Large Scale Optimization

Emmanouil Zachariadis, Assistant Professor
Department of Management Science and Technology, Athens University of Economics and Business
E: ezach@aueb.gr
A: Evelpidon 47A and Lefkados 33 street, 9th floor, Room 906, 11362, Athens, Greece

T: +30-210-8203 674

3rd Combinatorial Structure: Element grouping → Selection Problems

3rd Combinatorial Structure: Selection Problems

Selection Problems: Given a set of elements, select some of the elements such than a specific objective function is optimized. Due to associated constraints it is impossible for all elements to be selected.

SET COVERING PROBLEM

- Set Covering Problem:
  - Input:

    Set U (Universe) consisting of m elements, $U = \{1,2,3, \dots, m\}$

    Set $S_{comp} = \{S_1, S_2, \dots, S_n\}$, with $S_i \subset U$

    $\exists S \in S_{comp}, : u \in S, \forall u \in U,$

    For every element $u \in U$, there is at least one element (set S) that belongs to set S_{comp} and includes u
  - Objective: Find the smallest number of sets $S_i \in S_{comp}$, such that all elements of U are included.

SET COVERING PROBLEM

- Objective function:

$$\min \sum_{S \in S_{comp}} x_S$$

- Constraints:

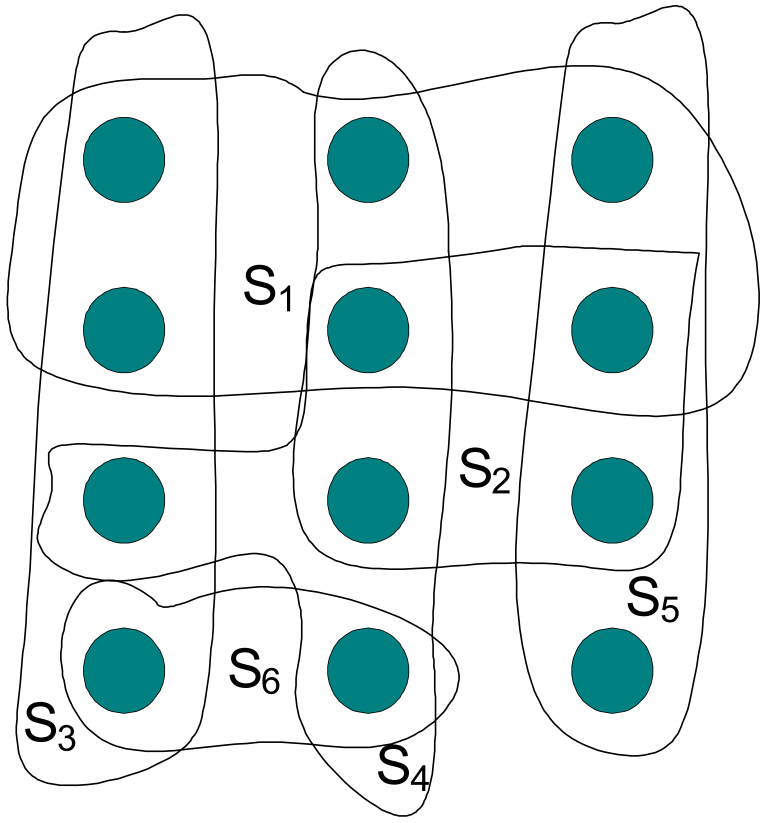
$$\sum_{S:u \in S} x_S \geq 1, \forall u \in U$$

For every element of U, find all subsets S, that contain it (S:u∈S). At least one of these subsets must be selected ($\sum_{S:u\in S} x_S \geq 1$).

$$x_S \in \{0,1\}, \forall S \in S_{comp}$$

A subset $\forall S \in S_{comp}$ whether it is selected or not

SET COVERING PROBLEM



A feasible solution with respect to the info of the figure is $S = \{S_1, S_4, S_5, S_3)$

The optimal solution with respect to the info of the figure is $S = \{S_3, S_4, S_5)$

SET COVERING PROBLEM EXAMPLE: Health Care Services

- There are 10 municipalities in the region of Athens that have requested a parked ambulance in a specific area of the municipality, in order to be able to serve emergencies in the area.
- The response of the Health Ministry is that the objective is not to assign an ambulance and each municipality, but to figure out a way to assign the ambulances to specific municipalities in a way that can serve the remaining areas as well in time.
- The quality of the road network containing the municipalities, as well as the inhabitants of each one is a determining factor for the ability of an assigned ambulance to serve neighboring municipalities.

SET COVERING PROBLEM EXAMPLE: Health Care Services

- An ambulance of municipality 1 can **also** serve municipality 2
- An ambulance of municipality 2 can **also** serve municipalities 1,3
- An ambulance of municipality 3 can **also** serve municipalities 4,5,6
- An ambulance of municipality 4 can **also** serve municipalities 2,3,9
- An ambulance of municipality 5 can **also** serve municipalities 4,6,8
- An ambulance of municipality 6 can **also** serve municipality 7
- An ambulance of municipality 7 can **also** serve municipality 8
- An ambulance of municipality 8 can **also** serve municipalities 5,7,10
- An ambulance of municipality 9 can **also** serve municipalities 4,8
- An ambulance of municipality 10 can **also** serve municipality 8

SET COVERING PROBLEM EXAMPLE: Health Care Services

- The objective of the Ministry of Health is to find the minimum number of ambulances required to be assigned to some municipalities such as the demand of all municipalities is covered.
- Design a Greedy Algorithm for solving the described problem.

SET COVERING PROBLEM EXAMPLE: Health Care Services

- Universe: A set of 10 municipalities, $U = \{1,2 , \ldots 10\}$
- Set S_{comp}: A set of subsets. Each subsets is related to one municipalities and contains the municipalities that can be served by assigning a vehicle to the related municipality.
- Therefore:

$$S_{comp} = \{S_1, S_1, \ldots , S_{10}\}$$

- where
  - S_1 : $\{1,2\}$
  - S_2 : $\{1,2,3\}$
  - S_3 : $\{3,4,5,6\}$
  - S_4 : $\{2,3,4,9\}$
  - S_5 : $\{4,5,6,8\}$
  - S_6 : $\{6,7\}$
  - S_7 : $\{7,8\}$
  - S_8 : $\{5,7,8,10\}$
  - S_9 : $\{4,8,9\}$
  - S_{10} : $\{8,10\}$

SET COVERING PROBLEM EXAMPLE: Health Care Services

- Objective Function
  $x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10}$

- Constraints:

$x1 + x2 \geq 1$ (Service for municipality 1)
$x1 + x2 + x4 \geq 1$ (Service for municipality 2)
$x2 + x3 + x4 \geq 1$ (Service for municipality 3)
$x3 + x4 + x5 + x9 \geq 1$ (Service for municipality 4)
$x3 + x5 + x8 \geq 1$ (Service for municipality 5)
$x3 + x5 + x6 \geq 1$ (Service for municipality 6)
$x6 + x7 + x8 \geq 1$ (Service for municipality 7)
$x5 + x7 + x8 + x9 + x10 \geq 1$ (Service for municipality 8)
$x4 + x9 \geq 1$ (Service for municipality 9)
$x8 + x10 \geq 1$ (Service for municipality 10)

$S1$: {1,2}
$S2$: {1,2,3}
$S3$: {3,4,5,6}
$S4$: {2,3,4,9}
$S5$: {4,5,6,8}
$S6$: {6,7}
$S7$: {7,8}
$S8$: {5,7,8,10}
$S9$: {4,8,9}
$S10$: {8,10}

SET COVERING PROBLEM EXAMPLE: Health Care Services

- **Solution structure**: A set of elements of set S_{comp}
- **Solution element**: A subset $S \in S_{comp}$, containing municipalities that can be served together
- **Selection criterion**: At each iteration select the subset S which maximizes the municipalities that are not covered by/serviced in the current partial solution.
- **Objective function**: The total number of elements S that are selected to cover all municipalities

SET COVERING PROBLEM EXAMPLE: Health Care Services

- **Iteration 1**: The are 4 sets (S_3,S_4,S_5,S_8) that satisfy the "selection criterion". We stochastically select set S_3, which means that the first ambulance will be assigned to municipality 3 and serve municipalities 3,4,5,6. Therefore, the partial solution is $\boldsymbol{s} = \{\boldsymbol{S_3}\}$.
- **Iteration 2**: Set $\boldsymbol{S_8}$ satisfies the "selection criterion". Therefore, the partial solution is $\boldsymbol{s} = \{\boldsymbol{S_3}, \boldsymbol{S_8}\}$.

SET COVERING PROBLEM EXAMPLE: Health Care Services

- **Iteration 3**: The are 2 sets ($\boldsymbol{S_1}, \boldsymbol{S_2}, \boldsymbol{S_4}$) that satisfy the “selection criterion”. We stochastically select set $\boldsymbol{S_1}$. Therefore, the partial solution is $\boldsymbol{s} = \{\boldsymbol{S_3}, \boldsymbol{S_8}, \boldsymbol{S_1}\}$.
- **Iteration 4**: The are 2 sets ($\boldsymbol{S_4}, \boldsymbol{S_9}$) that satisfy the “selection criterion”. We stochastically select set $\boldsymbol{S_4}$. Therefore, the partial solution is $\boldsymbol{s} = \{\boldsymbol{S_3}, \boldsymbol{S_8}, \boldsymbol{S_1}, \boldsymbol{S_4}\}$.

Therefore, the solution is $\boldsymbol{s} = \{\boldsymbol{S_3}, \boldsymbol{S_8}, \boldsymbol{S_1}, \boldsymbol{S_4}\}$and the objective function value is 4, therefore 4 ambulances need to be assigned to municipalities 3,8,1 and 4, such that all 10 municipalities are served.

Staff Selection/Recruitment

- An organizations has decided to recruit employees for a specific project. Each employ has multiple skills. The project requires the following skills to be completed:
    1. Communication
    2. Operation research experience
    3. IT experience
    4. Data mining
    5. Management
    6. Leadership
    7. Creative thinking
    8. Planning and organization skills
    9. Teamwork
    10. Personal values

Staff Selection/Recruitment

- There are 10 candidates, each one wit the following skills:

  Candidate 1 has skills 1 and 2,

  Candidate 2 has skills 1,2 and 3,

  Candidate 3 has skills 3,4,5 and 6,

  Candidate 4 has skills 2,3,4 and 9,

  Candidate 5 has skills 4,5,6 and 8,

  Candidate 6 has skills 6 and 7,

  Candidate 7 has skills 7 and 8,

  Candidate 8 has skills 5,7,8 and 10,

  Candidate 9 has skills 4,8,9 and 10 and

  Candidate 10 has skills 8 and 10.

- The organizations asks to create a greedy algorithm for finding the optimal combination of employees to complete this project. The objective is to hire the minimum required number of employees in order to save costs.

Staff Selection/Recruitment

- **Solution structure**: Number of groups of (sets S_j) of the 10 skills.
- **Solution element**: A group S_j of skills of an employ j.
- **Selection criterion**: In each iteration select a group S_j that included the most non covered skills from the currently selected candidates until all 10 skills are covered.
- **Objective function**: The total number of candidates that will be hired to work on the project.

Set covering problem representation as the ambulance selection problem. Same data, same Greedy algorithms and consequently same solution.

Staff Selection/Recruitment

- How would the problem setting be modified if the candidates require different salary and the objective functions was to minimize the total salaries?

Weighted Set Covering Problem

- Objective function:

$$\min \sum_{S \in S_{comp}} w_S \cdot x_S$$

- Constraints:

$$\sum_{S:u \in S} x_S \geq 1, \forall u \in U$$

For every element of U, find all subsets S, that contain it (S:u∈S).
At least one of these subsets must be selected ($\sum_{S:u\in S} x_S \geq 1$).

$$x_S \in \{0,1\}, \forall S \in S_{comp}$$

A subset $\forall S \in S_{comp}$ whether it is selected or not

Weighted Set Covering Problem

Universe: Skills($U = \{1,2,3, \ldots, 10\}$)

- Candidate employees

$$S_{comp} = \{S_1, S_1, \ldots, S_{10}\}$$

- where
  - $S_1 : \{1,2\}$
  - $S_2 : \{1,2,3\}$
  - $S_3 : \{3,4,5,6\}$
  - $S_4 : \{2,3,4,9\}$
  - $S_5 : \{4,5,6,8\}$
  - $S_6 : \{6,7\}$
  - $S_7 : \{7,8\}$
  - $S_8 : \{5,7,8,10\}$
  - $S_9 : \{4,8,9\}$
  - $S_{10} : \{8,10\}$

Weighted Set Covering Problem

- Objective function
  $1000 \cdot x_1 + 700 \cdot x_2 + 800 \cdot x_3 + 0 \cdot x_4 + 1150 \cdot x_5 + 780 \cdot x_6 + 800 \cdot x_7$
  $+ 1200 \cdot x_8 + 1500 \cdot x_9 + 750 \cdot x_9 + 1200 \cdot x_{10}$

- Constraints:

$$x1 + x2 \geq 1$$
$$x1 + x2 + x4 \geq 1$$
$$x2 + x3 + x4 \geq 1$$
$$x3 + x4 + x5 + x9 \geq 1$$
$$x3 + x5 + x8 \geq 1$$
$$x3 + x5 + x6 \geq 1$$
$$x6 + x7 + x8 \geq 1$$
$$x5 + x7 + x8 + x9 + x10 \geq 1$$
$$x4 + x9 \geq 1$$
$$x8 + x10 \geq 1$$

$S1$: {1,2}
$S2$: {1,2,3}
$S3$: {3,4,5,6}
$S4$: {2,3,4,9}
$S5$: {4,5,6,8}
$S6$: {6,7}
$S7$: {7,8}
$S8$: {5,7,8,10}
$S9$: {4,8,9}
$S10$: {8,10}

Weighted Set Covering Problem: Greedy Algorithm

- Step 1:
  If the weight of a candidate $S = S_{comp}$ is 0, then

  $$x_s = 1$$

  Ignore (Delete) all constraints containing x_s

  Comment: This selection has zero cost, it is free.
  Therefore, all constraints that include this candidate have no reason to remain, as the respected skills are already covered by the free employee.

Weighted Set Covering Problem: Greedy Algorithm

- Step 2:
  If the weight of a candidate $S \in S_{comp}$ is not 0
  AND
  The candidate is not included in any constraints, then

$$x_s = 0$$

Comment: We would pay an employee that does not cover any uncovered skill. No reason to be hired.

Weighted Set Covering Problem: Greedy Algorithm

- Step 3:
  Iteratively and until all constraints are deleted:

  1. For all $S \in S_{comp}$ that have not yet received x_S, calculate:

$$v_S = \frac{c_S}{d_S}$$

  with d_S the number of costraints containing S

  2. Find candidate S^* with the minimum value v_S

  3. Set $x_{S^*} = 1$

  4. Erase all constraints containing S^*

  5. Step 2 (Erase all non-necessary candidate employees)

Weighted Set Covering Problem

- Step 1

$$1000 \cdot x_1 + 700 \cdot x_2 + 800 \cdot x_3 + \mathbf{0} \cdot \boldsymbol{x_4} + 1150 \cdot x_5 + 780 \cdot x_6 + 800 \cdot x_7 + 1200 \cdot x_8 + 1500 \cdot x_9 + 750 \cdot x_9 + 1200 \cdot x_{10}$$

$$\boldsymbol{x_4 = 1}$$

- Constraints:

$x1 + x2 \geq 1$ (Covering skill 1)
~~$x1 + x2 + x4 \geq 1$~~ (Covering skill 2)
~~$x2 + x3 + x4 \geq 1$~~ (Covering skill 3)
~~$x3 + x4 + x5 + x9 \geq 1$~~ (Covering skill 4)
$x3 + x5 + x8 \geq 1$ (Covering skill 5)
$x3 + x5 + x6 \geq 1$ (Covering skill 6)
$x6 + x7 + x8 \geq 1$ (Covering skill 7)
$x5 + x7 + x8 + x9 + x10 \geq 1$ (Covering skill 8)
~~$x4 + x9 \geq 1$~~ (Covering skill 9)
$x8 + x10 \geq 1$ (Covering skill 10)

$S1$: {1,2}
$S2$: {1,2,3}
$S3$: {3,4,5,6}
$S4$: {2,3,4,9}
$S5$: {4,5,6,8}
$S6$: {6,7}
$S7$: {7,8}
$S8$: {5,7,8,10}
$S9$: {4,8,9}
$S10$: {8,10}

Weighted Set Covering Problem

- Step 2

There is no redundant candidate that is not contained in any constraints, therefore we proceed to Step 3

- Constraints :

$x1 + x2 \geq 1$ (Covering skill 1)
$x3 + x5 + x8 \geq 1$ (Covering skill 5)
$x3 + x5 + x6 \geq 1$ (Covering skill 6)
$x6 + x7 + x8 \geq 1$ (Covering skill 7)
$x5 + x7 + x8 + x9 + x10 \geq 1$ (Covering skill 8)
$x8 + x10 \geq 1$ (Covering skill 10)

$S1$: {1,2}
$S2$: {1,2,3}
$S3$: {3,4,5,6}
$S4$: {2,3,4,9}
$S5$: {4,5,6,8}
$S6$: {6,7}
$S7$: {7,8}
$S8$: {5,7,8,10}
$S9$: {4,8,9}
$S10$: {8,10}

Weighted Set Covering Problem

- Step 3 – Iteration 1

Calculate $v_S = \frac{c_s}{d_s}$

$$\begin{aligned} v_1 &= 1000/1 = 1000 \\ v_2 &= 700/1 = 700 \\ v_3 &= 800/2 = 400 \\ v_5 &= 1150/3 = 383.3 \\ v_6 &= 780/2 = 390 \\ v_7 &= 800/2 = 400 \\ \boldsymbol{v_8} &\boldsymbol{= 1200/4 = 300} \\ v_9 &= 750/1 = 750 \\ v_{10} &= 1200/2 = 600 \end{aligned}$$

$$\begin{aligned} &1000 \cdot x_1 + 700 \cdot x_2 \\ &+ 800 \cdot x_3 + \mathbf{0} \cdot \boldsymbol{x_4} \\ &+ 1150 \cdot x_5 + 780 \cdot x_6 \\ &+ 800 \cdot x_7 + 1200 \cdot x_8 \\ &+ 1500 \cdot x_9 + 750 \cdot x_9 \\ &+ 1200 \cdot x_{10} \end{aligned}$$

- Constraints:

$$\begin{gathered} x1 + x2 \geq 1 \\ x3 + x5 + x8 \geq 1 \\ x3 + x5 + x6 \geq 1 \\ x6 + x7 + x8 \geq 1 \\ x5 + x7 + x8 + x9 + x10 \geq 1 \\ x8 + x10 \geq 1 \end{gathered}$$

Weighted Set Covering Problem

- Step 3 – Iteration 1
  Therefore, we choose 8, $\boldsymbol{x_8 = 1}$
  Delete all constraints containing candidate 8 $(x8)$

$x1 + x2 \geq 1$ (Covering skill 1)
~~$x3 + x5 + x8 \geq 1$~~ (Covering skill 5)
$x3 + x5 + x6 \geq 1$ (Covering skill 6)
~~$x6 + x7 + x8 \geq 1$~~ (Covering skill 7)
~~$x5 + x7 + x8 + x9 + x10 \geq 1$~~ (Covering skill 8)
~~$x8 + x10 \geq 1$~~ (Covering skill 10)

For each candidate s that is not contained in the constraints, we set $x_s = 0$,
Therefore,

$$x_7 = 0, x_9 = 0, x_{10} = 0$$

Weighted Set Covering Problem

- Step 3 – Iteration 2

Calculate $v_S = \frac{c_s}{d_s}$

$$v_1 = 1000/1 = 1000$$
$$\boldsymbol{v_2 = 700/1 = 700}$$
$$v_3 = 800/1 = 800$$
$$v_5 = 1150/1 = 1150$$
$$v_6 = 780/1 = 780$$

$$1000 \cdot x_1 + 700 \cdot x_2 + 800 \cdot x_3 + \mathbf{0} \cdot \boldsymbol{x_4} + 1150 \cdot x_5 + 780 \cdot x_6 + 800 \cdot x_7 + 1200 \cdot x_8 + 1500 \cdot x_9 + 750 \cdot x_9 + 1200 \cdot x_{10}$$

- Constraints:

$$x1 + x2 \geq 1$$
$$x3 + x5 + x6 \geq 1$$

Weighted Set Covering Problem

- Step 3 – Iteration 2
  Therefore, we choose 2, $\boldsymbol{x_2} = \mathbf{1}$
  Delete all constraints containing candidate 8 2

~~$x1 + x2 \geq 1$~~ (Covering skill 1)
$x3 + x5 + x6 \geq 1$ (Covering skill 6)

For each candidate s that is not contained in the constraints, we set $x_s = 0$,
Therefore,

$$x_1 = 0$$

Weighted Set Covering Problem

- Step 3 – Iteration 3

Calculate $v_S = \frac{c_s}{d_s}$

$$v_3 = 800$$
$$v_5 = 1150/1 = 1150$$
$$\boldsymbol{v_6 = 780/1 = 780}$$

$$1000 \cdot x_1 + 700 \cdot x_2 + 800 \cdot x_3 + \mathbf{0} \cdot \boldsymbol{x_4} + 1150 \cdot x_5 + 780 \cdot x_6 + 800 \cdot x_7 + 1200 \cdot x_8 + 1500 \cdot x_9 + 750 \cdot x_9$$

- Constraints:

$$x3 + x5 + x6 \geq 1$$

Weighted Set Covering Problem

- Step 3 – Iteration 2
  Therefore, we choose 6, $\boldsymbol{x_6 = 1}$
  Delete all constraints containing candidate 8 6

  $$\cancel{x3 + x5 + x6 \geq 1}$$

  Algorithm termination (All constraints are deleted, i.e., there are no skills that are not covered)

- Final solution:

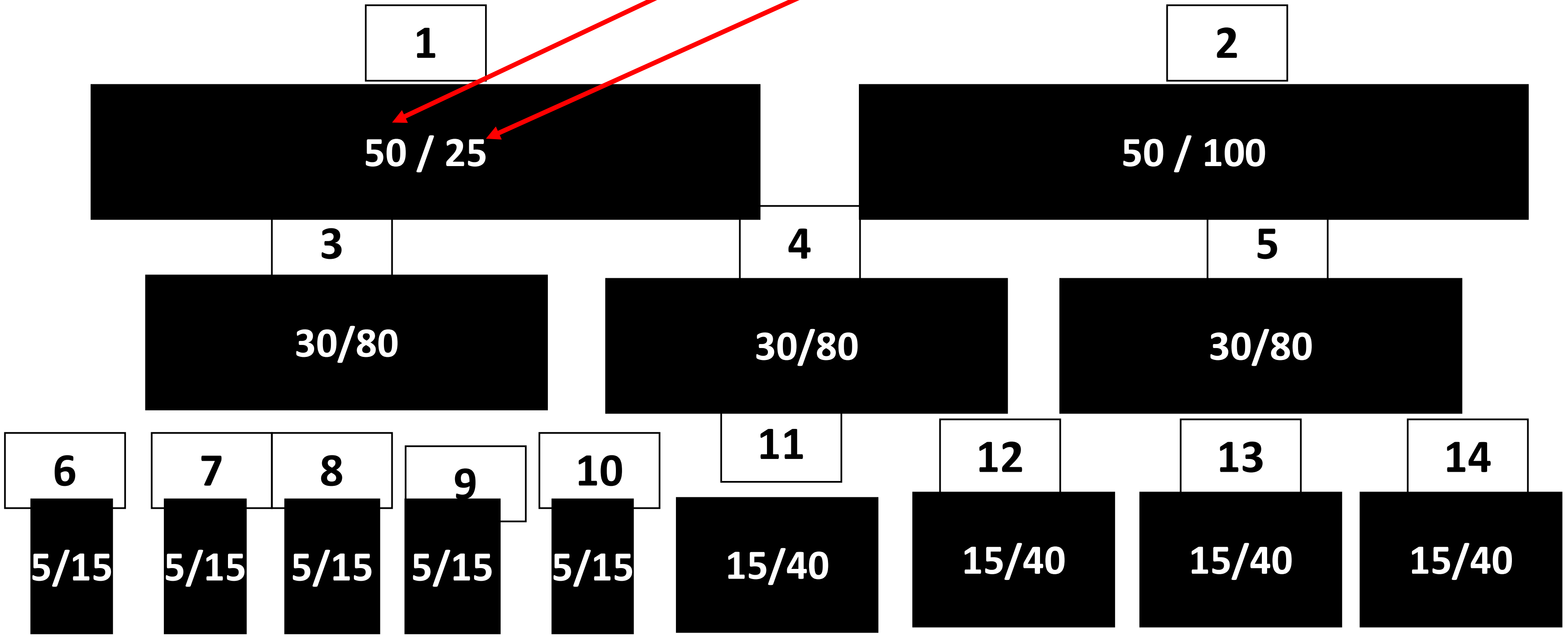
  $$s = \{S_2, S_4, S_6, S_8\}$$

  Objective function:

  $$z(s) = 700 + 0 + 780 + 1200 = 2680$$

Problem

Given a set of 14 containers, each one having a specific weight and specific profit, determine the containers that will be transported by a ship with capacity of 120 weight units. The objective is to maximize the total value of the containers transported.



Solution by Greedy Algorithm

- Solution structure: A number of selected containers.
- Solution element: A container added to a partial solution.
- Selection criterion: At each iteration select the container with the highest profit/size rate if it respects the ship capacity

Comment: Sort containers in decreasing profit/size order, and iterate through the sorted list

- Objective function: Sum the value of the selected containers.

Solution by Greedy Algorithm

–Iteration 1. As more that one containers satisfy the ”Selection Criterion” in the best way (the minimum fraction for Iteration 1 has containers 6,7,8,9 and 10), we randomly select one of these. We randomly select container 6, then the partial solution is S:{6}.

–Iteration 2. Similarly, S:{6,10}.

–Iteration 3. Similarly, S:{6,10,7}.

–Iteration 4. Similarly, S:{6,10,7,8}.

–Iteration 5. Similarly, S:{6,10,7,8,9}.

Solution by Greedy Algorithm

- Iteration 6. Similarly, S:{6,10,7,8,9,11}.
- Iteration 7. Similarly, S:{6,10,7,8,9,11,12}.
- Iteration 8. Similarly, S:{6,10,7,8,9,11,12,13}.
- Iteration 9. Similarly, S:{6,10,7,8,9,11,12,13,14}.
- Iteration 10. Similarly, S:{6,10,7,8,9,11,12,13,14,3}.

The total value of the selected containers is

$$z(s) = 15 + 15 + 15 + 15 + 15 + 40 + 40 + 40 + 40 + 80 = 315$$

Strategic Problem

- Consider a road network with various nodes (crossroads)
- We need to design a bus route from point A to point B such that the travel time between these locations is minimized.
- We do not need to visit all network/graph nodes (!)
  - Just transit from node A to node B
- Each pair of nodes is connected by an arc/edge
- Every arc/edge is associated with the time which is required to travel between the two corresponding nodes.

Problem

- What is the solution structure of the problem?
- How can we solve the problem?

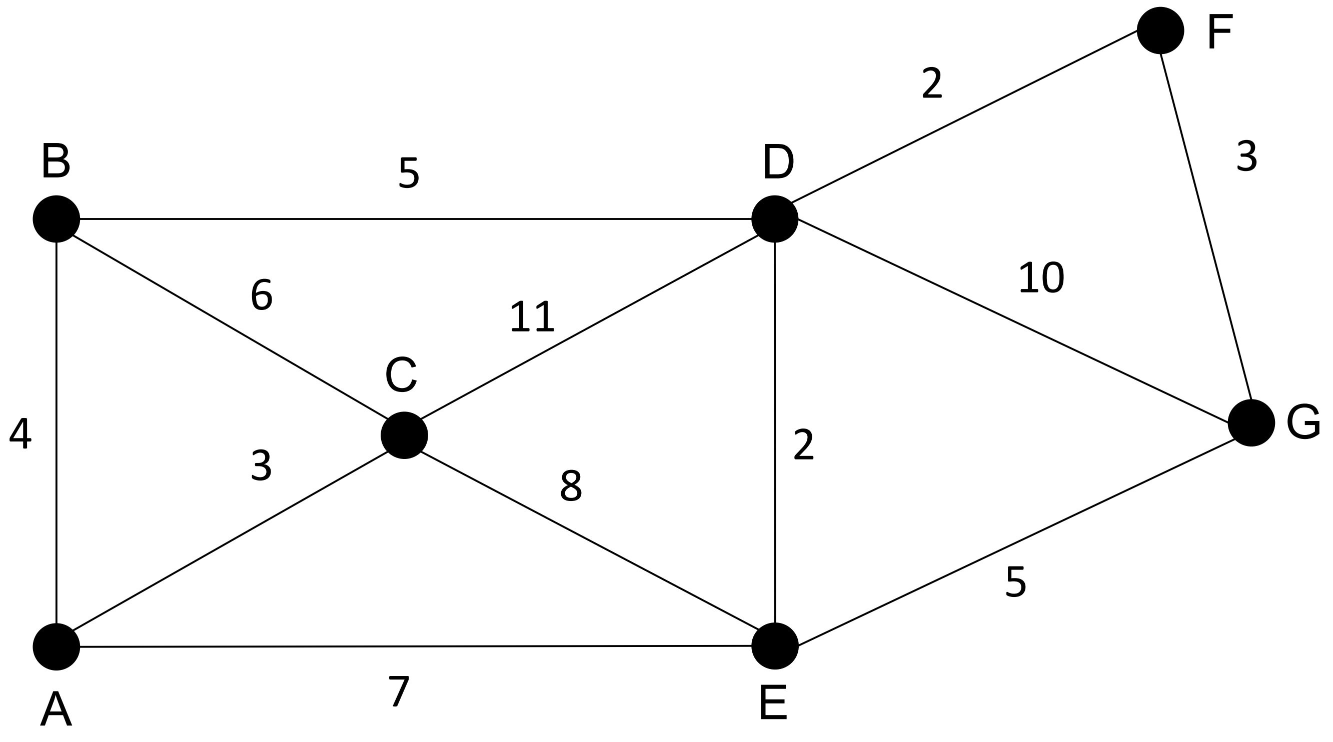
Shortest Path Problems

- This problem is known as Shortest Path Problem
- The problem is classified in a broad category of graph exploring problems
- It plays a crucial role for the following applications:
  - GPS Navigation
  - Distance matrix construction
  - Required for column generation methodologies of the Operations Research field
- Most familiar example
  - Google Maps

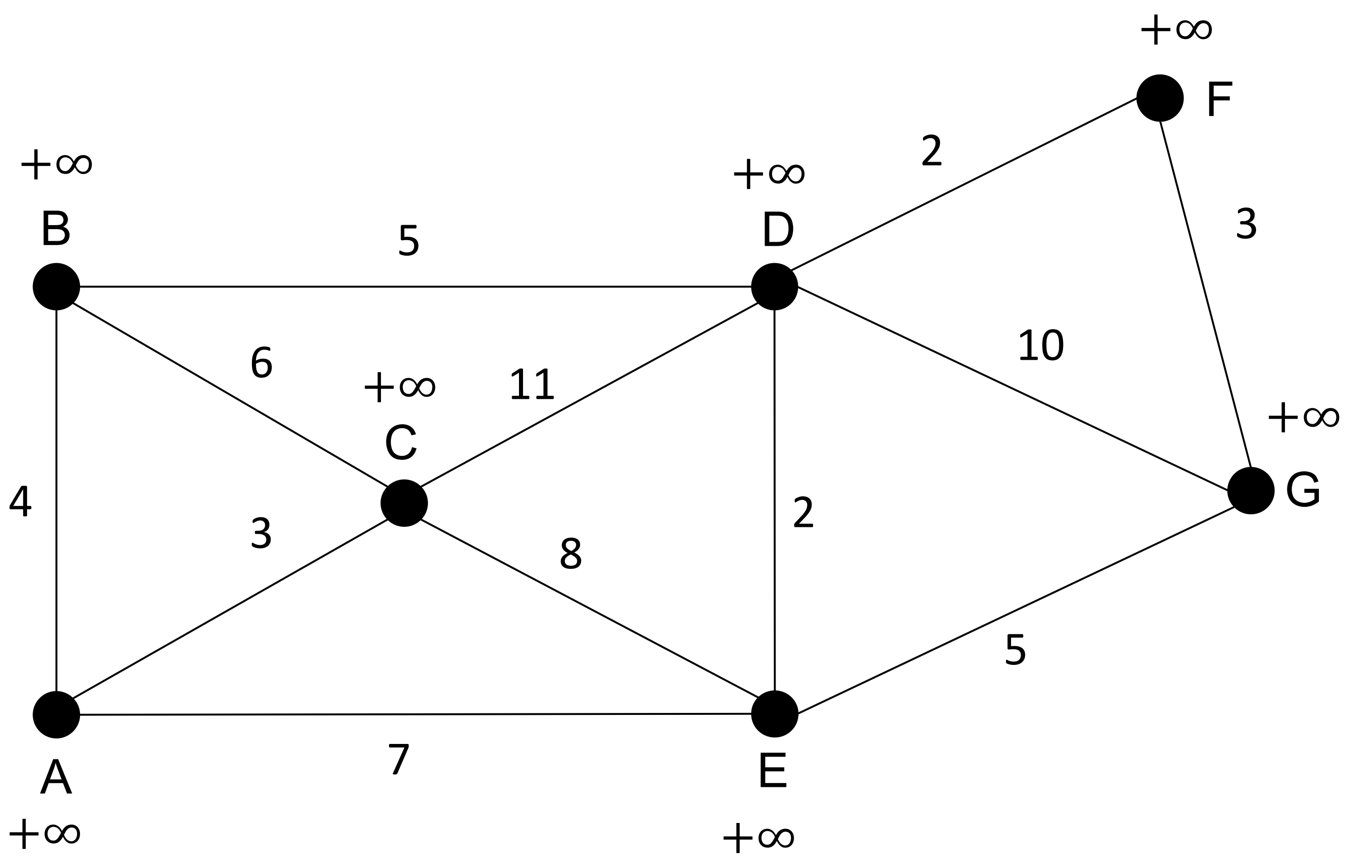
Shortest Path Problems

- For Shortest Path Problems there is a Greedy Algorithms that guarantees the construction of the optimal solution
- This algorithm is named the Dijkstra algorithm
- Algorithm steps
  - Initialize nodes $\forall i \epsilon N$ with cost $dist_i = +\infty$
  - Set cost of starting node equal to $dist_s = 0$
  - Visited Nodes $V \leftarrow \{s\}$
  - Iteratively, while $t \notin V$ (not reached the destination node)
    - Select a node n with the minimum cost among the non yet visited nodes
      - Set the cost of node n equal to this minimum cost
      - $V \leftarrow V \cup \{s\}$
    - Update the cost for all nodes that are connected n and have not been visited yet
      - If the cost of visiting n via s is less than the current cost, set this value as the cost of the neighbor

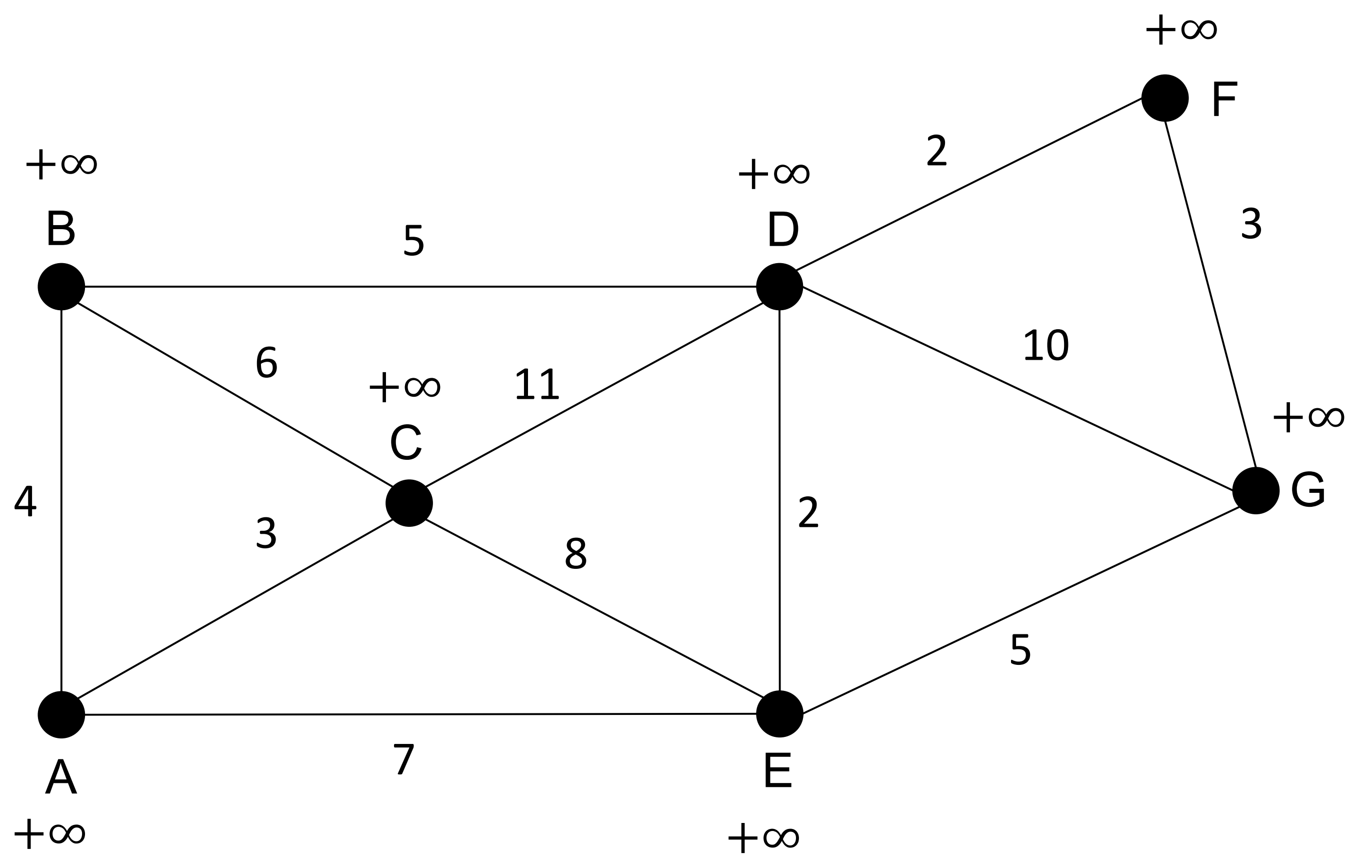
Input



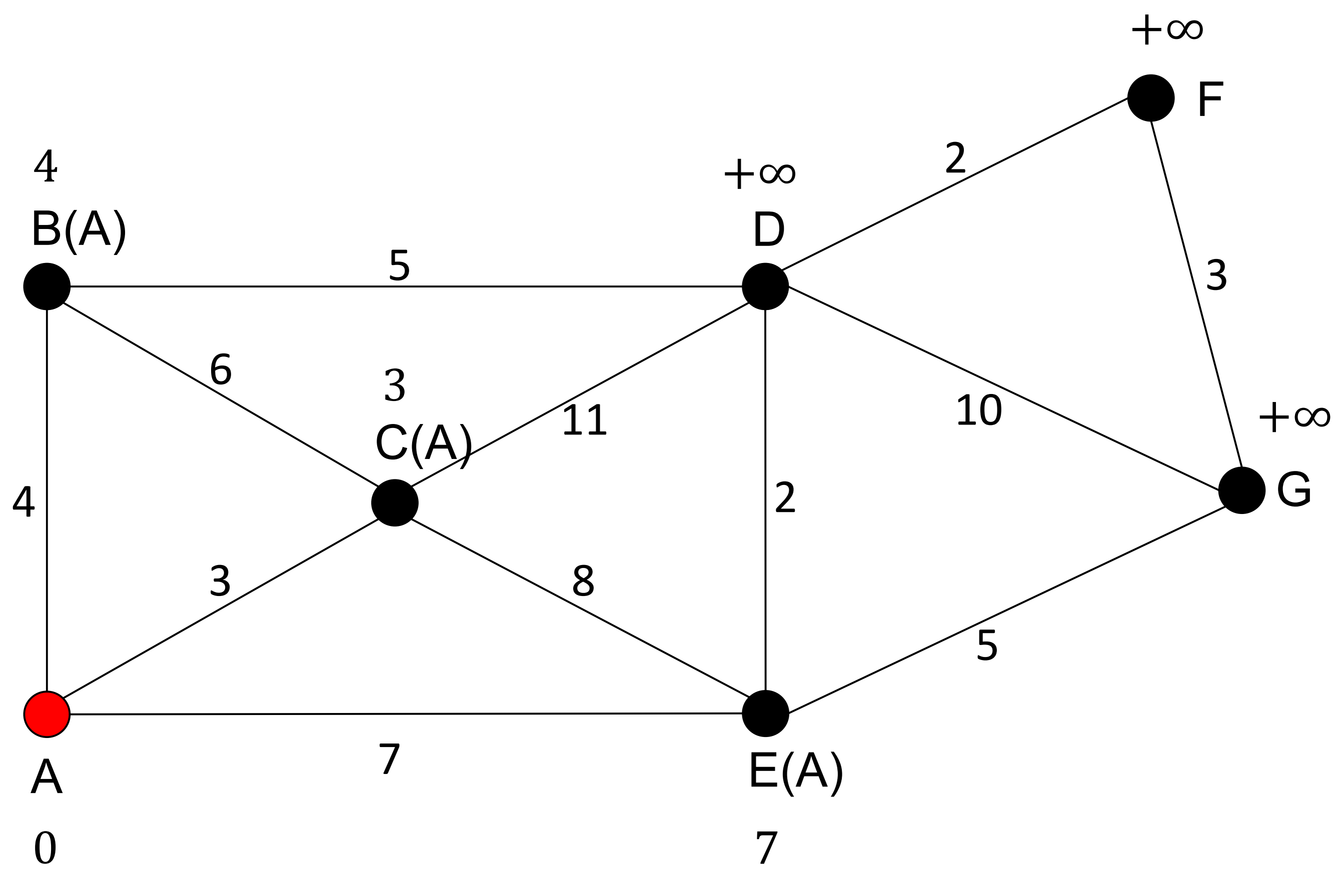
Dijkstra Initialization



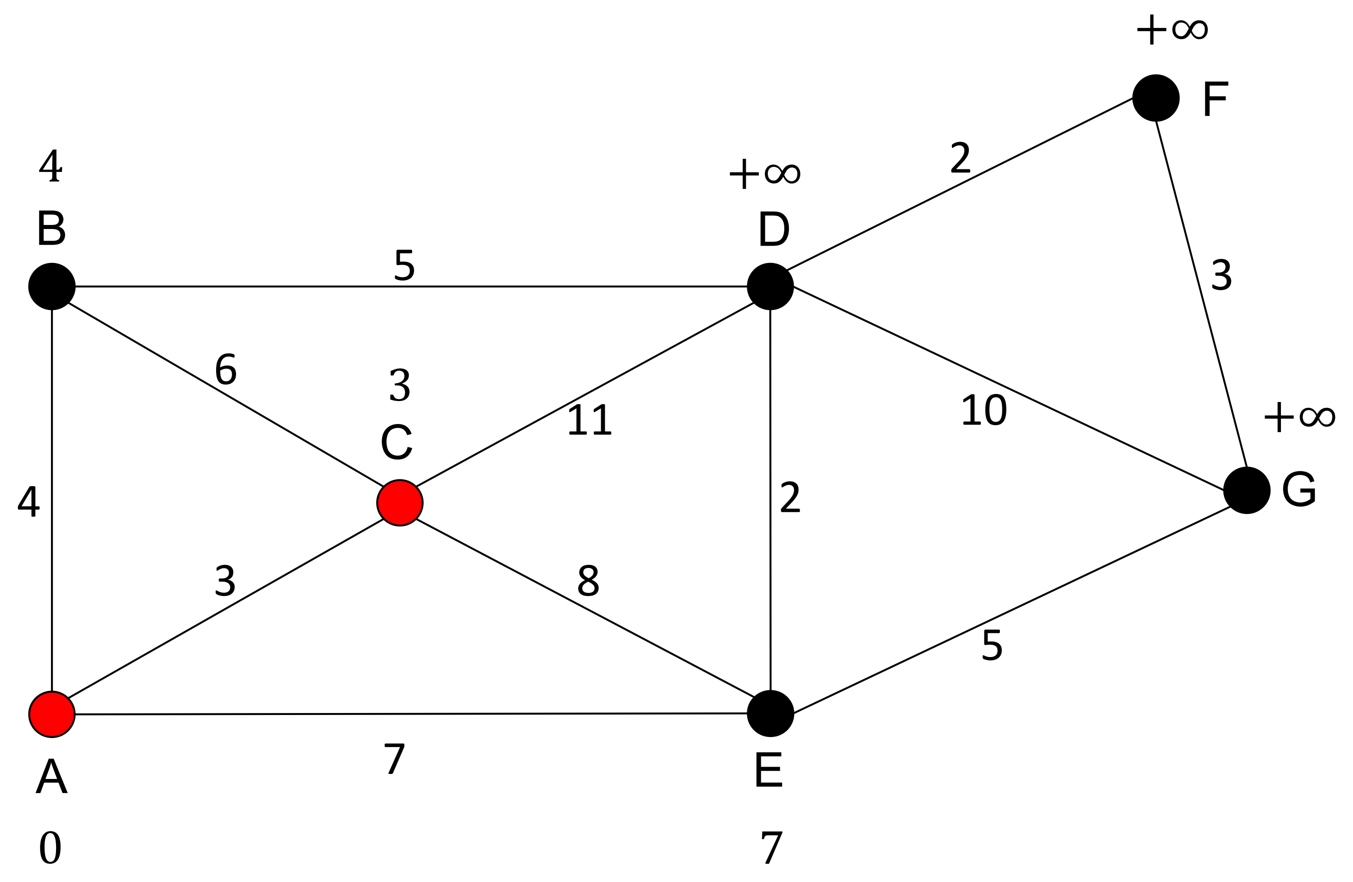
Dijkstra Greedy Algorithm



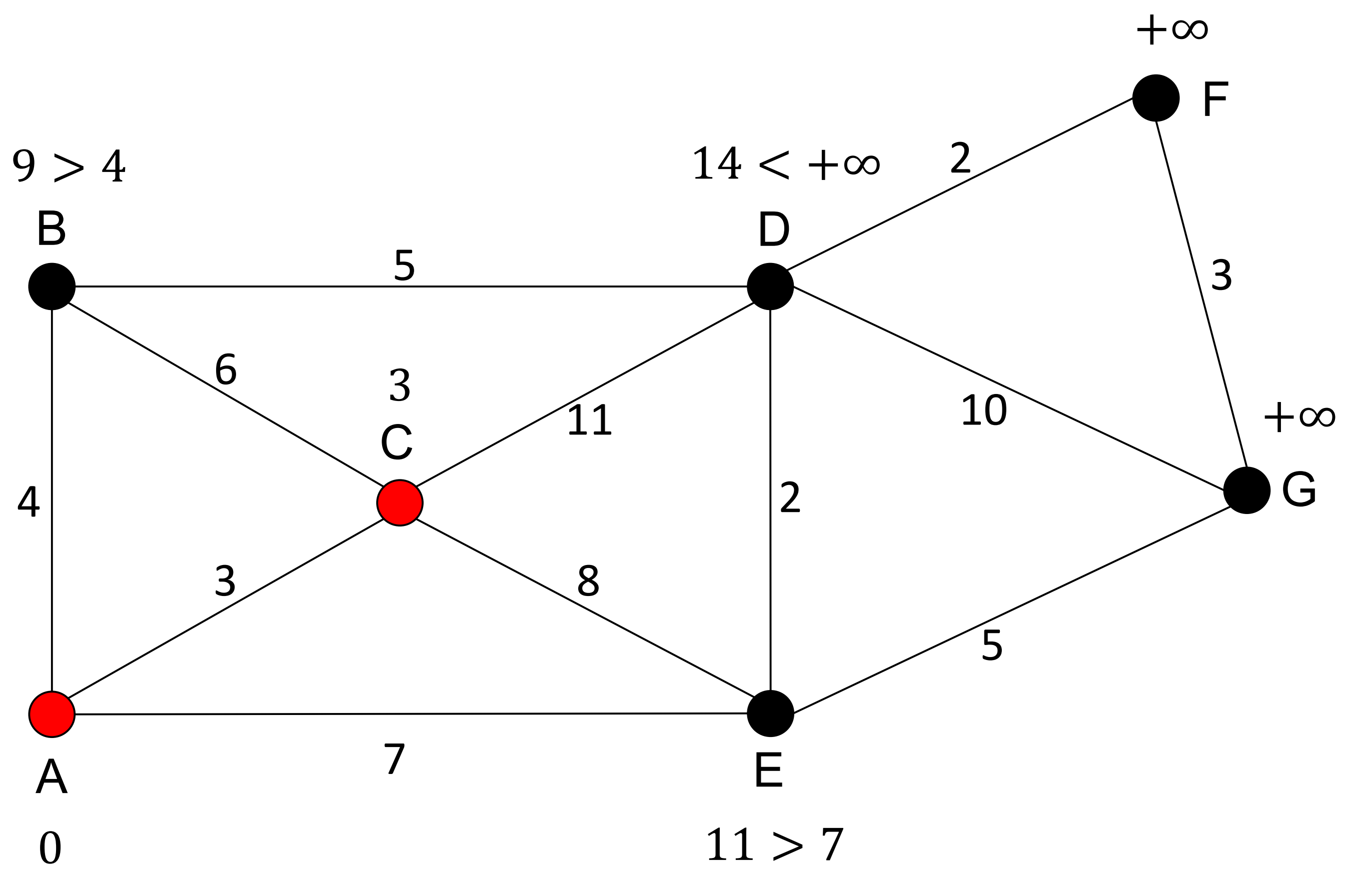
Dijkstra Greedy Algorithm



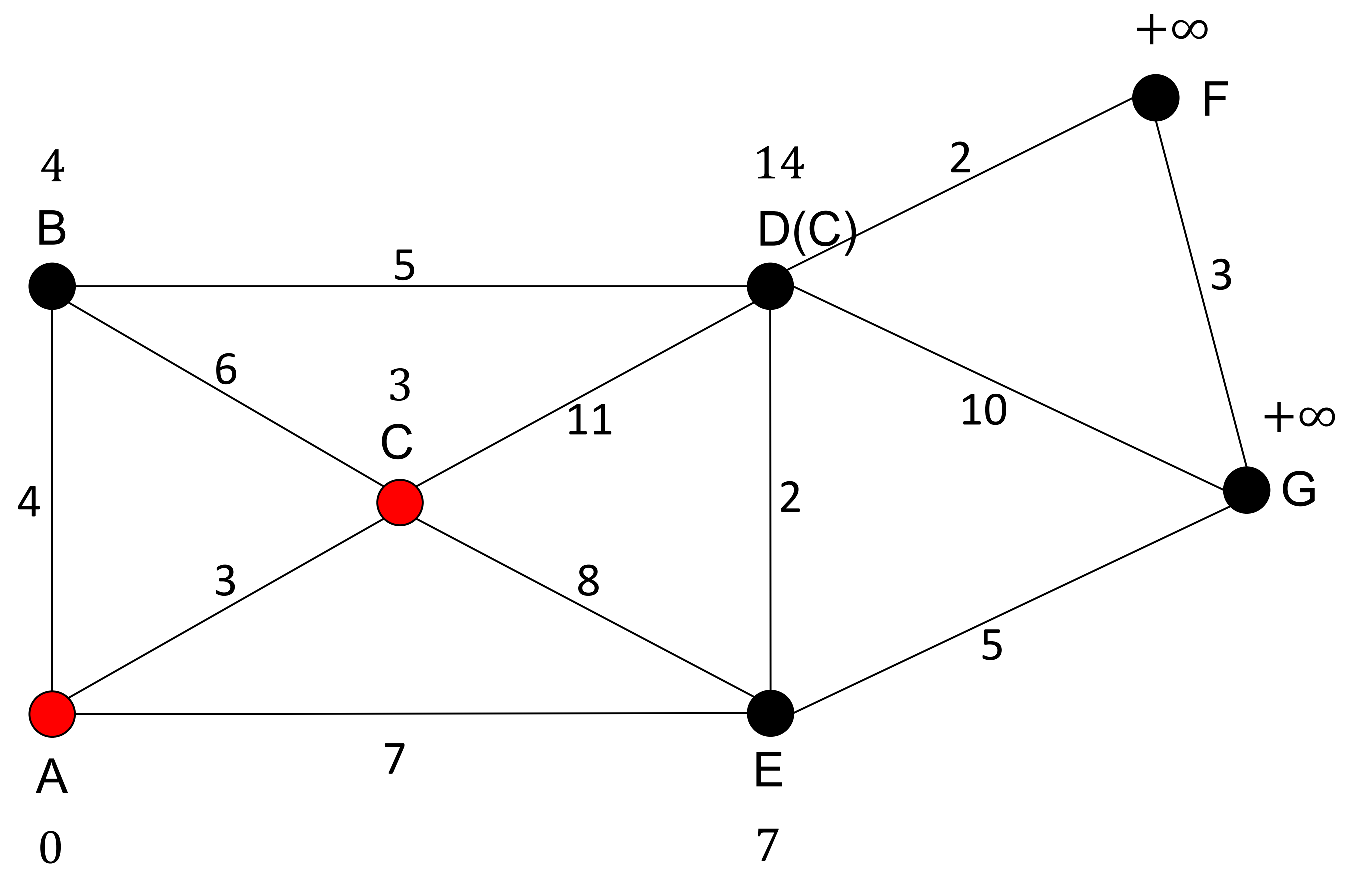
Dijkstra Greedy Algorithm



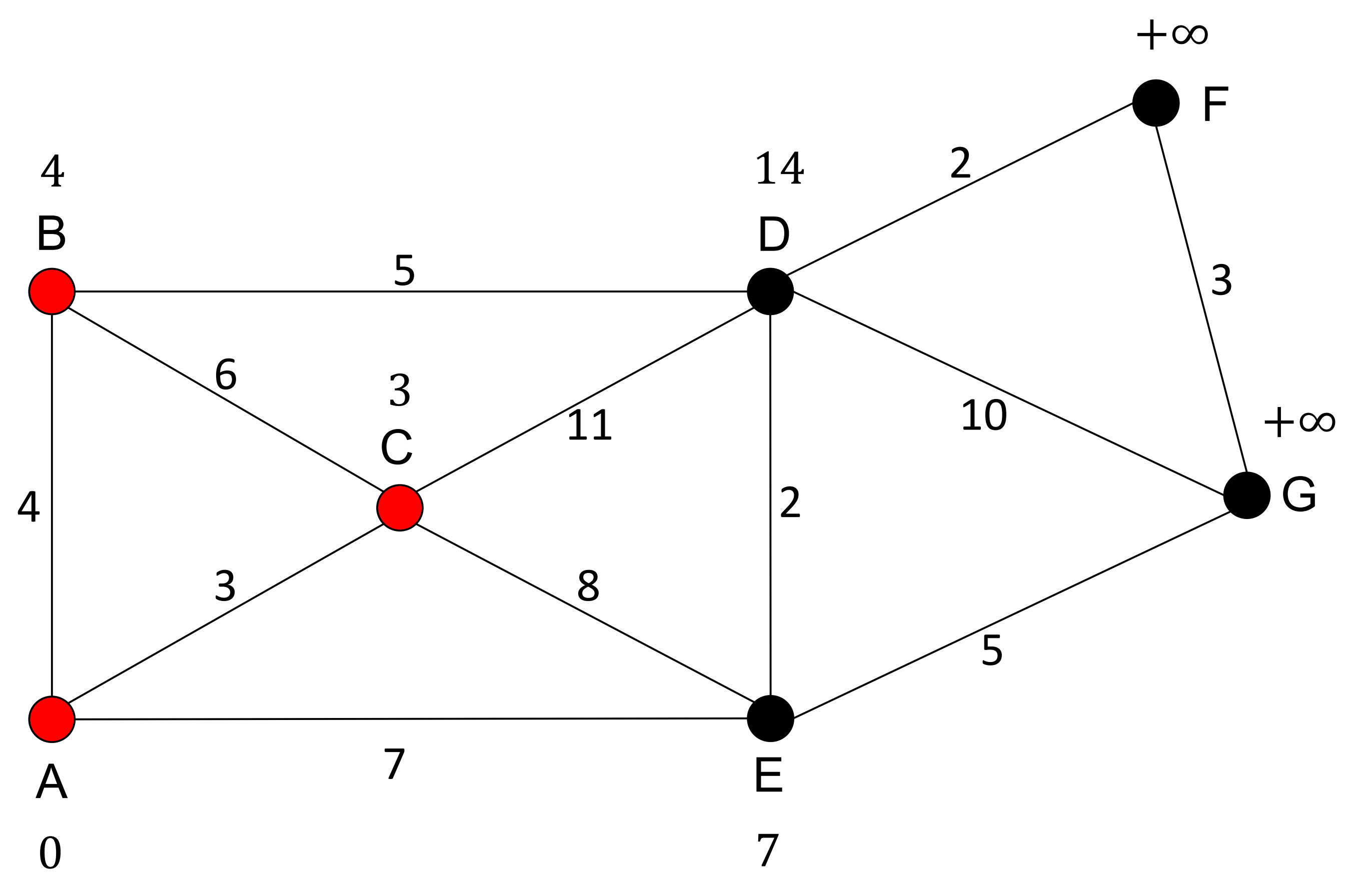
Dijkstra Greedy Algorithm



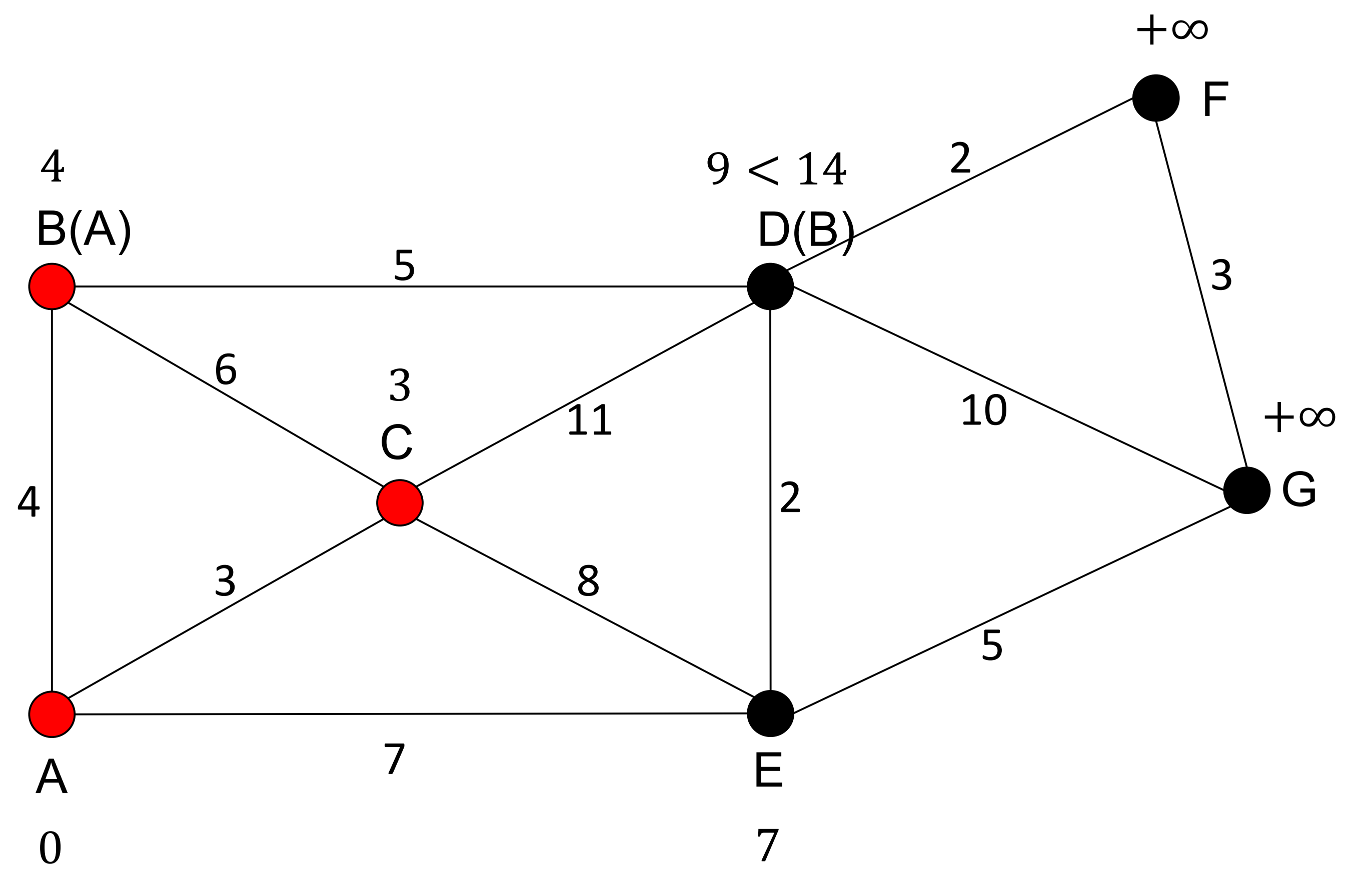
Dijkstra Greedy Algorithm



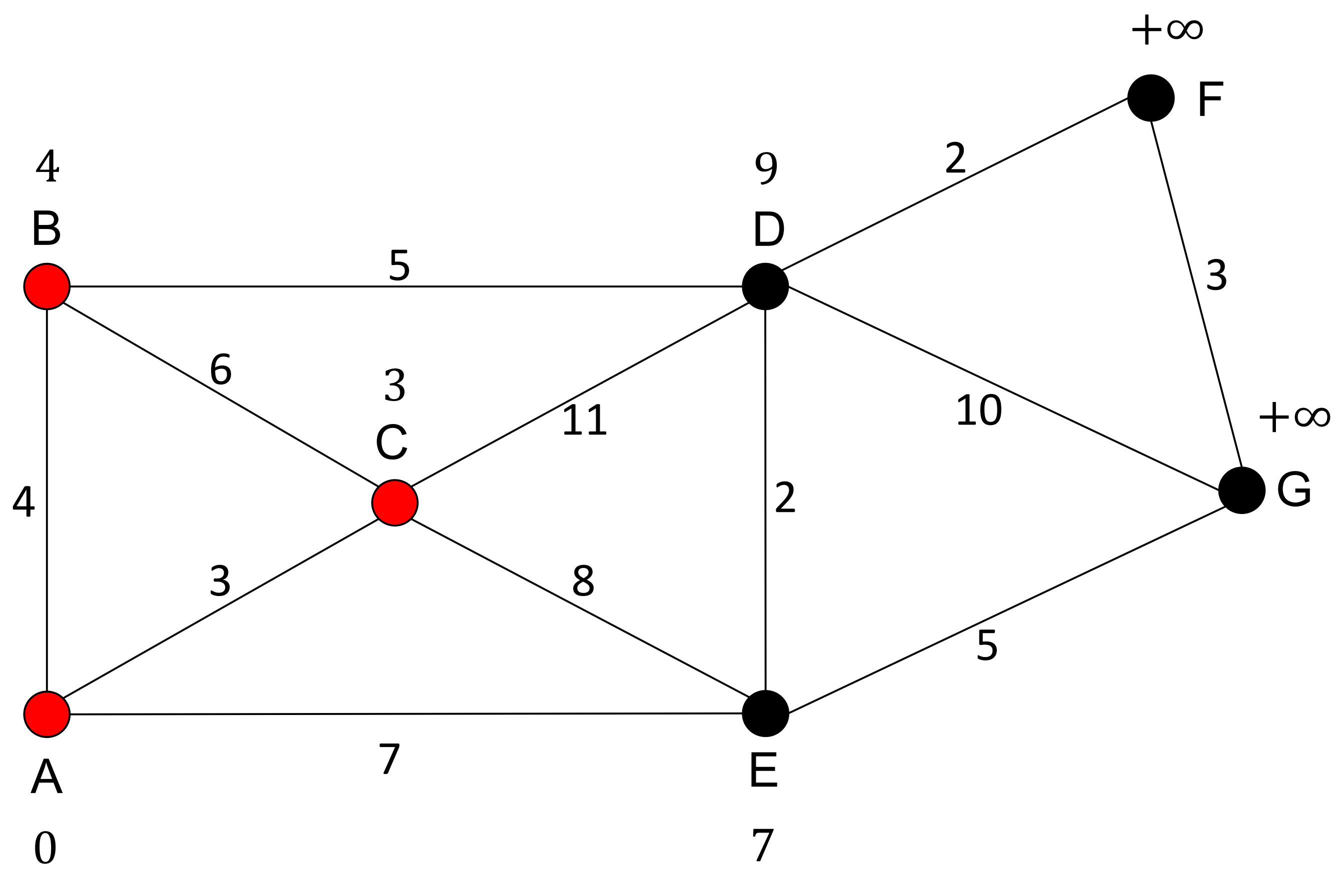
Dijkstra Greedy Algorithm



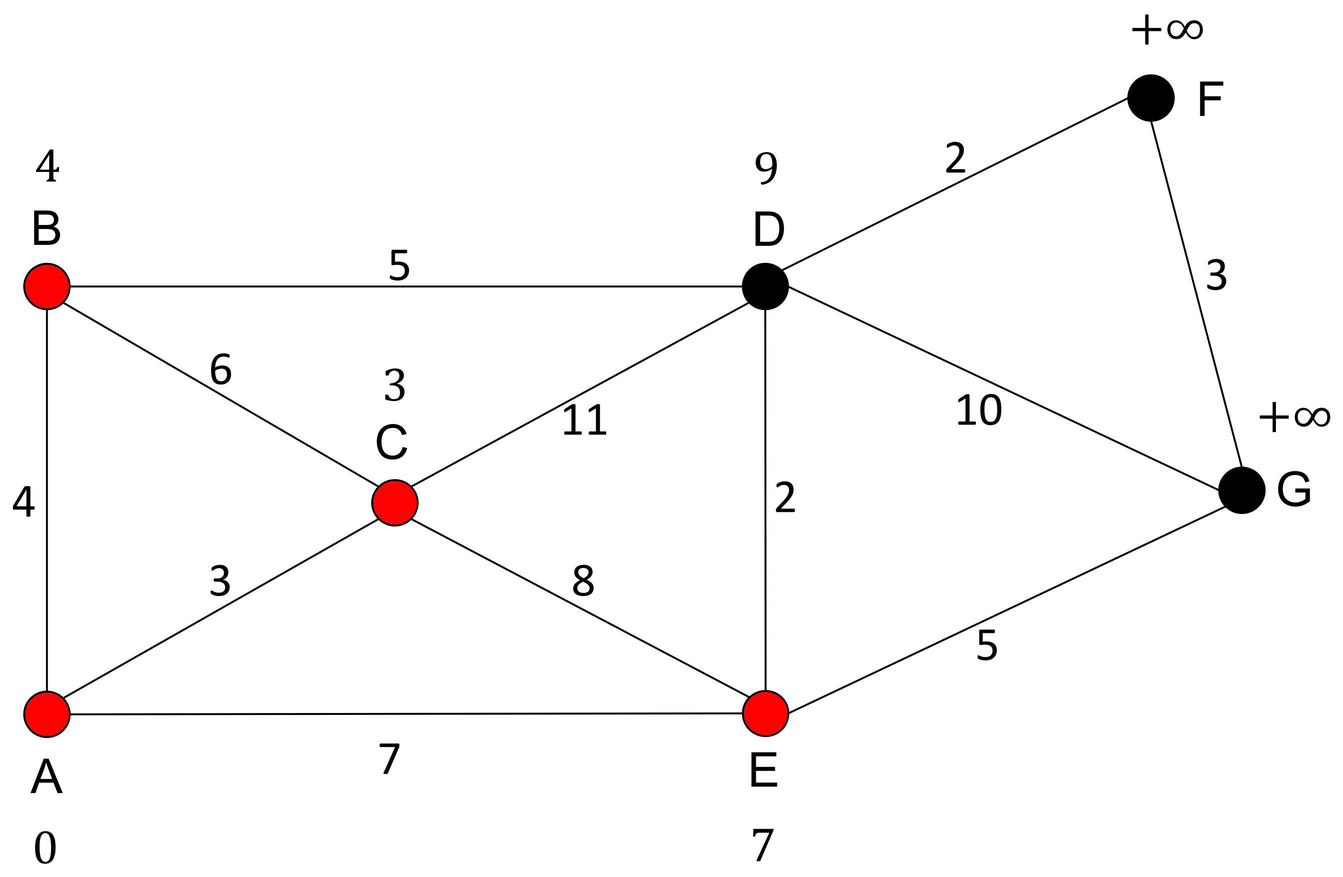
Dijkstra Greedy Algorithm



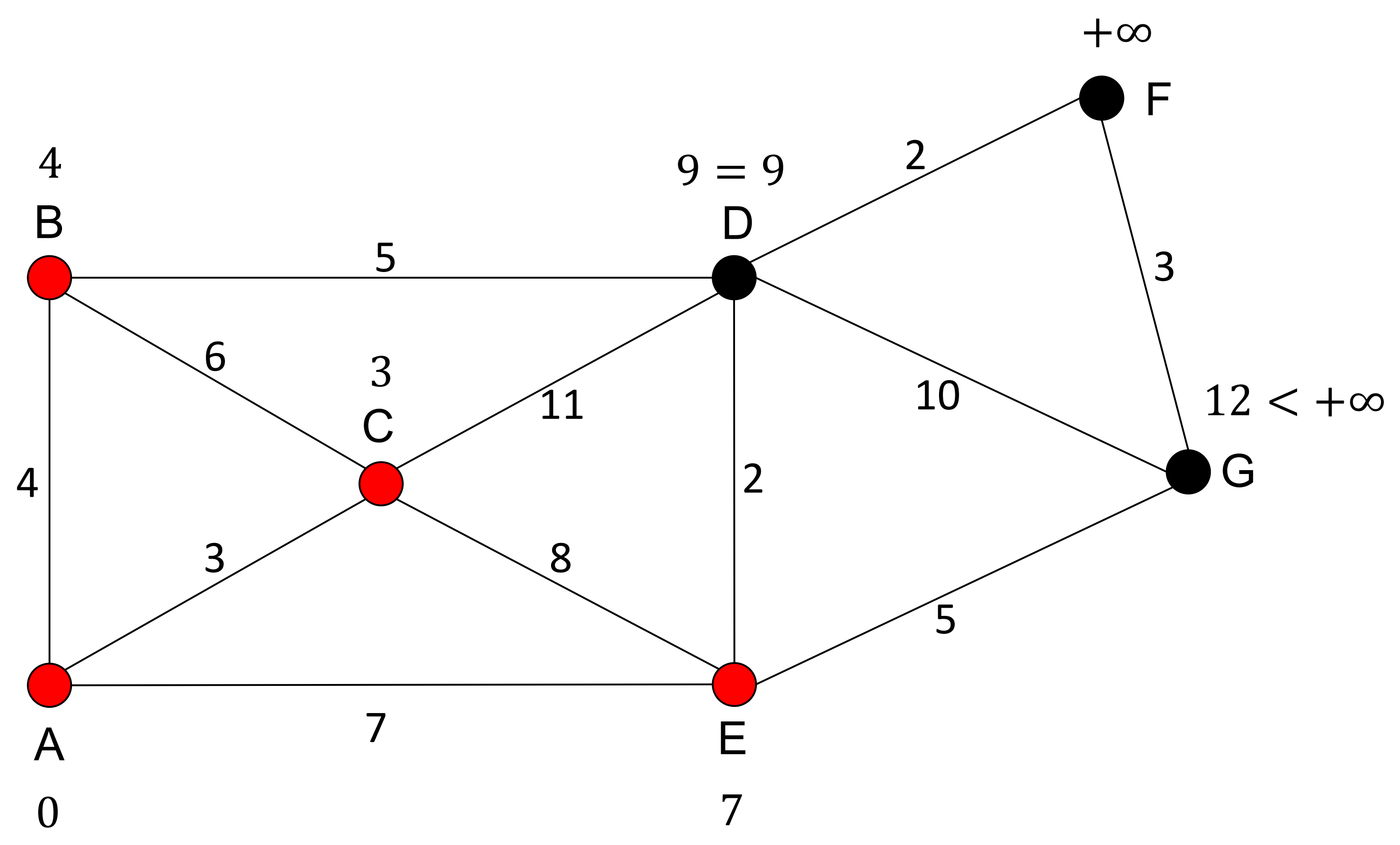
Dijkstra Greedy Algorithm



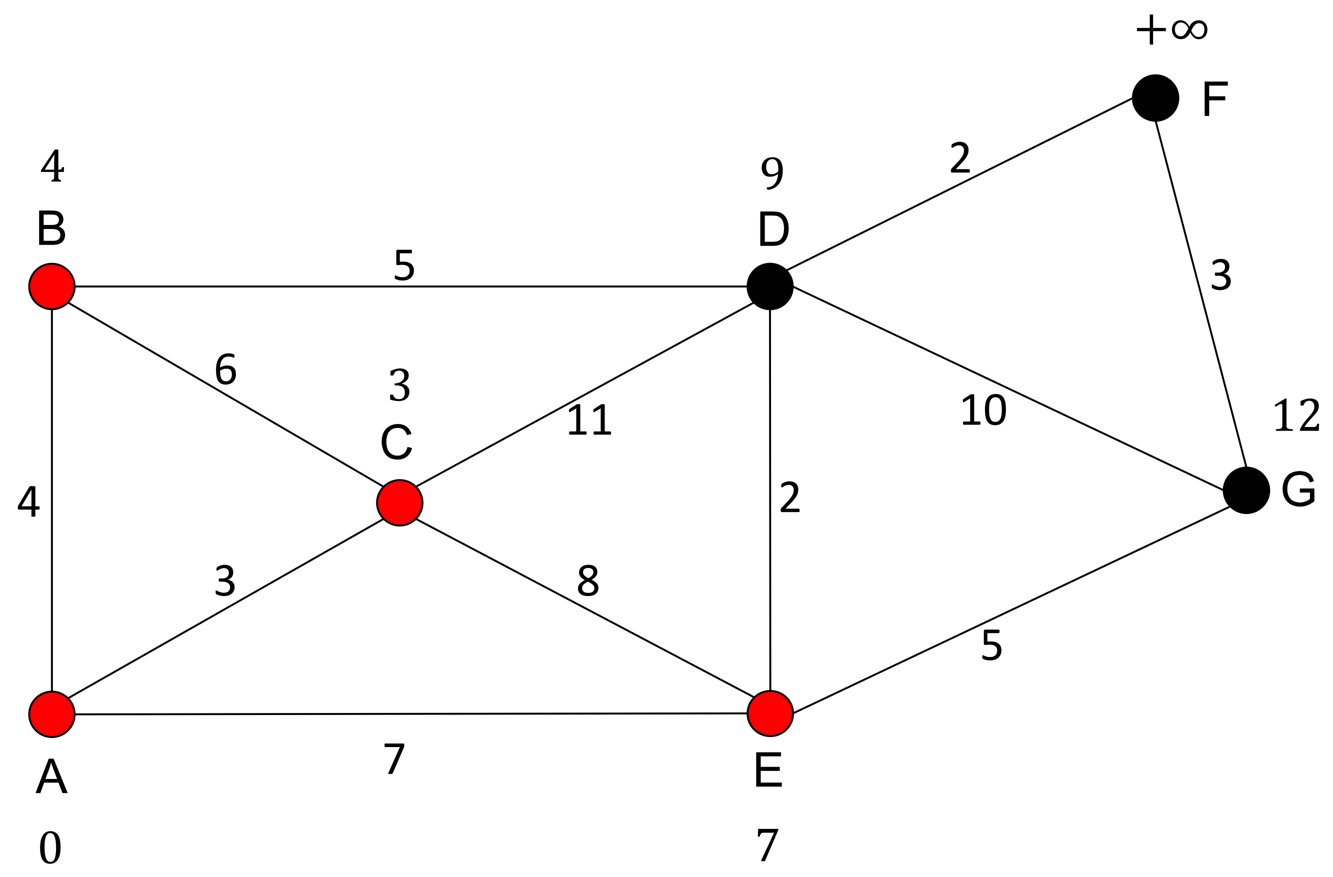
Dijkstra Greedy Algorithm



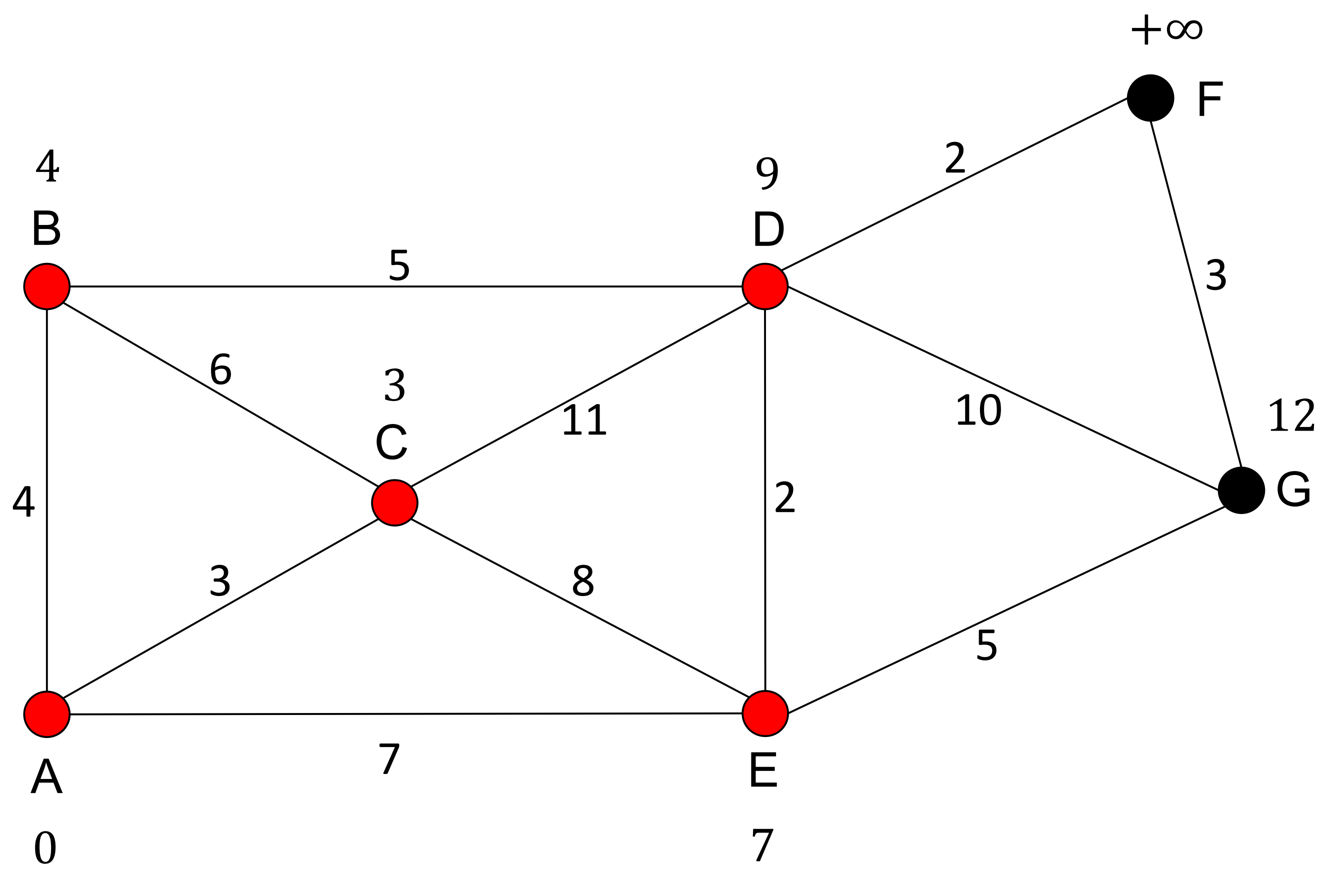
Dijkstra Greedy Algorithm



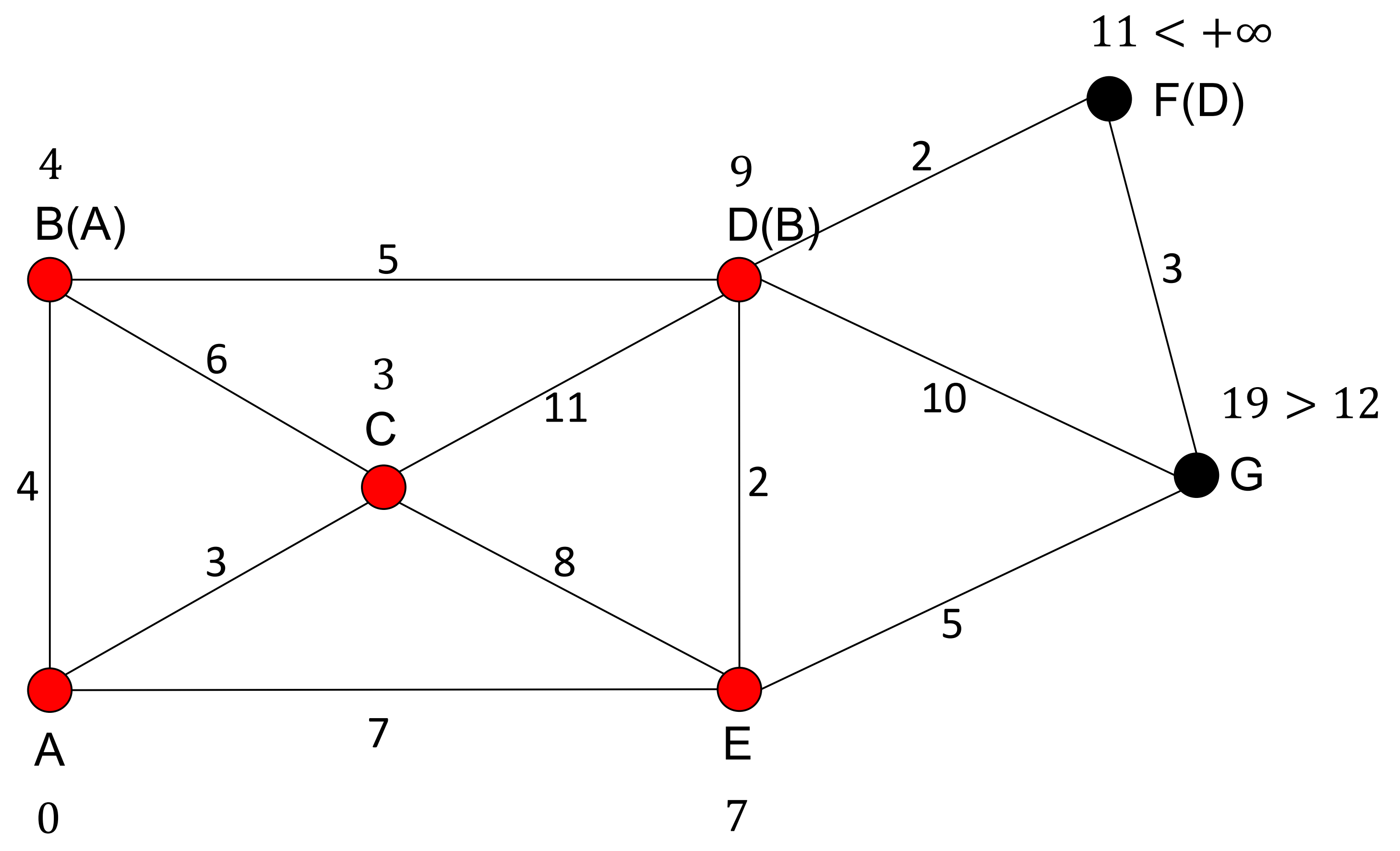
Dijkstra Greedy Algorithm



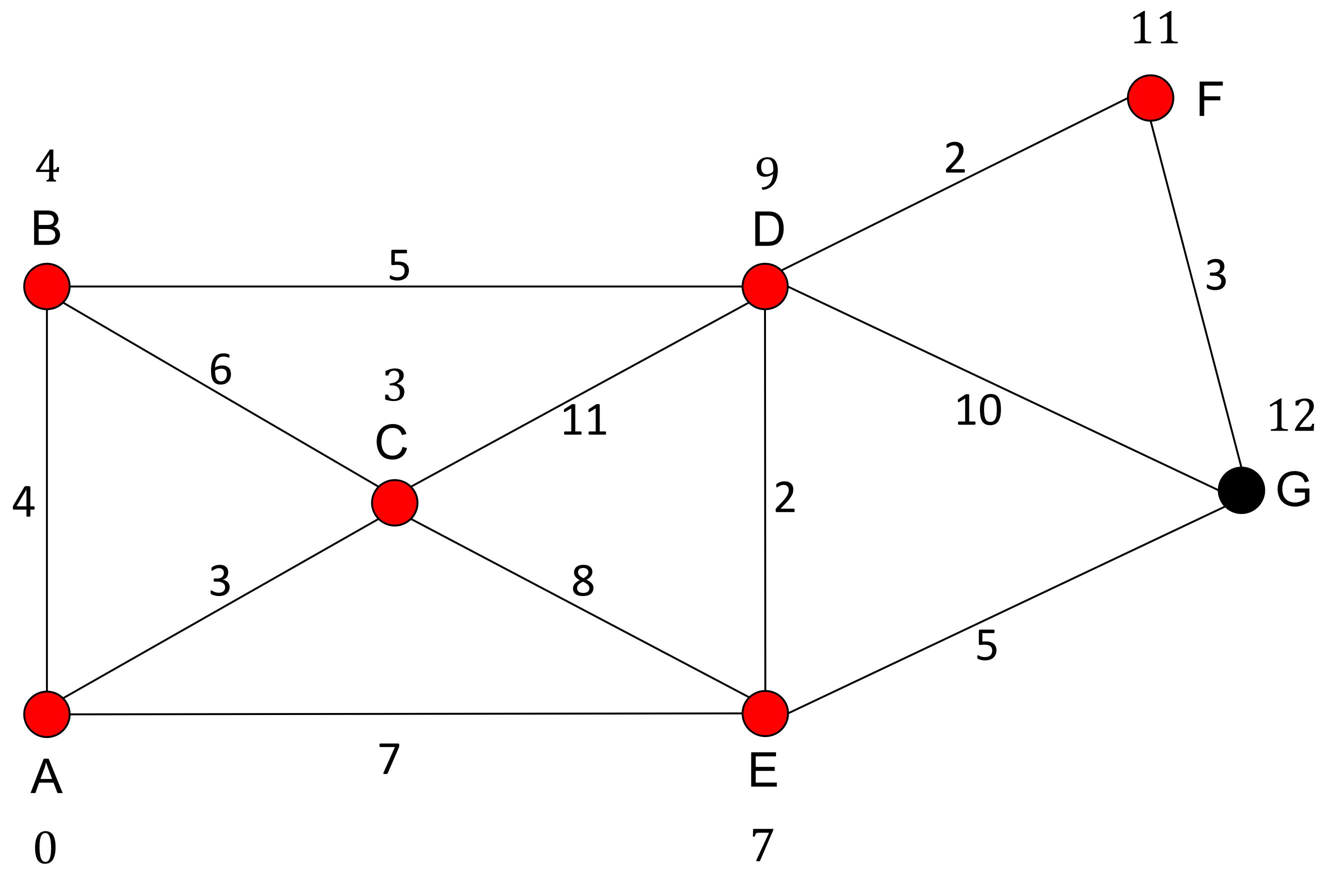
Dijkstra Greedy Algorithm



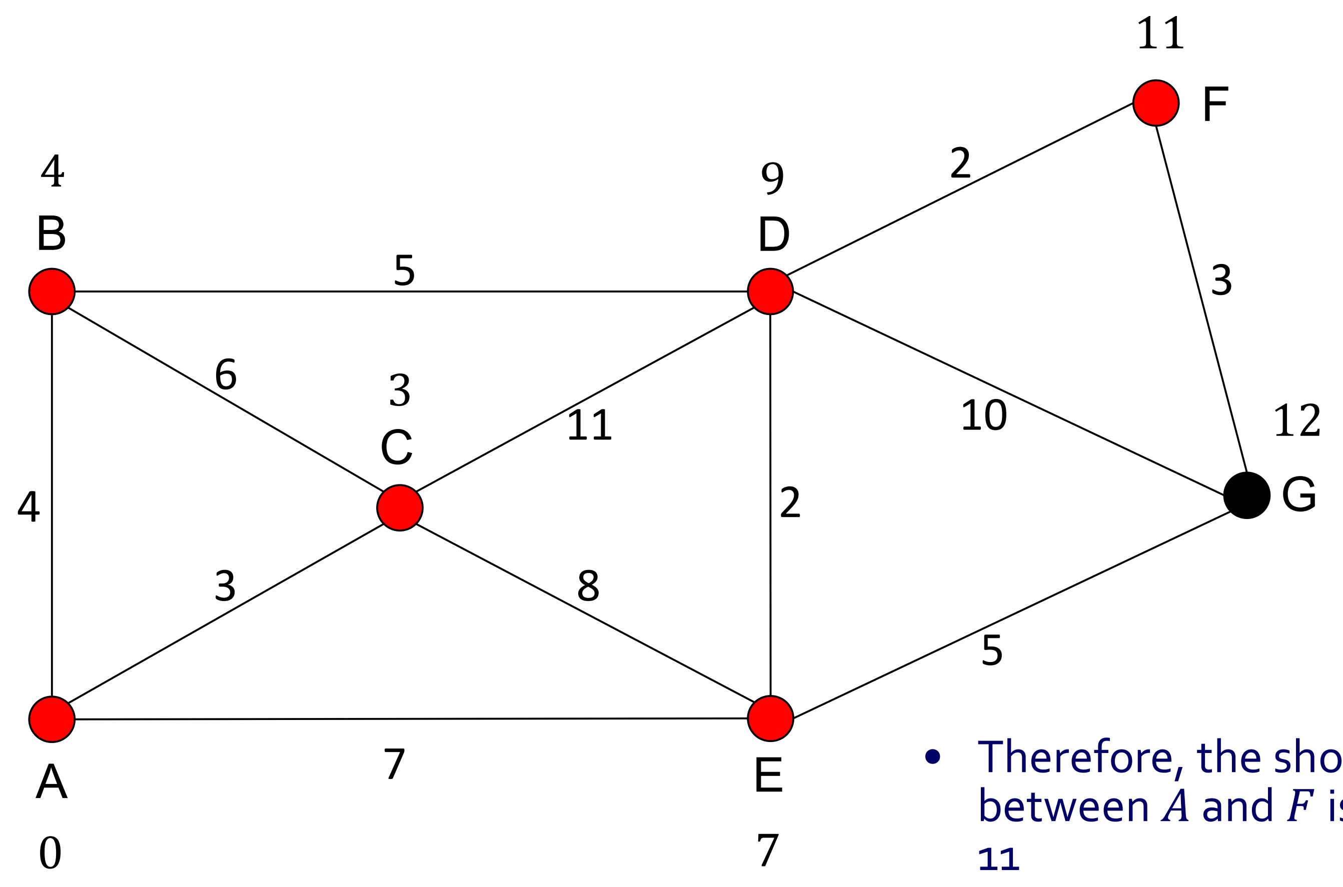
Dijkstra Greedy Algorithm



Dijkstra Greedy Algorithm



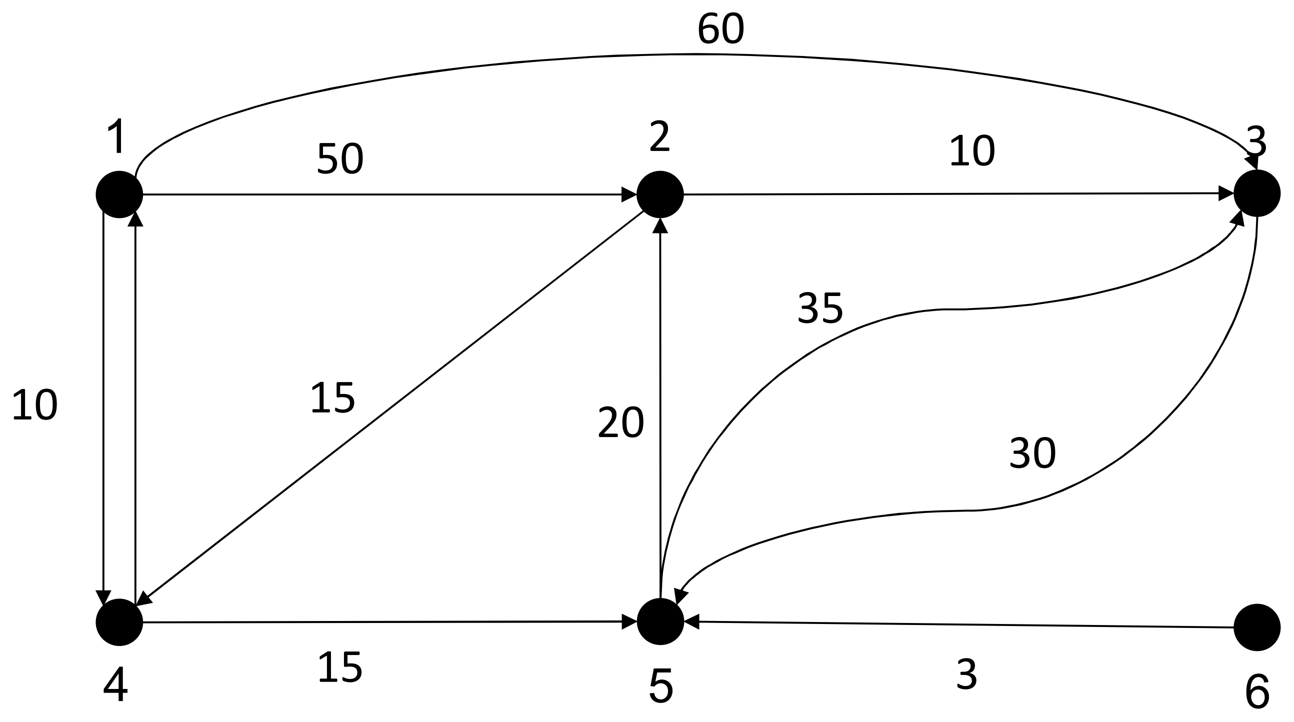
Dijkstra Greedy Algorithm



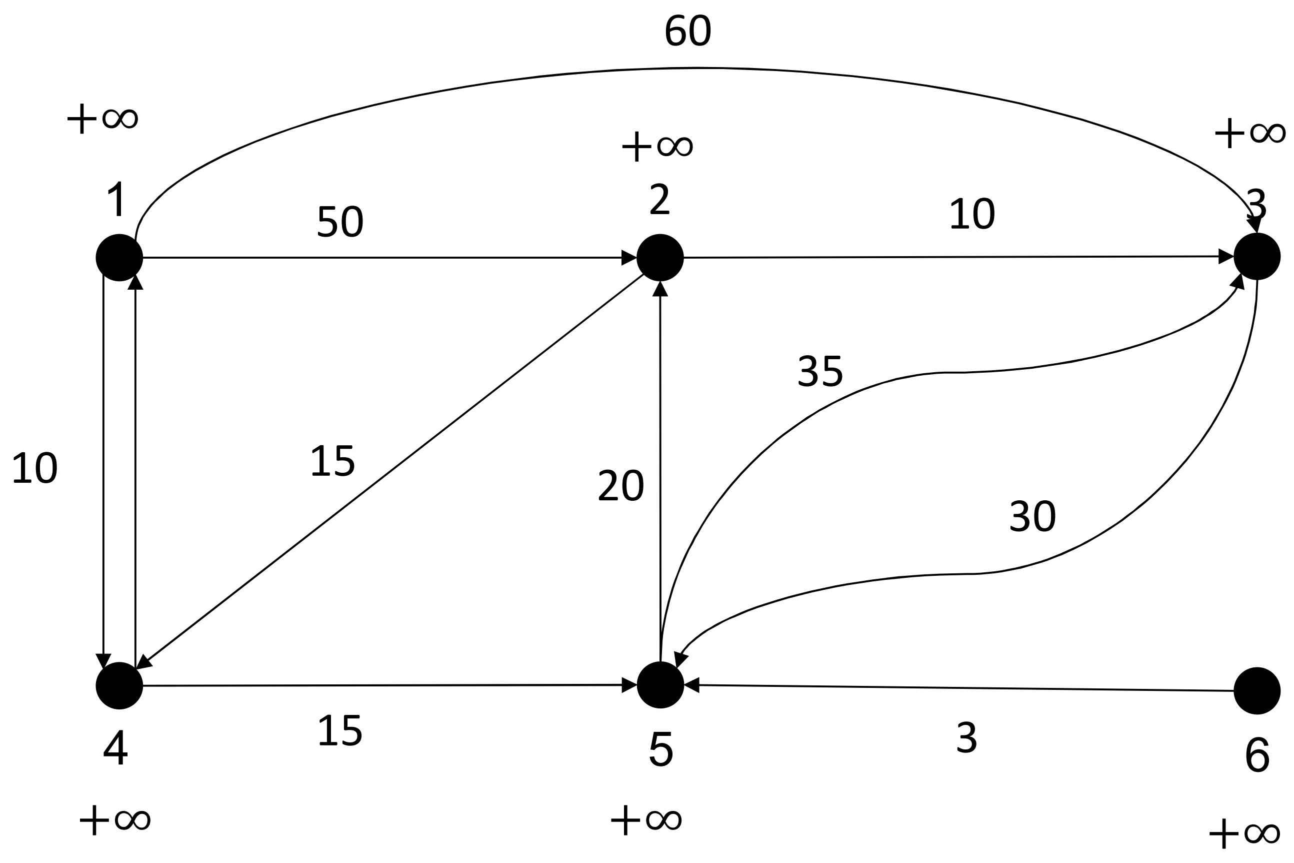
- Therefore, the shortest path between A and F is the ABDF with 11

Shortest Path Example

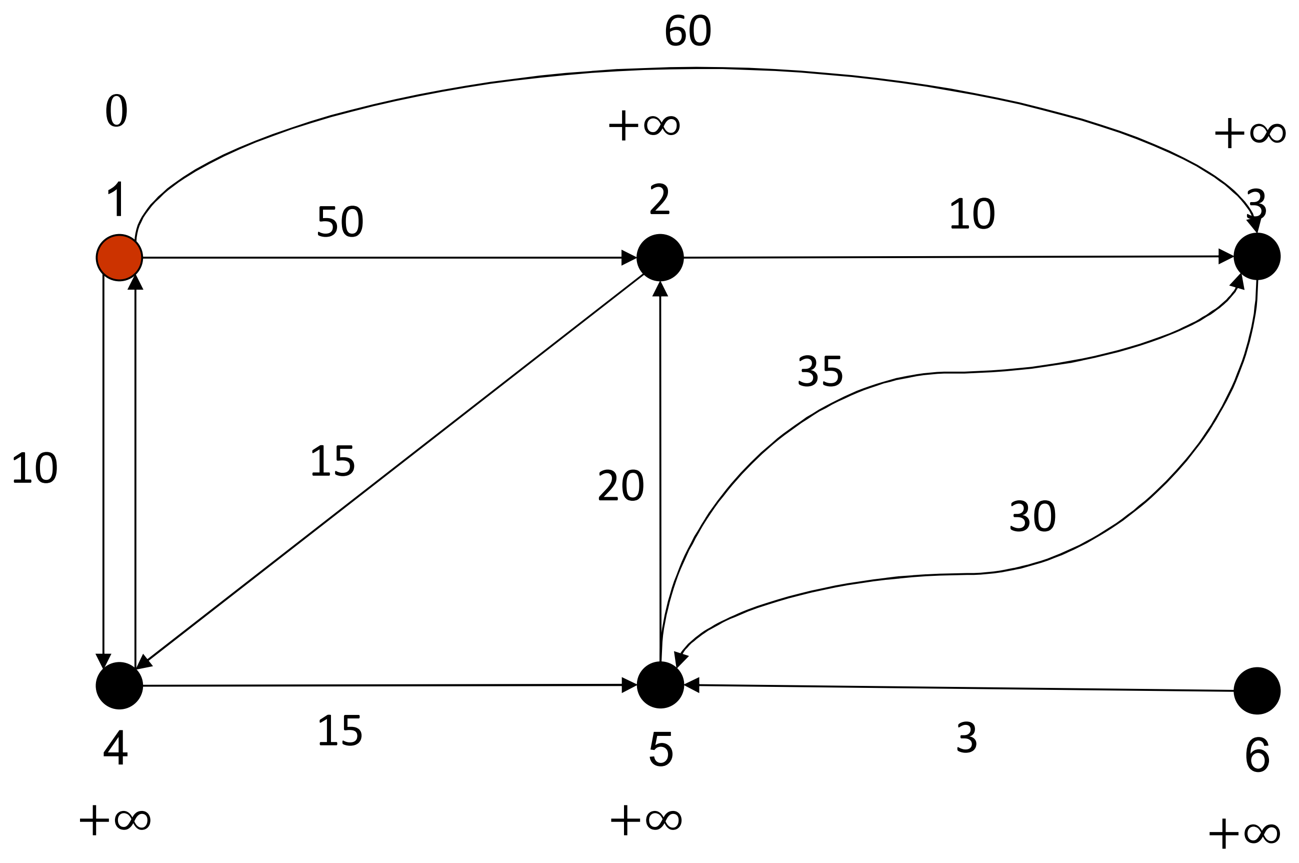
Find the shortest path starting from node 1 and ending in node 3



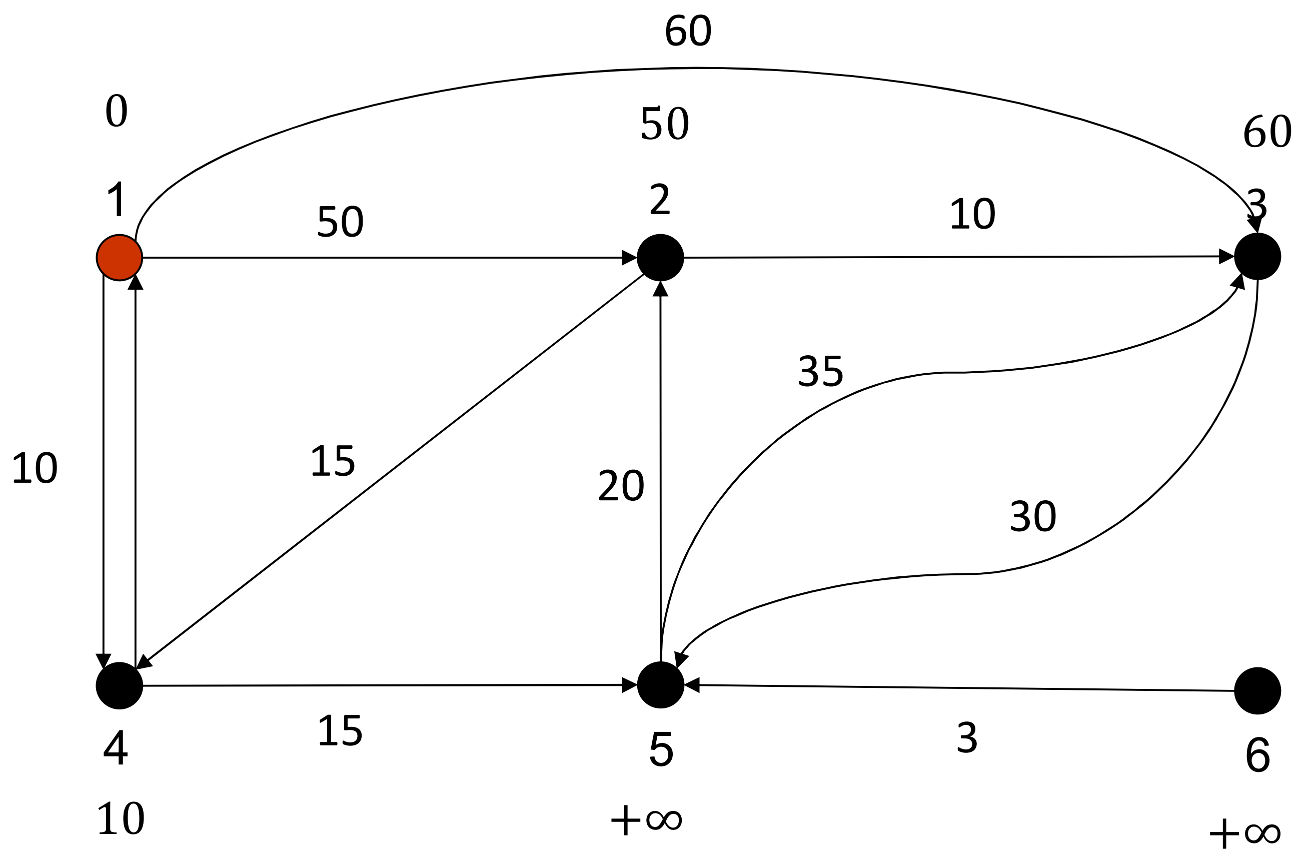
Initialization



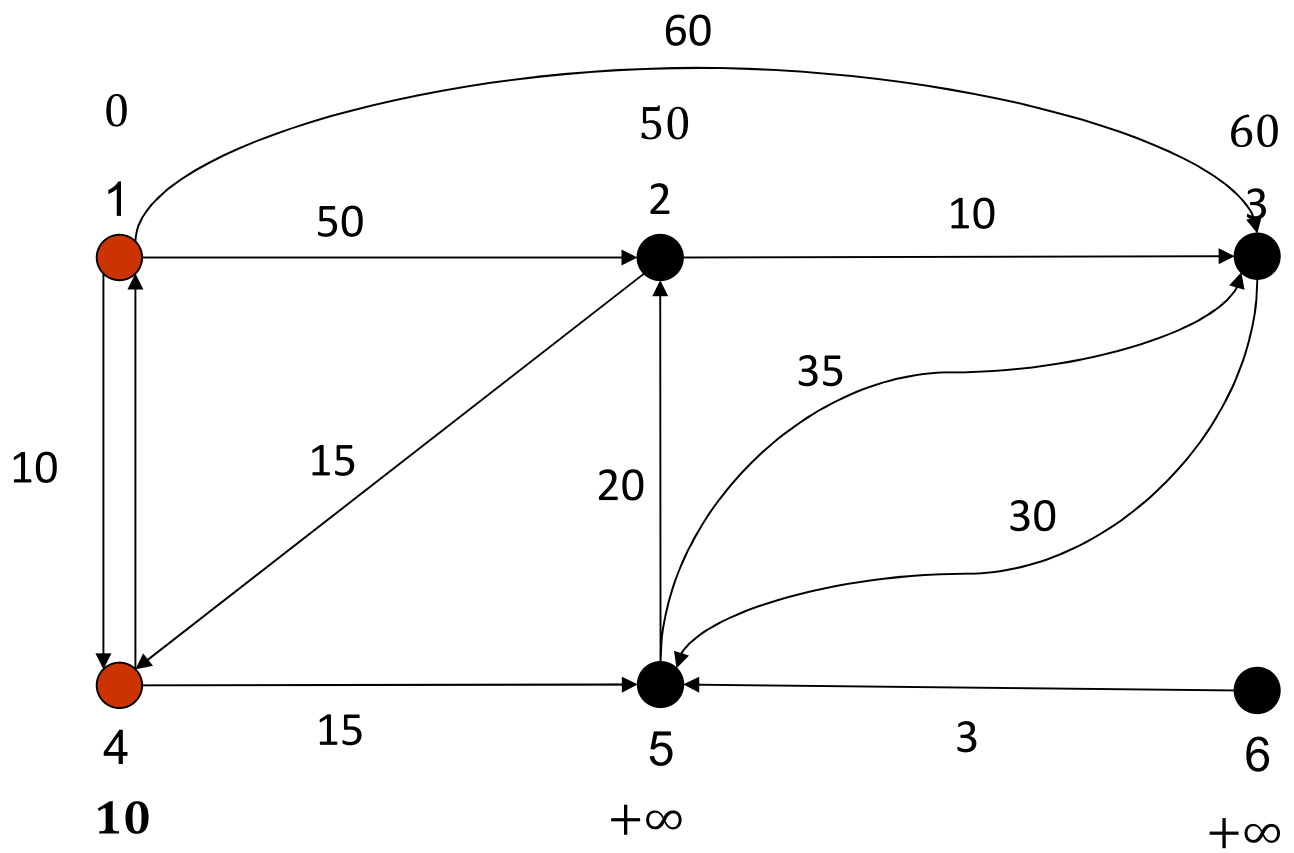
Initialization



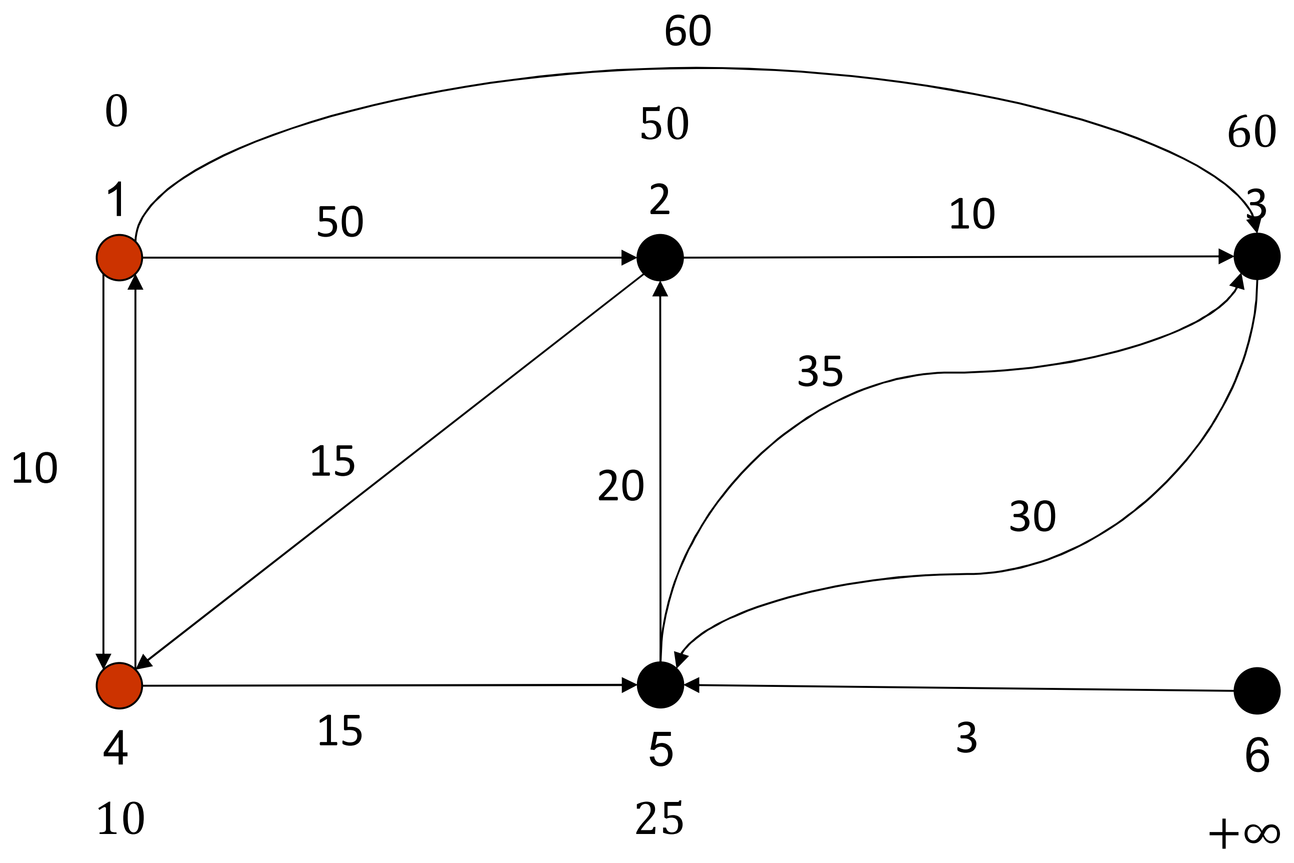
Update the distances of all visited nodes



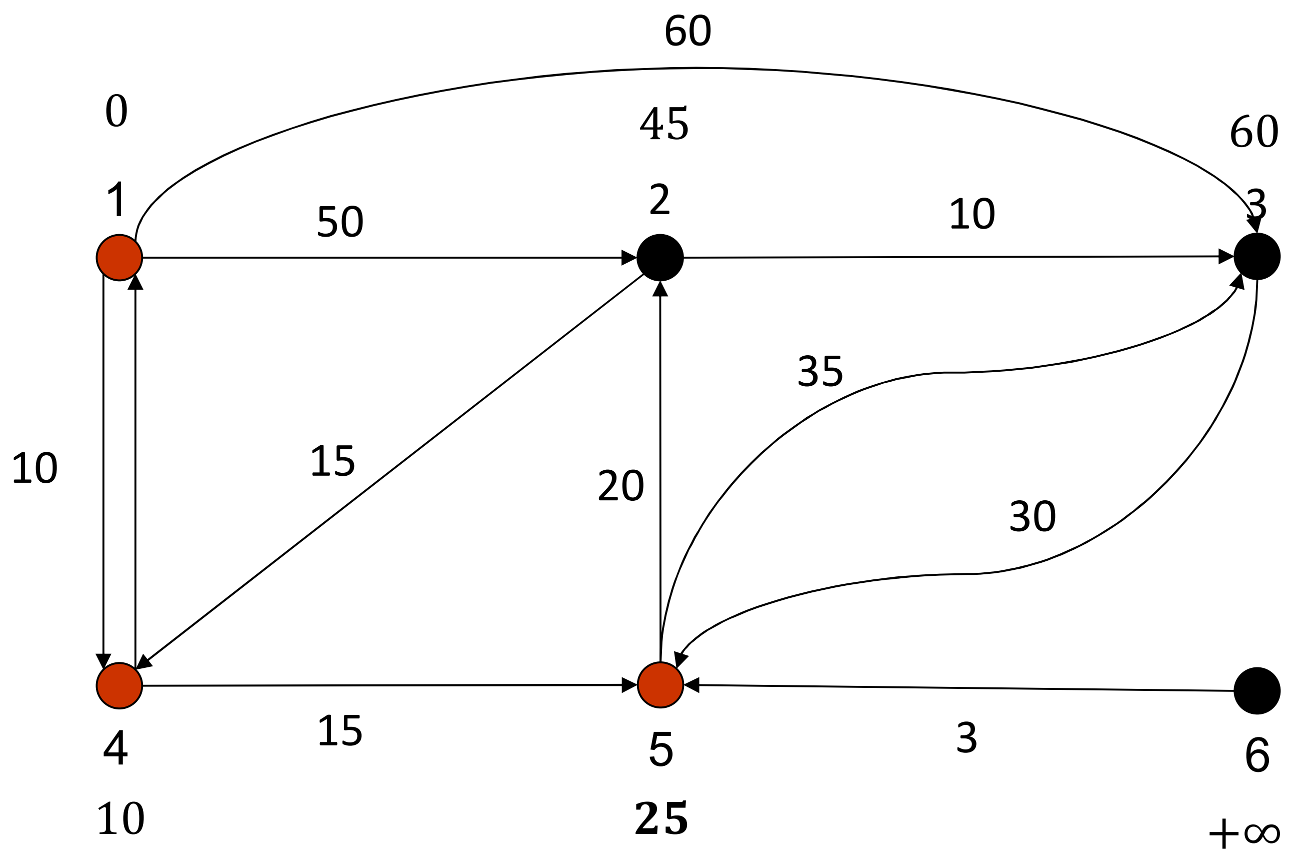
Selection of shortest distance



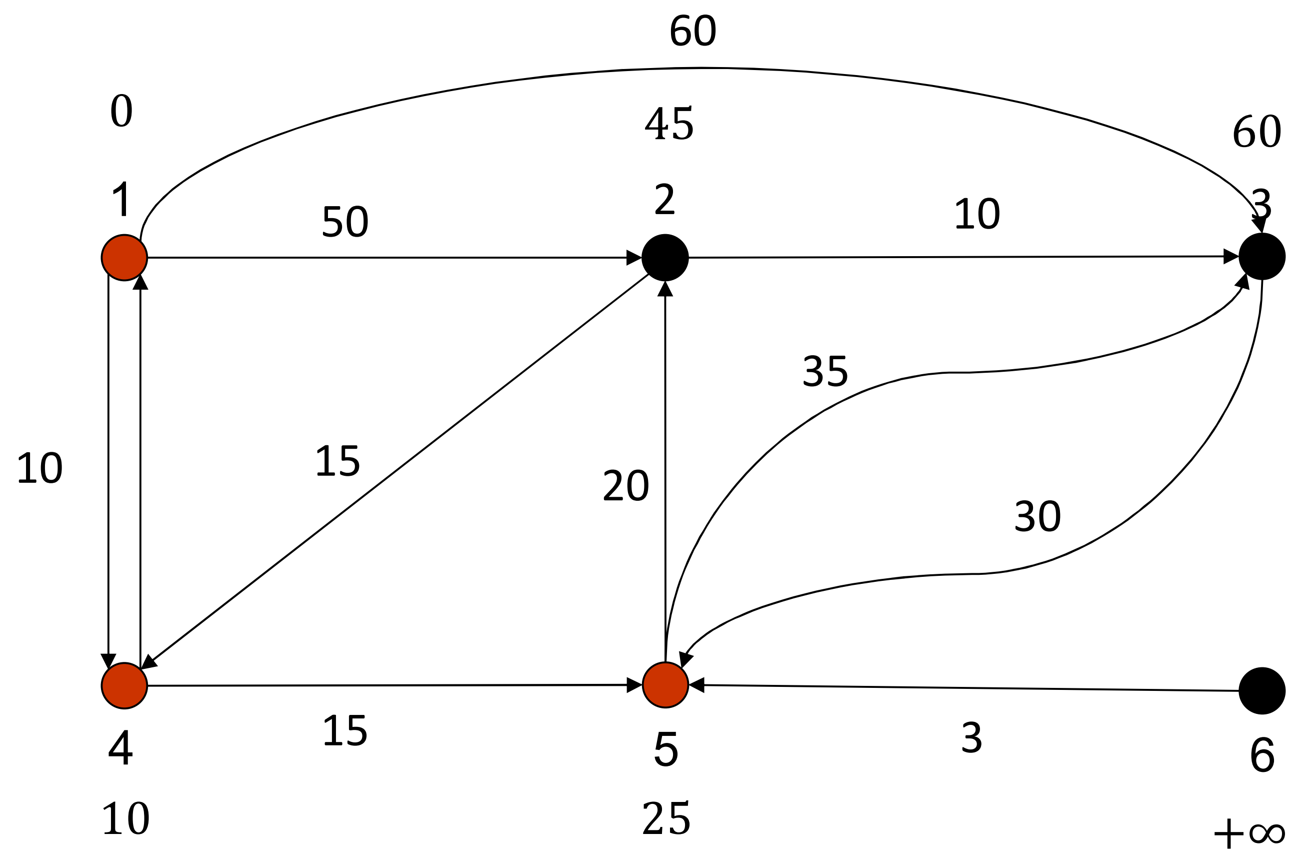
Update the distances of all visited nodes



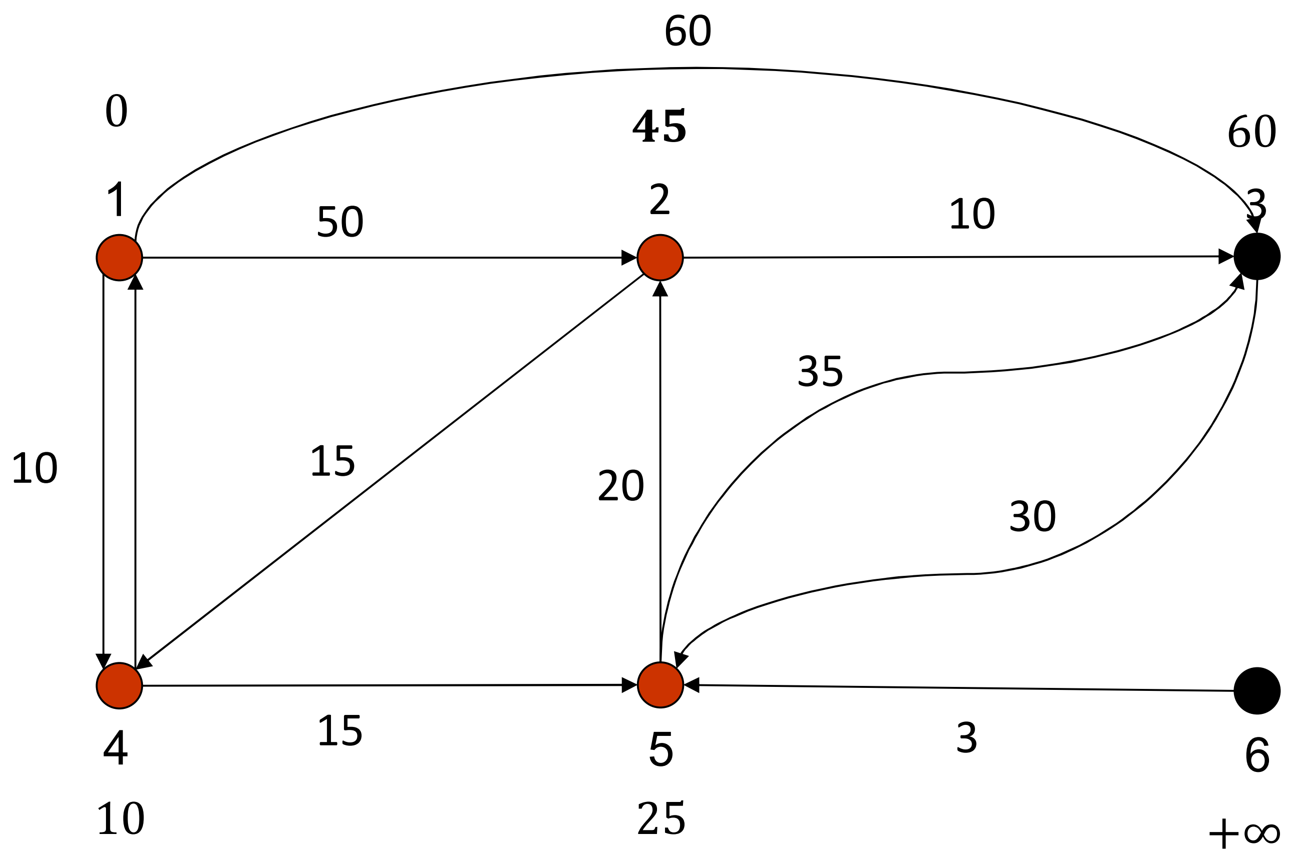
Selection of shortest distance



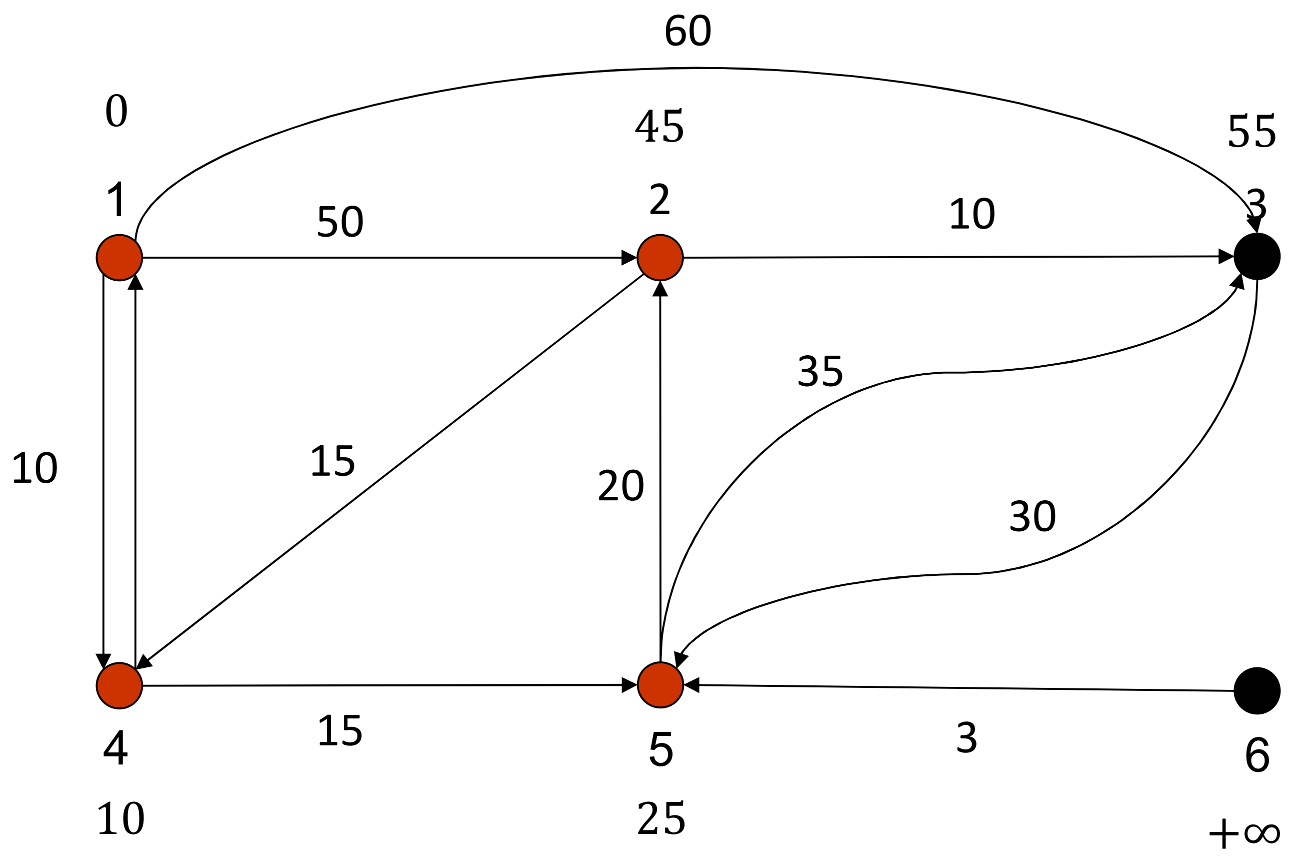
Update the distances of all visited nodes



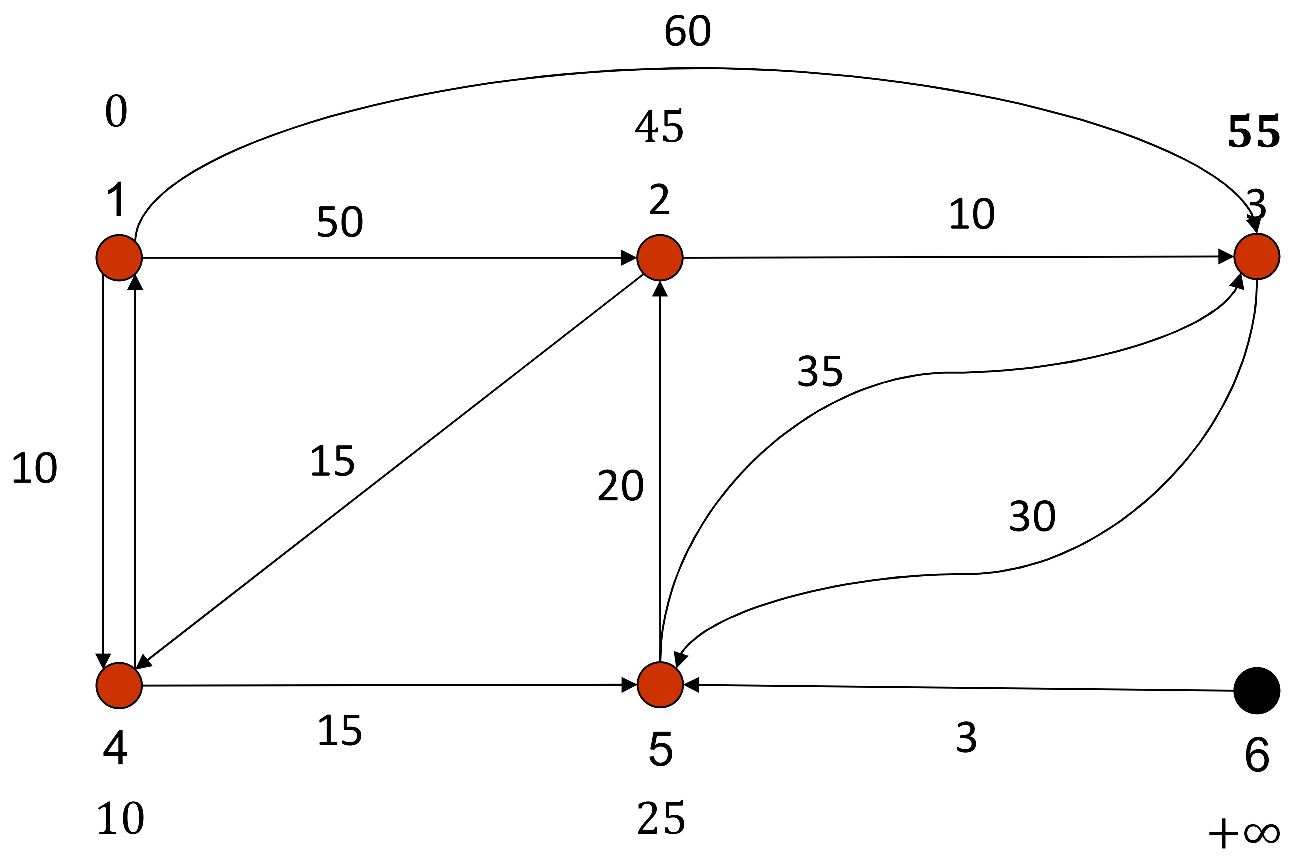
Selection of shortest distance



Update the distances of all visited nodes



Update the distances of all visited nodes



Algorithm termination: The shortest path is 1-4-5-2-3 with cost 55

Problem

- Let's consider another example
- Assume a factory with 5 machines
- We have to install a hydraulic system to provide the required pressure to each machine.
- Therefore, we have to connect all machines, with pipes
- It is not necessary, for each pair of machines to be connected
- However, it is necessary to have a pipe path from any machine to any machine.

Problem Minimum Spanning Tree

- The problem is known as minimum spanning tree
- The Minimum Spanning Tree is defined on an undirected graph.
- The graph consists a of a set of nodes and a set candidate of edges
- The objective is to select edges such that
  - A path between every pair of nodes exists
  - The total weight of the selected arcs is the minimum

Basic terminology for undirected graphs

- Undirected graphs
  - The arcs have no directions (edges)
- Path between two nodes
  - Sequence of different nodes (or edges) that connect those two nodes
- A path that starts and ends at the same node is named a circle
- Two nodes are connected, when there is at least one path that connects them (undirected graphs!)
- An undirected graph is called connected, if every pair of nodes is connected
- An undirected graph, where each pair of nodes is connected with exactly one path is called a tree

Basic terminology for undirected graphs

- Spanning tree of a set of n nodes is a connected undirected graph that does not contain cycles and contains all n nodes
- Every spanning tree has exactly $n - 1$ edges
  - Minimum number of edges to have a connected graph
  - Maximum number of edges not to have cycles

In our example we search for a spanning tree with one more property:

Minimization of the total cost of the selected edges

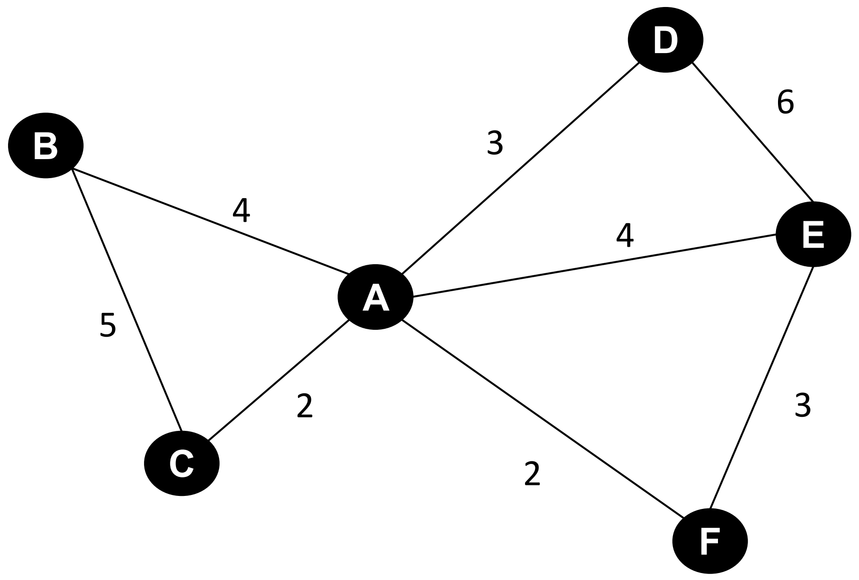
Minimum Spanning Tree Algorithms

- For the Minimum Spanning tree problem, there are two especially popular greedy algorithms that guarantee optimal solution generation
- They guarantee that the tree that will be constructed is the actual minimum spanning tree
  - Prims Algorithm
  - Kruskall Algorithm

Prims Algorithm

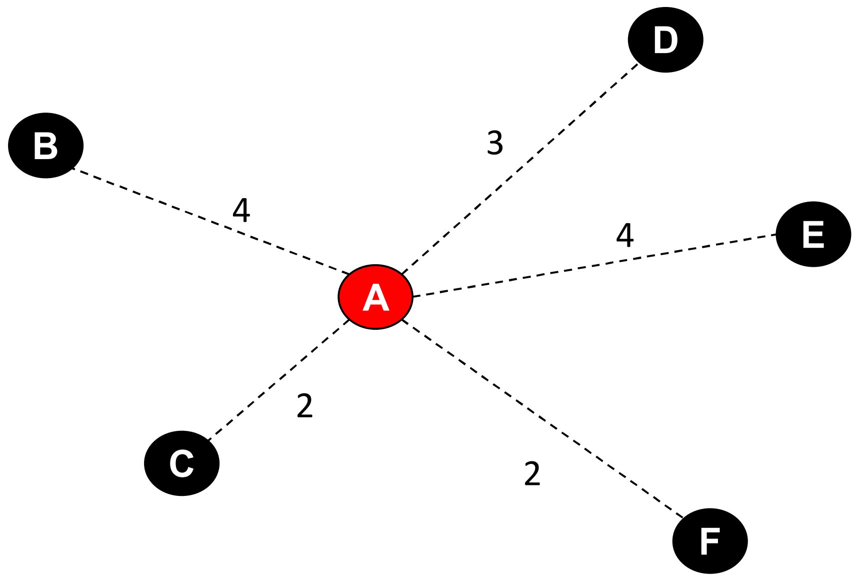
- Starting with a connected graph of n nodes with given weights, name the edges as candidate edges
- Randomly choose a node (A)
- Find the closest node(B)
- $E_S \leftarrow E_S \cup \{(A, B)\}$
- $N_C \leftarrow N_C \cup \{A\} \cup \{B\}$
- The set of selected arcs $E_S = \emptyset$
- While $(|E_S| < n - 1)$
  - Find the shortest edge that connects a node of N_C (A) with a node of $N - N_C$ (B)
  - $E_S \leftarrow E_S \cup \{(A, B)\}$
  - $N_C \leftarrow N_C \cup \{B\}$

Prims Algorithm



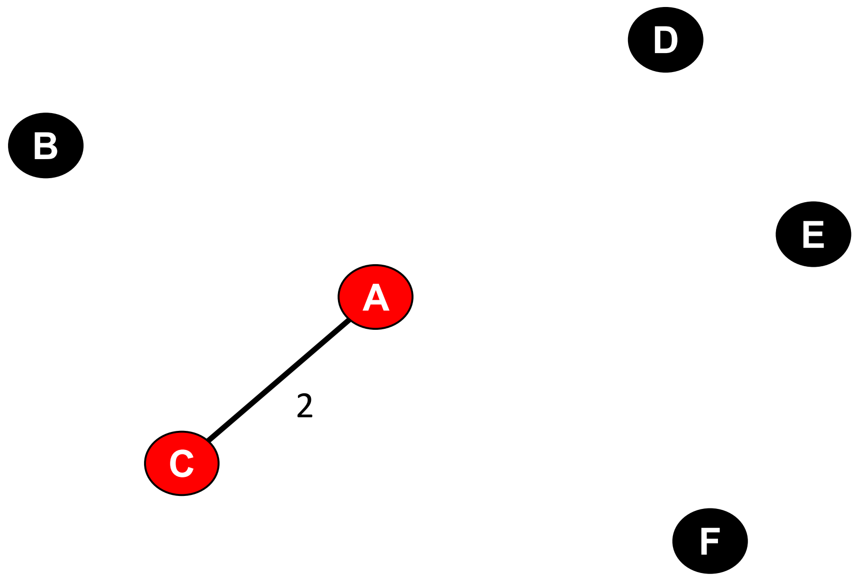
Prims Algorithm

- Random node selection: A



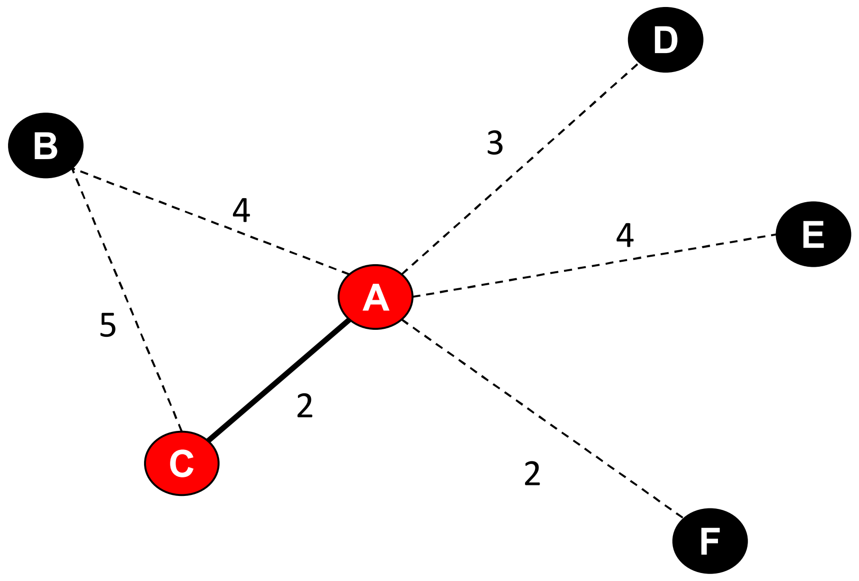
Prims Algorithm

- Selection of shortest edge {A, C}
- $E_S = \{(A, \mathrm{C})\}$
- $N_C = \{A, C\}$



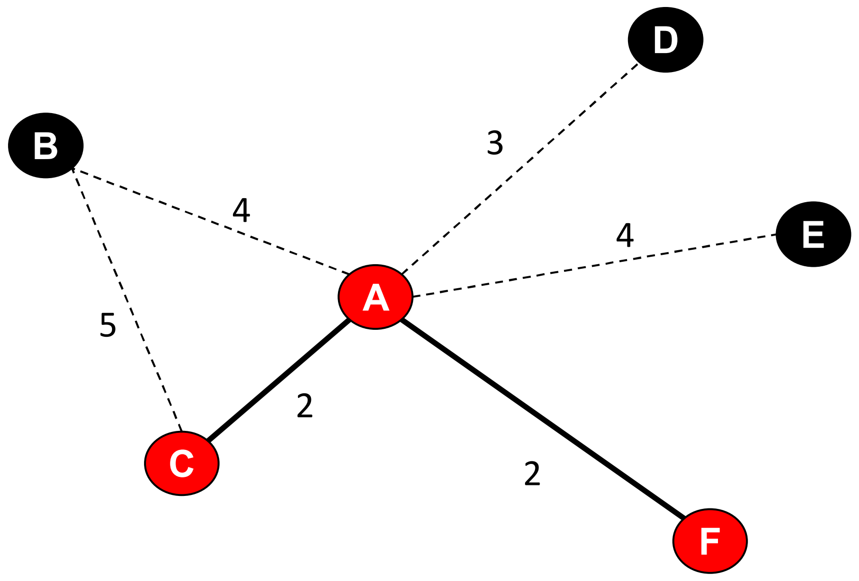
Prims Algorithm

- Compare all the edges from nodes of N_C to nodes of $\mathrm{N} - N_C$



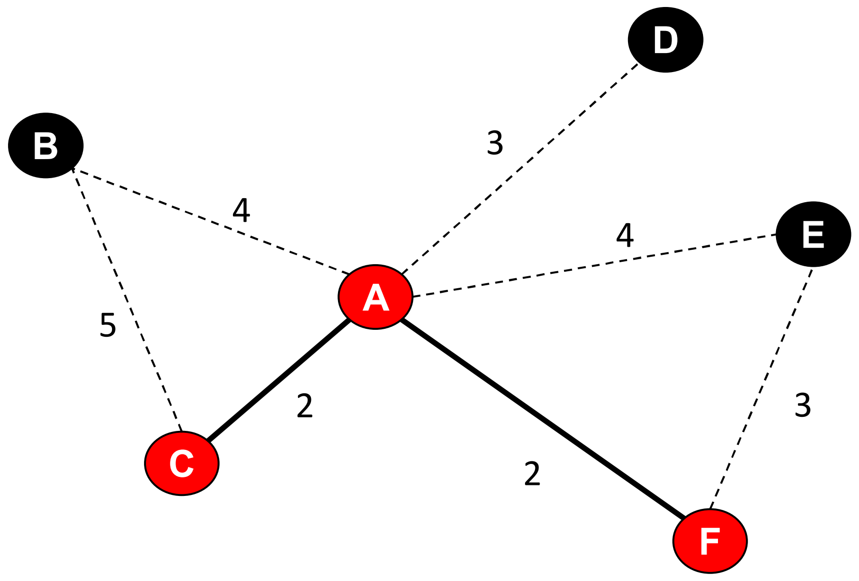
Prims Algorithm

- Selection of shortest edge {A, C}
- $E_S = \{(A, \mathrm{C}), (A, F)\}$
- $N_C = \{A, C, F\}$



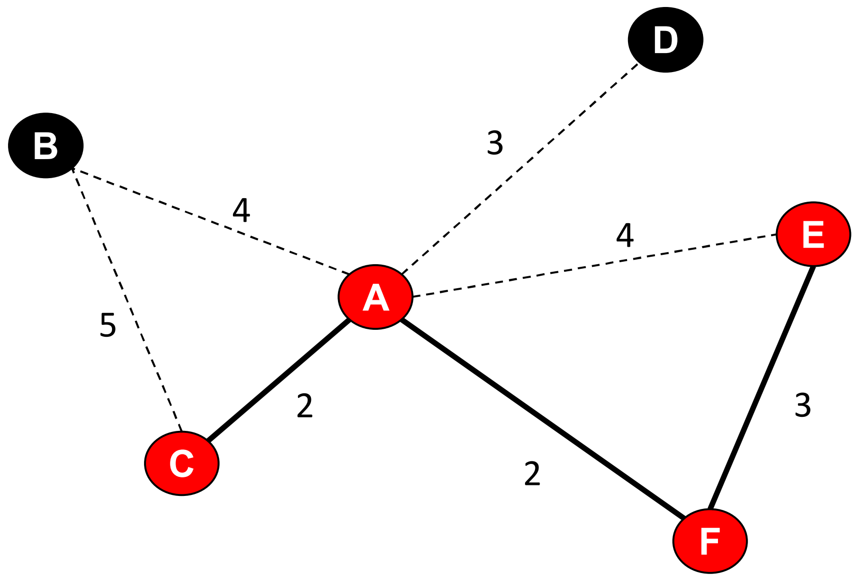
Prims Algorithm

- Compare all the edges from nodes of N_C to nodes of $\mathrm{N} - N_C$



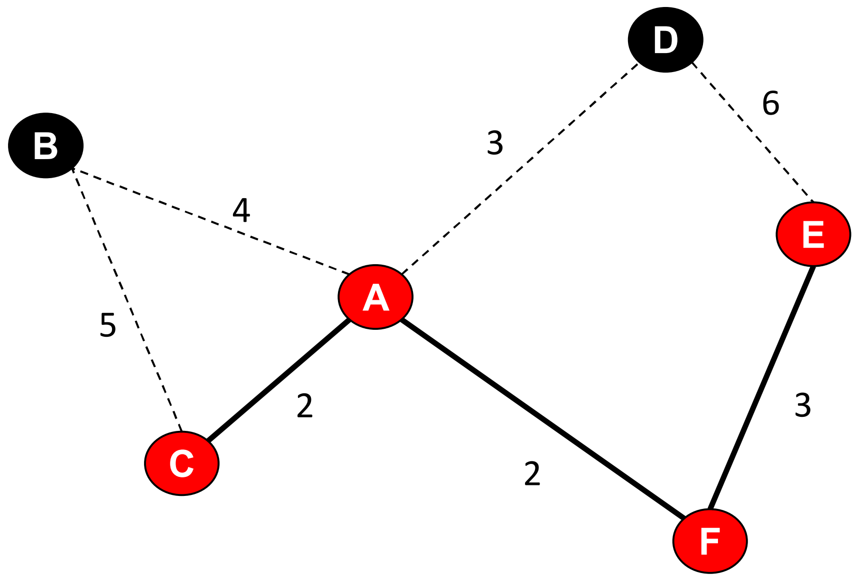
Prims Algorithm

- Selection of shortest edge {A, C}
- $E_S = \{(A, \mathrm{C}), (A, F), (F, E)\}$
- $N_C = \{A, C, F, E\}$



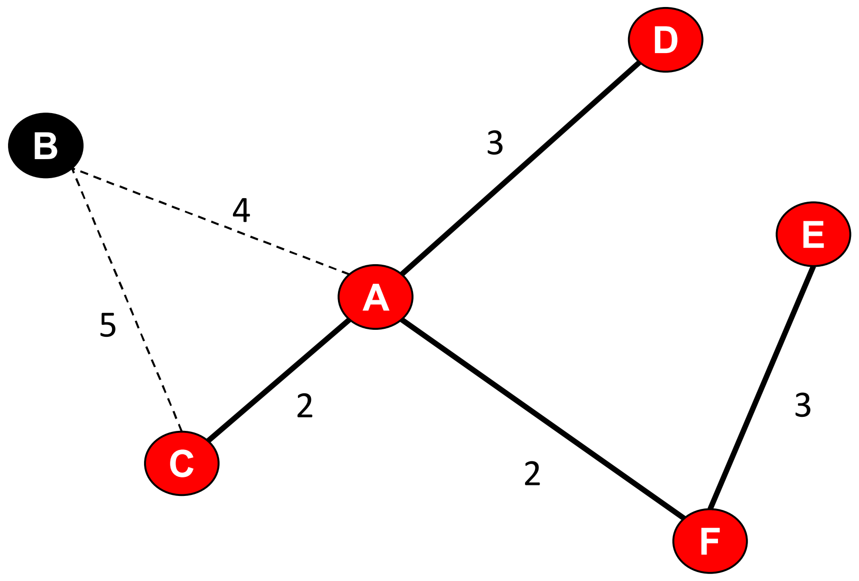
Prims Algorithm

- Compare all the edges from nodes of N_C to nodes of $\mathrm{N} - N_C$



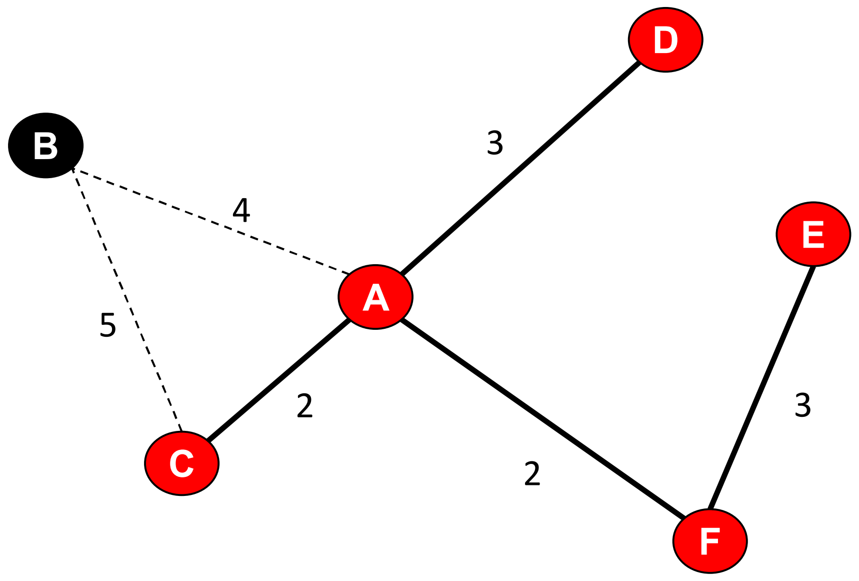
Prims Algorithm

- Selection of shortest edge {A, C}
- $E_S = \{(A, \mathrm{C}), (A, F), (F, E), (A, D)\}$
- $N_C = \{A, C, F, E, D\}$



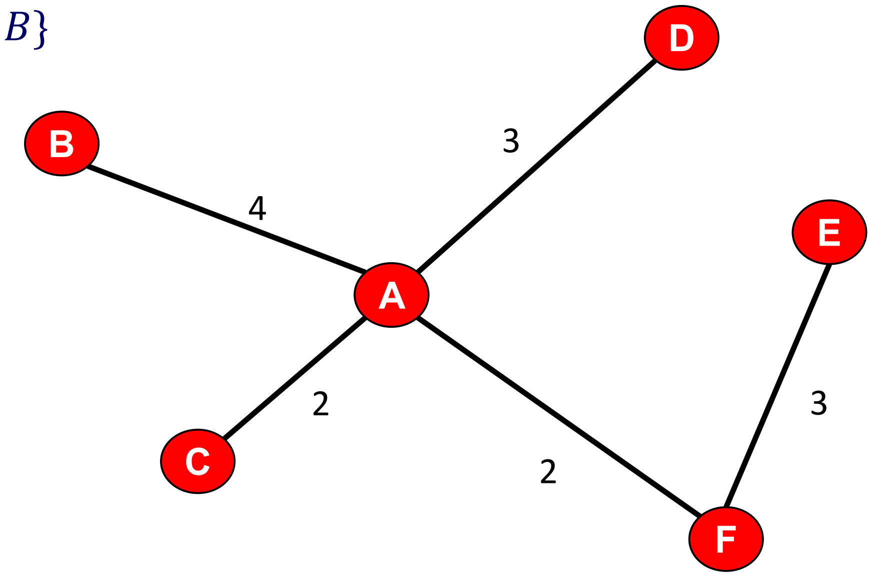
Prims Algorithm

- Compare all the edges from nodes of N_C to nodes of $\mathrm{N} - N_C$



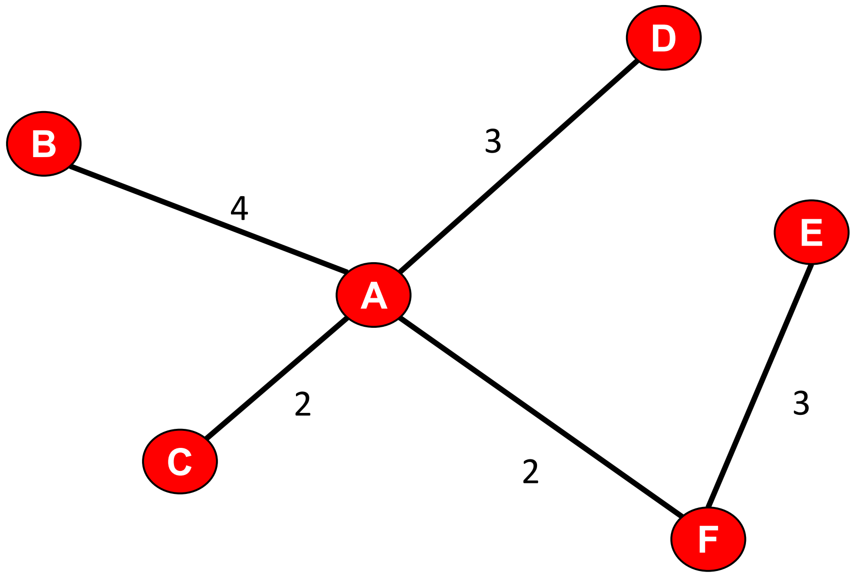
Prims Algorithm

- Selection of shortest edge {A, C}
- $E_S = \{(A, \mathrm{C}), (A, F), (F, E), (A, D), (A, B)\}$
- $N_C = \{A, C, F, E, D, B\}$



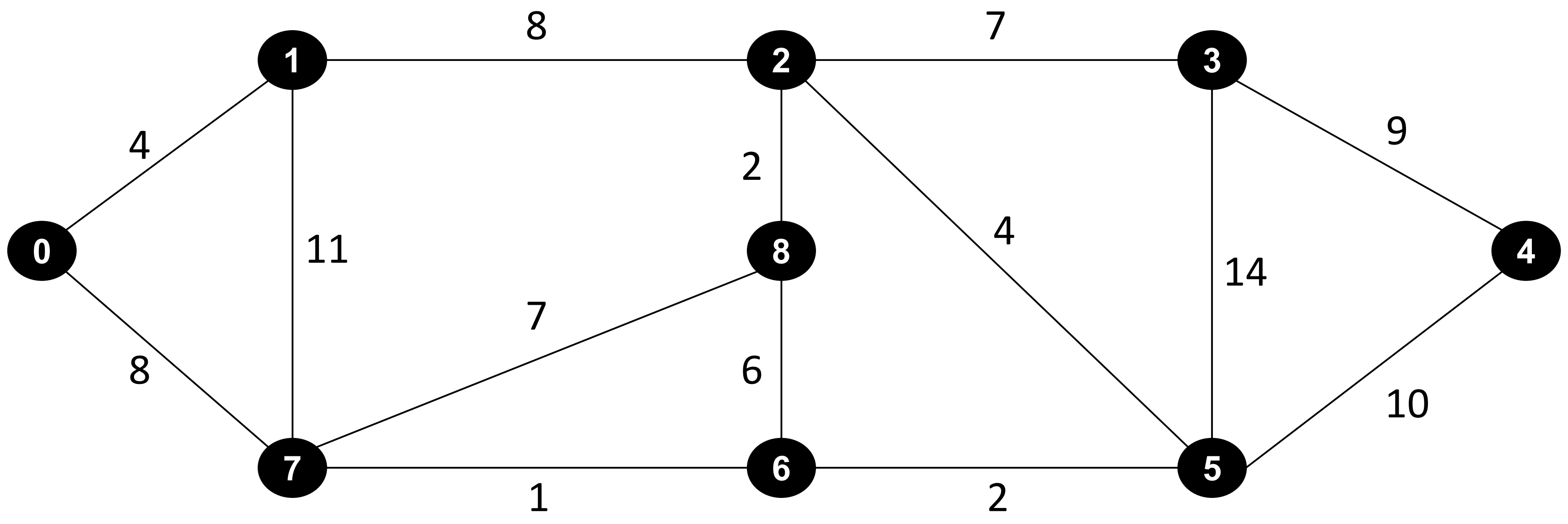
Prims Algorithm

- Algorithm termination
- $E_S = \{(A, \mathrm{C}), (A, F), (F, E), (A, D), (A, B)\}$
- $N_C = \{A, C, F, E, D, B\}$
- *Final cost: 14*



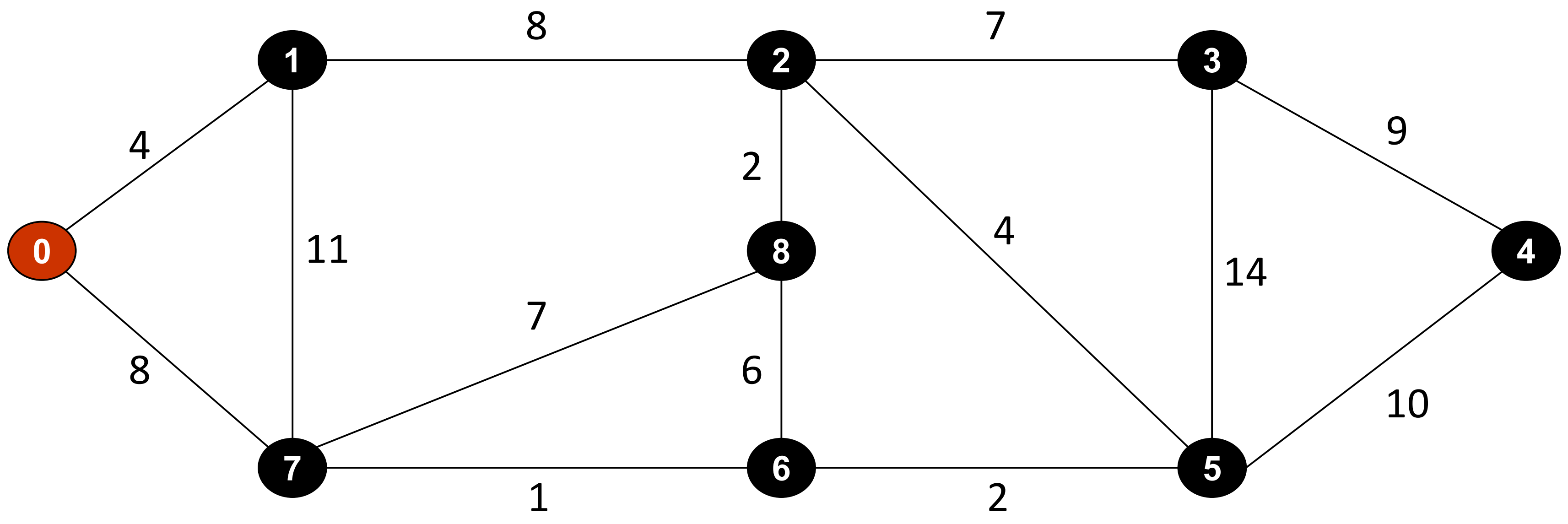
Prims Algorithm

- Apply the Prims algorithm to the graph below, in order to construct the Minimum Spanning Tree of all 9 nodes



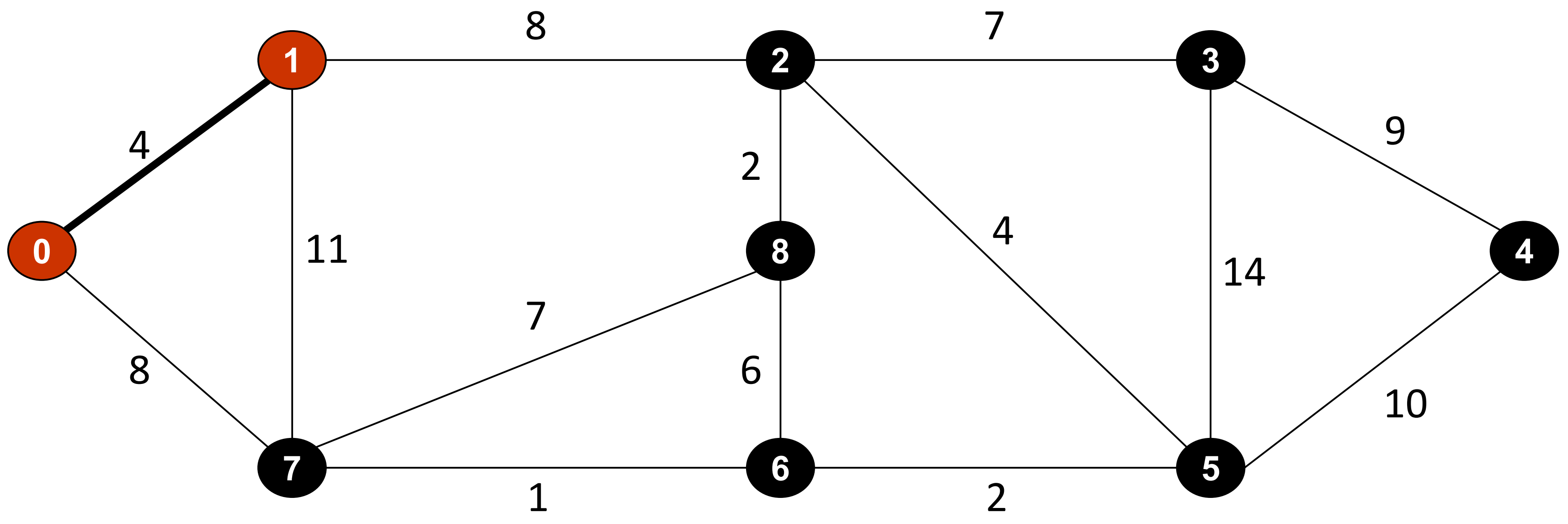
Prims Algorithm

- Initialization
- Random choice of node 0



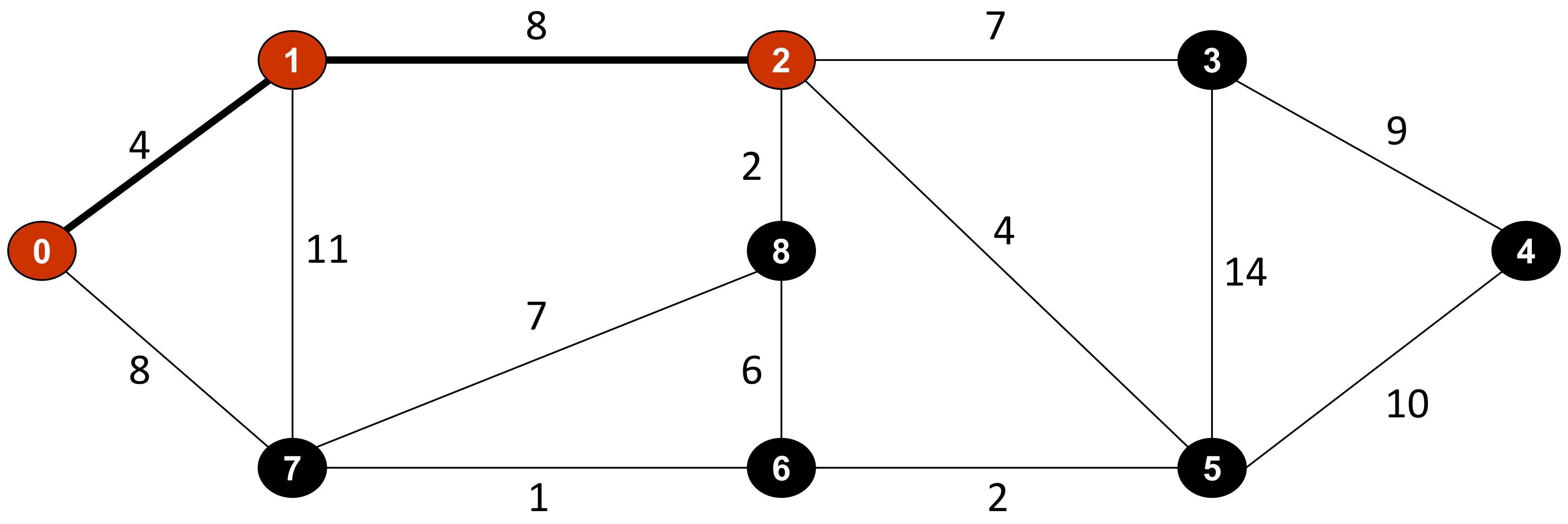
Prims Algorithm

- Selection of edge 0-1
  - $N_C \leftarrow \{0\} \cup \{B\}$
  - $E_S \leftarrow \{0,1\}$



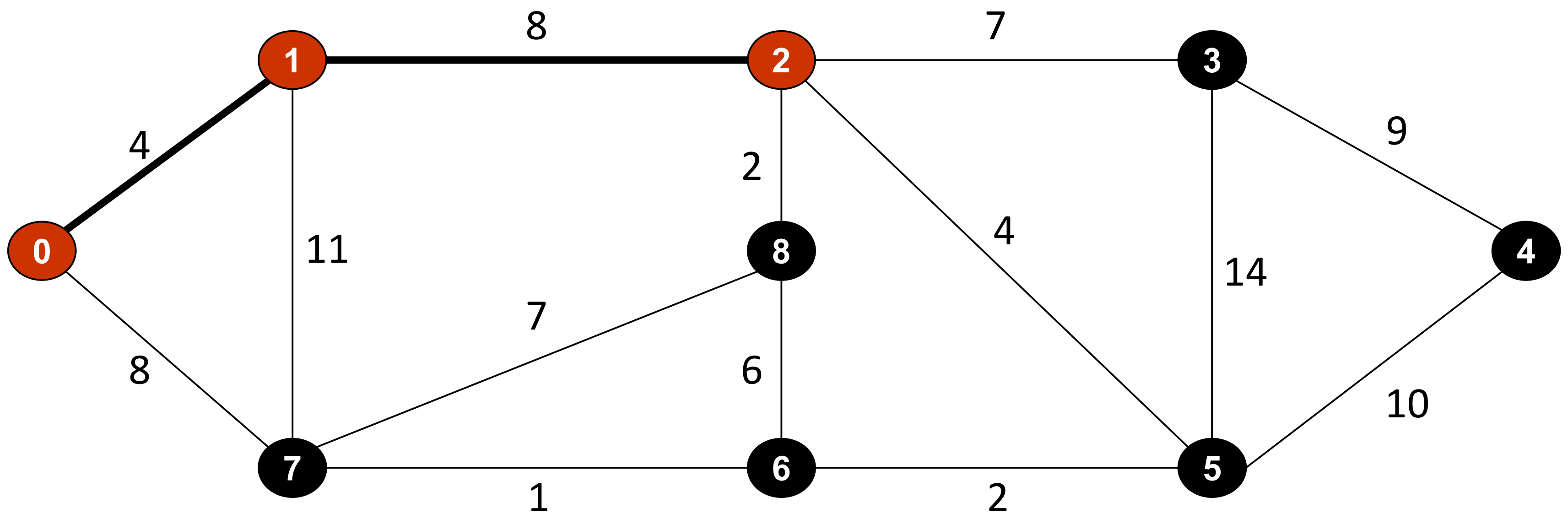
Prims Algorithm

- Selection of “cheapest” (shortest) edge



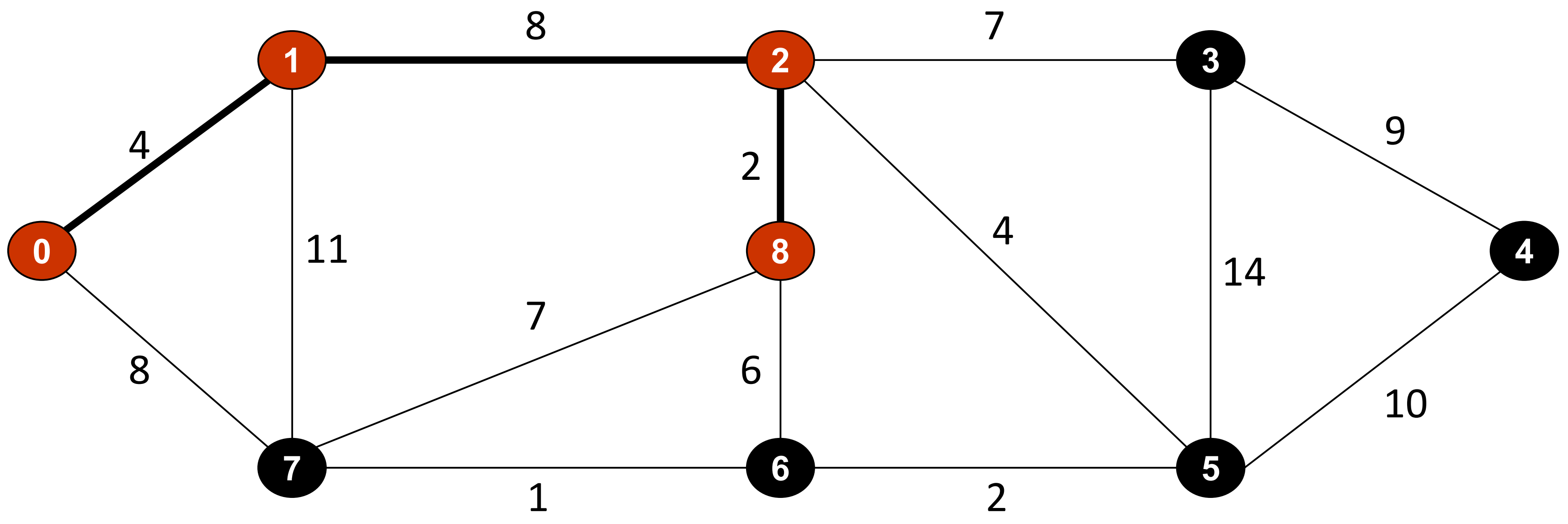
Prims Algorithm

- Selection of edge 1-2
  - $N_C \leftarrow \{0,1\} \cup \{2\}$
  - $E_S \leftarrow \{(0,1)\} \cup \{(1,2)\}$



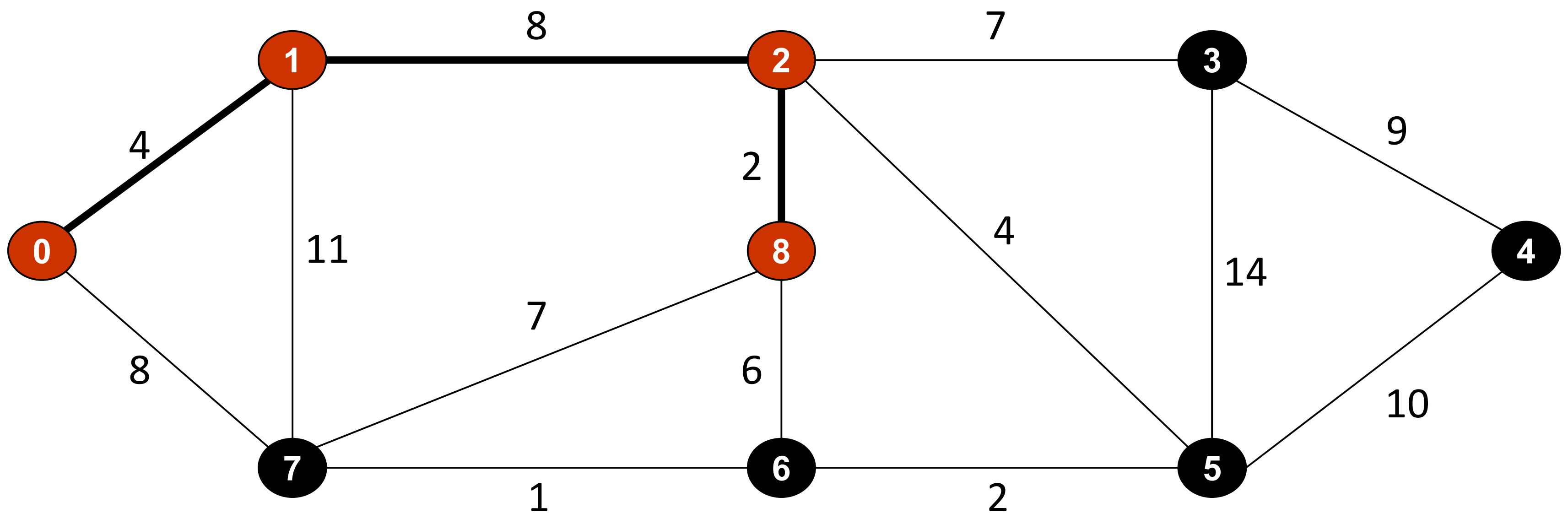
Prims Algorithm

- Selection of “cheapest” (shortest) edge



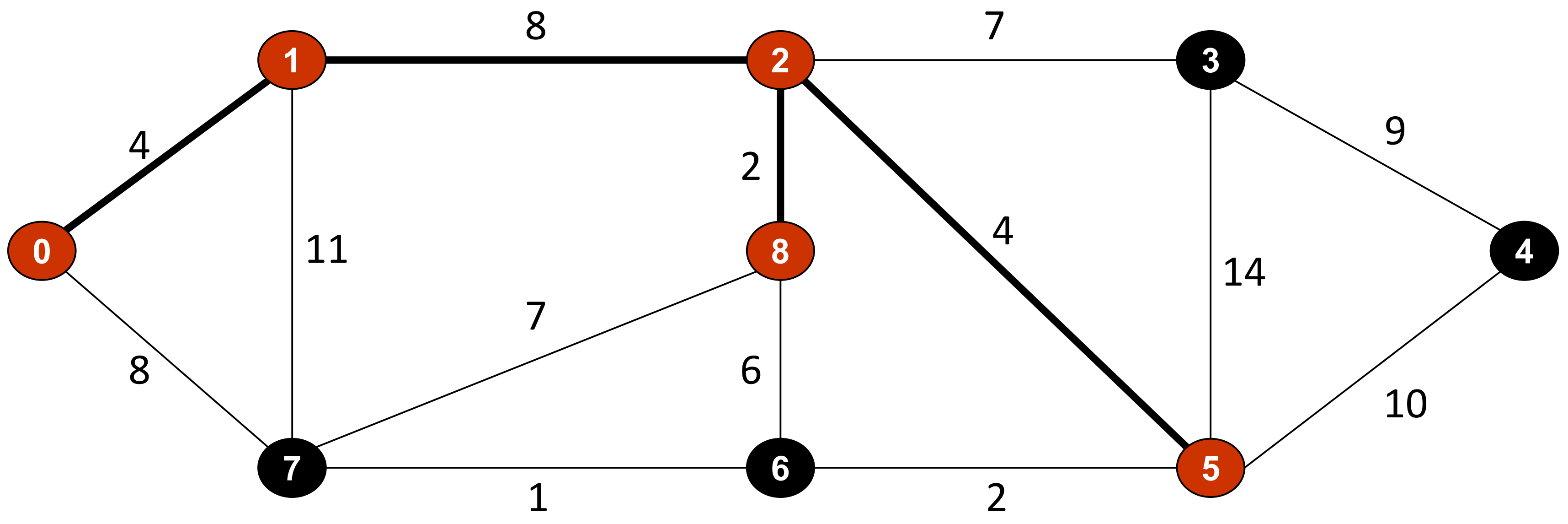
Prims Algorithm

- Selection of edge 2-8
  - $N_C \leftarrow \{0,1,2\} \cup \{8\}$
  - $E_S \leftarrow \{(0,1),(1,2)\} \cup \{(2,8)\}$



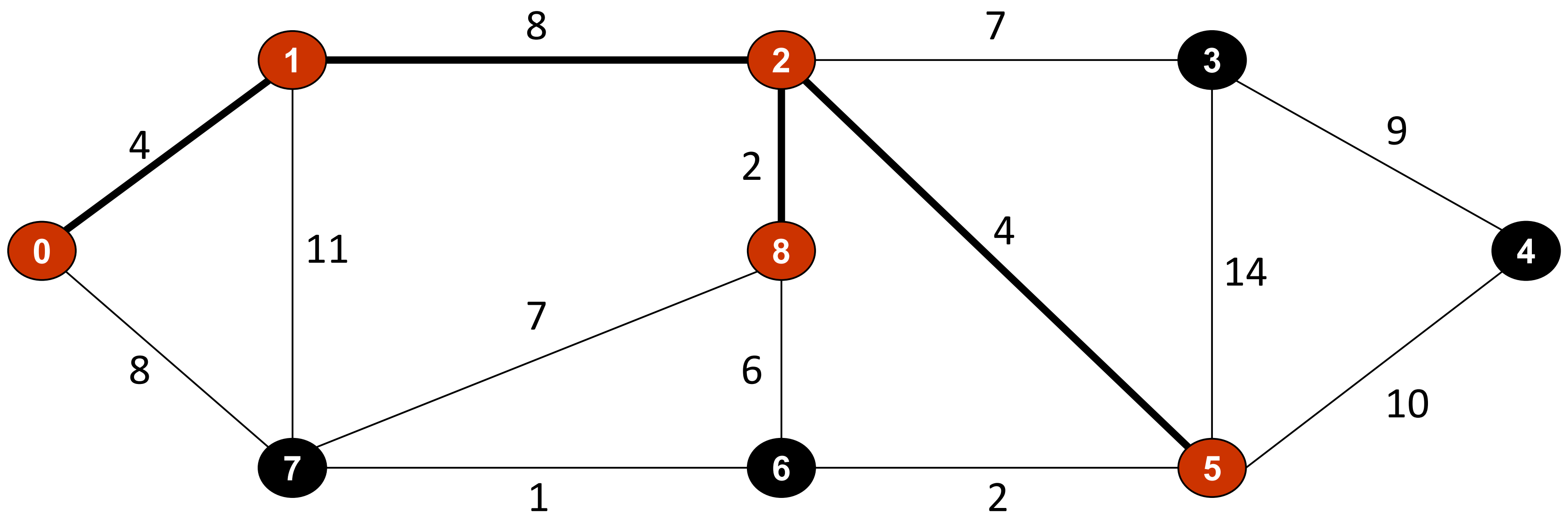
Prims Algorithm

- Selection of “cheapest” (shortest) edge



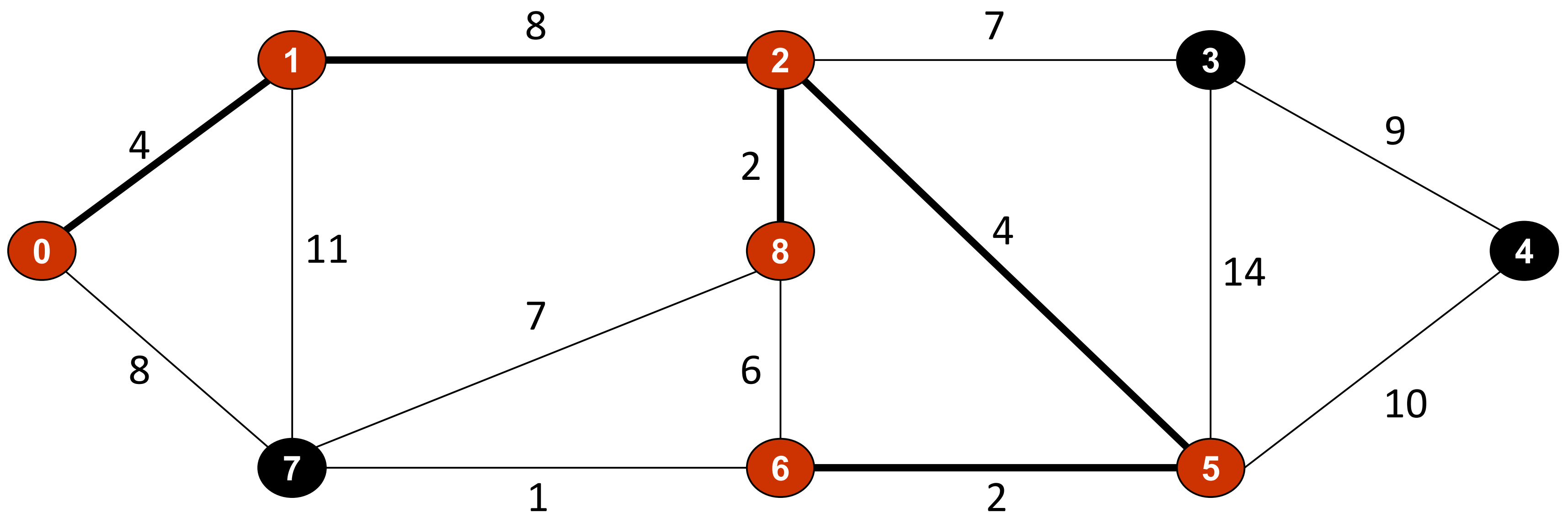
Prims Algorithm

- Selection of edge 2-5
  - $N_C \leftarrow \{0,1,2,8\} \cup \{5\}$
  - $E_S \leftarrow \{(0,1),(1,2),(2,8)\} \cup \{(2,5)\}$



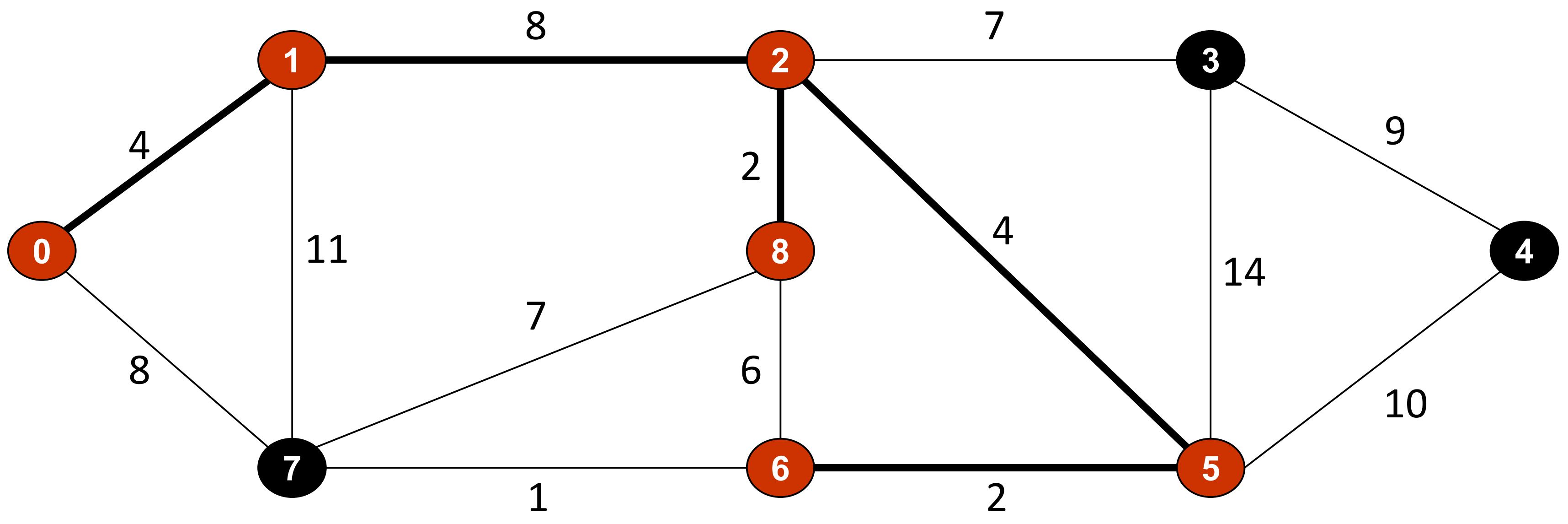
Prims Algorithm

- Selection of “cheapest” (shortest) edge



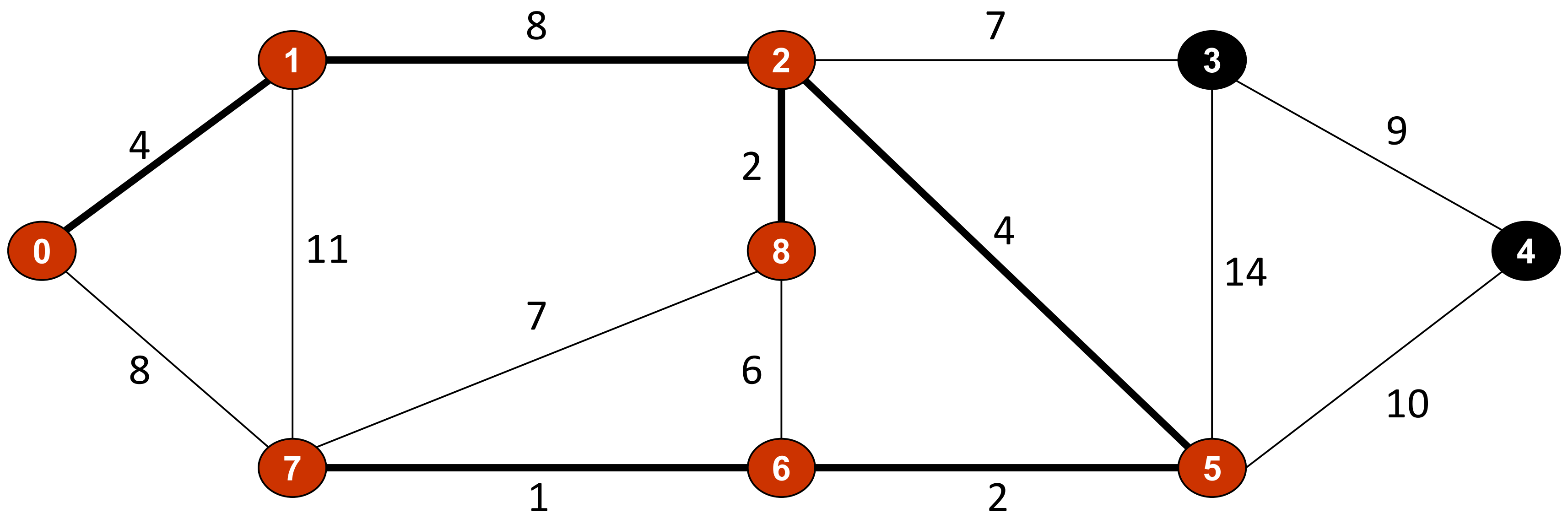
Prims Algorithm

- Selection of edge 5-6
  - $N_C \leftarrow \{0,1,2,8,5\} \cup \{5\}$
  - $E_S \leftarrow \{(0,1),(1,2),(2,8),(2,5)\} \cup \{(5,6)\}$



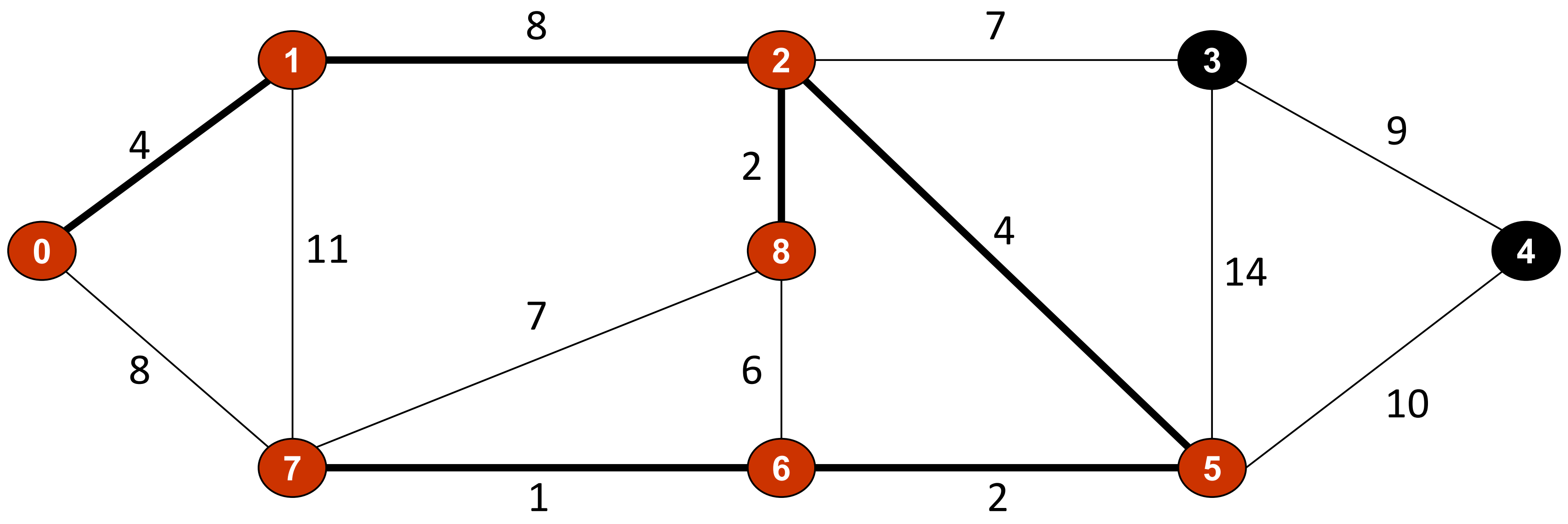
Prims Algorithm

- Selection of “cheapest” (shortest) edge



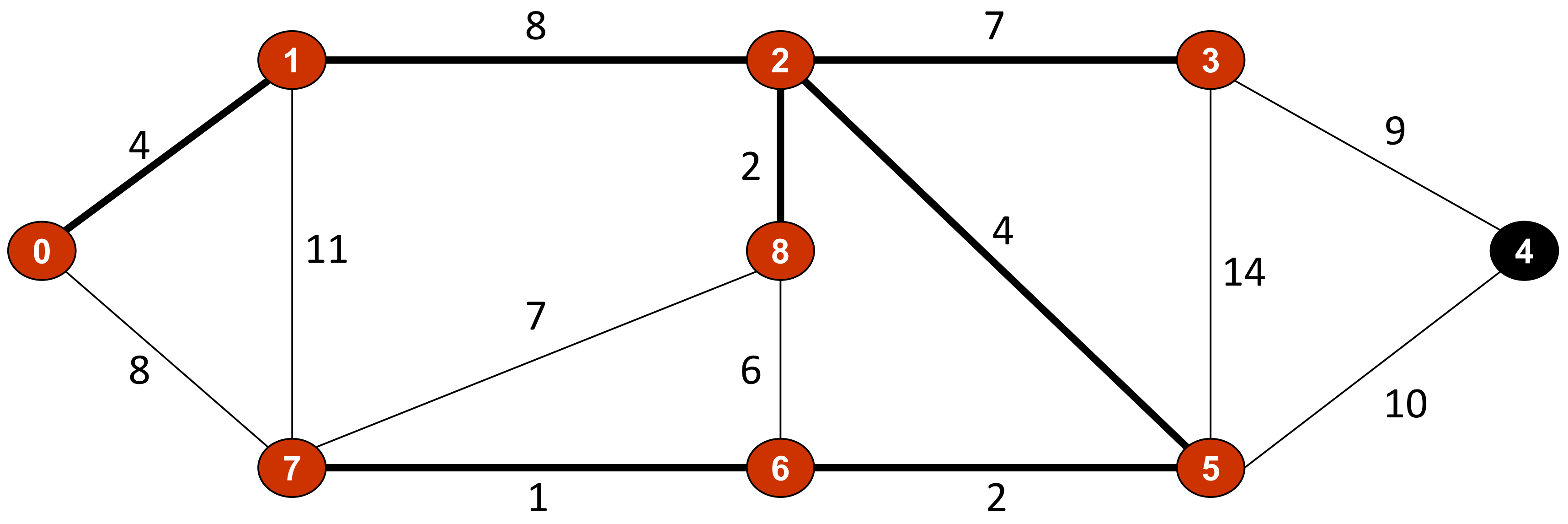
Prims Algorithm

- Selection of edge 6-7
  - $N_C \leftarrow \{0,1,2,8,5,6\} \cup \{7\}$
  - $E_S \leftarrow \{(0,1),(1,2),(2,8),(2,5),(5,6)\} \cup \{(6,7)\}$



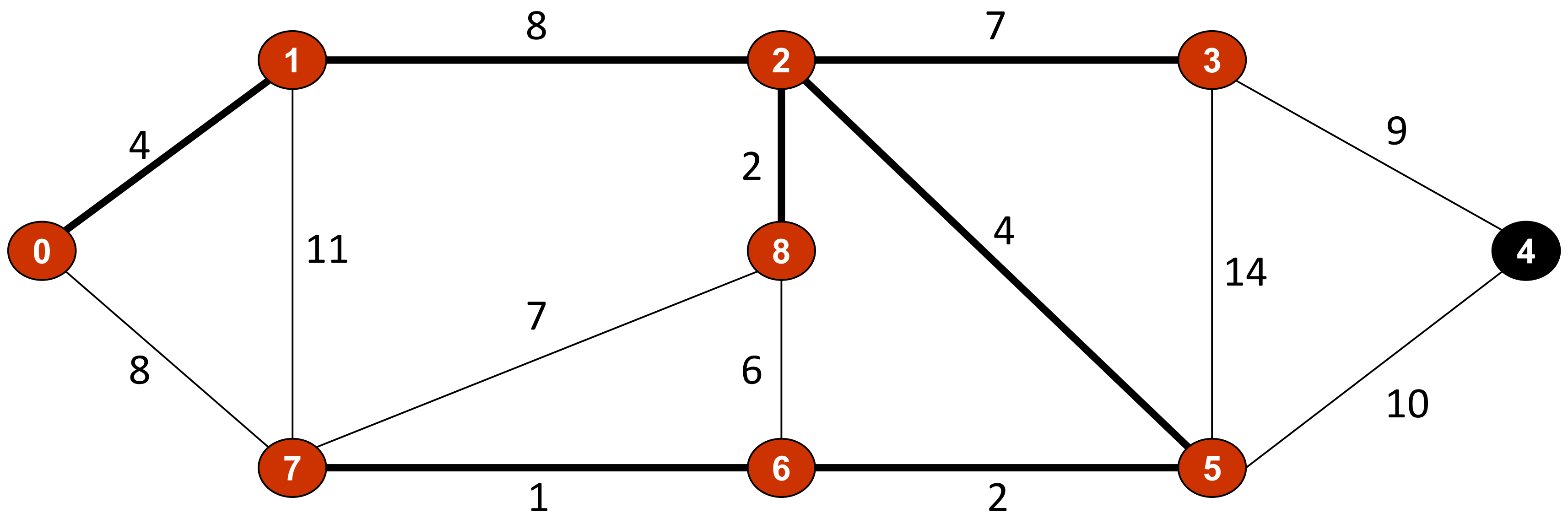
Prims Algorithm

- Selection of “cheapest” (shortest) edge



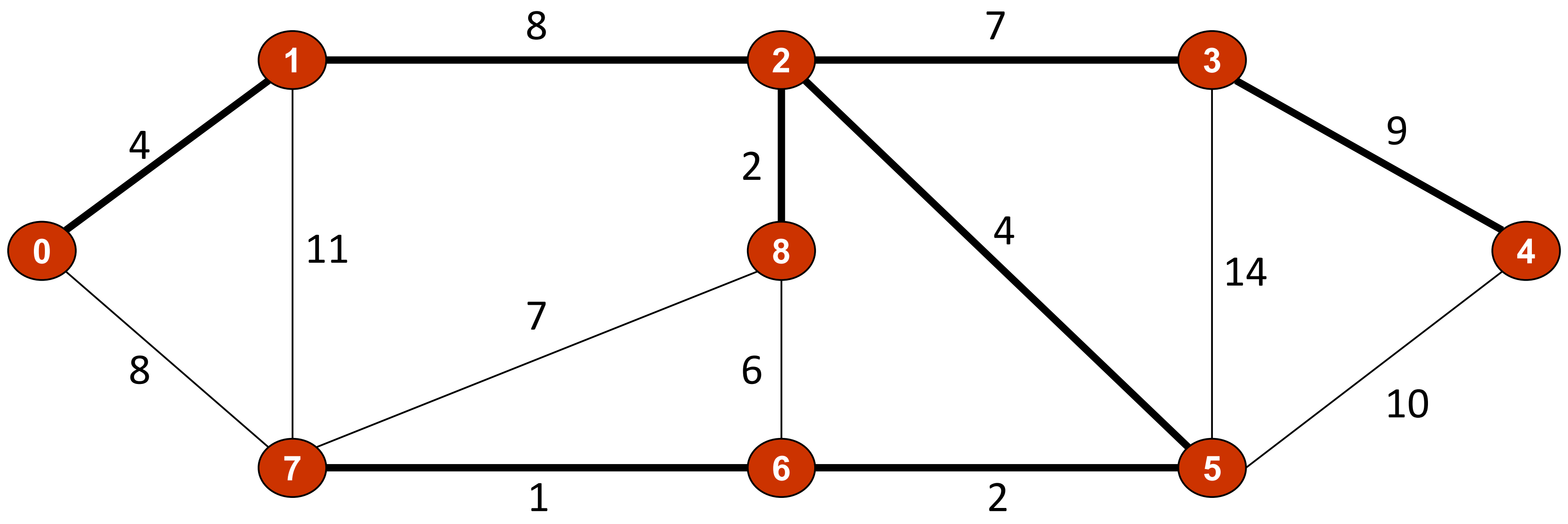
Prims Algorithm

- Selection of edge 2-3
  - $N_C \leftarrow \{0,1,2,8,5,6,7\} \cup \{3\}$
  - $E_S \leftarrow \{(0,1), (1,2), (2,8), (2,5), (5,6), (6,7)\} \cup \{(2,3)\}$



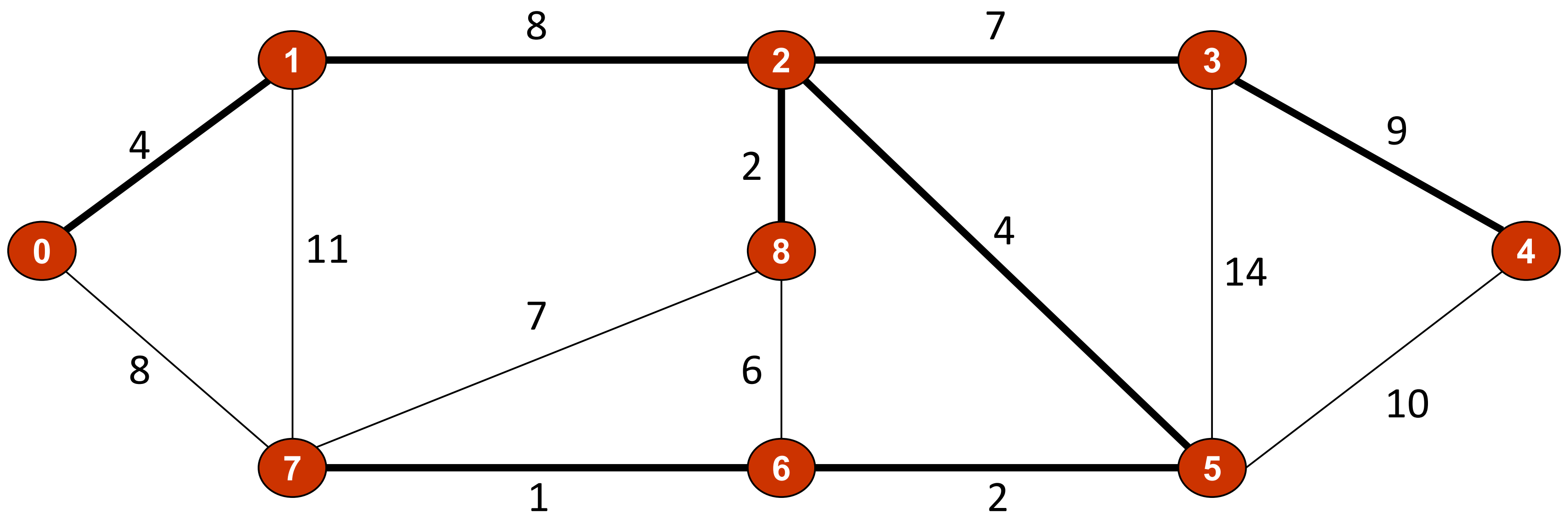
Prims Algorithm

- Selection of “cheapest” (shortest) edge



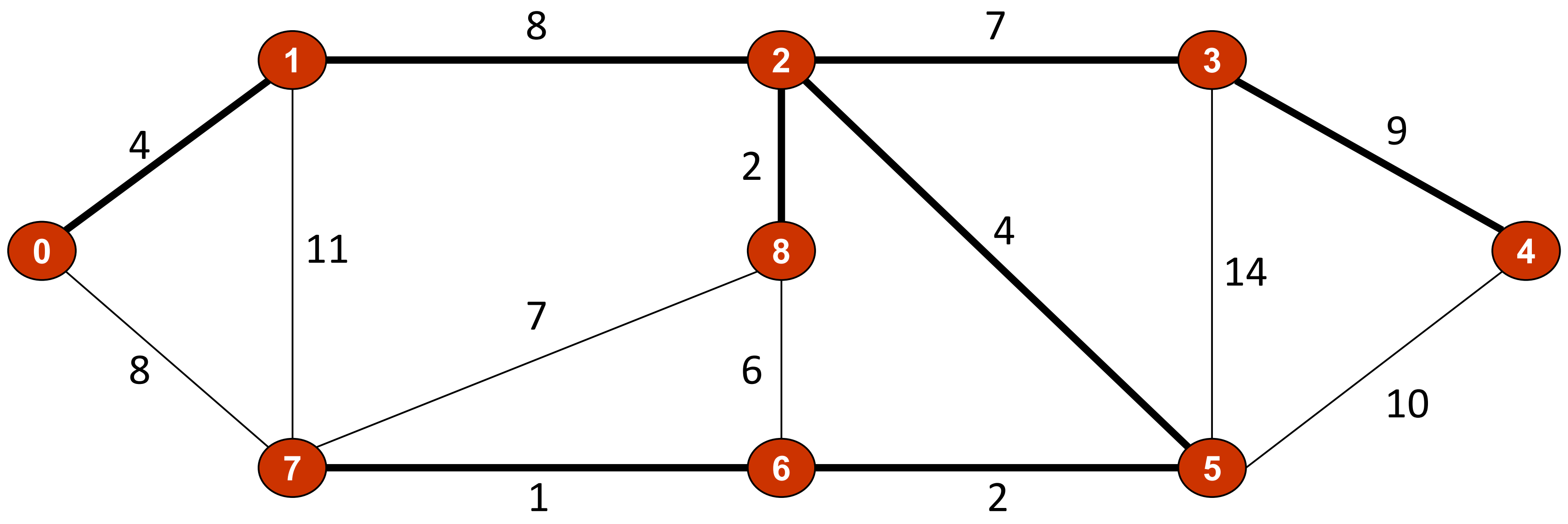
Prims Algorithm

- Selection of edge 3-4
  - $N_C \leftarrow \{0,1,2,8,5,6,7,3\} \cup \{3\}$
  - $E_S \leftarrow \{(0,1),(1,2),(2,8),(2,5),(5,6),(6,7),(2,3)\} \cup \{(3,4)\}$



Prims Algorithm

- $E_S \leftarrow \{(0,1),(1,2),(2,8),(2,5),(5,6),(6,7),(2,3),(3,4)\}$
- Algorithm termination: $|E_S| = n - 1 = 9 - 1 = 8$
- Total cost: 35



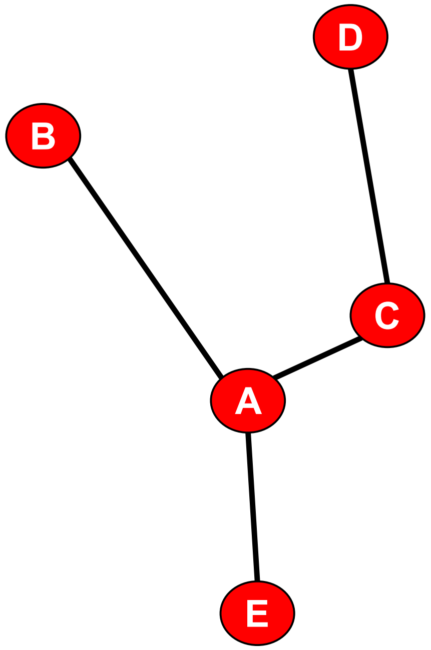
Kruskal’s Algorithm

- *The steps of the Kruskal algorithm are the following:*

1. *Sort all candidate connections/edges*
2. *For each edge e*
    1. *Check if the selection of the edge create a circle in the tree*
    2. *If it does not create a cycle,add this connection to* E_S $(E_S \leftarrow E_S \cup e)$
    3. If $|E_S| = n - 1$, terminate the algorithm

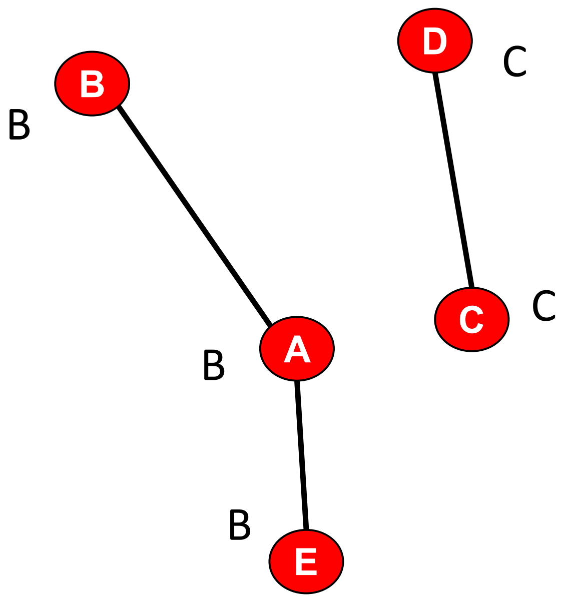
Kruskal’s Algorithm

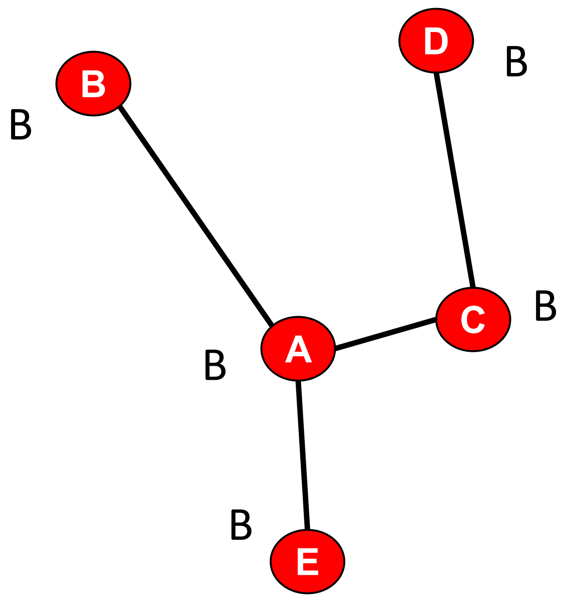
- The check for cycle appearance during the selection of an edge is a challenging step (if we want it to be efficient)
- Let's consider a tree (right)
- The selection of edge (B, D) would create a cycle
- We observe that every connection of two nodes that belong to the same tree creates a cycle
- Therefore, if we would design an algorithm that checks for the creation of cycles during the selection of an edge, we would practically design an algorithm that checks if the two nodes of the selected candidate edge belong to the same tree
- Therefore, each node must contain info that uniquely defines the identity of the tree that the node belongs to
- In order to achieve this, every node that belongs to a tree is associated with a common node that is considered the tree representative



Kruskal's Algorithm

- However, the representation requirement for each node that belongs to a tree creates the following problem:
- How is the representative of the nodes that belong to two merging trees selected?
- E.g. When the two trees below merge to create a new tree, we require all nodes of the new tree to have the same representative node



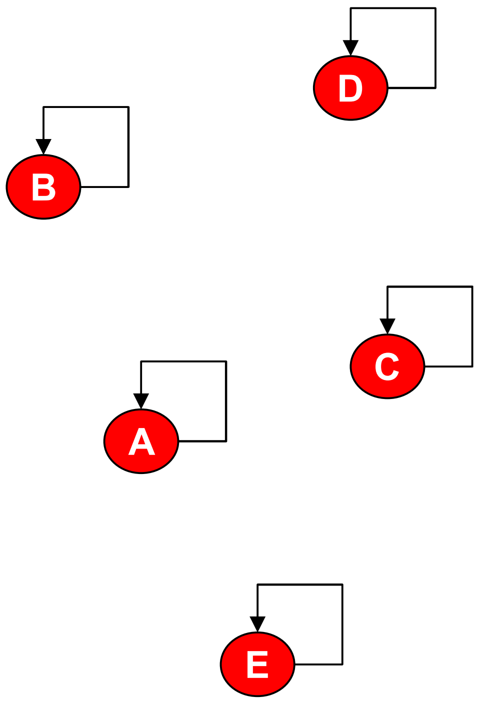


Kruskal’s Algorithm

- In order to achieve this we adopt the following idea:
- Every node has a “parent” node When the node is not connected to any other nodes, the node parent is the node itself
  - A node is the representative of a set of nodes , if its parent node is the node itself
- Every time two nodes A and B are connected, we find the representatives of the nodes $r(A)$ and $r(B)$
- One of $r(A)$ and $r(B)$ is selected as the parent of both $r(A)$ and $r(B)$
- Therefore, whenever we want to find the representative of a node, we follow the sequence of parents
- We trace the representative, when we reach a node whose parent is the node itself

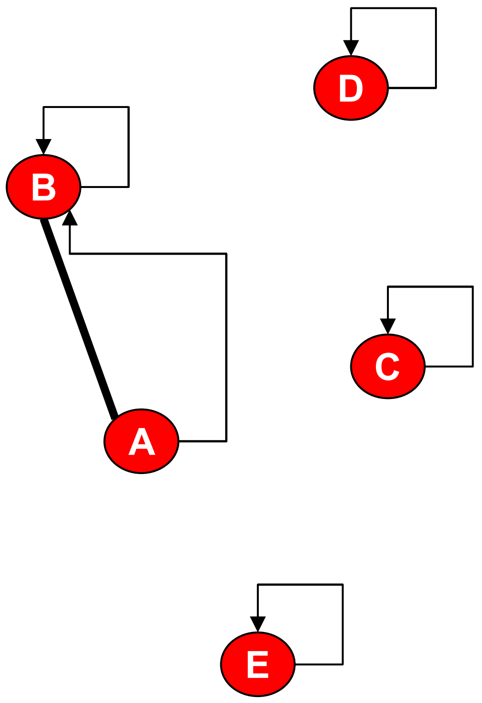
Kruskal’s Algorithm

- Initially, the nodes are not connected, and therefore each one of them, is connected to itself
- Therefore, the representative of each node is the node itself



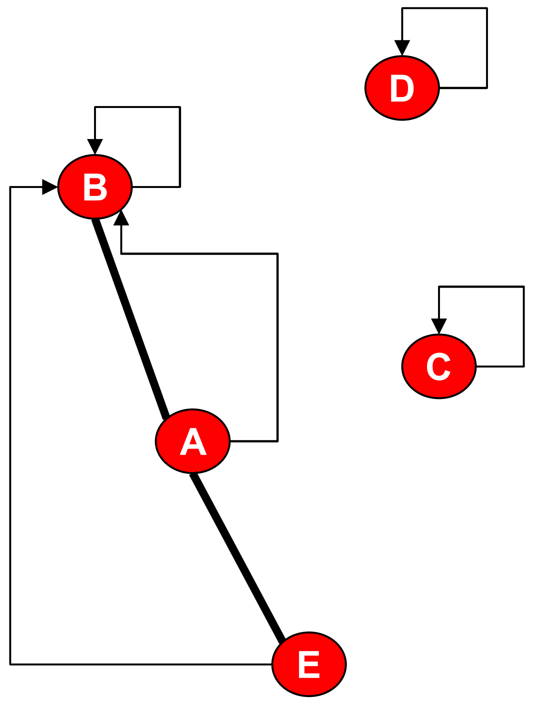
Kruskal’s Algorithm

- Connect A with B
- Assume that node A and node B are connected with edge (A, B)
- Representative of A: A
- Representative of B: B
- Parent of A: B



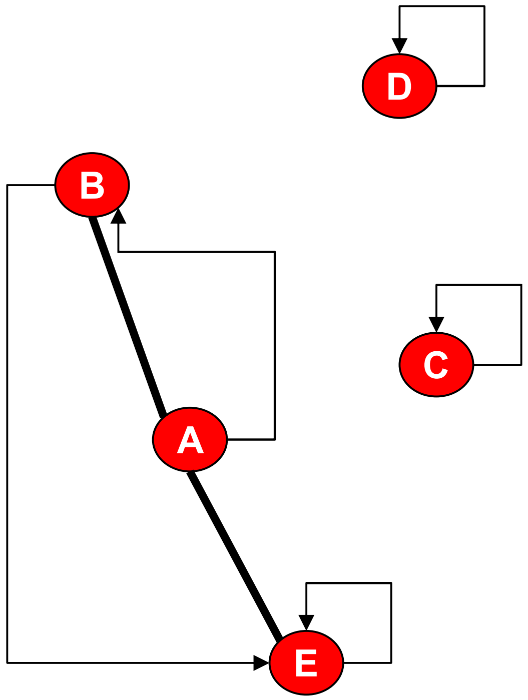
Kruskal’s Algorithm

- Connect A with E
- Assume that node A and node E are connected with edge (A, E)
- Representative of A: B
- Representative of E: E
- Parent of E: B
- Representative of A B E: node B



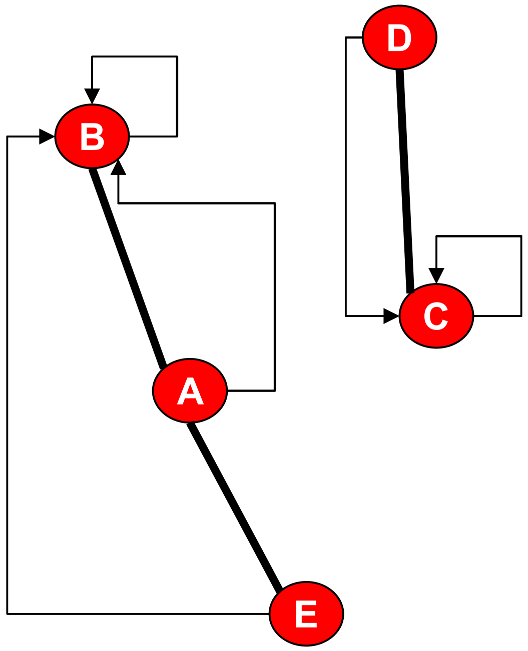
Kruskal’s Algorithm

- **COMMENT – Alternative parent selection**
- Connect A with E
- Assume that node A node E are connected with edge
- Representative of A: B
- Representative of E: E
- Parent of B: E
- Representative of A B E: node E
- In practice, we prefer the previous parent setting as it reduces the tree depth (Union by rank)



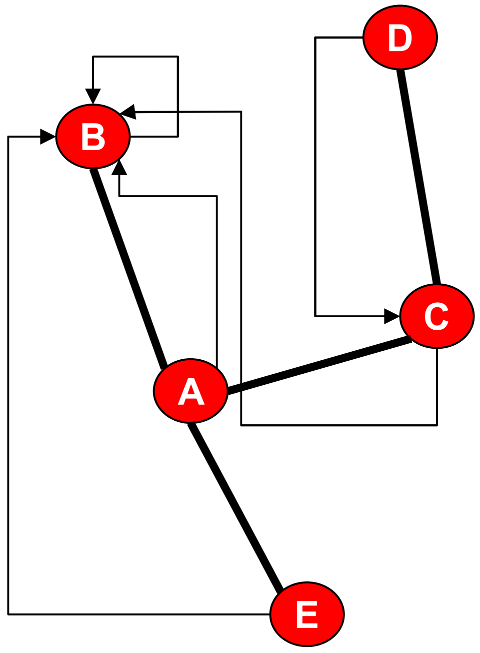
Kruskal’s Algorithm

- Connect C with D
- Assume that node C node D are connected with edge (C, D)
- Representative of C, is C
- Representative of D, is D
- Parent of D: C
- Representative of A B C: node B
- Representative of D C: node C

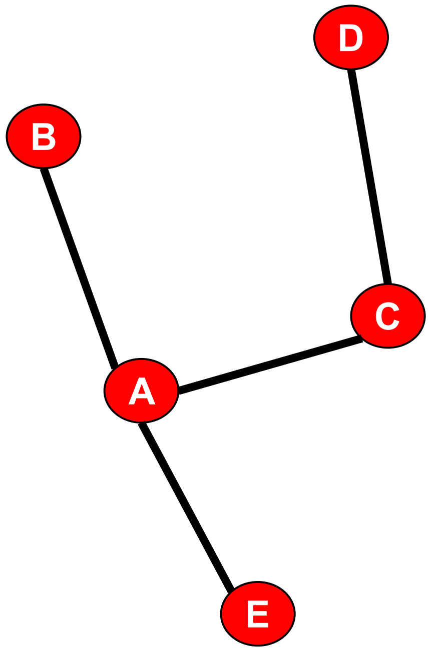


Kruskal’s Algorithm

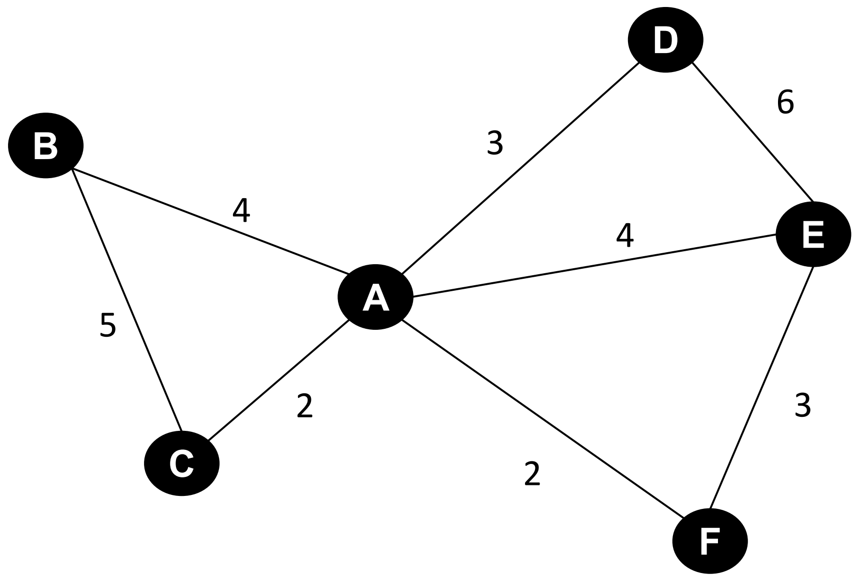
- Connect A with C
- Assume that node A node C are connected with edge (A, C)
- Representative of A, is B
- Representative of C, is C
- Parent of C: B
- Representative of A B C D E: node B



Kruskal’s Algorithm



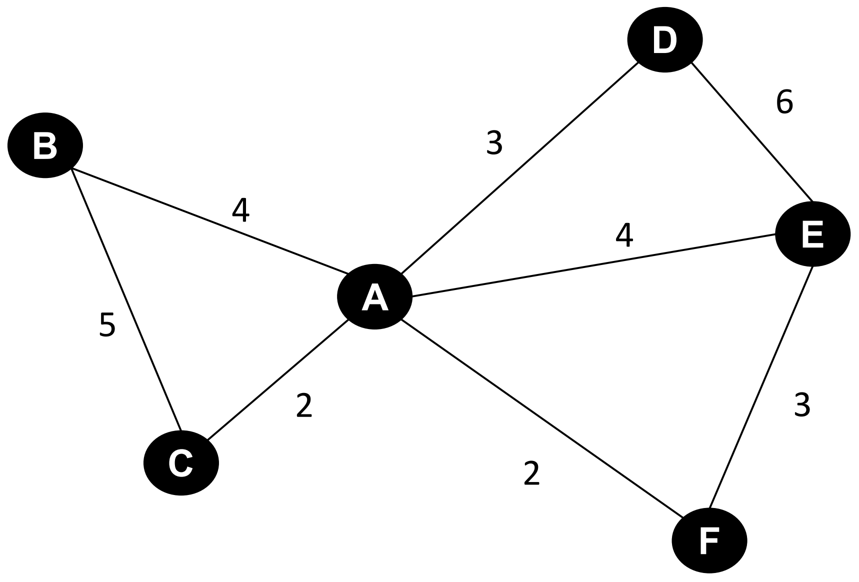
Kruskal’s Algorithm



Kruskal’s Algorithm

- Edge sorting (ascending)

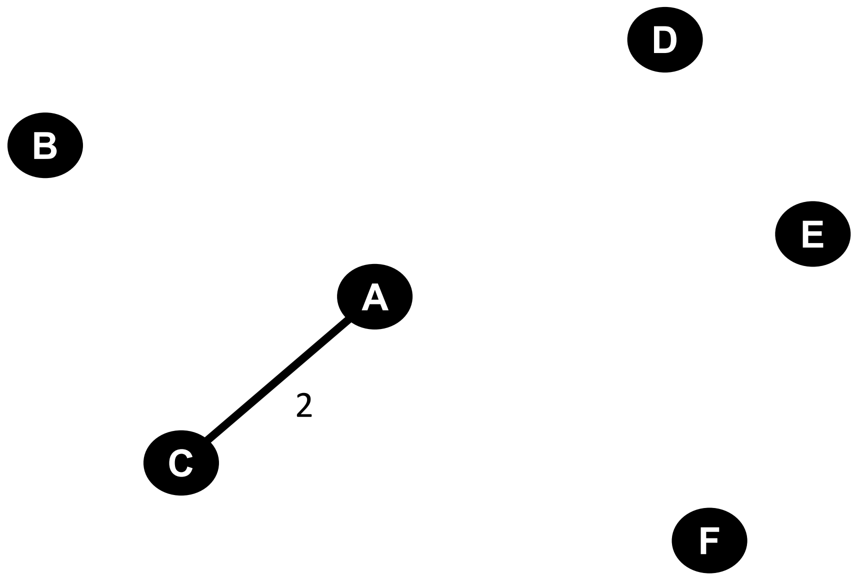
Edge	Cost
AC	2
AF	2
EF	3
AD	3
AE	4
AB	4
BC	5
DE	6



Kruskal’s Algorithm

- Select AC

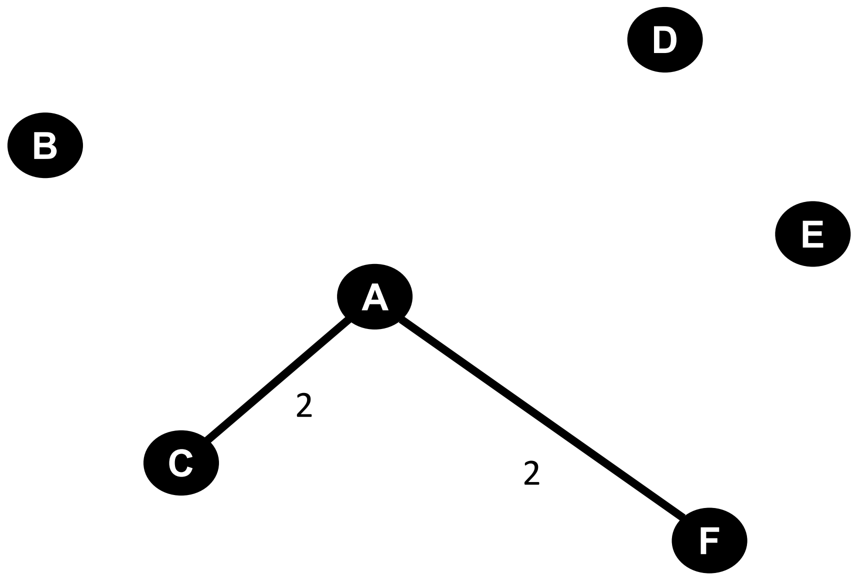
Edge	Cost
AC	**2**
AF	2
EF	3
AD	3
AE	4
AB	4
BC	5
DE	6



Kruskal’s Algorithm

- Select AF

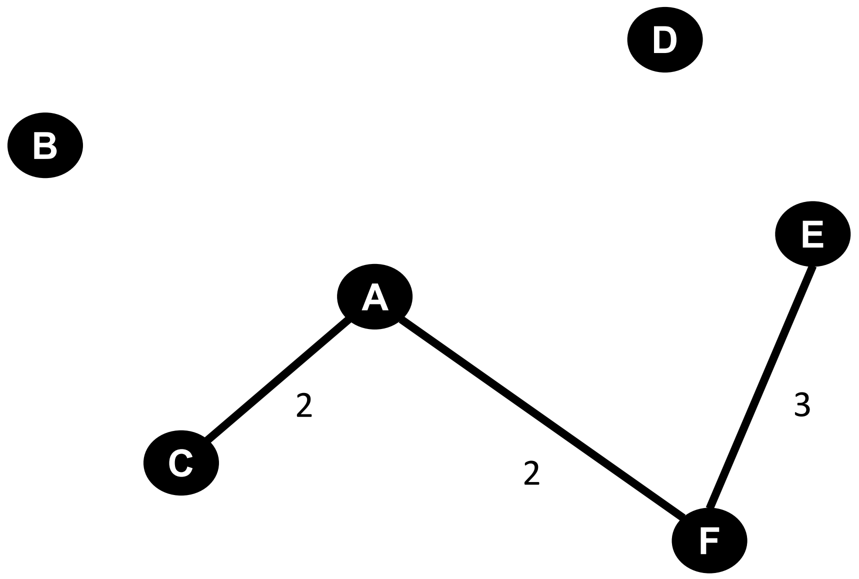
Edge	Cost
AC	**2**
AF	**2**
EF	3
AD	3
AE	4
AB	4
BC	5
DE	6



Kruskal's Algorithm

- Select EF

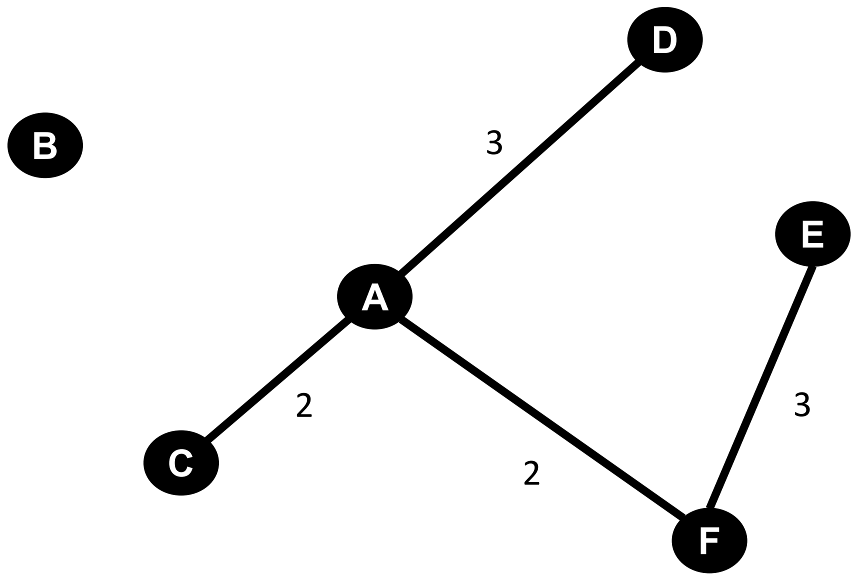
Edge	Cost
AC	**2**
AF	**2**
EF	**3**
AD	3
AE	4
AB	4
BC	5
DE	6



Kruskal's Algorithm

- Select AD

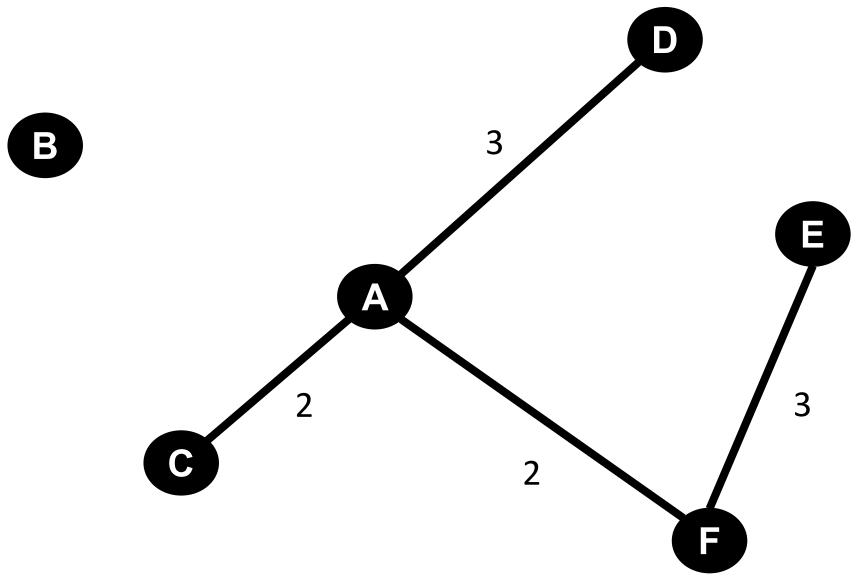
Edge	Cost
AC	**2**
AF	**2**
EF	**3**
AD	**3**
AE	4
AB	4
BC	5
DE	6



Kruskal’s Algorithm

- Edge AE is not selected as it creates a cycle
- It connects the nodes A and E that belong to the same tree

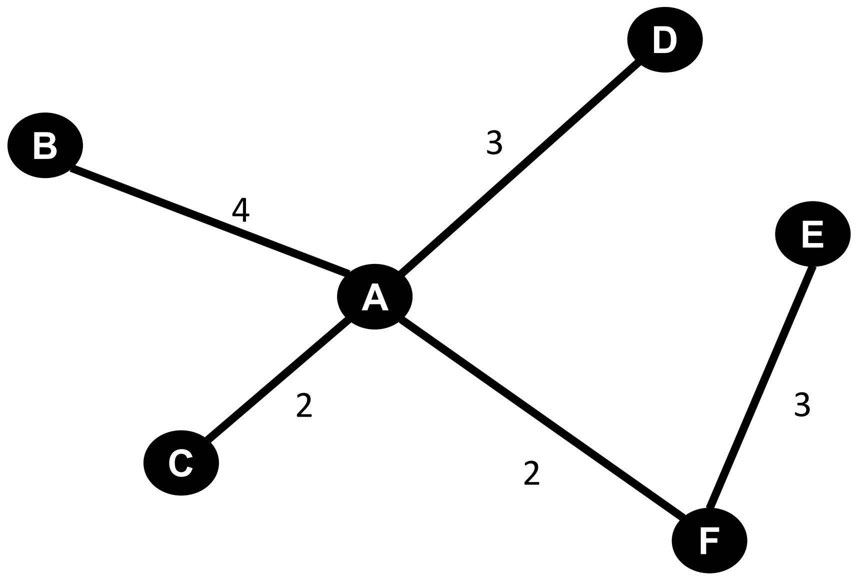
Edge	Cost
AC	**2**
AF	**2**
EF	**3**
AD	**3**
AE	4
AB	4
BC	5
DE	6



Kruskal's Algorithm

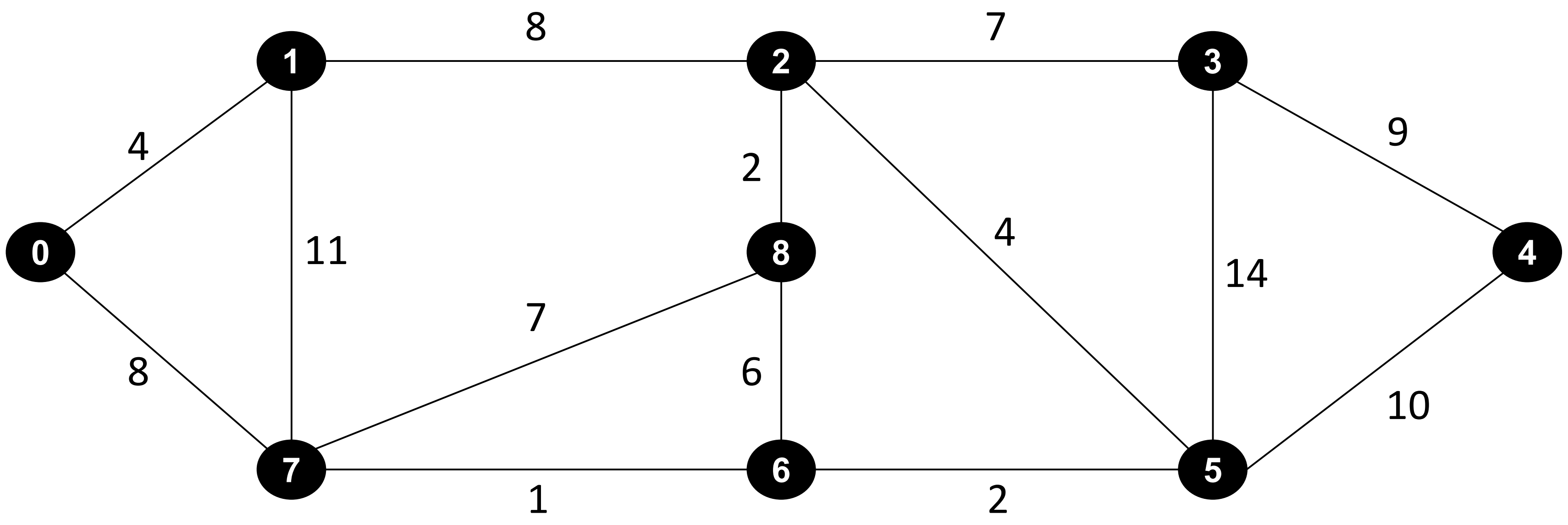
- Select AB
- Algorithm termination
- Final solution cost: 14

Edge	Cost
AC	**2**
AF	**2**
EF	**3**
AD	**3**
AE	4
AB	**4**
BC	5
DE	6



Kruskal’s Algorithm

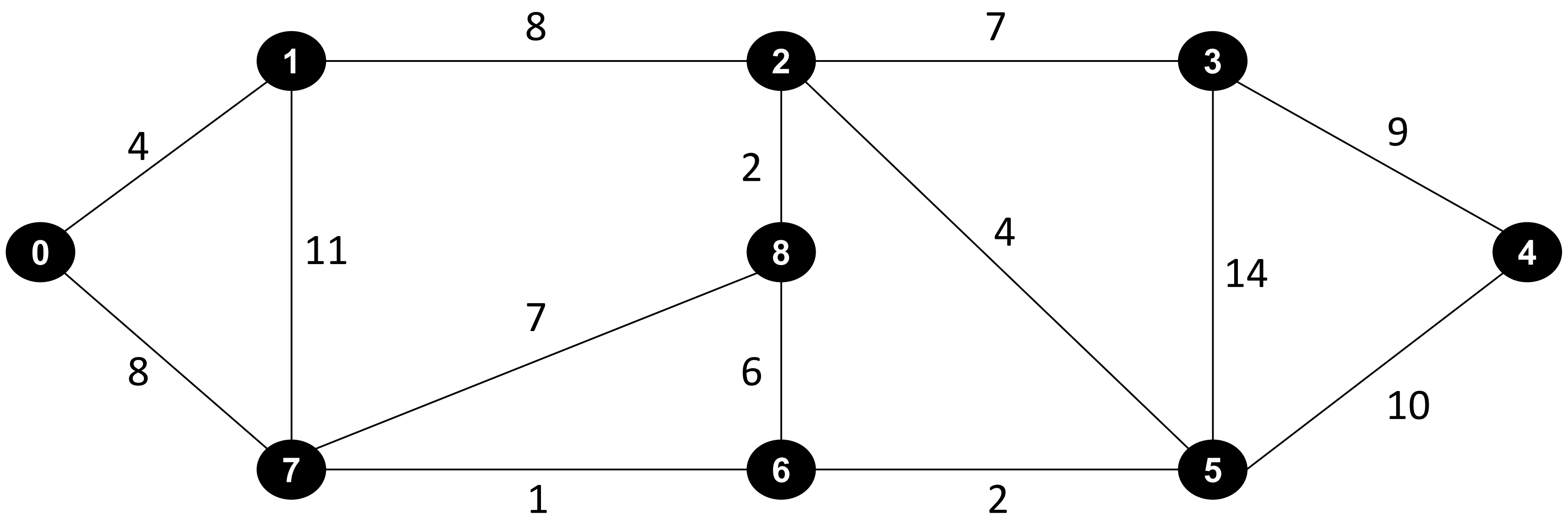
- Apply the Kruskal’s algorithm to the graph below, in order to construct the Minimum Spanning Tree of all 9 nodes



Kruskal’s Algorithm

- Apply the Kruskal’s algorithm to the graph below, in order to construct the Minimum Spanning Tree of all 9 nodes

(6,7):1	(5,6):2	(2,8):2	(0,1):4	(2,5):4	(6,8):6	(2,3):7
(7,8):7	(1,2):8	(0,7):8	(3,4):9	(4,5):10	(1,7):11	(3,5):14


0
1
2
3
4
5
6
7
8
4
8
11
8
7
2
7
6
1
2
4
9
14
10