Operations Research

Chapter 6

Linear Programming Networks

Linear Programming: Networks

- Network Analysis
- Network Terminology
- Network Problems
- Examples of Network Problems
- Minimum Spanning Tree
- Simple Greedy Procedure
- Kruskal’s Algorithm
- Directed and Undirected Arcs, Networks
- Shortest Route Problem
- Dijkstra's Algorithm

Network Analysis

- Network Analysis.- OR technique(s) for the solution of problems based on the visual and conceptual representation of the interrelationships between the components of a system.
- A **network (graph)** consists of a set of points and a set of lines connecting certain points. The points are called **nodes** (vertices). The lines are called **arcs** (edges, branches).
- The arcs may have certain flow. If the flow can go in only one direction in the arc, the arc is said to be **directed arc**, if not the arc is **undirected** (or links).
  - Examples of Network Applications

Nodes	Arcs	Flow
Intersections	Roads	Vehicles
Airports	Air Routes	Aircraft
Switching Points	Wires, Channels	Messages
Pumping Stations	Pipes	Fluid
Work Centers	Material Handling Systems	Jobs

Network Terminology

- A network that has only directed arcs is said to be a **directed network**. If all the arcs are undirected, the network is **undirected**.
- A **path** between two nodes is a sequence of arcs connecting them. A **directed path** from node i to node j allows flow from node i to j. An **undirected path** from i to j allows flow in either direction.

Network Terminology

- A path that begins and ends at the same node is called a **cycle**. A cycle can be directed or undirected if the path is directed or undirected.
- Two nodes are **connected** if the network contains at least one undirected path between them
- A **connected network** is a network where every pair of nodes is connected.
- A **tree** is a connected network with no cycles. A spanning tree is a tree connecting all the nodes of a network.
- **Arc capacity** is the maximum amount of flow that can be carried on a directed arc.
- In a **supply node** the flow out exceeds the flow in. The opposite occurs in a **demand node**. In a transshipment node, flow in = flow out.

Network Problems

- Some problems typically solved by using network methods:
  1. Minimum Spanning Tree
  2. Shortest Route
  3. Maximum Flow
  4. Critical Path

Examples of Network Problems

- **Minimum Spanning Tree-** The objective is to connect all of the nodes of a network with links that minimize the total "cost" ("cost" could be time, distance, etc.)
  - Common Examples:
    - Design of computer networks
    - Design of a network of roads
- **Shortest Route-** The objective is to minimize the "cost" to go from a Node A (the origin) to a Node B (the destination)
  - Common Examples:
    - Selection of highways to go from City A to City B such that the total traveled distance is minimized.
    - Deciding what branches of a pipeline to build to connect City A to City B such that the total cost is minimized.

Examples of Network Problems

- **Maximum Flow-** The objective is to maximize the flow that can be sent from one node (the source) to another (the sink)
  - Common Examples:
    - Traffic management
    - Logistics systems
- **Critical Path-** The objective is to minimize the time it takes to complete the total project
  - Common Example:
    - Project Management (CPM & PERT Networks)

Minimum Spanning Tree

- A minimum spanning tree is the set of the arcs of a network that have the shortest total length while providing a route (path) between each pair of nodes.
- More formally:
  - Let $G = (V,E)$ represent a network, where V is the node set and E is the arc set and $w(e)$ is the cost (or weight) of constructing edge $e \in E$.
  - **Problem**: find the set of edges that form a spanning tree T for G where the total weight

$$w(T) = \sum_{e \in T} w(e)$$

    is as small as possible.
  - For a spanning tree to be a minimum spanning tree, it has to satisfy the path optimality conditions

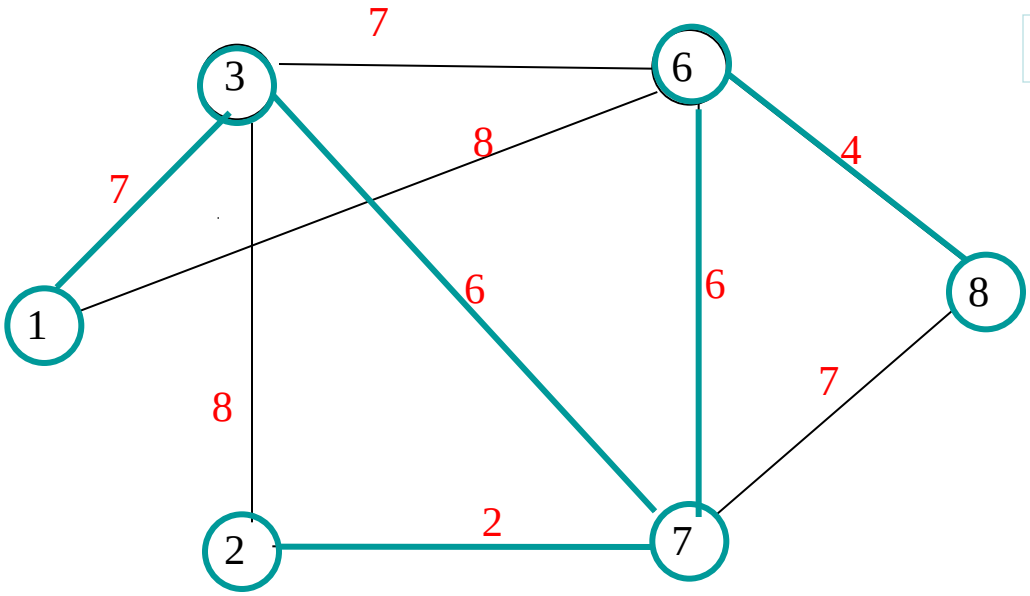
Applications?

- Constructing highways or railroads spanning several cities
- Laying pipelines connecting offshore drilling sites, refineries, and consumer markets
- Designing computer networks
- Others?

Simple Greedy Procedure

- Procedure:
  1. Select any node arbitrarily and then connect it to the nearest distinct node.
  2. Identify the unconnected node that is the closest to a connected node, and then connect these two nodes (ties may be broken arbitrarily). Repeat this until all nodes have been connected.

Example1



Select any node

Connect the closest node

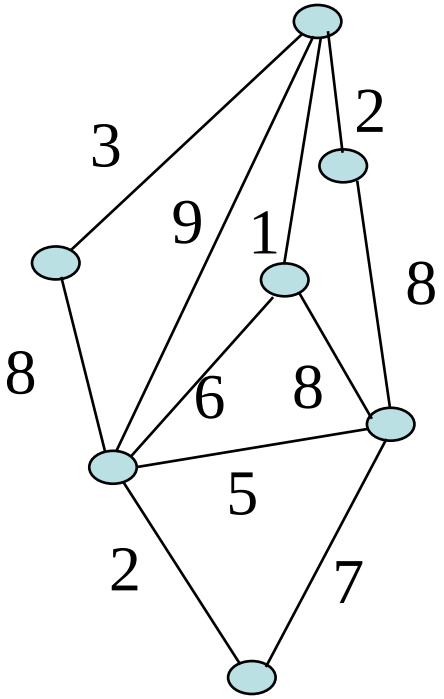
Select the closest node to any of the connected nodes

Select the closest node to any of the connected nodes

$V_{13}, V_{37}, V_{27}, V_{67}, V_{68}=1$ $w(T) = 7+6+2+6+4=25$

Example2

- Apply the greedy procedure to solve the minimum spanning tree for the graph shown below:

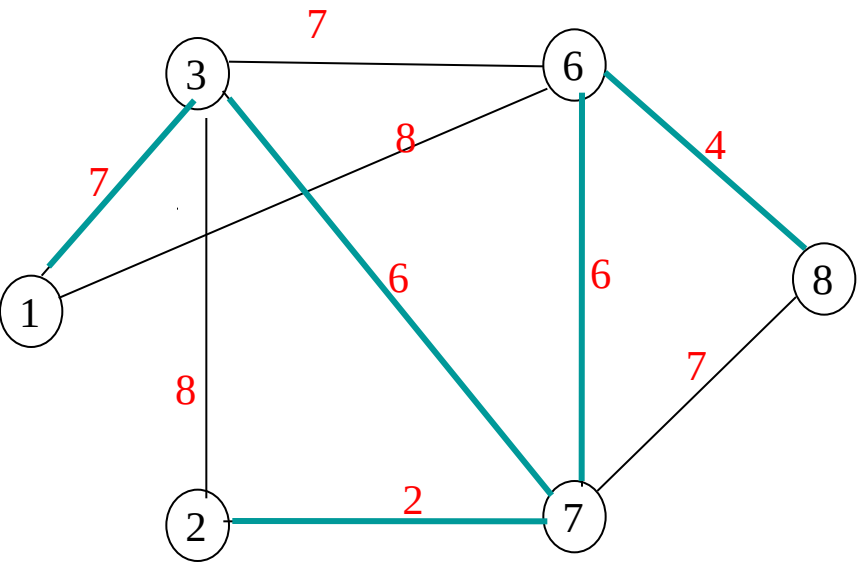


Kruskal's Algorithm

(a more refined greedy algorithm)

- A more efficient algorithm (and also optimal) for obtaining the minimum spanning tree
- Sort all the arcs in non-decreasing order of cost
- Define a set, LIST, that is the set of arcs chosen as part of a minimum spanning tree
- Initially LIST is empty
- Examine the arcs in the sorted order one by one and check whether adding the arc we are currently examining to LIST creates a cycle with the arcs already in LIST. If not, add arc to LIST, otherwise discard it.
- Terminate procedure when the cardinality of LIST equals the number of vertices less one. The result is minimum spanning tree T*

Example3



Iteration	List
0	$\varnothing$
1	V_{27}
2	V_{27}, V_{68}
3	V_{27}, V_{68}, V_{37}
4	$V_{27}, V_{68}, V_{37}, V_{67}$
5	$V_{27}, V_{68}, V_{37}, V_{67}, V_{13}$
6	

Order	Arc
1	2-7
2	6-8
3	3-7
4	6-7
5	3-6
6	1-3
7	7-8
8	1-6
9	2-3

From the sorted list, select the least cost arc

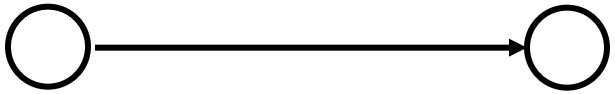
From the sorted list select the least cost arc

Selecting this arc, will create a cycle→we skip it

Since the number of arcs is n-1 (number of nodes –1), we stop

Directed and Undirected Arcs

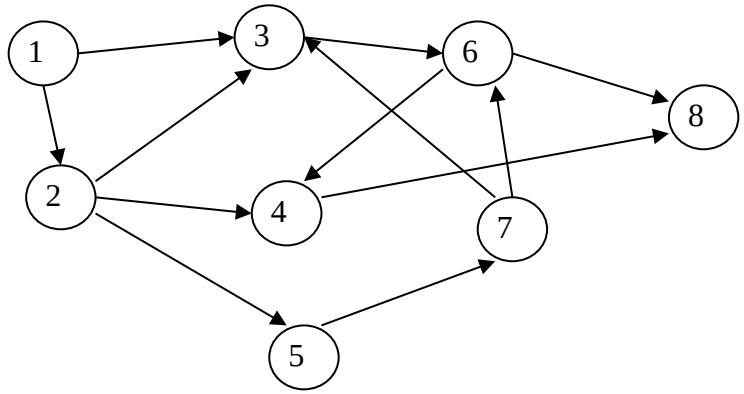
- Directed arcs- used to represent cases when flow is allowed only one direction. Examples: traffic in one-way streets, water flow in channels, etc. They are graphically represented with arrows.



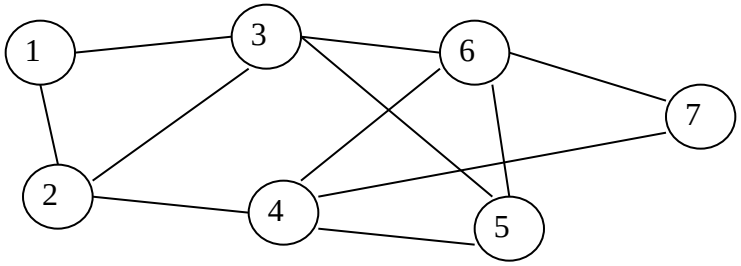
- Undirected Arcs (link)- used to represent cases when flow is allowed in both directions. Examples: traffic in two-way streets, pipes allowing flow in both direction, etc. They are graphically represented with a simple straight line.

Directed and Undirected Networks

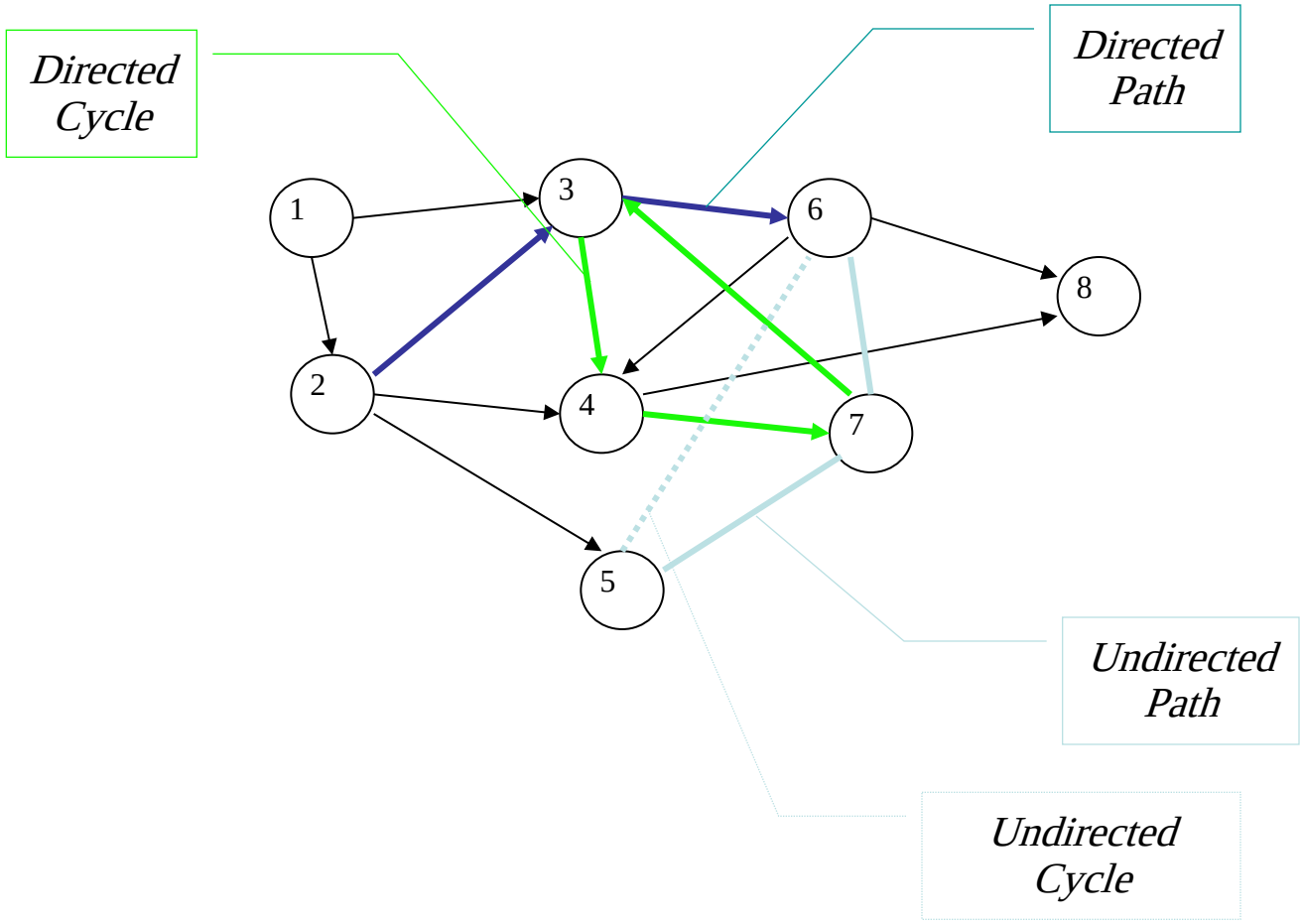
- Directed Network



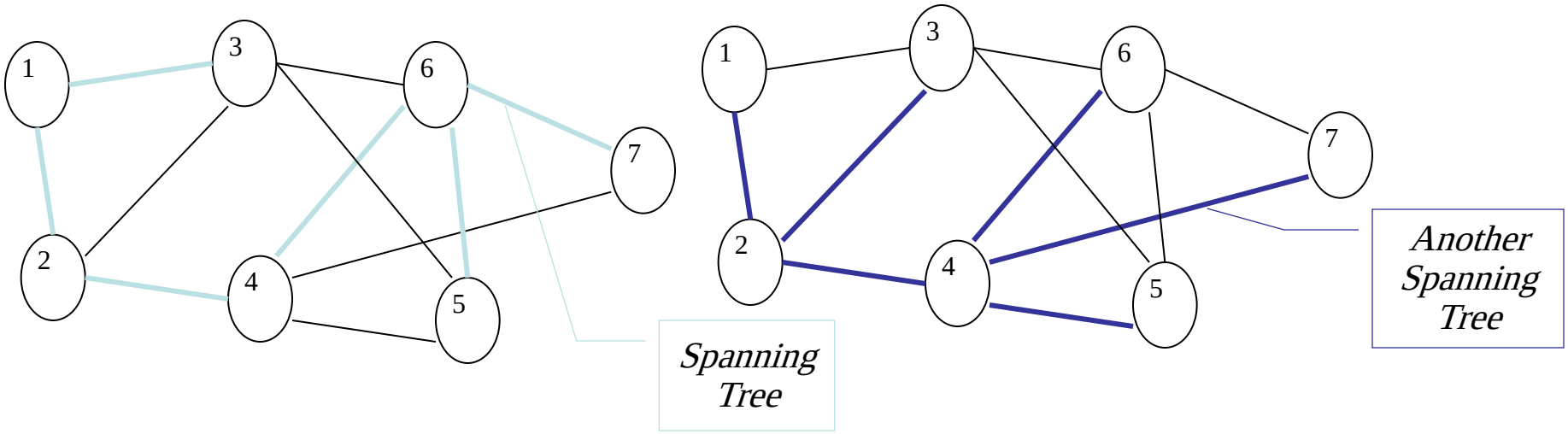
- Undirected Network



Paths


Directed Cycle
Directed Path
1
2
3
4
5
6
7
8
Undirected Path
Undirected Cycle

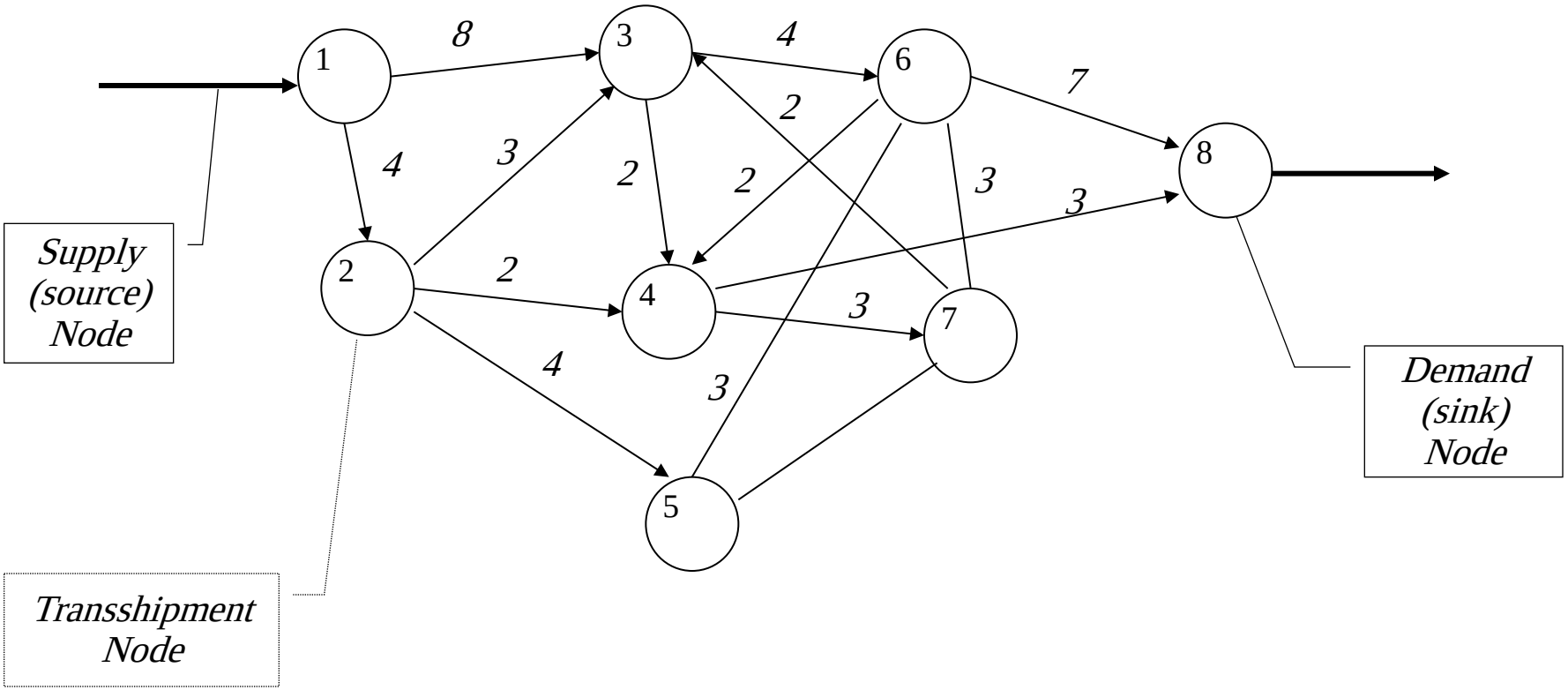
Connected Network and Spanning Tree



- ***Suppose:*** *G=(V,E) is a connected graph with vertex set* V *and edge set* E*, and* T *is a subset of* E *such that:*
  - *Every vertex of* G *is connected by an edge in* T
  - *The edges in* T *form a tree (i.e. there are no cycles)*

then T *is a spanning tree for* G*.*

Capacitated Network

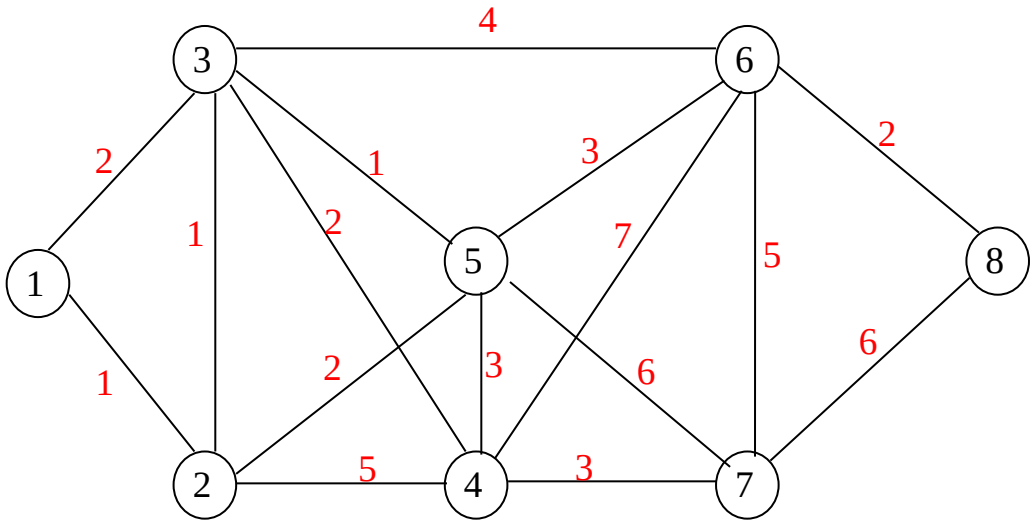


Networks and Graphs

- There is a close relationship between networks and graphs
- Commonly, a network is defined as a graph whose arcs have a certain flow. This is usually accompanied by "parameters" which describe costs or bounds.
- Usually, when we talk about graphs (graph theory) we are interested in the topological (spatial relationship) properties related to the graphical representation of the problem.
- We are more interested in the solution of engineering problems involving some kind of flow in the arcs of the graph (distance, cost, etc) and typically there will be parameters associated with the graph. Thus, the term "network" rather than "graph" is most appropriate. However, we will rely on some results and concepts from graph theory to frame the discussion of the problem at hand.

Shortest Route Problem

- *Objective: To determine the shortest route (path) from the source node to a destination node.*
- *Alternatively, we could define the shortest path problem as determining how to send 1 unit of flow as cheaply as possible from the source node to another node in an incapacitated network.*
- *Corresponding to each arc (*i,j*), there is a non-negative number* d_{ij} *called the distance from* i *to* j *(*$d_{ij} = \infty$ *if we cannot get from* i *to* j *directly).*
- *Example: Find the shortest route between Nodes 1 and 8.*



Applications

- Want to send a truck from a DC to a customer.
- Want to send a message across a telecommunications network.
- What does “shortest” mean in these cases?

Dijkstra's Algorithm

- *Dijkstra's algorithm is an efficient method to find the shortest path between two nodes.*
- *Some of the characteristics of the algorithm:*
  - *Assigns a temporary (unsolved) or permanent (solved) label to each node in the network.*
  - *The temporary (unsolved) label is an upper bound on the shortest distance from the source (origin) node to that node.*
  - *The shortest route from the source to a node is given by the permanent label.*

Steps of the Shortest Path (Dikjstra's) Algorithm

- Objective of the n^{th} iteration: Find the n^{th} nearest node to the origin.
- Input for the n^{th} iteration: n-*1* nearest nodes to the origin (solved at the previous iteration), including their shortest path and the distance from the origin (call these solved nodes).
- Candidates for the n^{th} nearest node: Each solved node that is directly connected by a link to one or more unsolved nodes provides one candidate - the unsolved node with the shortest connecting link (ties provide additional candidates).
- Calculation of the n^{th} nearest node: For each such solved node and its candidate, add the distance between them and the distance of the shortest path from the origin to the solved node. The candidate with the smallest such total distance is the n^{th} nearest node (ties provide additional solved nodes), and its shortest path is the one generating this distance.

Assumptions of Dijsktra's Algorithm

- The arc lengths are integers
- The network contains a directed path from node *s* (the source) to every other node in the network
- The network does not contain a negative cycle (you cannot get back to the origin)
- The network is directed

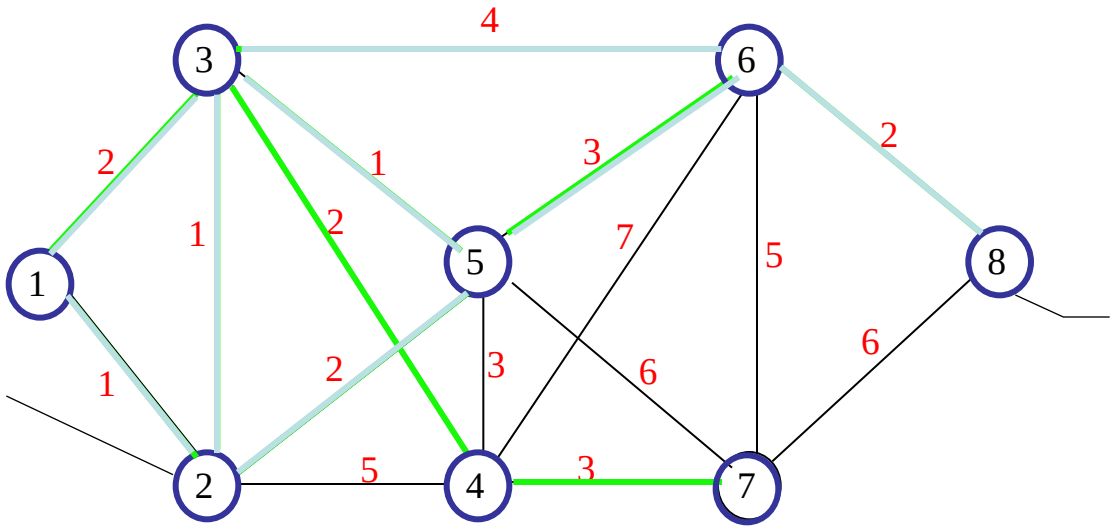
Basic Idea of the Algorithm

- Maintains a directed out-tree T rooted at s that spans the nodes with finite distance labels.
- Maintains this tree using predecessor indices and the following invariant property:
  - Every tree arc (i,j) satisfies the condition $d(j)=d(I)+c_{ij}$ with respect to the current distance labels
- The operation of selecting a minimum temporary distance label is termed the *node selection* operation.
- The operation of checking whether the current labels for nodes I and j satisfy $d(j)>d(I)+c_{ij}$ and, if so, then setting $d(j)=d(I)+c_{ij}$ is the called the **distance update** operation

Dijkstra's Algorithm Notation

- Finds shortest path from the source node to all other nodes in the network
- Notation
  - d(i) is the distance label for node i
  - S is the set of permanently labeled nodes
  - S' is the set of temporarily labeled nodes
  - s is the source node
  - c_{ij} is the cost (distance) on arc (i,j)
  - T is the shortest path tree on G
  - pred(j) = i if $(i,j) \in T$
  - $A(i)$ is the arc adjacency list of node i

Example4



Permanent Label (solved)

Once we have reached the destination, we map the shortest route

n	Solved nodes directly connected to Unsolved nodes	Closest connected unsolved node	Total Distance Involved	n^{th} Nearest node	Minimum Distance	Last Connection
1	*1*	*2*	*1*	*2*	*1*	*1-2*
2	*1* *2*	*3* *3*	*2* *1+1*	*3* *3*	*2* *2*	*1-3* *2-3*
3	*2* *3*	*5* *5*	*1+2* *2+1*	*5* *5*	*3* *3*	*2-5* *3-5*
4	*2* *3* *5*	*4* *4* *4*	*1+5* *2+2* *3+3*	*4*	*4*	*3-4*