Operational Research and Optimisation

AQ052-3-M-ORO and VD1

Network Models

Topic & Structure of the lesson

- Minimal spinning tree algorithm
- Shortest route algorithm

Learning Outcomes

- A the end of this topic, You should be able to apply appropriate network algorithms to find the shortest path to the destination.

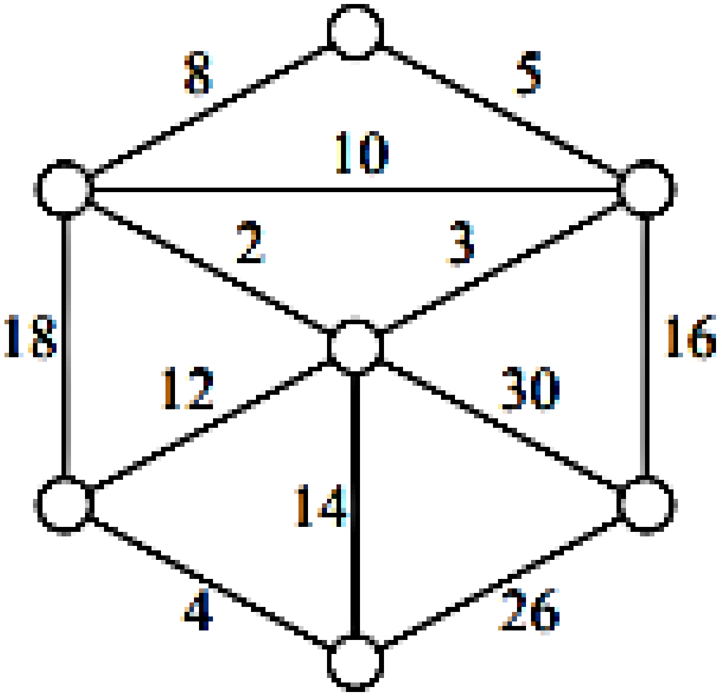
Key Terms you must be able to use

If you have mastered this topic, you should be able to use the following terms correctly in your assignments and exams:

Example of Network Problem

Using
(i) Shortest path algorithm (Dijkstra algorithm)
(ii) Prims minimum spanning tree
(iii) Kruskal minimum spanning tree
(iv) Boruvka algorithm

to find the shortest path from one node to another.

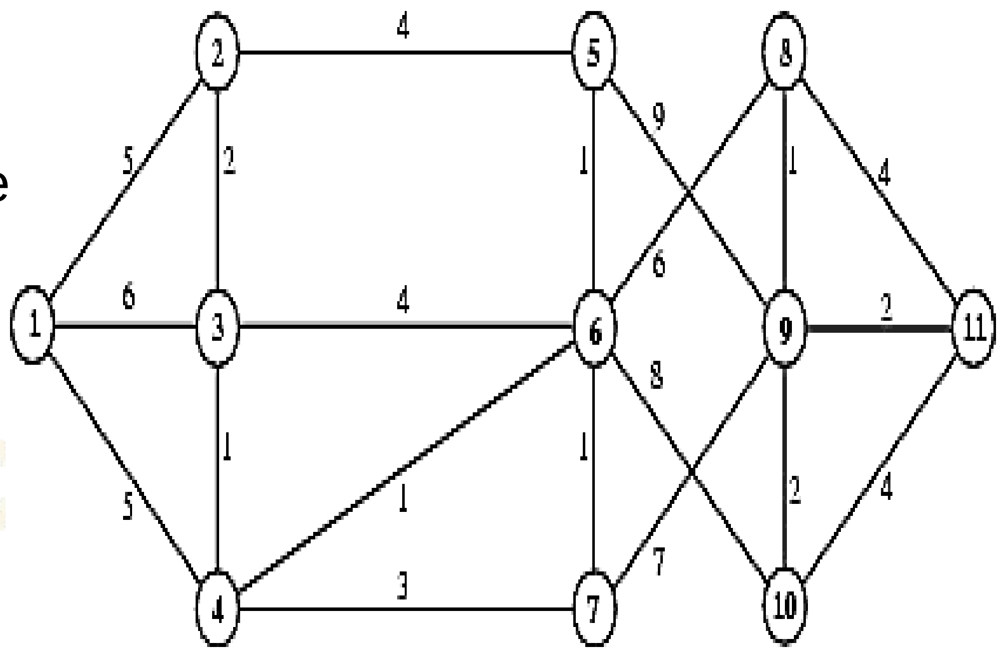


Exercise

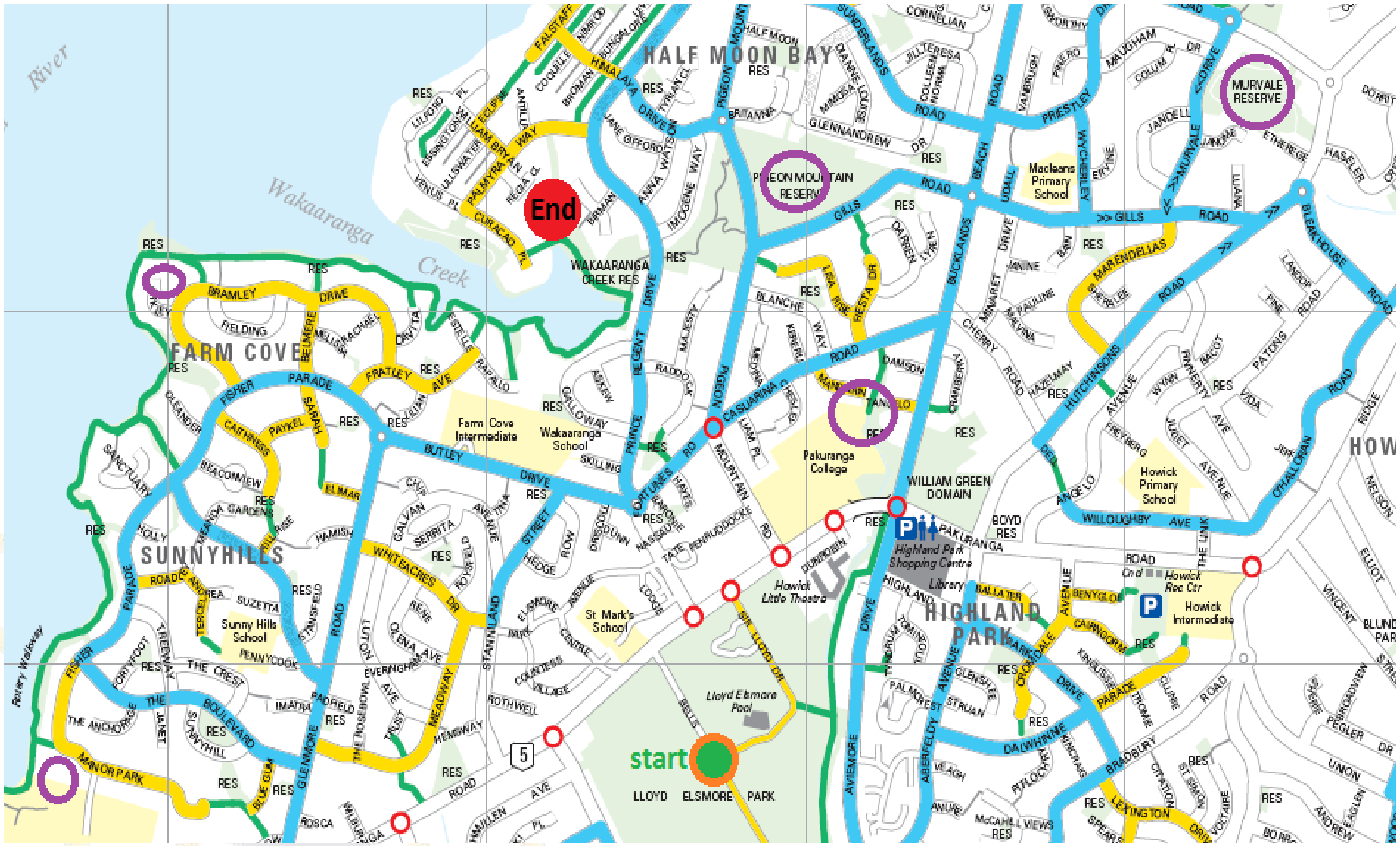
Using the following algorithms:

(i) Shortest path algorithm (Dijkstra algorithm)
(ii) Prims minimum spanning tree
(iii) Kruskal minimum spanning tree
(iv) Boruvka algorithm

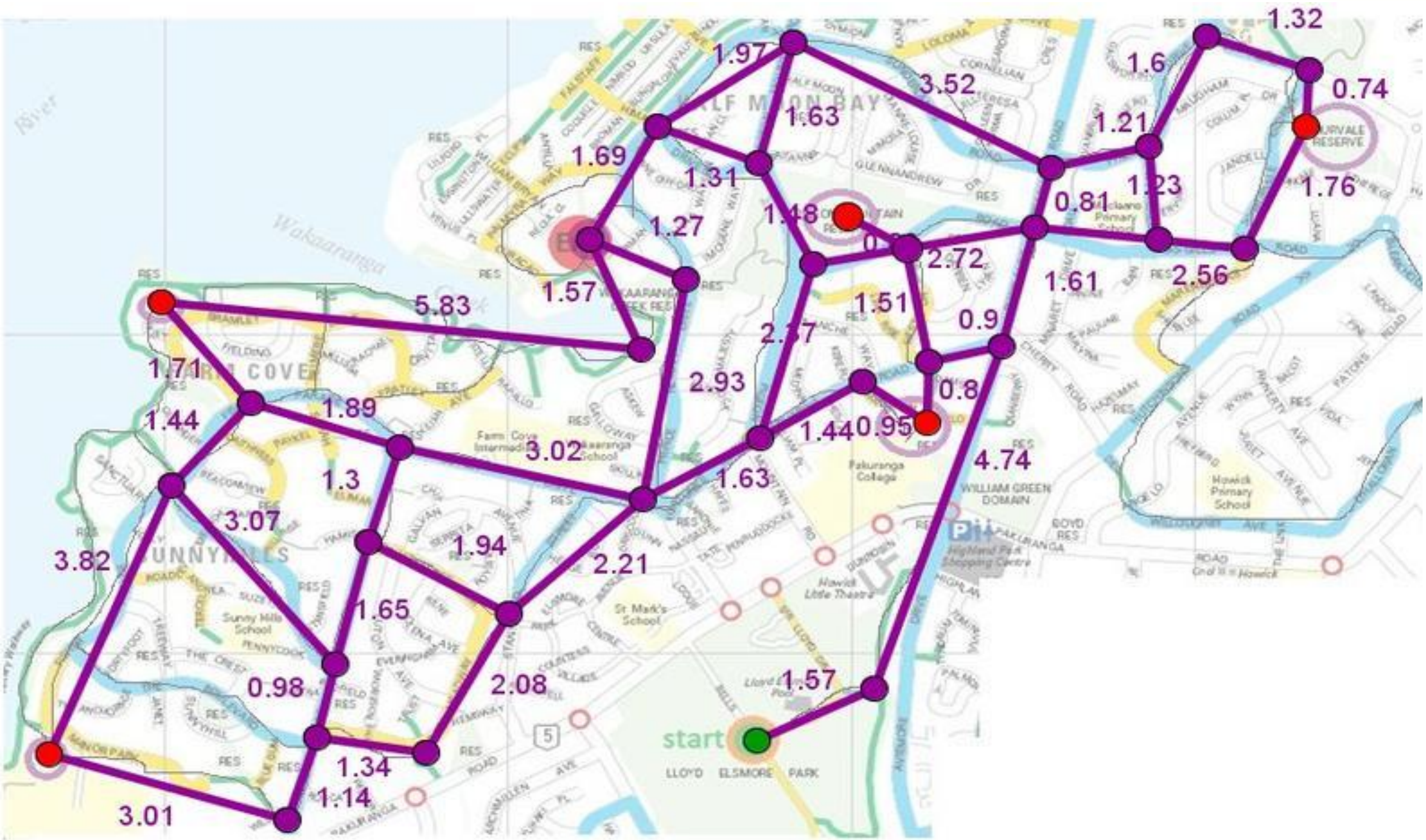
to find the shortest path from Node 1 to Node 11.



Example of Maps



Networks


1.32
1.97
1.6
0.74
3.52
1.63
1.21
1.69
1.31
1.23
1.76
0.81
1.27
1.48
2.72
2.56
1.57
1.61
5.83
1.51
0.9
1.71
2.93
0.8
1.89
1.44
1.44
0.95
3.02
4.74
1.3
1.63
3.07
1.94
3.82
2.21
1.65
1.57
0.98
2.08
start
1.34
1.14
3.01

Example of Solution

Shortest Path Problem

- The <u>shortest path problem</u> is concerned with finding the shortest path in a network from one node (or set of nodes) to another node (or set of nodes).
- If all arcs in the network have nonnegative values then a labeling algorithm can be used to find the shortest paths from a particular node to all other nodes in the network.
- The criterion to be minimized in the shortest-route problem is not limited to distance even though the term "shortest" is used in describing the procedure. Other criteria include time and cost. (Neither time nor cost are necessarily linearly related to distance.)

Shortest-Path Algorithm

Note: We use the notation [ ] to represent a permanent label and ( ) to represent a tentative label.

- Step 1: Assign node 1 the permanent label [0,*S*]. The first number is the distance from node 1; the second number is the preceding node. Since node 1 has no preceding node, it is labeled *S* for the starting node.

- Step 2: Compute tentative labels, (*d*,*n*), for the nodes that can be reached directly from node 1. *d* = the direct distance from node 1 to the node in question -- this is called the distance value. *n* indicates the preceding node on the route from node 1 -- this is called the preceding node value.

Shortest-Path Algorithm

- Step 3: Identify the tentatively labeled node with the smallest distance value. Suppose it is node *k*. Node *k* is now permanently labeled (using [,] brackets). If all nodes are permanently labeled, GO TO STEP 5.

- Step 4: Consider all nodes without permanent labels that can be reached directly from the node *k* identified in Step 3. For each, calculate the quantity *t*, where

$$t = (\text{arc distance from node } k \text{ to node } i) + (\text{distance value at node } k).$$

Shortest-Path Algorithm

- Step 4: (continued)
  - If the non-permanently labeled node has a tentative label, compare *t* with the current distance value at the tentatively labeled node in question.
    - If $t <$ distance value of the tentatively labeled node, replace the tentative label with (t,k).
    - If $t \geq$ distance value of the tentatively labeled node, keep the current tentative label.
  - If the non-permanently labeled node does not have a tentative label, create a tentative label of (t,k) for the node in question.

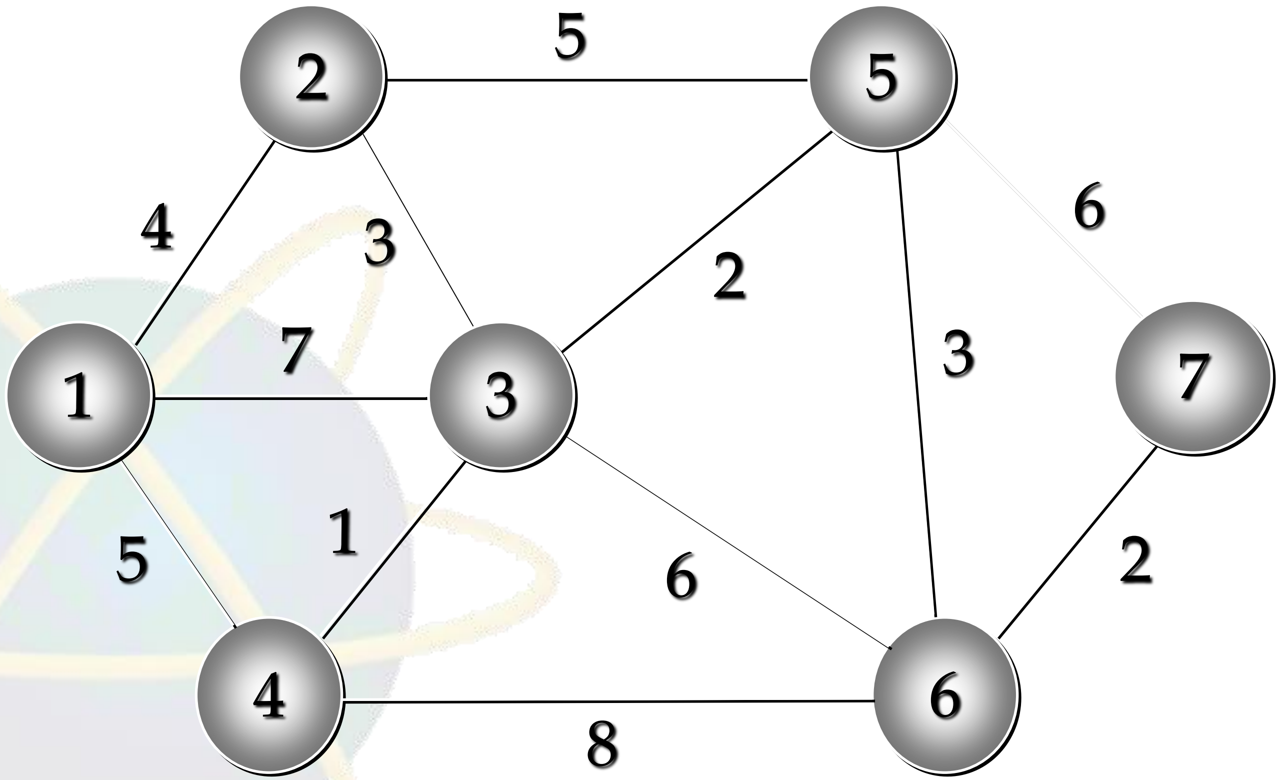
In either case, now GO TO STEP 3.

Shortest-Path Algorithm

- Step 5: The permanent labels identify the shortest distance from node 1 to each node as well as the preceding node on the shortest route. The shortest route to a given node can be found by working backwards by starting at the given node and moving to its preceding node. Continuing this procedure from the preceding node will provide the shortest route from node 1 to the node in question.

Example: Shortest Path

- Find the Shortest Path From Node 1 to All Other Nodes in the Network:

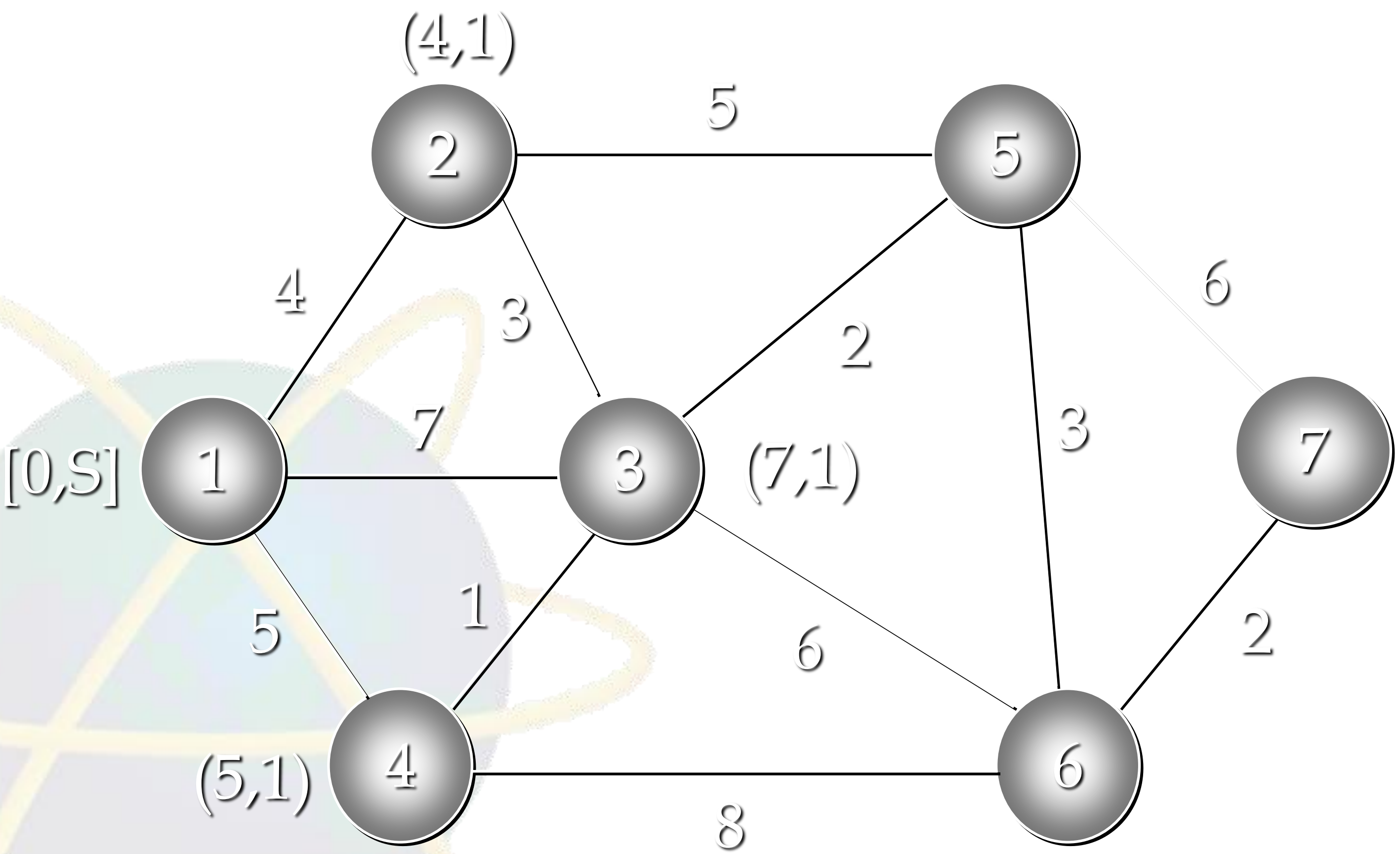


Example: Shortest Path

- Iteration 1
  - Step 1: Assign Node 1 the permanent label [0,S].
  - Step 2: Since Nodes 2, 3, and 4 are directly connected to Node 1, assign the tentative labels of (4,1) to Node 2; (7,1) to Node 3; and (5,1) to Node 4.

Example: Shortest Path

- Tentative Labels Shown

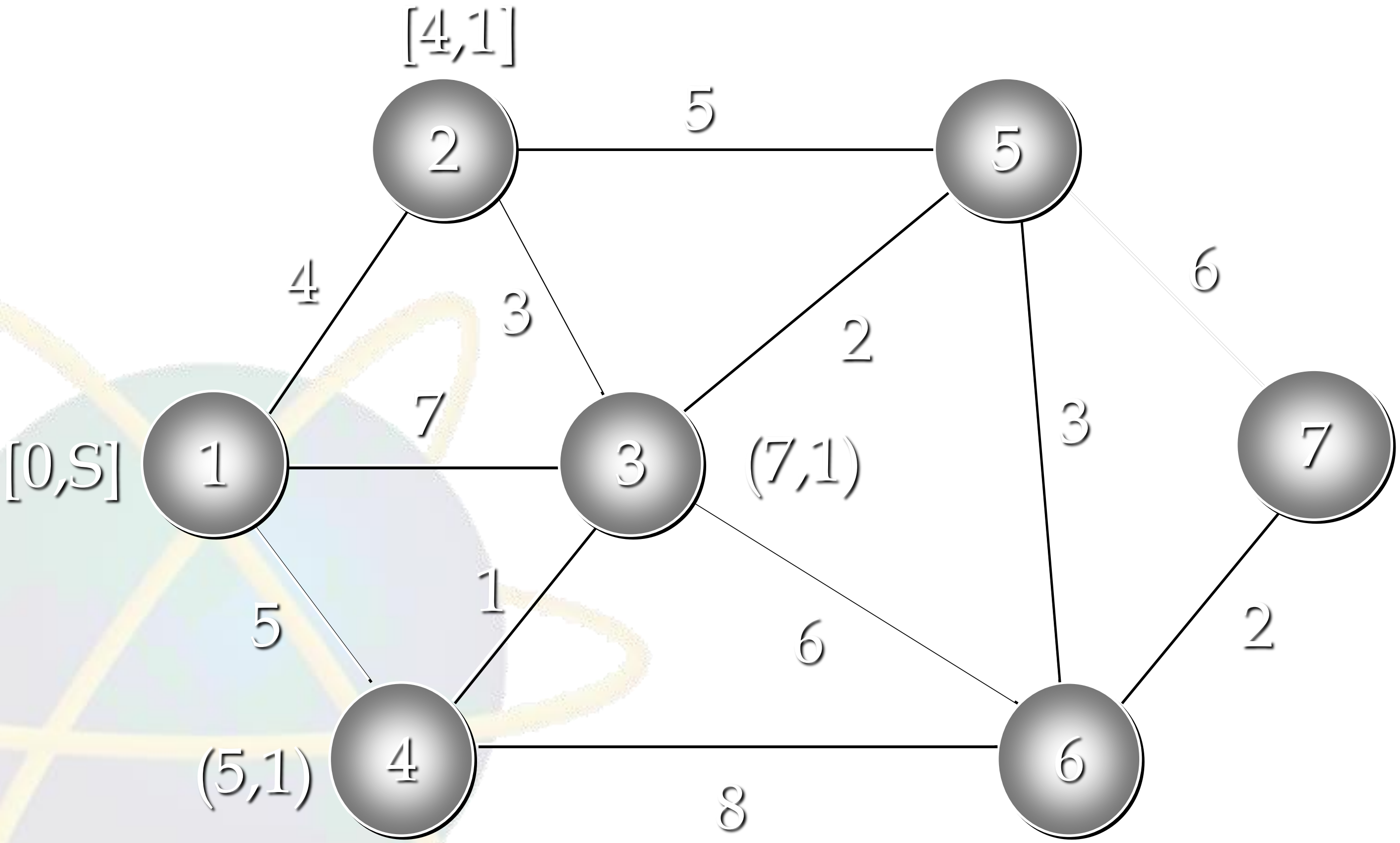


Example: Shortest Path

- Iteration 1
  - Step 3: Node 2 is the tentatively labeled node with the smallest distance (4) , and hence becomes the new permanently labeled node.

Example: Shortest Path

- Permanent Label Shown



Example: Shortest Path

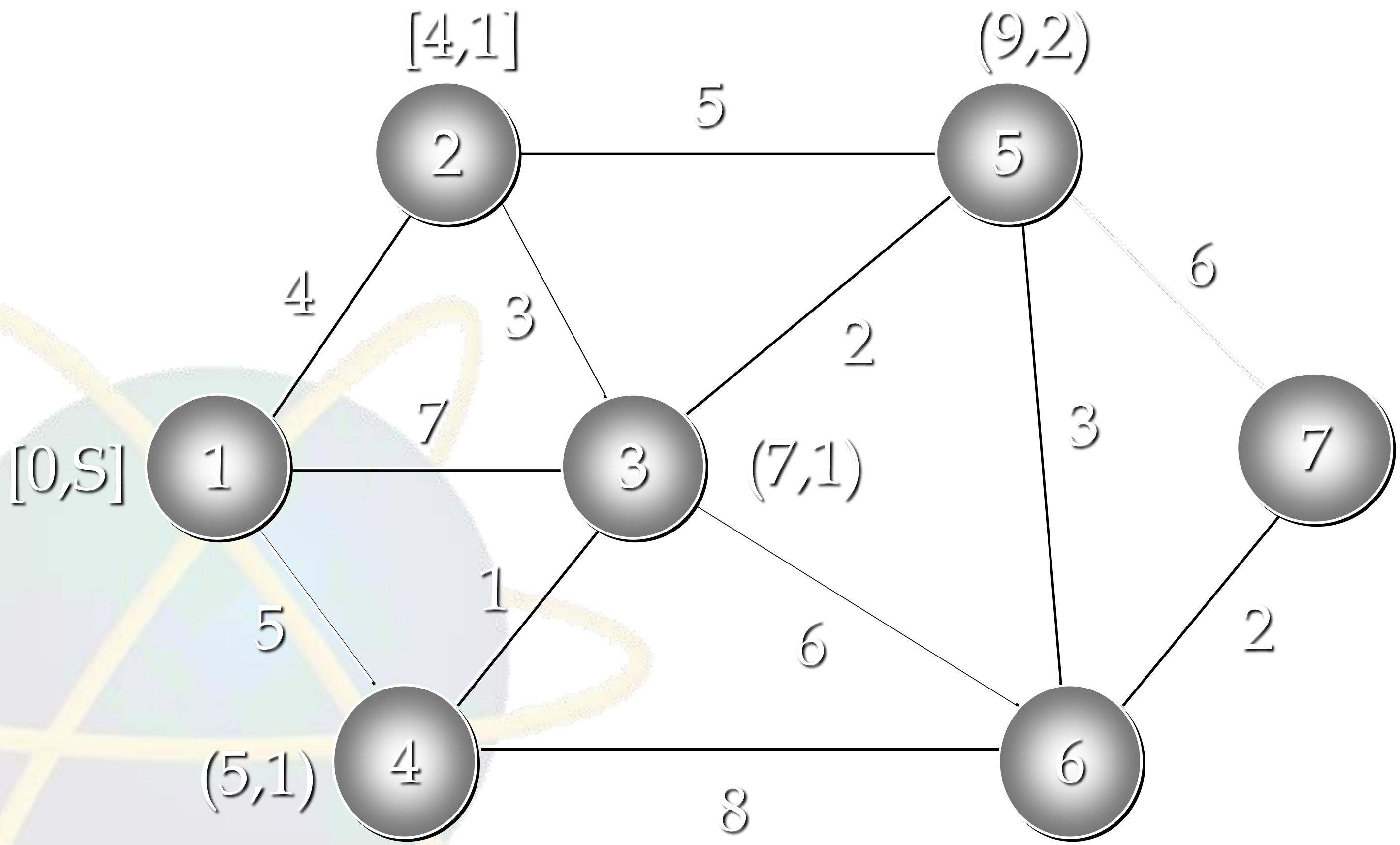
- Iteration 1
  - Step 4: For each node with a tentative label which is connected to Node 2 by just one arc, compute the sum of its arc length plus the distance value of Node 2 (which is 4).

    Node 3: 3 + 4 = 7 (not smaller than current label; do not change.)

    Node 5: 5 + 4 = 9 (assign tentative label to Node 5 of (9,2) since node 5 had no label.)

Example: Shortest Path

- Iteration 1, Step 4 Results

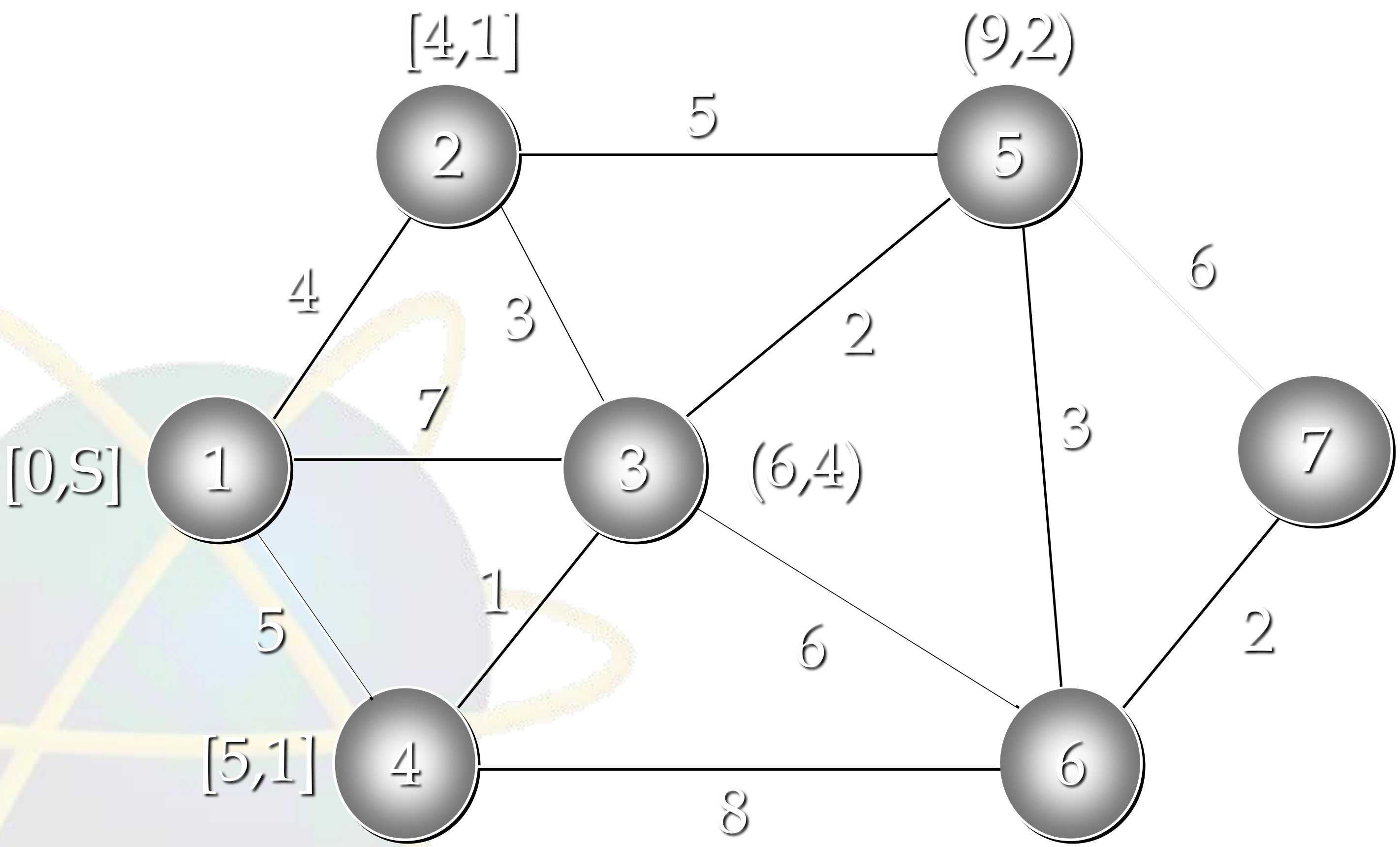


Example: Shortest Path

- Iteration 2
  - Step 3: Node 4 has the smallest tentative label distance (5). It now becomes the new permanently labeled node.

Example: Shortest Path

- Iteration 2, Step 3 Results



Example: Shortest Path

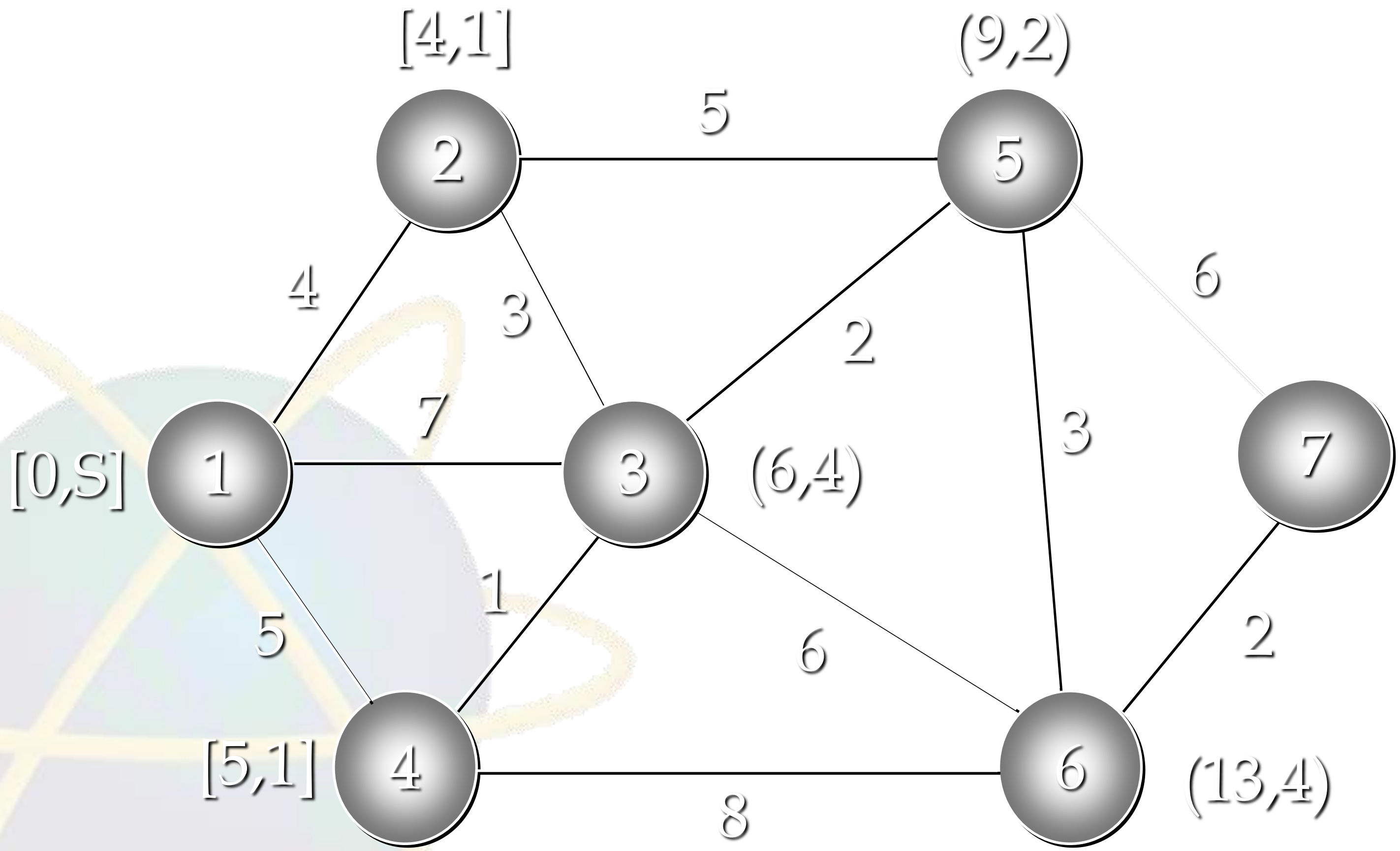
- Iteration 2
  - Step 4: For each node with a tentative label which is connected to node 4 by just one arc, compute the sum of its arc length plus the distance value of node 4 (which is 5).

    Node 3: 1 + 5 = 6 (replace the tentative label of node 3 by (6,4) since 6 < 7, the current distance.)

    Node 6: 8 + 5 = 13 (assign tentative label to node 6 of (13,4) since node 6 had no label.)

Example: Shortest Path

- Iteration 2, Step 4 Results

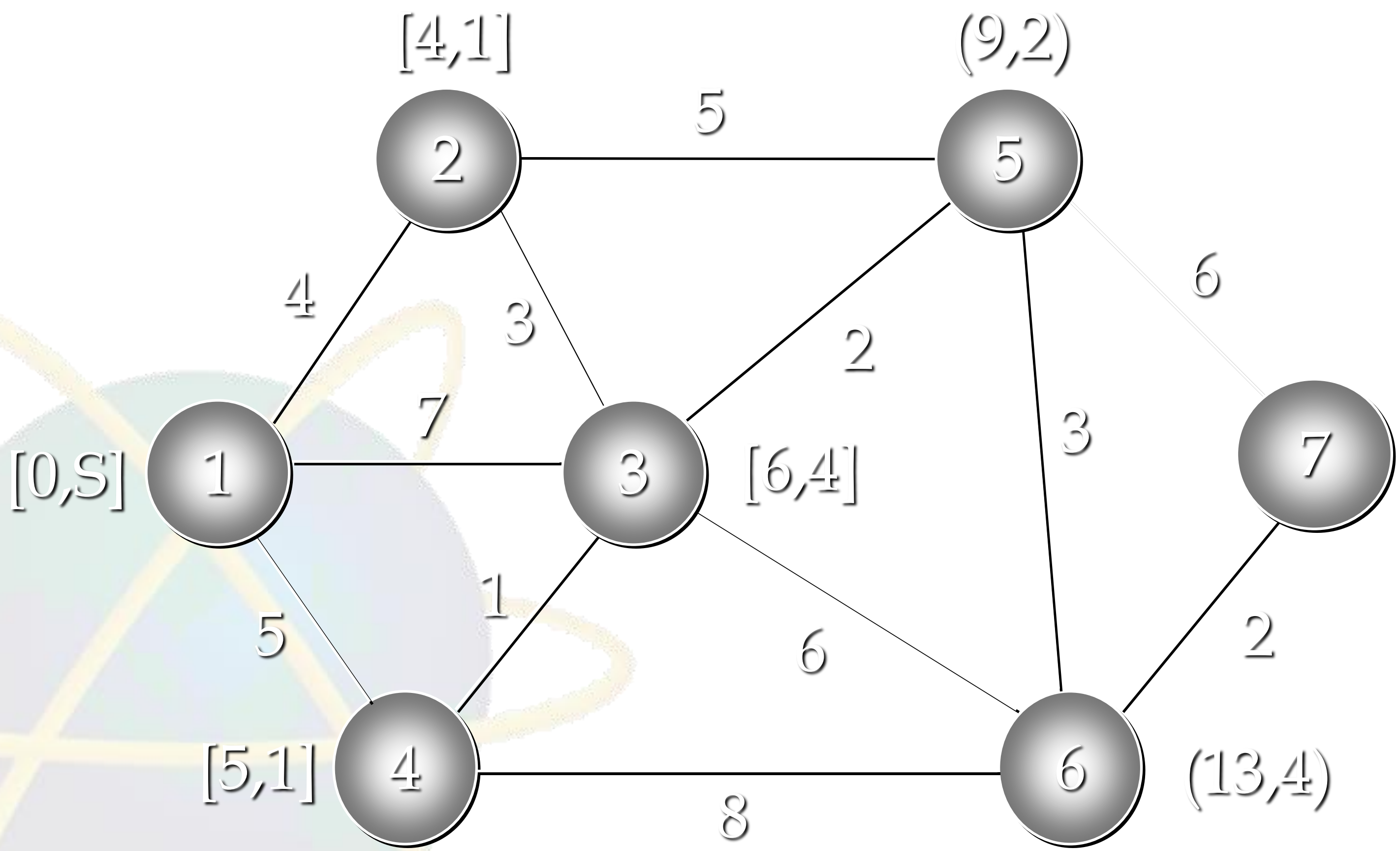


Example: Shortest Path

- Iteration 3
  - Step 3: Node 3 has the smallest tentative distance label (6). It now becomes the new permanently labeled node.

Example: Shortest Path

- Iteration 3, Step 3 Results



Example: Shortest Path

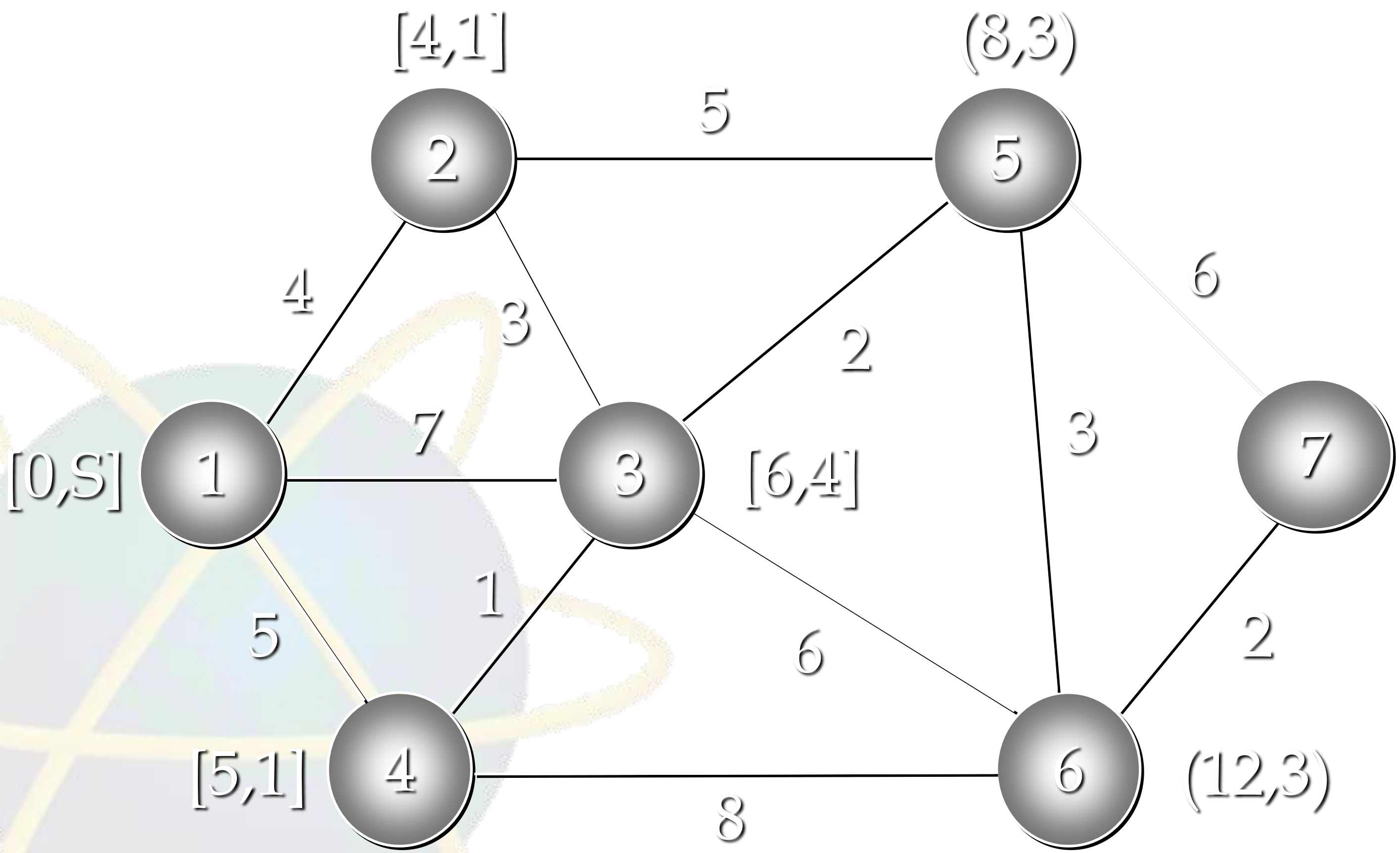
- Iteration 3
  - Step 4: For each node with a tentative label which is connected to node 3 by just one arc, compute the sum of its arc length plus the distance to node 3 (which is 6).

    Node 5: 2 + 6 = 8 (replace the tentative label of node 5 with (8,3) since 8 < 9, the current distance)

    Node 6: 6 + 6 = 12 (replace the tentative label of node 6 with (12,3) since 12 < 13, the current distance)

Example: Shortest Path

- Iteration 3, Step 4 Results

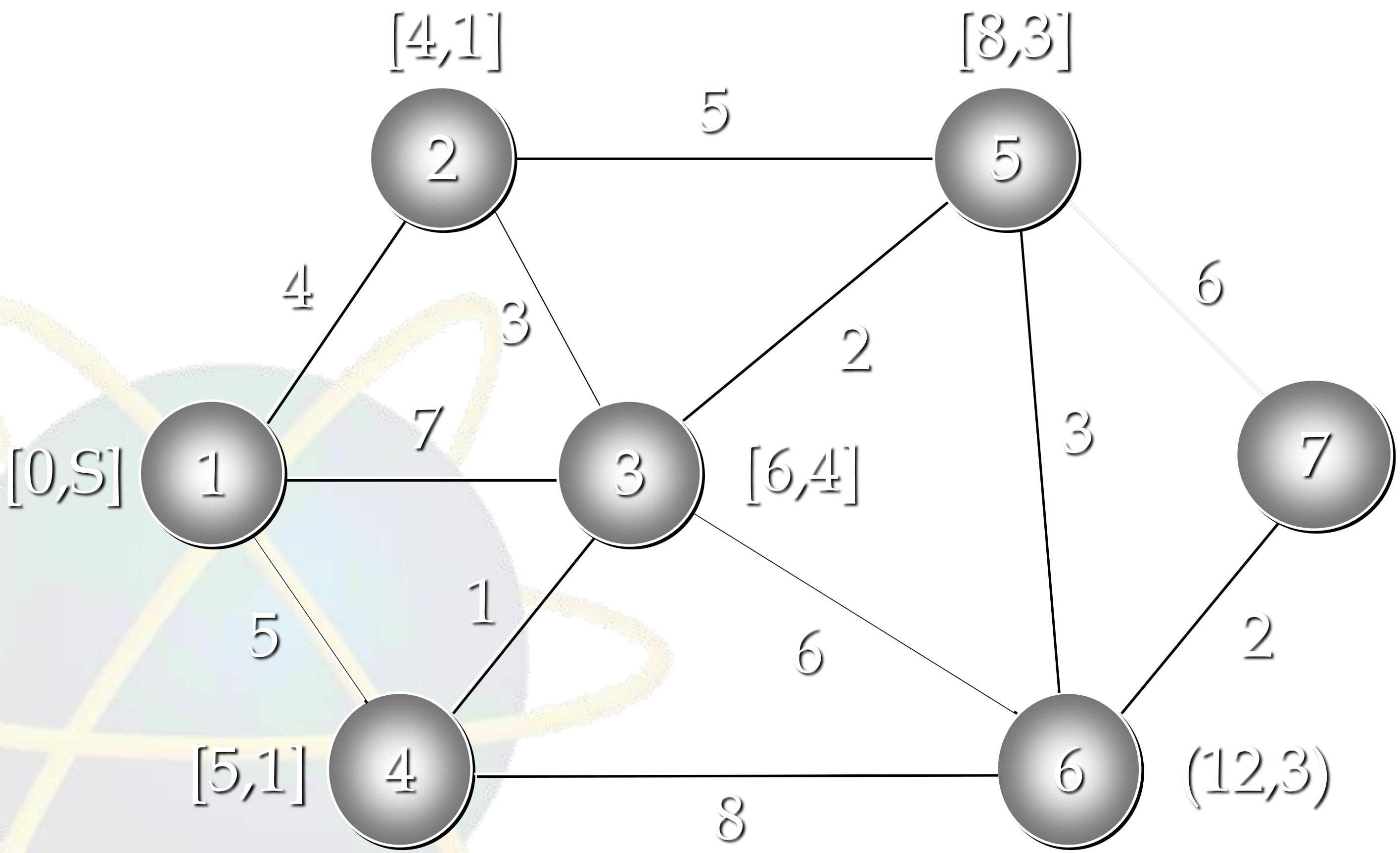


Example: Shortest Path

- Iteration 4
  - Step 3: Node 5 has the smallest tentative label distance (8). It now becomes the new permanently labeled node.

Example: Shortest Path

- Iteration 4, Step 3 Results



Example: Shortest Path

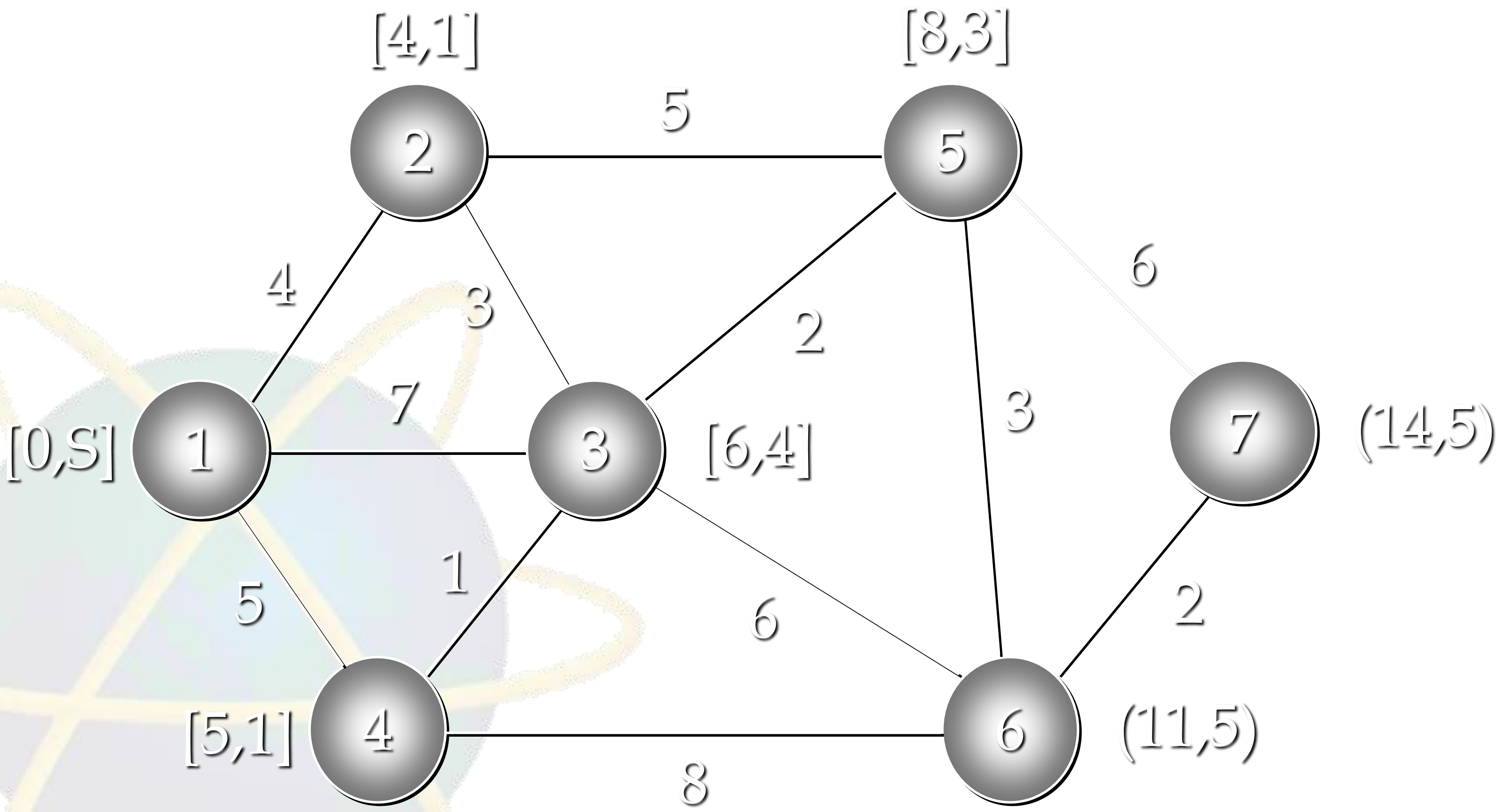
- Iteration 4
  - Step 4: For each node with a tentative label which is connected to node 5 by just one arc, compute the sum of its arc length plus the distance value of node 5 (which is 8).

    Node 6: 3 + 8 = 11 (Replace the tentative label with (11,5) since 11 < 12, the current distance.)

    Node 7: 6 + 8 = 14 (Assign

Example: Shortest Path

- Iteration 4, Step 4 Results

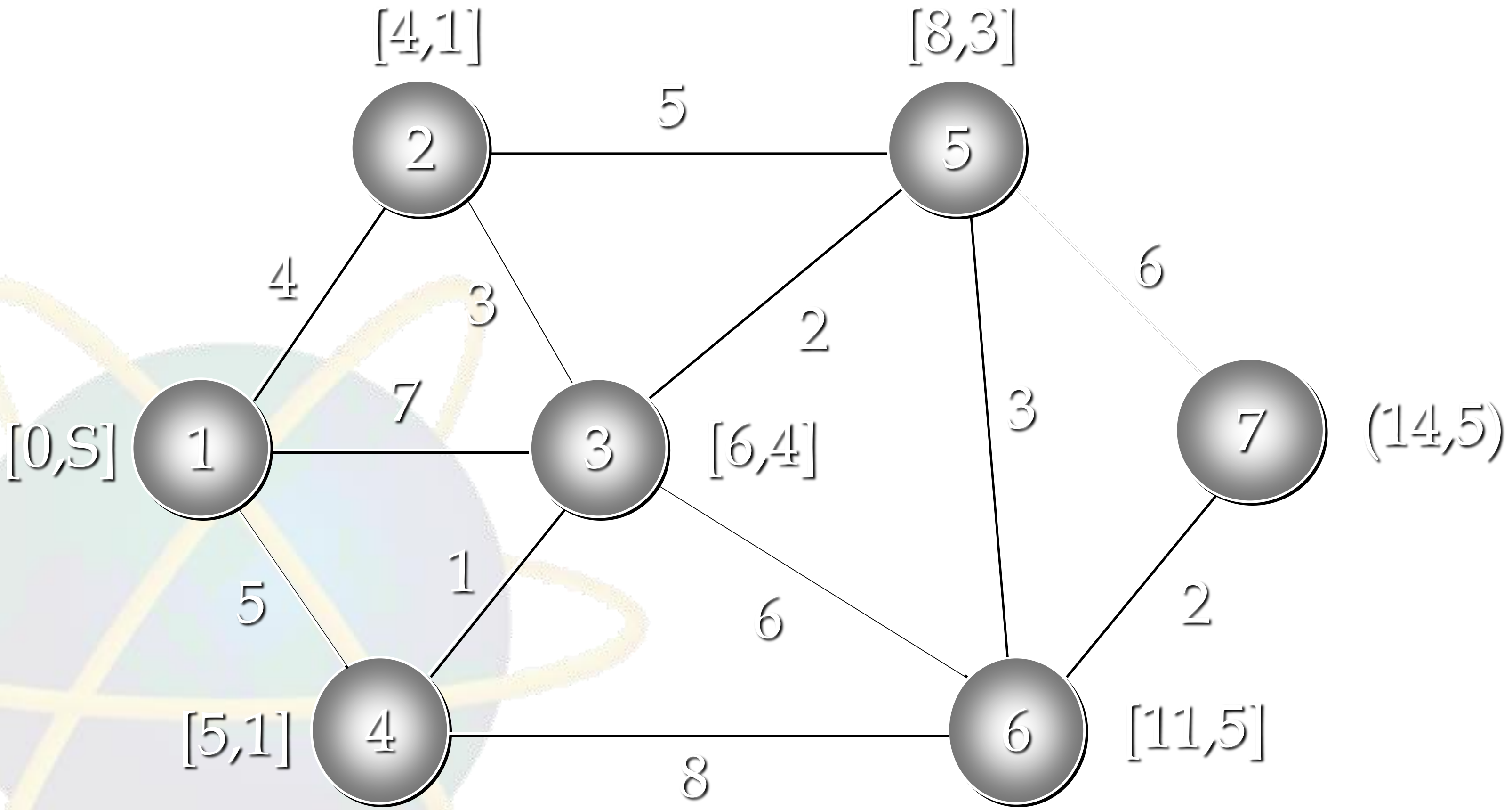


Example: Shortest Path

- Iteration 5
  - Step 3: Node 6 has the smallest tentative label distance (11). It now becomes the new permanently labeled node.

Example: Shortest Path

- Iteration 5, Step 3 Results



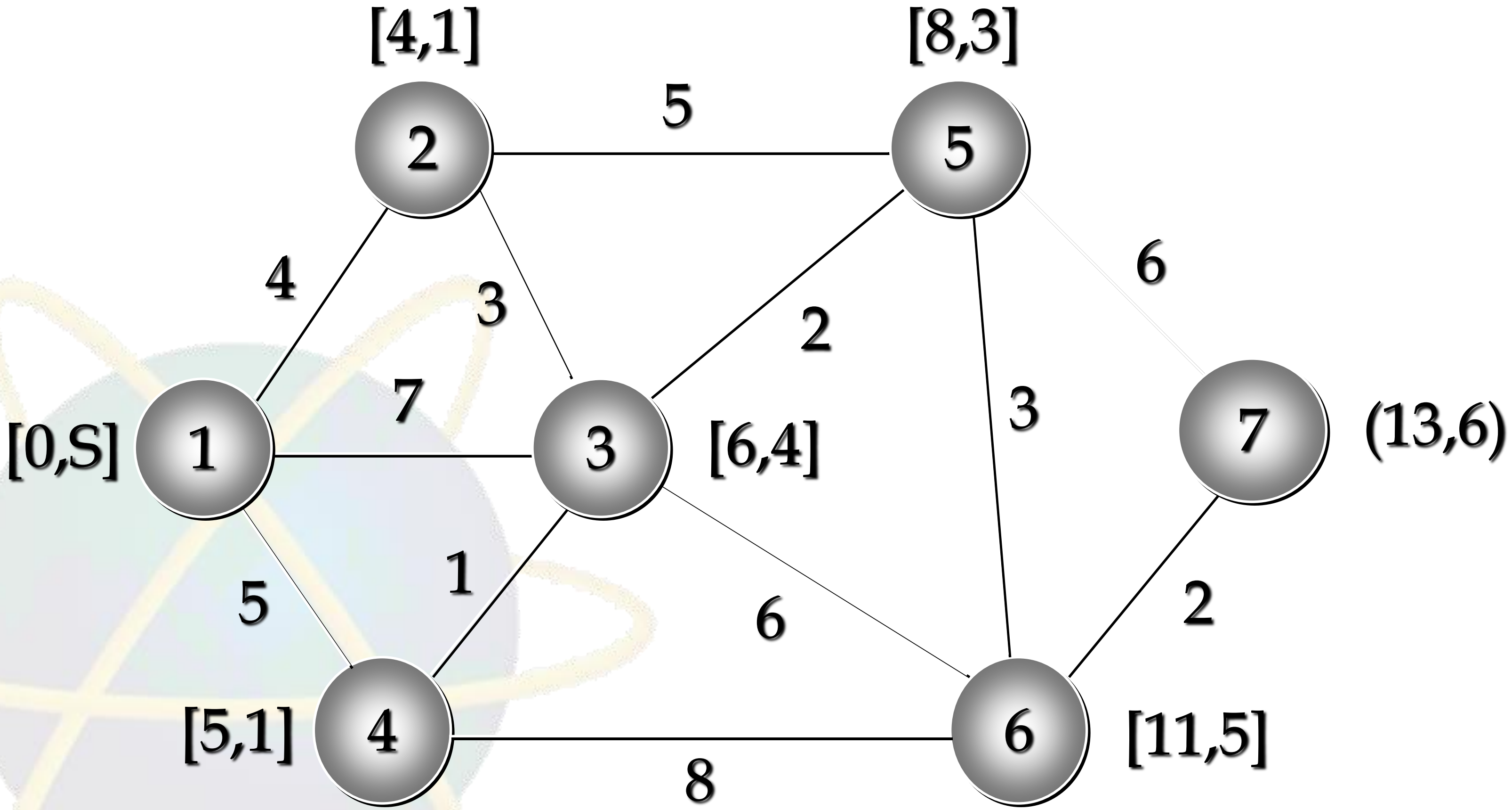
Example: Shortest Path

- Iteration 5
  - Step 4: For each node with a tentative label which is connected to Node 6 by just one arc, compute the sum of its arc length plus the distance value of Node 6 (which is 11).

    Node 7: 2 + 11 = 13 (replace the tentative label with (13,6) since 13 < 14, the current distance.)

Example: Shortest Path

- Iteration 5, Step 4 Results



Example: Shortest Path

- Iteration 6
  - Step 3: Node 7 becomes permanently labeled, and hence all nodes are now permanently labeled. Thus proceed to summarize in Step 5.
  - Step 5: Summarize by tracing the shortest routes backwards through the permanent labels.

Example: Shortest Path

- Solution Summary

Node	Minimum Distance	Shortest Route
2	4	1-2
3	6	1-4-3
4	5	1-4
5	8	1-4-3-5
6	11	1-4-3-5-6
7	13	1-4-3-5-6-7

Minimal Spanning Tree Problem

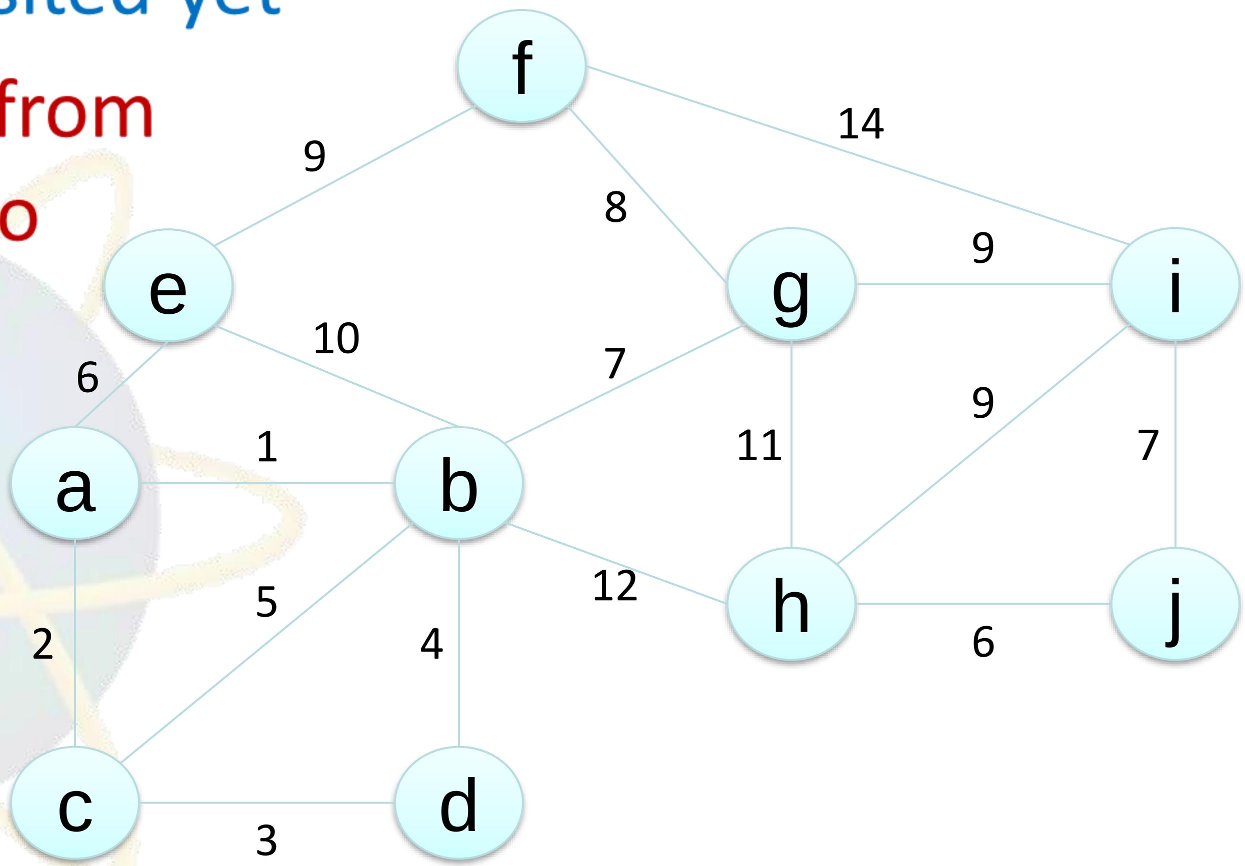
- A tree is a set of connected arcs that does not form a cycle.
- A spanning tree is a tree that connects all nodes of a network.
- The minimal spanning tree problem seeks to determine the minimum sum of arc lengths necessary to connect all nodes in a network.
- The criterion to be minimized in the minimal spanning tree problem is not limited to distance even though the term "closest" is used in describing the procedure. Other criteria include time and cost. (Neither time nor cost are necessarily linearly related to distance.)

Building a minimum spanning tree – Prim's algorithm

- Prim's algorithm takes a graph G = (V, E) and builds an MCST ***T***
- PrimMCST(V, E)
  - Pick an arbitrary node **r** from V
  - Add **r** to *T*
  - While *T* contains < |V| nodes
    - Find a **minimum weight edge (u, v)** where $\mathbf{u} \in \boldsymbol{T}$ and $\mathbf{v} \notin \boldsymbol{T}$
    - Add node v to *T*

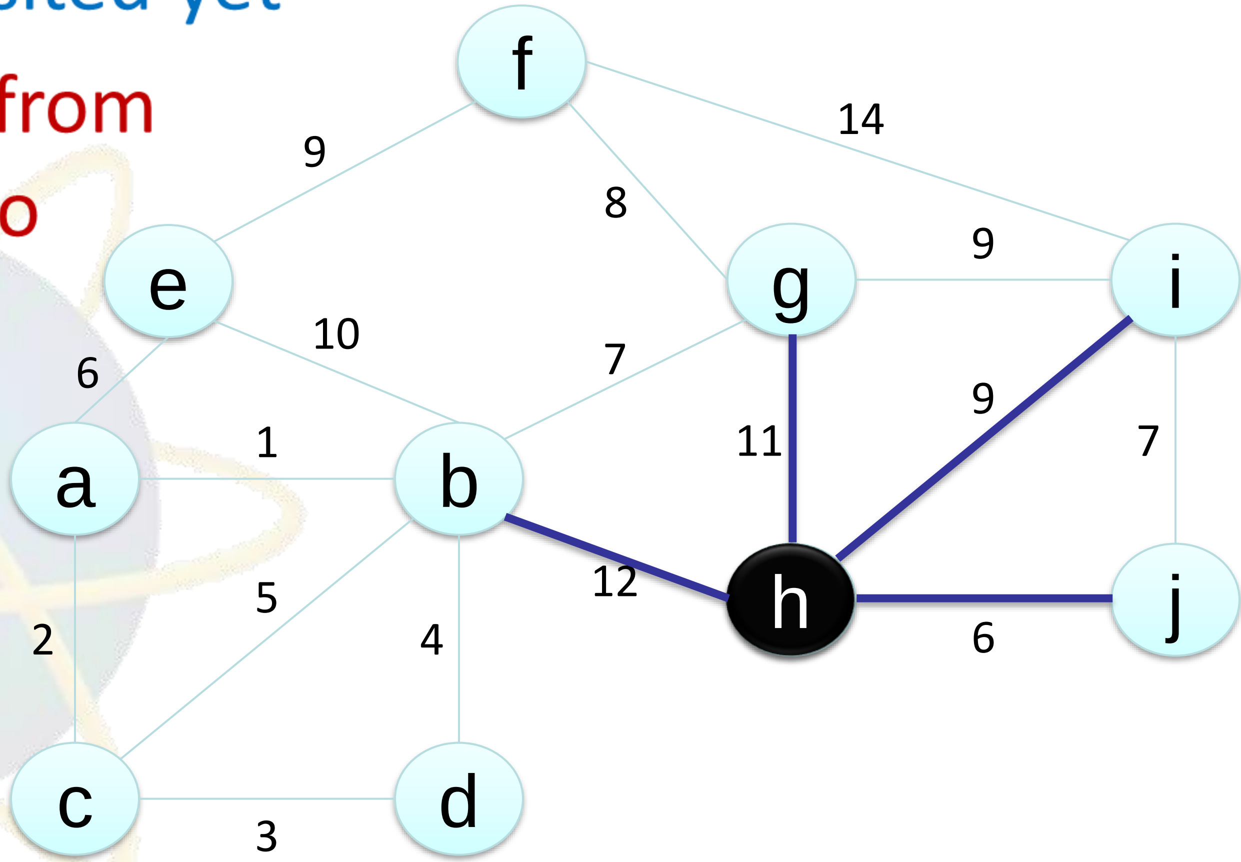
Example

- Start at an arbitrary node, say, h.
- **Blue:** not visited yet
- **Red:** edges from nodes $\in T$ to nodes $\notin T$
- **Black:** in T



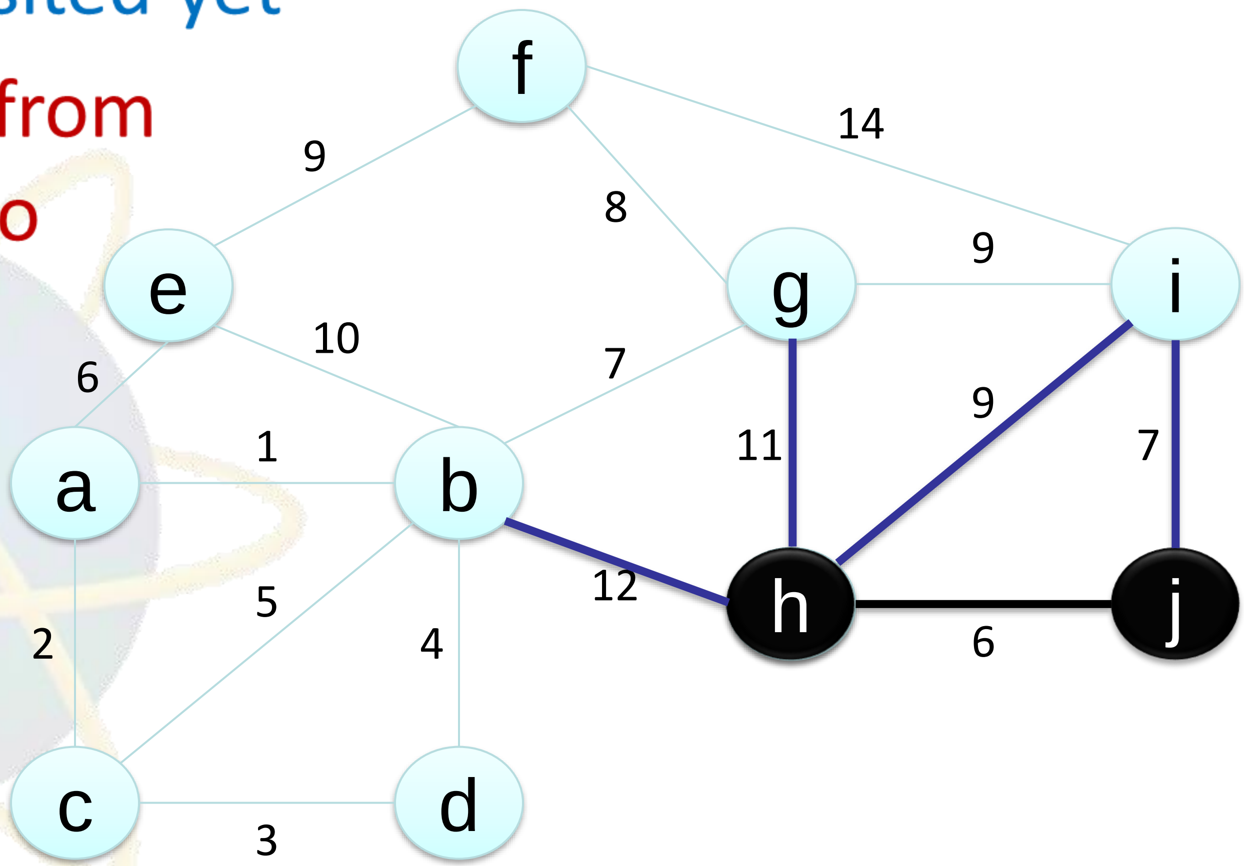
Prim's algorithm

- Start at an arbitrary node, say, h.
- **Blue:** not visited yet
- **Red:** edges from nodes $\in T$ to nodes $\notin T$
- **Black:** in T



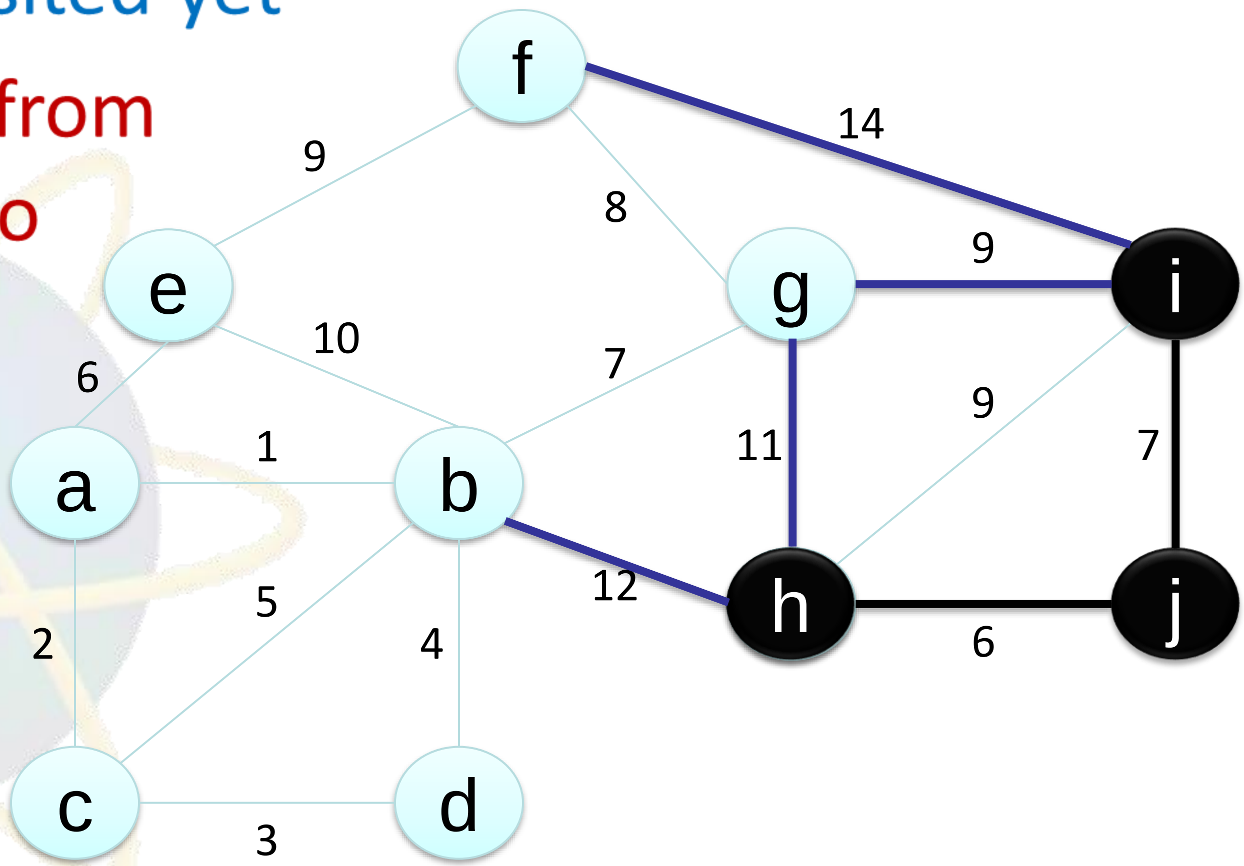
Prim's algorithm

- Start at an arbitrary node, say, h.
- **Blue:** not visited yet
- **Red:** edges from nodes $\in T$ to nodes $\notin T$
- **Black:** in T



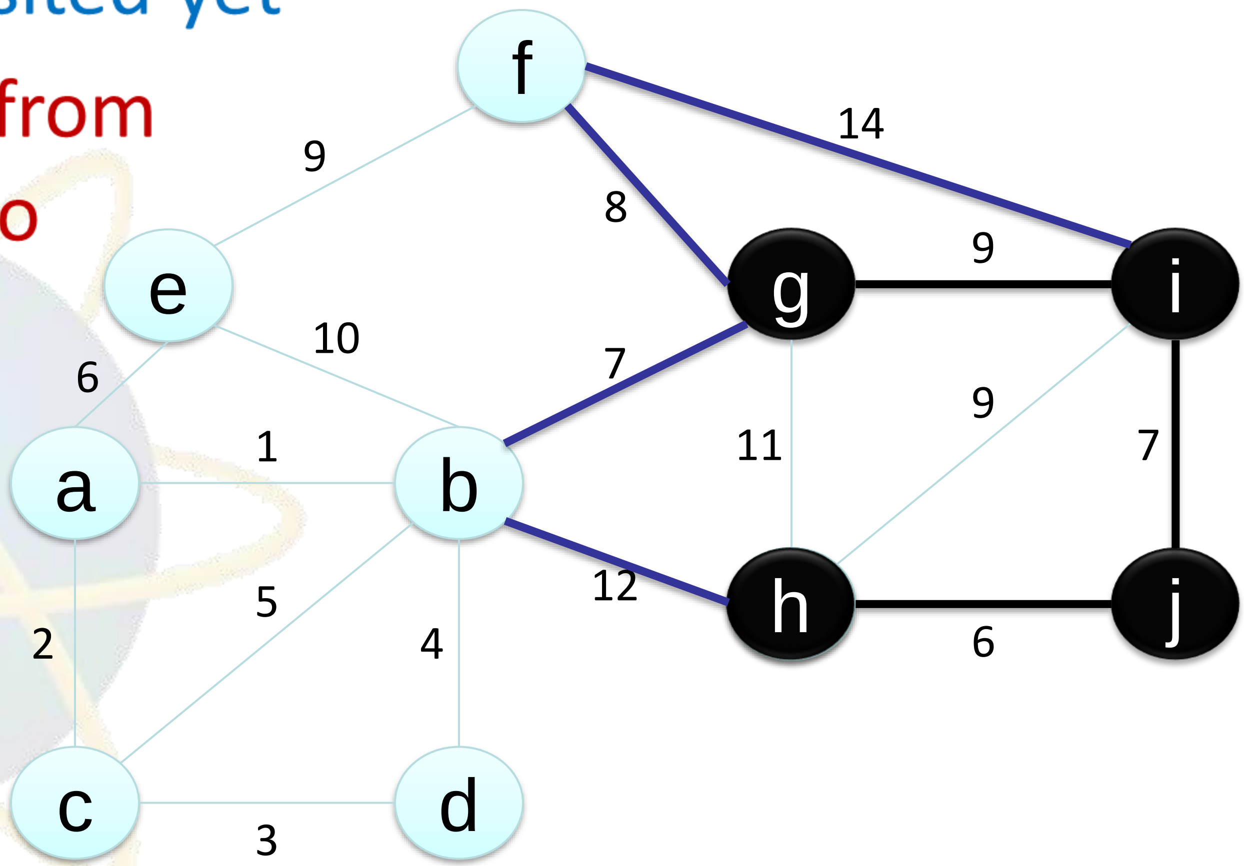
Prim's algorithm

- Start at an arbitrary node, say, h.
- **Blue:** not visited yet
- **Red:** edges from nodes $\in T$ to nodes $\notin T$
- **Black:** in T



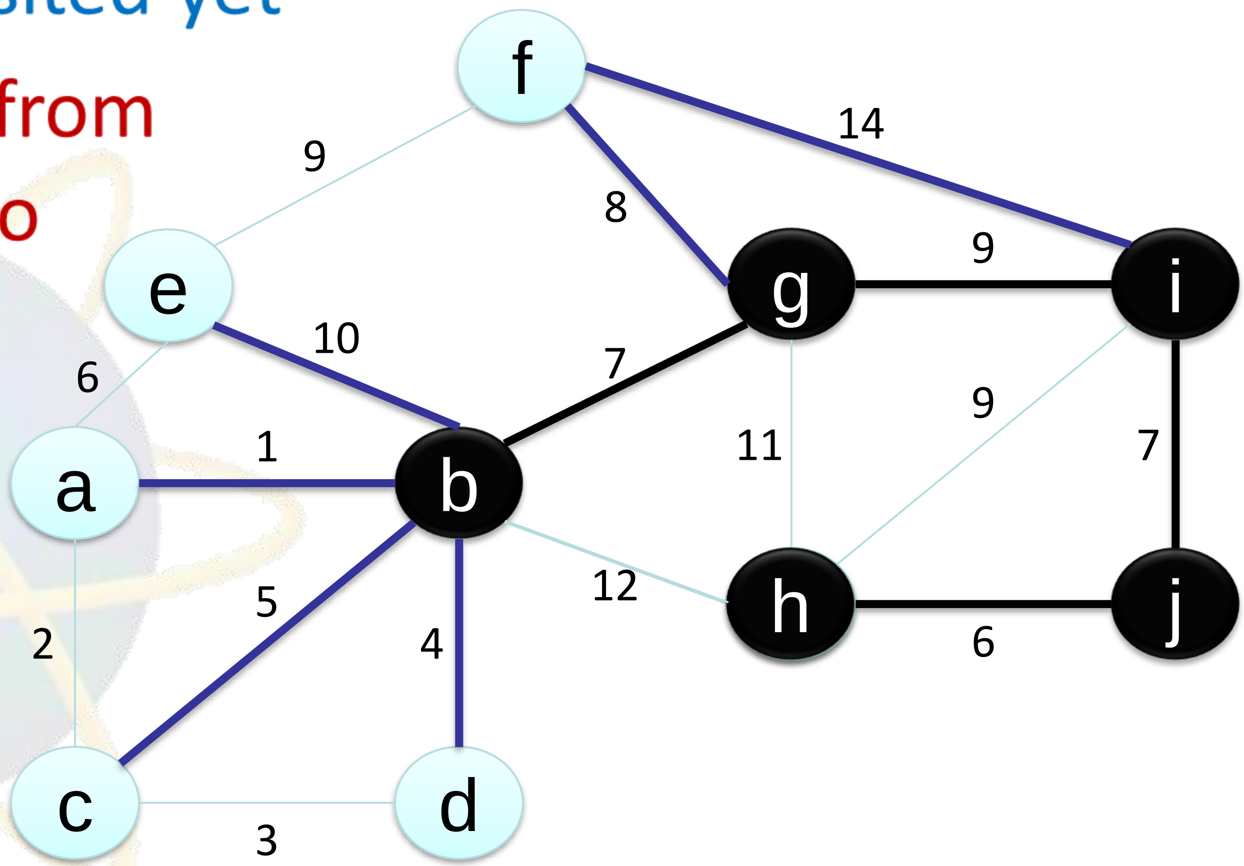
Prim's algorithm

- Start at an arbitrary node, say, h.
- **Blue:** not visited yet
- **Red:** edges from nodes $\in T$ to nodes $\notin T$
- **Black:** in T



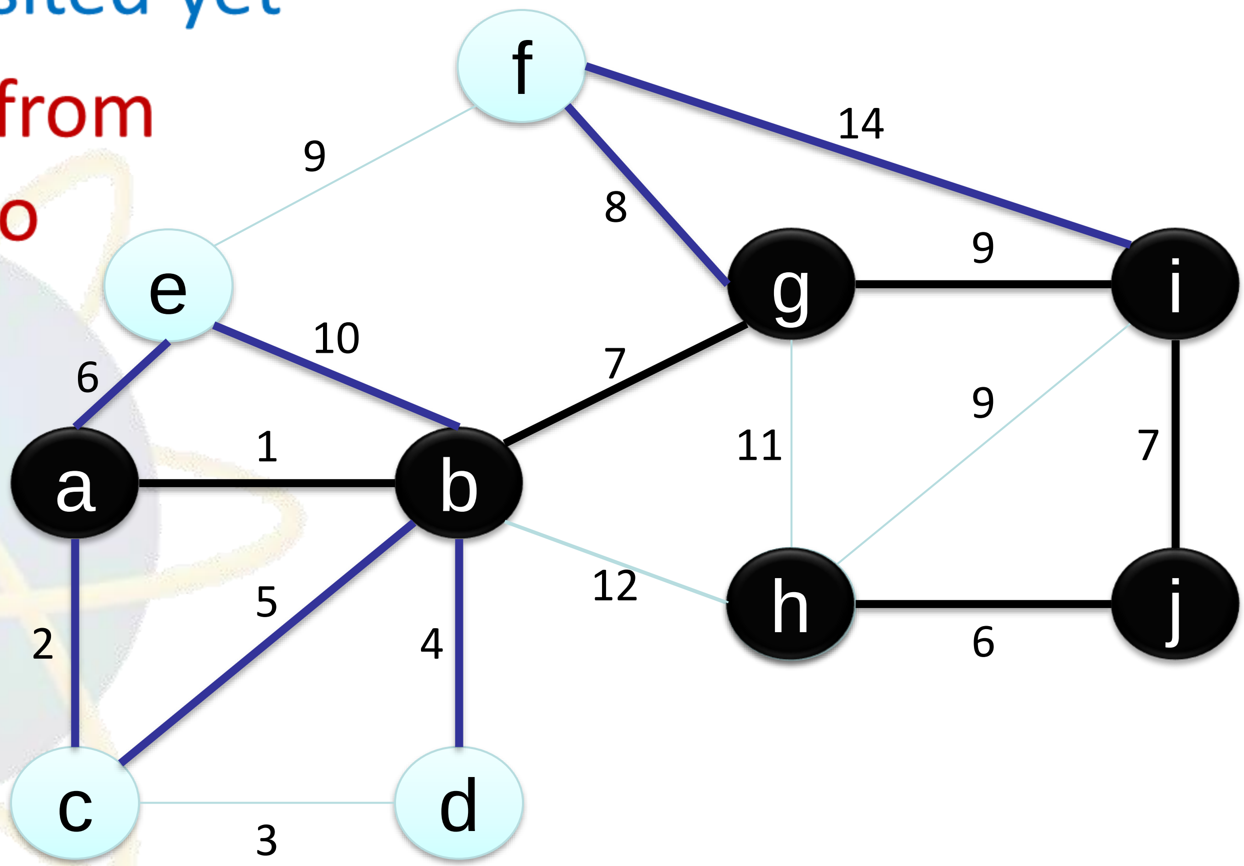
Prim's algorithm

- Start at an arbitrary node, say, h.
- **Blue:** not visited yet
- **Red:** edges from nodes $\in T$ to nodes $\notin T$
- **Black:** in T



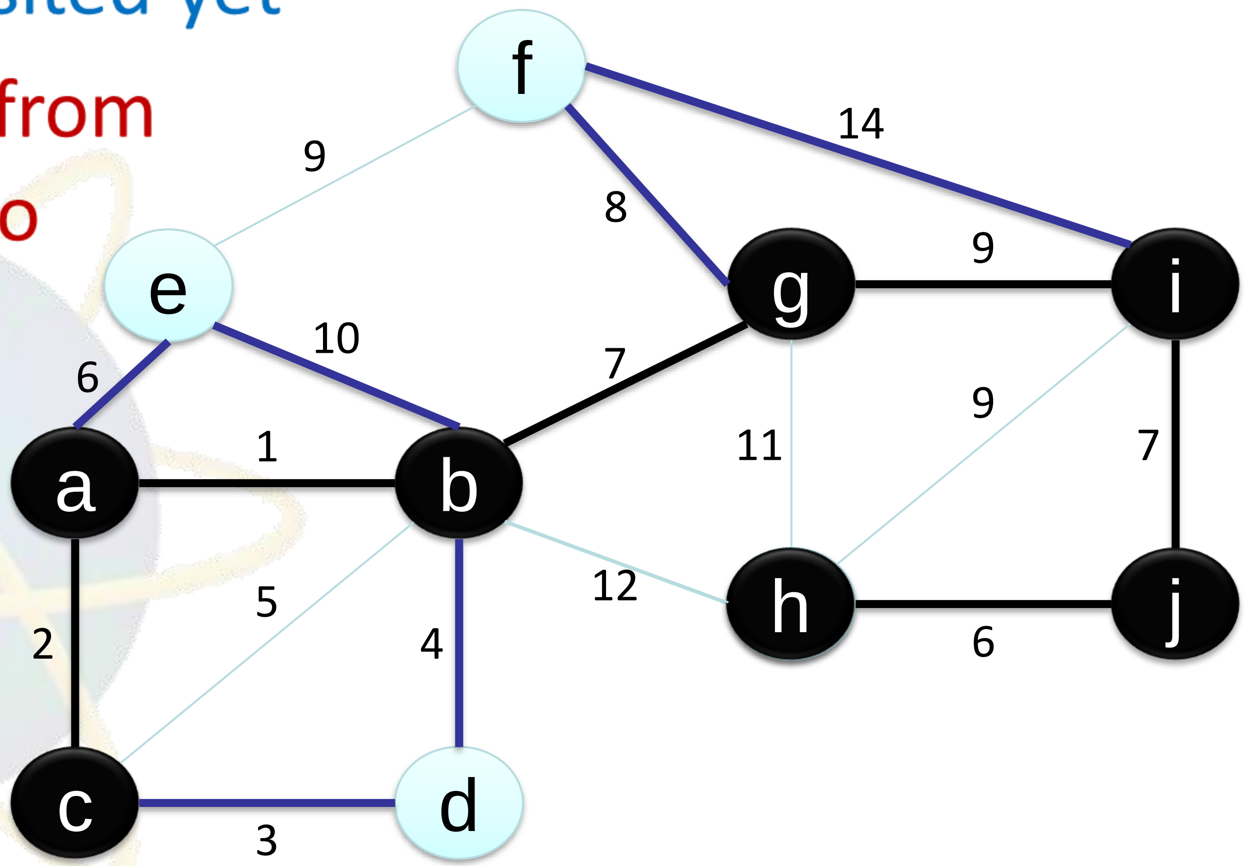
Prim’s algorithm

- Start at an arbitrary node, say, h.
- **Blue:** not visited yet
- **Red:** edges from nodes $\in T$ to nodes $\notin T$
- **Black:** in T



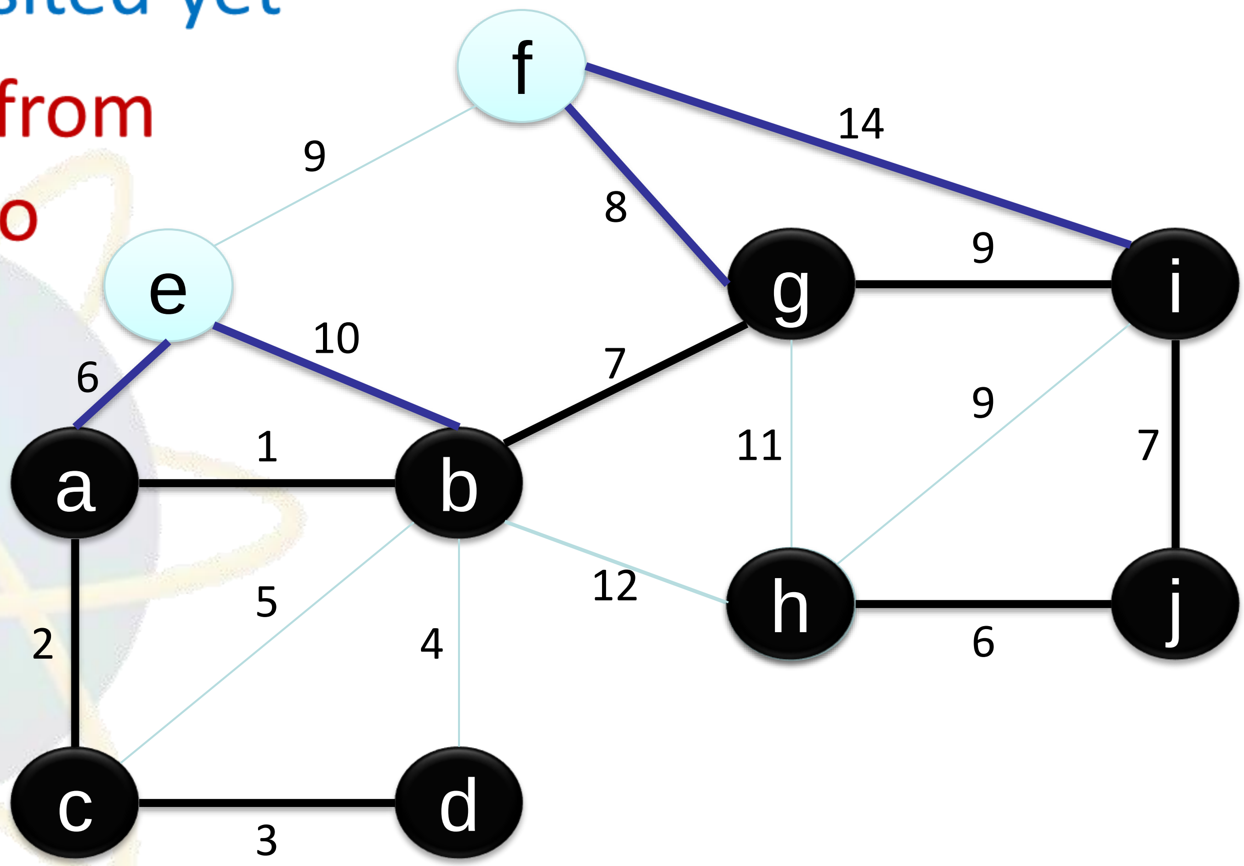
Prim's algorithm

- Start at an arbitrary node, say, h.
- **Blue:** not visited yet
- **Red:** edges from nodes $\in T$ to nodes $\notin T$
- **Black:** in T



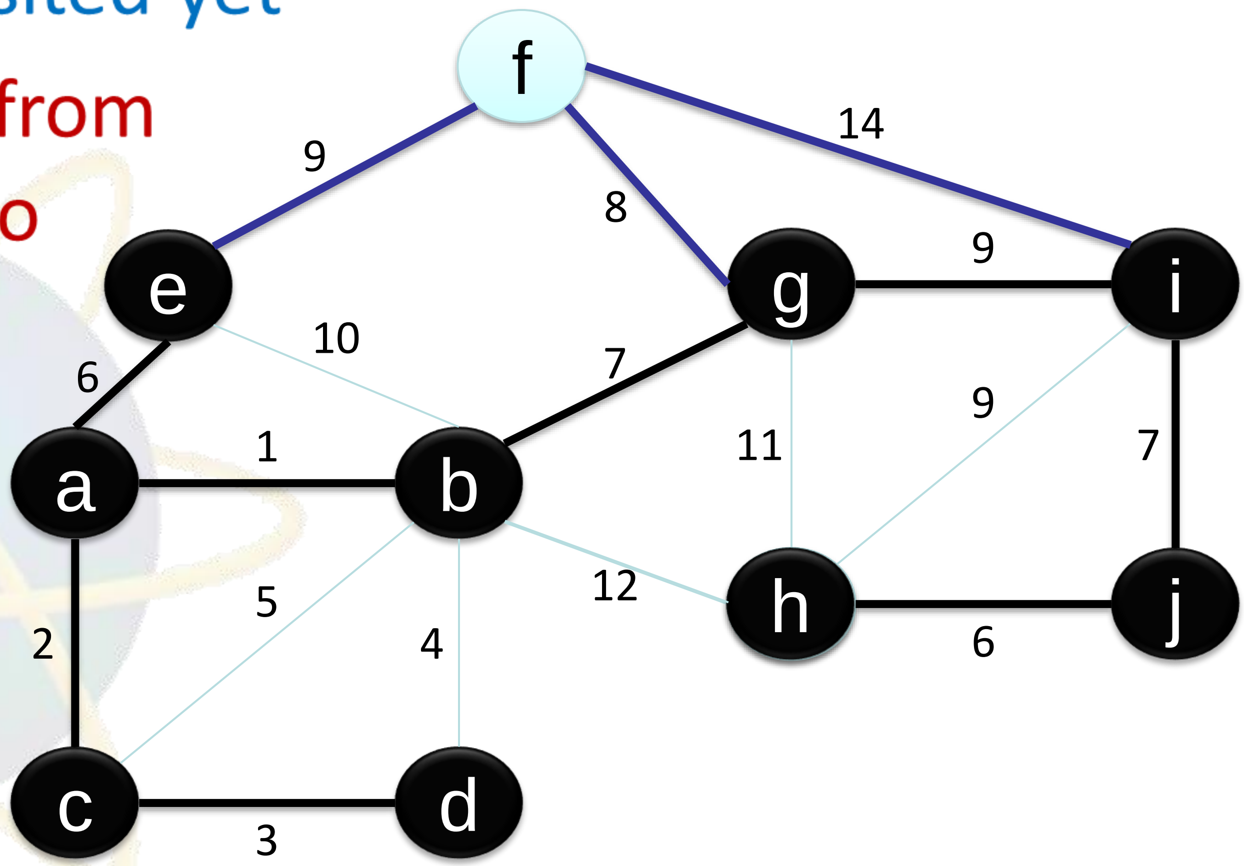
Prim's algorithm

- Start at an arbitrary node, say, h.
- **Blue:** not visited yet
- **Red:** edges from nodes $\in T$ to nodes $\notin T$
- **Black:** in T



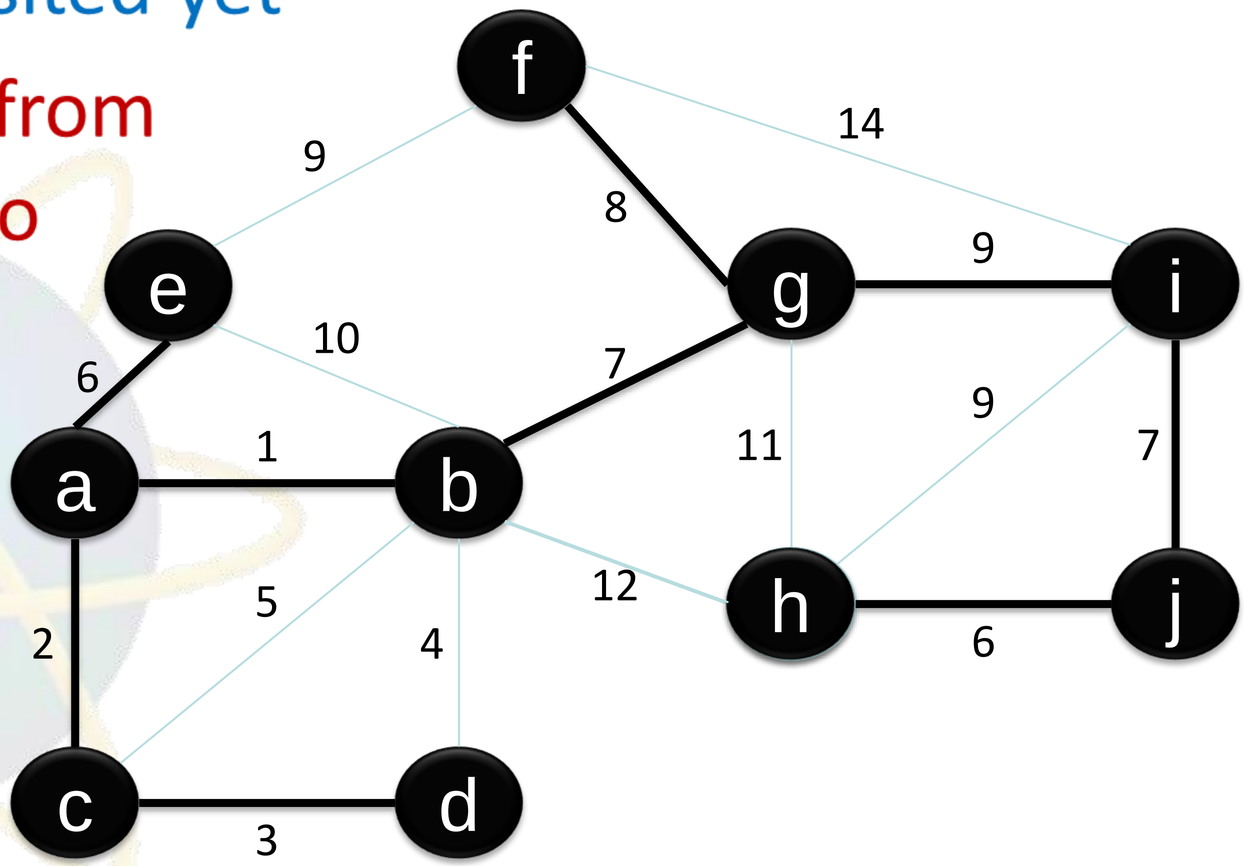
Prim's algorithm

- Start at an arbitrary node, say, h.
- **Blue:** not visited yet
- **Red:** edges from nodes $\in T$ to nodes $\notin T$
- **Black:** in T



Prim’s algorithm

- Start at an arbitrary node, say, h.
- **Blue:** not visited yet
- **Red:** edges from nodes $\in T$ to nodes $\notin T$
- **Black:** in T
- Minimum Cost: 47



Minimal Spanning Tree Algorithm – Kruskal's algorithm

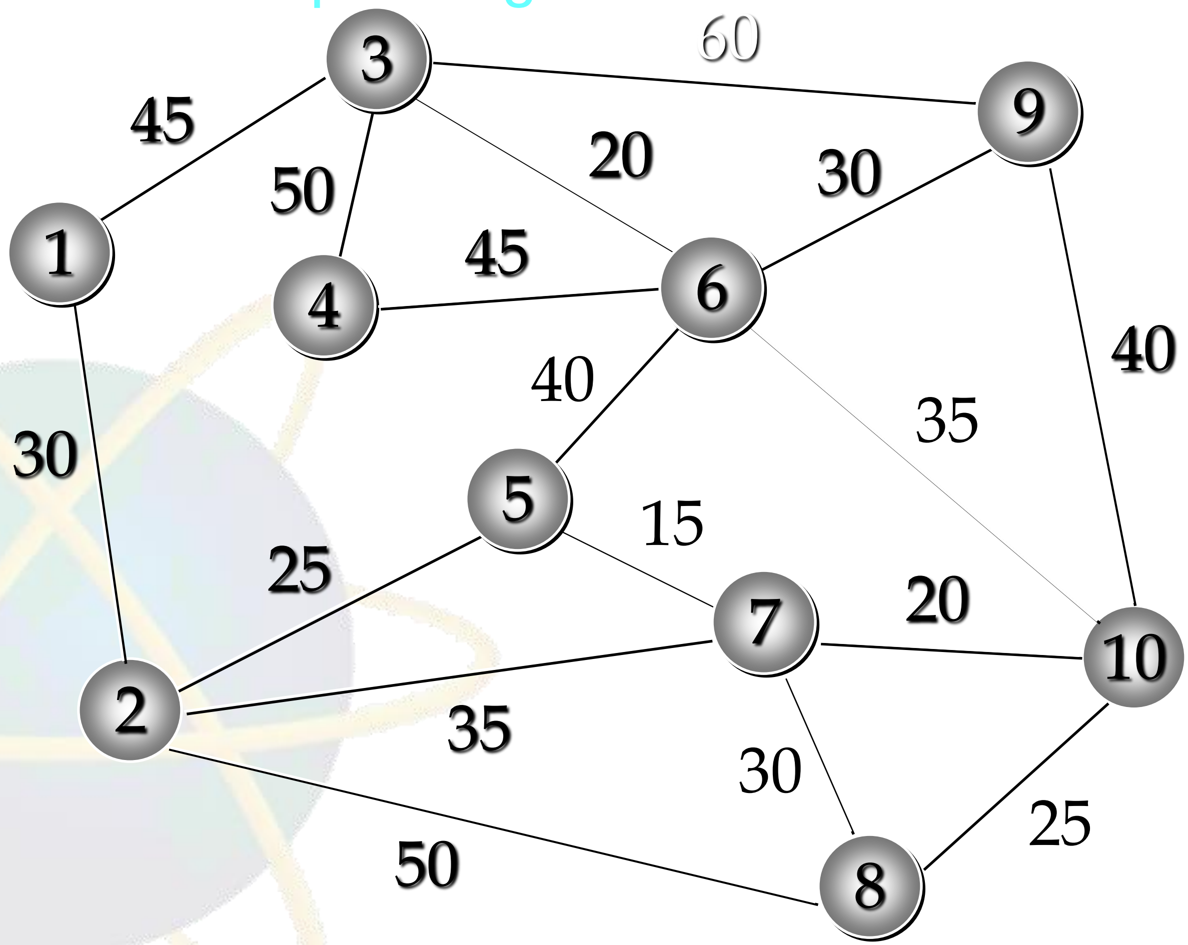
- Step 1: Arbitrarily begin at any node and connect it to the closest node. The two nodes are referred to as connected nodes, and the remaining nodes are referred to as unconnected nodes.

- Step 2: Identify the unconnected node that is closest to one of the connected nodes (break ties arbitrarily). Add this new node to the set of connected nodes. Repeat this step until all nodes have been connected.

- Note: A problem with *n* nodes to be connected will require *n* - 1 iterations of the above steps.

Example: Minimal Spanning Tree

- Find the Minimal Spanning Tree:



Example: Minimal Spanning Tree

- Iteration 1: Arbitrarily selecting node 1, we see that its closest node is node 2 (distance = 30). Therefore, initially we have:

  Connected nodes: 1,2

  Unconnected nodes: 3,4,5,6,7,8,9,10

  Chosen arcs: 1-2

- Iteration 2: The closest unconnected node to a connected node is node 5 (distance = 25 to node 2). Node 5 becomes a connected node.

  Connected nodes: 1,2,5

  Unconnected nodes: 3,4,6,7,8,9,10

  Chosen arcs: 1-2, 2-5

Example: Minimal Spanning Tree

- Iteration 3: The closest unconnected node to a connected node is node 7 (distance = 15 to node 5). Node 7 becomes a connected node.

  Connected nodes: 1,2,5,7

  Unconnected nodes: 3,4,6,8,9,10

  Chosen arcs: 1-2, 2-5, 5-7

- Iteration 4: The closest unconnected node to a connected node is node 10 (distance = 20 to node 7). Node 10 becomes a connected node.

  Connected nodes: 1,2,5,7,10

  Unconnected nodes: 3,4,6,8,9

  Chosen arcs: 1-2, 2-5, 5-7, 7-10

Example: Minimal Spanning Tree

- Iteration 5: The closest unconnected node to a connected node is node 8 (distance = 25 to node 10). Node 8 becomes a connected node.
  Connected nodes: 1,2,5,7,10,8
  Unconnected nodes: 3,4,6,9
  Chosen arcs: 1-2, 2-5, 5-7, 7-10, 10-8

- Iteration 6: The closest unconnected node to a connected node is node 6 (distance = 35 to node 10). Node 6 becomes a connected node.
  Connected nodes: 1,2,5,7,10,8,6
  Unconnected nodes: 3,4,9
  Chosen arcs: 1-2, 2-5, 5-7, 7-10, 10-8, 10-6

Example: Minimal Spanning Tree

- Iteration 7: The closest unconnected node to a connected node is node 3 (distance = 20 to node 6). Node 3 becomes a connected node.

  Connected nodes: 1,2,5,7,10,8,6,3

  Unconnected nodes: 4,9

  Chosen arcs: 1-2, 2-5, 5-7, 7-10, 10-8, 10-6, 6-3

- Iteration 8: The closest unconnected node to a connected node is node 9 (distance = 30 to node 6). Node 9 becomes a connected node.

  Connected nodes: 1,2,5,7,10,8,6,3,9

  Unconnected nodes: 4

  Chosen arcs: 1-2, 2-5, 5-7, 7-10, 10-8, 10-6, 6-3, 6-9

Example: Minimal Spanning Tree

- Iteration 9: The only remaining unconnected node is node 4. It is closest to connected node 6 (distance = 45).

  Thus, the minimal spanning tree (displayed on the next slide) consists of:

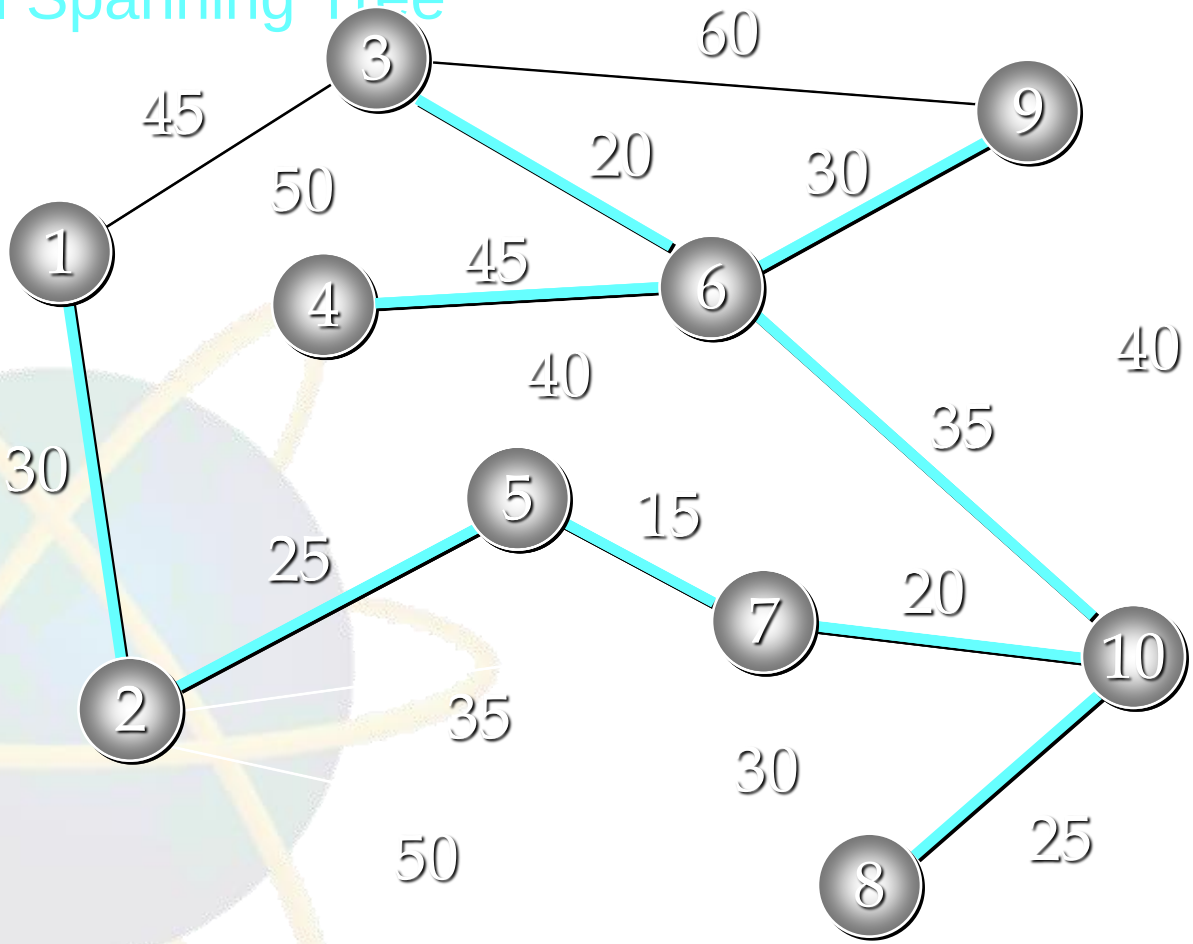
  Arcs: 1-2, 2-5, 5-7, 7-10, 10-8, 10-6, 6-3, 6-9, 6-4

  Values: 30 + 25 + 15 + 20 + 25 + 35 + 20 + 30 + 45

  = 245

Example: Minimal Spanning Tree

- Optimal Spanning Tree

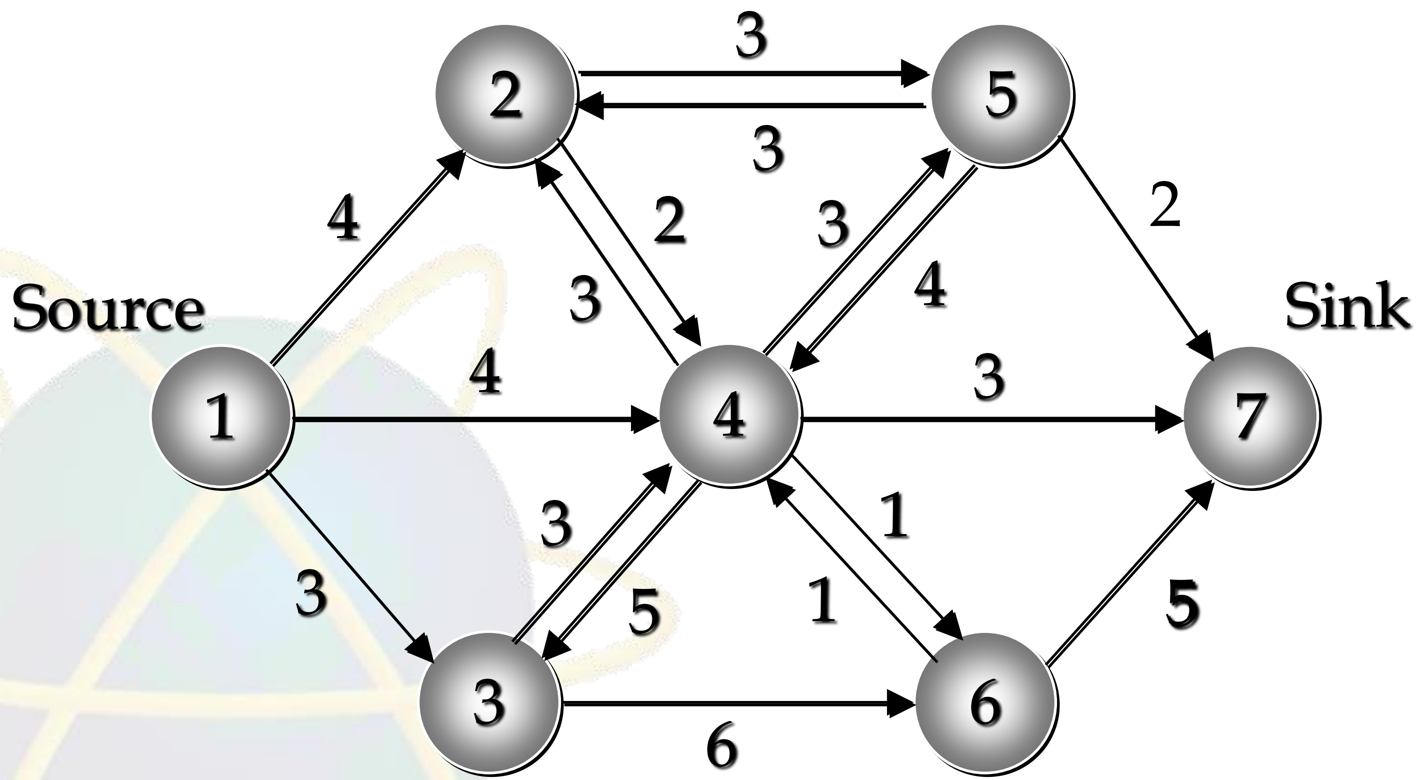


Maximal Flow Problem

- The maximal flow problem is concerned with determining the maximal volume of flow (vehicles, messages, fluid etc) from one node (called the source) to another node (called the sink).
- In the maximal flow problem, each arc has a maximum arc flow capacity which limits the flow through the arc. When we do not specify capacities for the nodes, we do assume that the flow out of a node is equal to the flow into the node.

Example: Maximal Flow

- Network Model

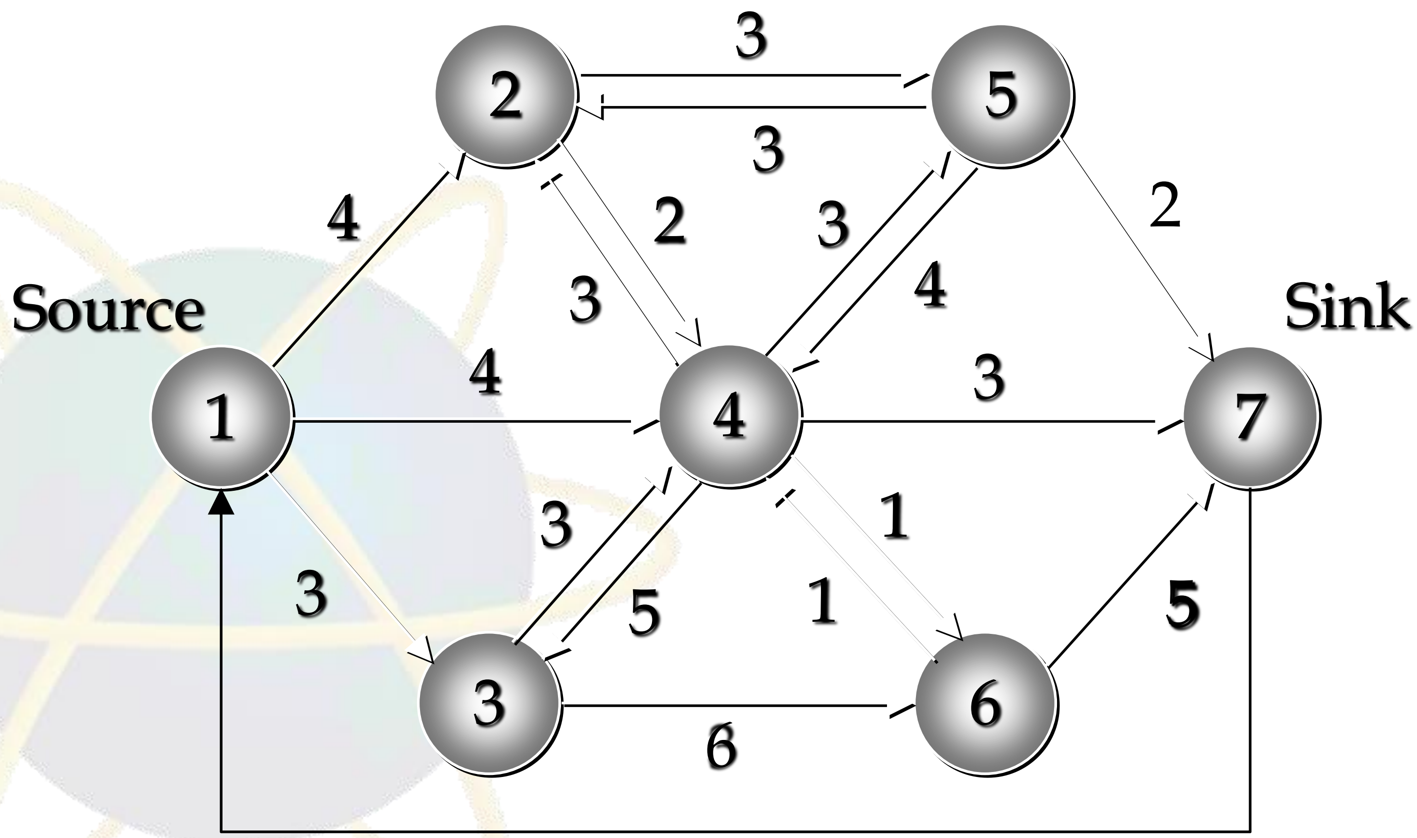


Example: Maximal Flow

- A transshipment model can be developed for the maximal flow problem.
- We will add an arc from node 7 back to node 1 to represent the total flow through the network.
- There is no capacity on the newly added 7-1 arc.
- We want to maximize the flow over the 7-1 arc.

Example: Maximal Flow

- Modified Network Model



Maximal Flow Problem

- LP Formulation
  - There is a variable for every arc.
  - There is a constraint for every node; the flow out must equal the flow in.
  - There is a constraint for every arc (except the added sink-to-source arc); arc capacity cannot be exceeded.
  - The objective is to maximize the flow over the added, sink-to-source arc.

Maximal Flow Problem

- LP Formulation

Max x_{k1} (k is sink node, 1 is source node)

s.t. $\sum_i x_{ij} - \sum_j x_{ji} = 0$ (conservation of flow)

$x_{ij} \leq c_{ij}$ (c_{ij} is capacity of ij arc)

$x_{ij} \geq 0$, for all i and j (non-negativity)

(x_{ij} represents the flow from node i to node j)

Example: Maximal Flow

- LP Formulation
  - 18 variables (for 17 original arcs and 1 added arc)
  - 24 constraints
    - 7 node flow-conservation constraints
    - 17 arc capacity constraints (for original arcs)

Example: Maximal Flow

- LP Formulation
  - Objective Function

    Max x_{71}

  - Node Flow-Conservation Constraints

    $x_{71} - x_{12} - x_{13} - x_{14} = 0$ (flow in & out of node 1)

    $x_{12} + x_{42} + x_{52} - x_{24} - x_{25} = 0$ (node 2)

    $x_{13} + x_{43} - x_{34} - x_{36} = 0$ (etc.)

    $x_{14} + x_{24} + x_{34} + x_{54} + x_{64} - x_{42} - x_{43} - x_{45} - x_{46} - x_{47} = 0$

    $x_{25} + x_{45} - x_{52} - x_{54} - x_{57} = 0$

    $x_{36} + x_{46} - x_{64} - x_{67} = 0$

    $x_{47} + x_{57} + x_{67} - x_{71} = 0$

Example: Maximal Flow

- LP Formulation (continued)
  - Arc Capacity Constraints

$x_{12} \leq 4 \quad x_{13} \leq 3 \quad x_{14} \leq 4$

$x_{24} \leq 2 \quad x_{25} \leq 3$

$x_{34} \leq 3 \quad x_{36} \leq 6$

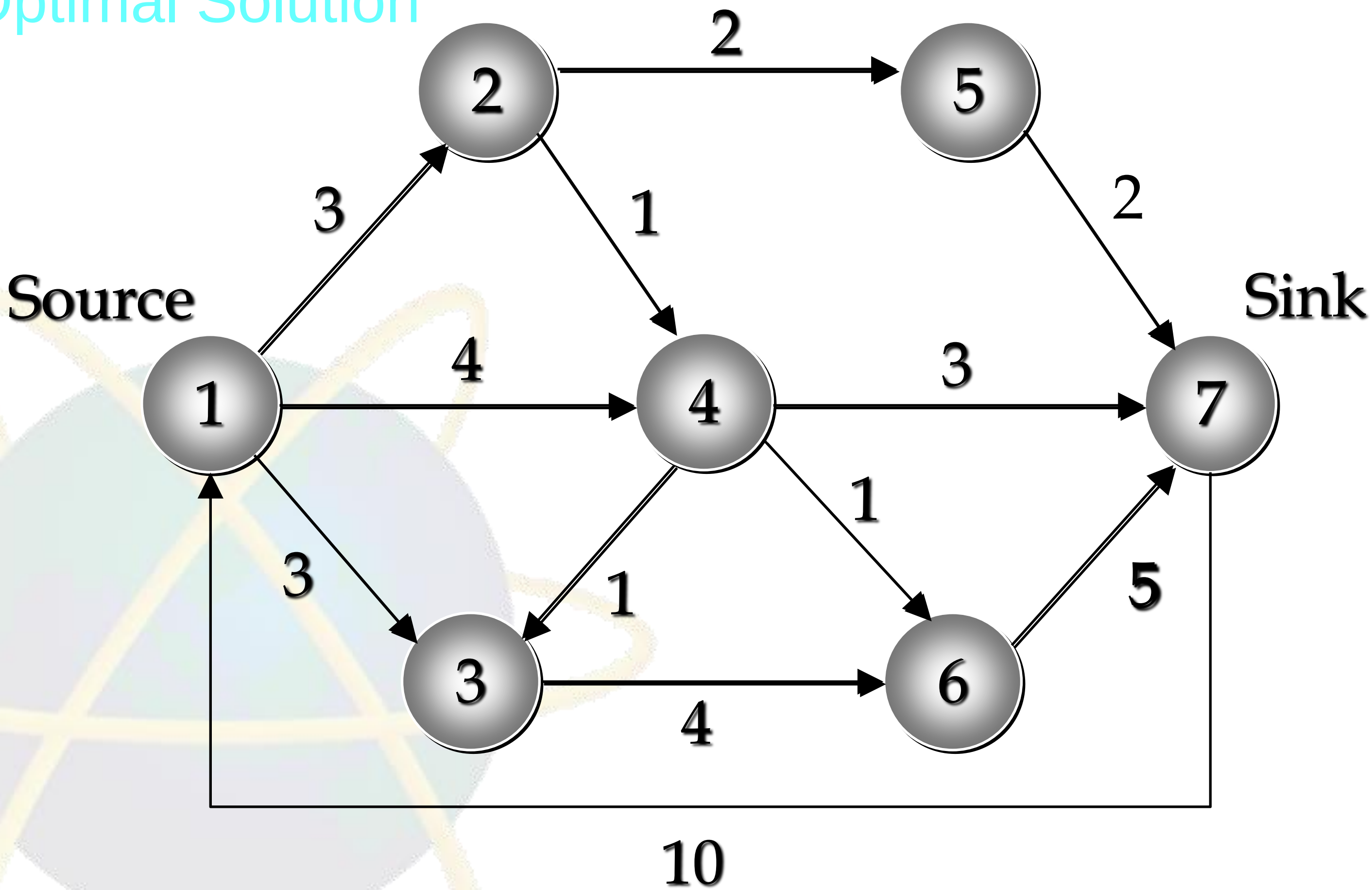
$x_{42} \leq 3 \quad x_{43} \leq 5 \quad x_{45} \leq 3 \quad x_{46} \leq 1 \quad x_{47} \leq 3$

$x_{52} \leq 5 \quad x_{54} \leq 5 \quad x_{57} \leq 5$

$x_{64} \leq 5 \quad x_{67} \leq 5$

Example: Maximal Flow

- Optimal Solution



Quick Review Question

Follow Up Assignment

Summary of Main Teaching Points

Question and Answer Session

Q & A

Next Lesson

Decision Analysis