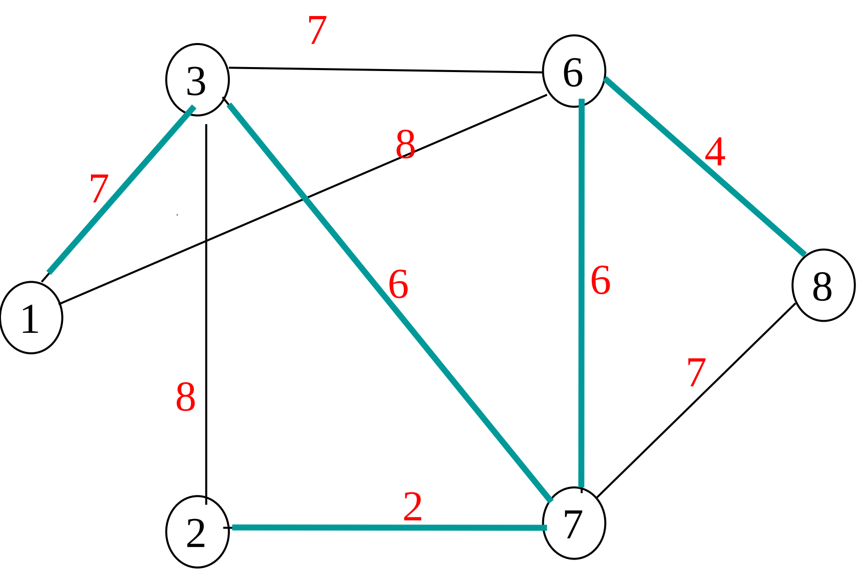
Kruskal's Algorithm

(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



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

Iteration	List
0	∅
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

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

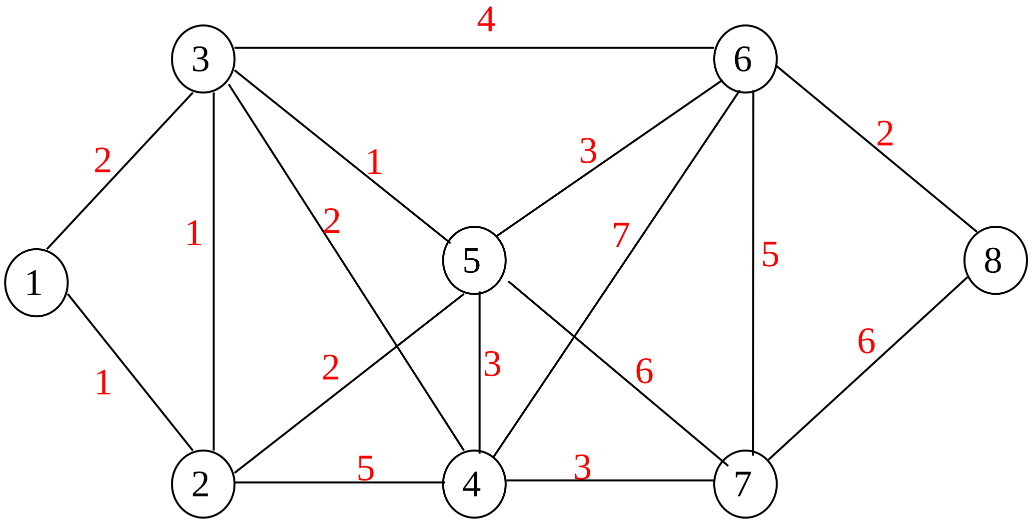
Shortest Route Problem

- *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