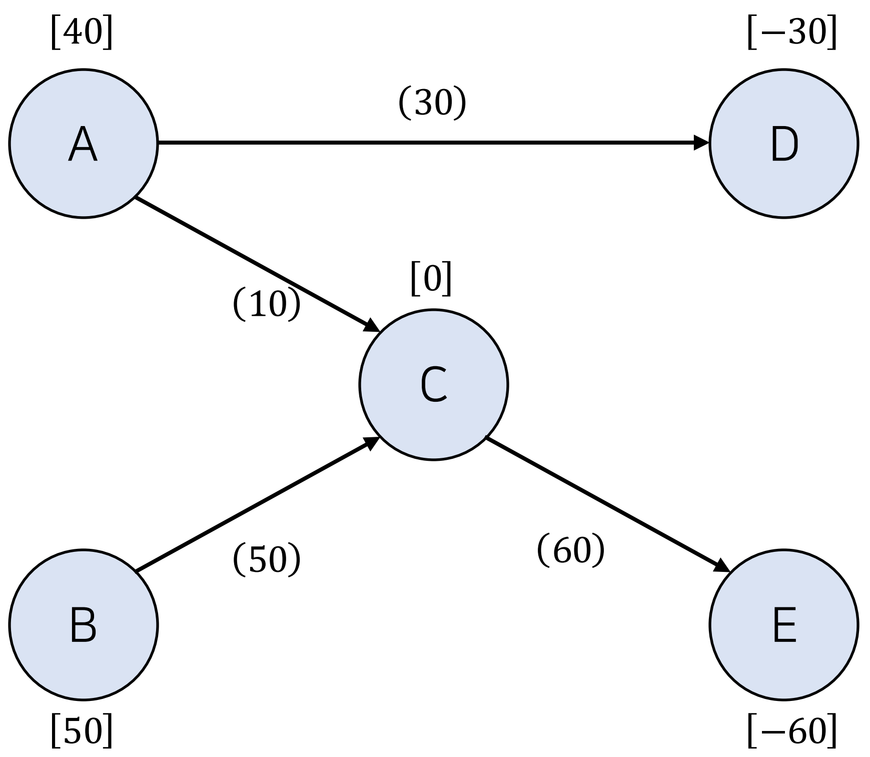
- Determining the Leaving BV and the new BFS
  - When choosing the entering BV, identify the cycle created and the resulting chain reaction as the entering BV increases.
  - Find the minimum value of θ at which any arc in the chain hits a non-negativity or upper-bound constraint.
    - ✓ $x_{AC} = \theta \leq u_{AC} = \infty$
    - ✓ $x_{CE} = 50 + \theta \leq u_{CE} = 80 \rightarrow \theta \leq 30$
    - ✓ $x_{DE} = 10 - \theta \geq 0 \rightarrow \theta \leq 10$
    - ✓ $x_{AD} = 40 - \theta \geq 0 \rightarrow \theta \leq 40$
  - Therefore, the leaving BV is x_{DE}, and we obtain the new BFS by applying $\theta = 10$.



The Network Simplex Method

- Determining the Leaving BV and the new BFS
  - Identify the cycle and the chain reaction induced by increasing the entering BV.
  - Among the resulting changes, find the smallest θ at which any arc hits a non-negativity or upper-bound constraint.
    - ✓ $x_{AC} = \theta \leq u_{AC} = \infty$
    - ✓ $x_{CE} = 50 + \theta \leq u_{CE} = 80 \rightarrow \theta \leq 30$
    - ✓ $x_{DE} = 10 - \theta \geq 0 \rightarrow \theta \leq 10$
    - ✓ $x_{AD} = 40 - \theta \geq 0 \rightarrow \theta \leq 40$
  - Therefore, the leaving BV is x_{DE} and we obtain a new BFS by applying $\theta = 10$.

A new BFS

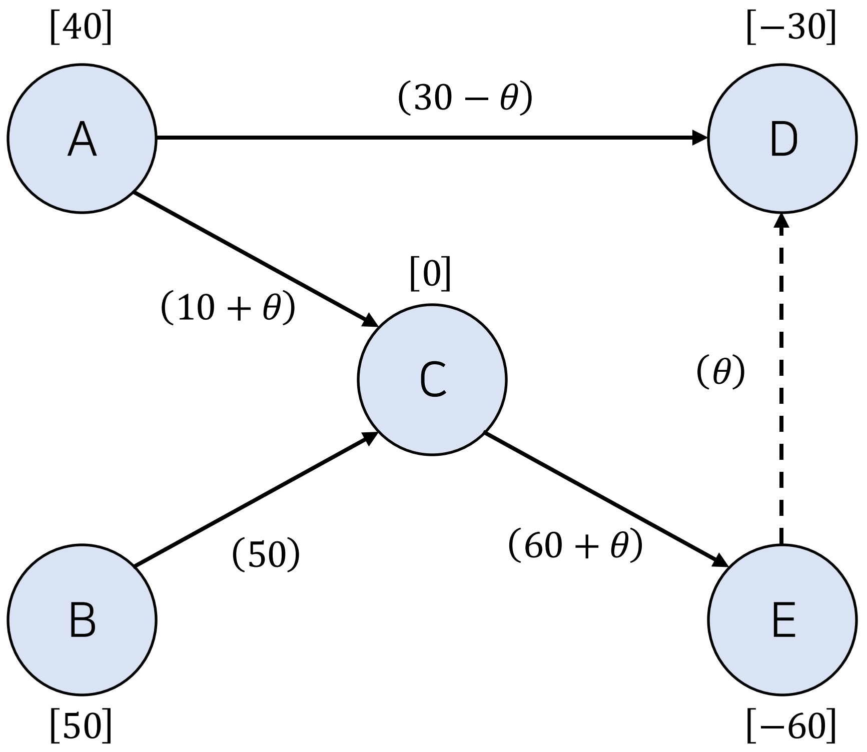


The Network Simplex Method

- Iterations of the Network Simplex Method

NBV	Created Cycle	ΔZ
B→A	BA-AC-BC	$-2\theta + 4\theta - 3\theta = -1$
D→E	DE-CE-AC-AD	$3\theta - \theta - 4\theta + 9\theta = 7\theta$
E→D	ED-AD-AC-CE	$2\theta - 9\theta + 4\theta + \theta = -2$

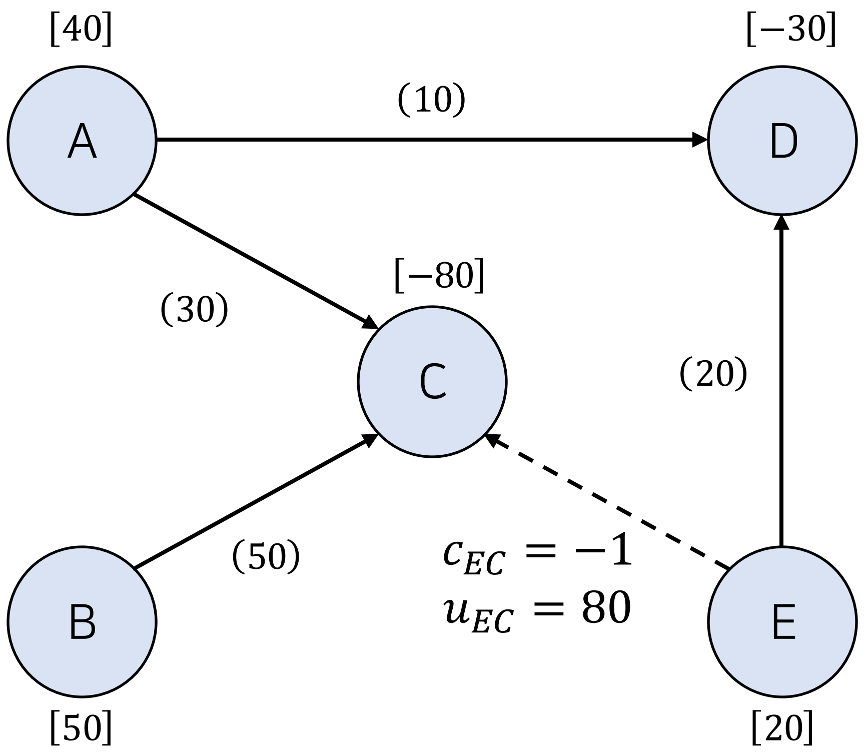
  - Entering BV: x_{ED}
  - Leaving BV: x_{CE}
    - ✓ $x_{ED} = \theta \leq u_{ED} = \infty$
    - ✓ $x_{AD} = 30 - \theta \geq 0 \rightarrow \theta \leq 30$
    - ✓ $x_{AC} = 10 + \theta \leq u_{AC} = \infty$
    - ✓ $x_{CE} = 60 + \theta \leq u_{CE} = 80 \rightarrow \theta \leq 20$
      - Upon reaching the upper bound, replace x_{CE} with $u_{CE} - y_{CE}$.



The Network Simplex Method

- Iterations of the Network Simplex Method
  - Entering BV: x_{ED}
  - Leaving BV: x_{CE}
    - ✓ $x_{ED} = 10 + \theta \leq u_{ED} = \infty$
    - ✓ $x_{AD} = 30 - \theta \geq 0 \rightarrow \theta \leq 30$
    - ✓ $x_{AC} = 10 + \theta \leq u_{AC} = \infty$
    - ✓ $x_{CE} = 60 + \theta \leq u_{CE} = 80 \rightarrow \theta \leq 20$
      - Upon reaching the upper bound, replace x_{CE} with $u_{CE} - y_{CE}$ and set $y_{CE} = 0$
      - Replace the arc $C \rightarrow E$ with $E \rightarrow C$, then $c_{EC} = -1, u_{EC} = 80$.
      - Update $b_C = [-80], b_E = [20]$
  - Determine the new BFS by applying $\theta = 20$

A new BFS

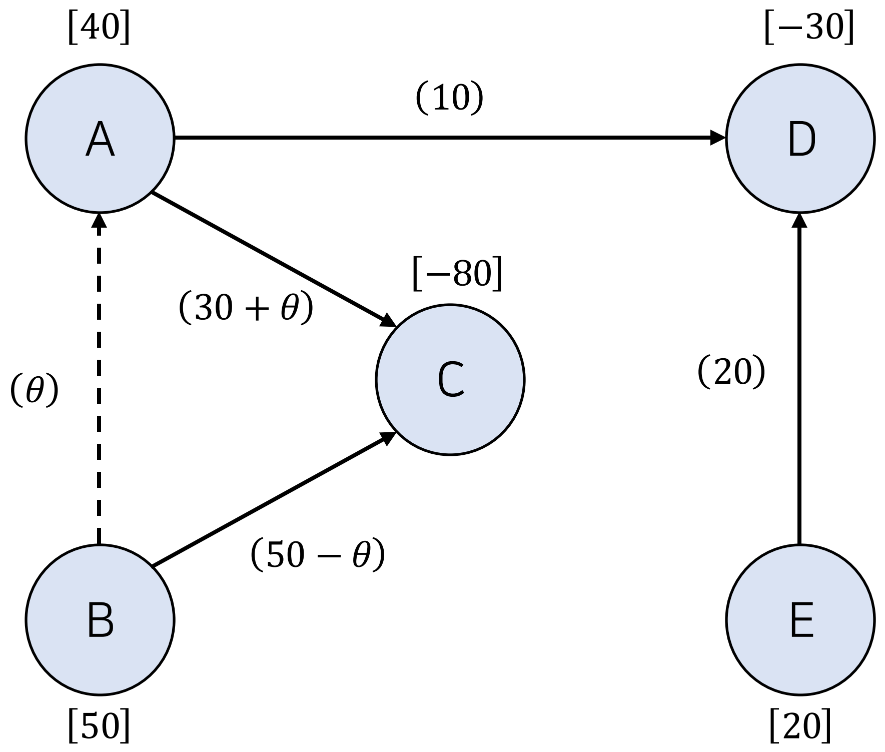


The Network Simplex Method

- Iterations of the Network Simplex Method

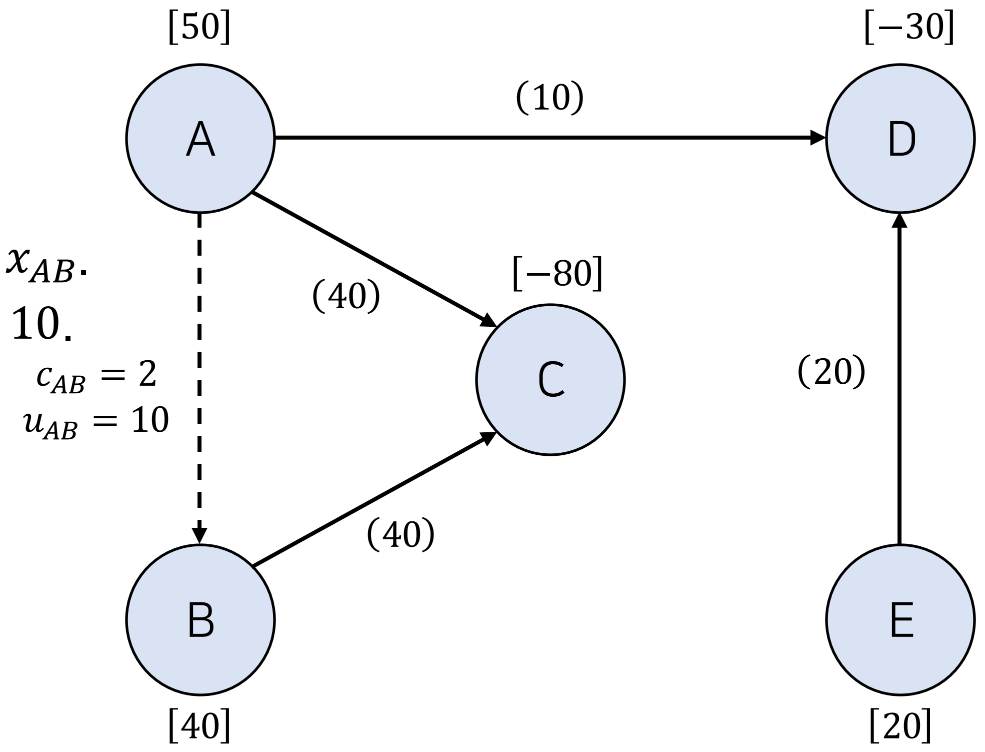
NBV	Created Cycle	ΔZ
B→A	BA-AC-BC	$-2\theta + 4\theta - 3\theta = -1$
D→E	DE-ED	$3\theta + 2\theta = 5\theta$
E→C	EC-AC-AD-ED	$-\theta - 4\theta + 9\theta - 2\theta = 2\theta$

- Entering BV: x_{BA}
- Leaving BV: x_{CE}
  - ✓ $y_{AB} = \theta \leq u_{BA} = 10 \rightarrow \theta \leq 10$
    - Upon reaching the upper bound, replace y_{AB} with $u_{BA} - x_{AB}$.
  - ✓ $x_{AC} = 30 + \theta \leq u_{AC} = \infty$
  - ✓ $x_{BC} = 50 - \theta \geq 0 \rightarrow \theta \leq 50$
- New BFS: Apply $\theta = 10$



The Network Simplex Method

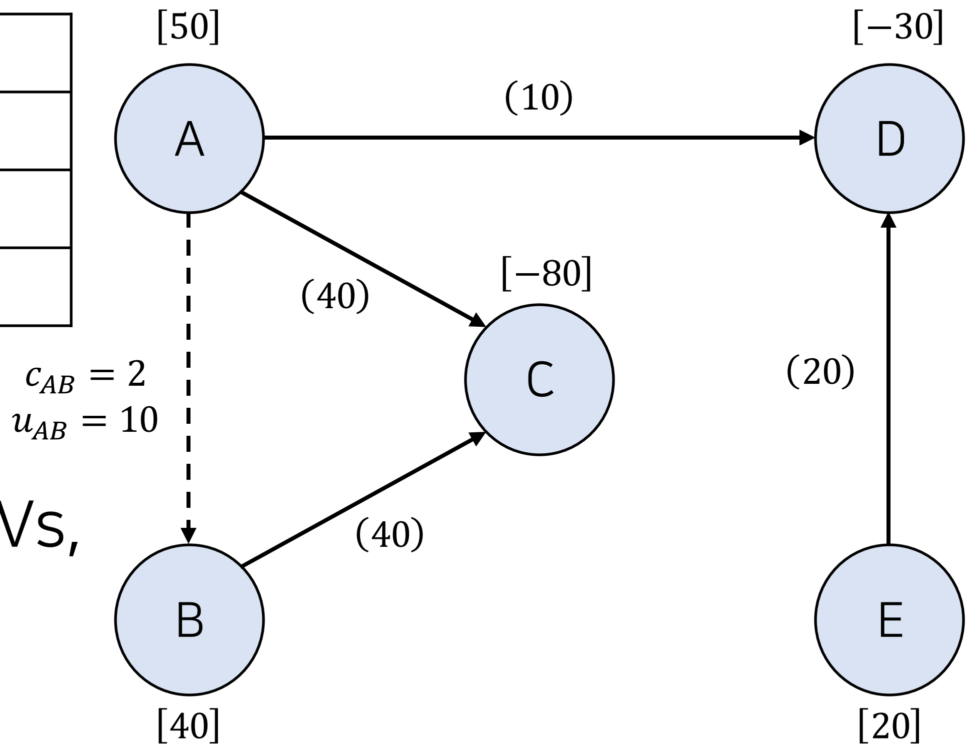
- Iterations of the Network Simplex Method
  - Entering BV: x_{BA}
  - Leaving BV: x_{CE}
    - ✓ $y_{AB} = \theta \leq u_{BA} = 10 \rightarrow \theta \leq 10$
      - Upon reaching the upper bound, replace y_{AB} with $u_{BA} - x_{AB}$.
      - After replacing arc $B \rightarrow A$ with $A \rightarrow B$, set $c_{AB} = 2, u_{AB} = 10$.
      - Update $b_A = [50], b_B = [40]$.
    - ✓ $x_{AC} = 30 + \theta \leq u_{AC} = \infty$
    - ✓ $x_{BC} = 50 - \theta \geq 0 \rightarrow \theta \leq 50$
  - New BFS: Apply $\theta = 10$



The Network Simplex Method

- Iterations of the Network Simplex Method

NBV	Created Cycle	ΔZ
A→B	AB-BC-AC	$2\theta + 3\theta - 4\theta = 1$
D→E	DE-ED	$3\theta + 2\theta = 5\theta$
E→C	EC-AC-AD-ED	$-\theta - 4\theta + 9\theta - 2\theta = 2\theta$



- Optimality check: For all candidate entering BVs, no improvement in Z is possible → optimal

Q&A