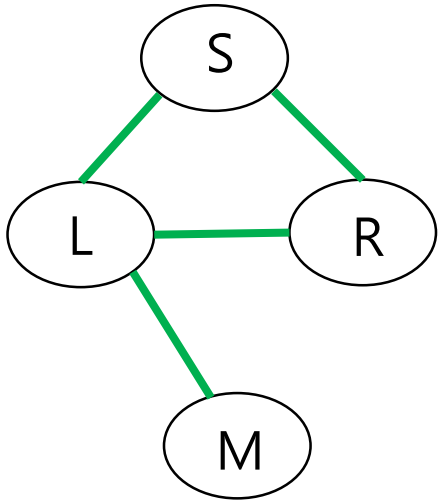
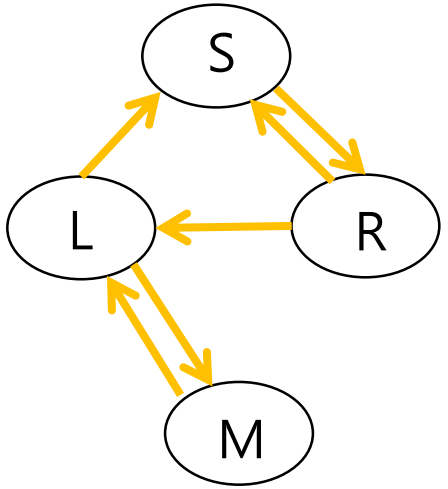
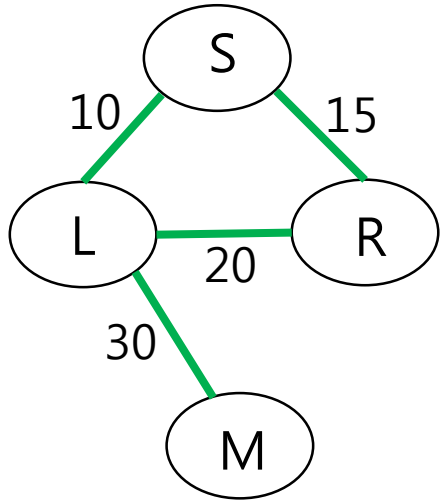
Chap 6. Graphs

Graph

- 망(network) 구조의 데이터를 표현하기 위한 자료 구조
  - Set of nodes
  - Set of edges
- Terminologies
  - Node, vertex
  - Edge, arc: a pair of nodes
  - Undirected graph
  - Directed graph
  - Complete graph
  - Subgraph
  - Path
  - Cycle, acyclic graph
  - (Connected) component
  - Strongly connected component
  - Degree of a node
    - In-degree
    - Out-degree
  - Adjacency
  - Weighted graph

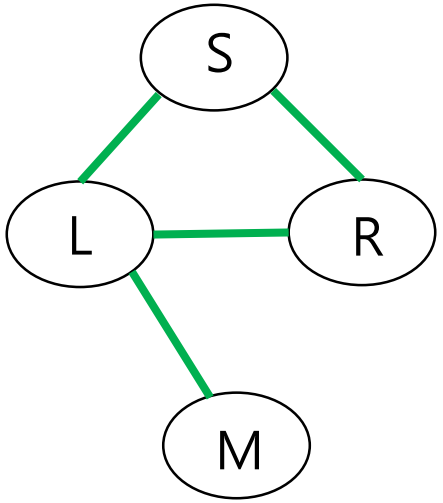




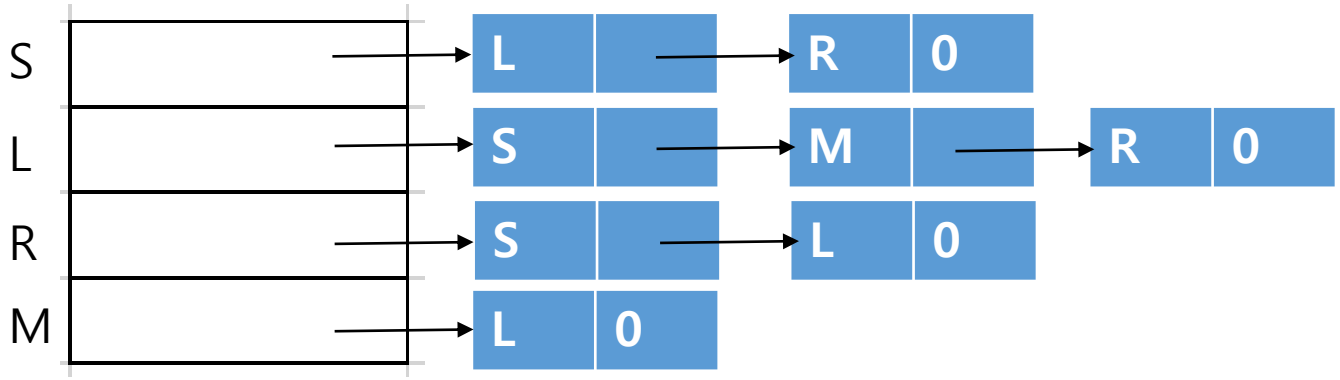


Representations of a graph

- Adjacency matrix
- Adjacency list

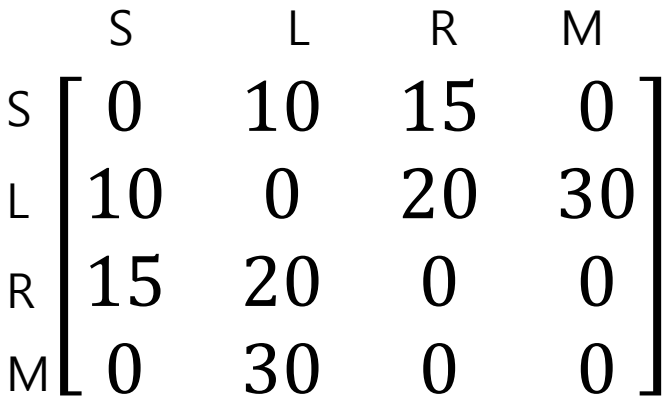


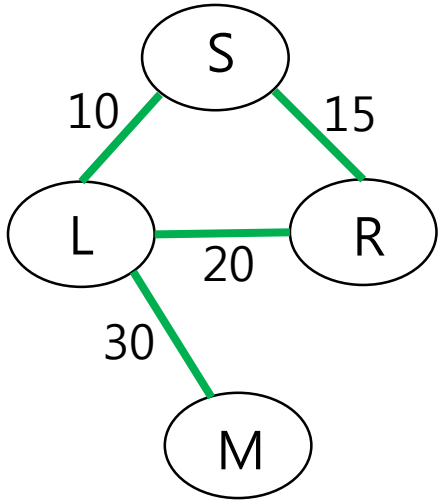
	S	L	R	M
S	0	1	1	0
L	1	0	1	1
R	1	1	0	0
M	0	1	0	0

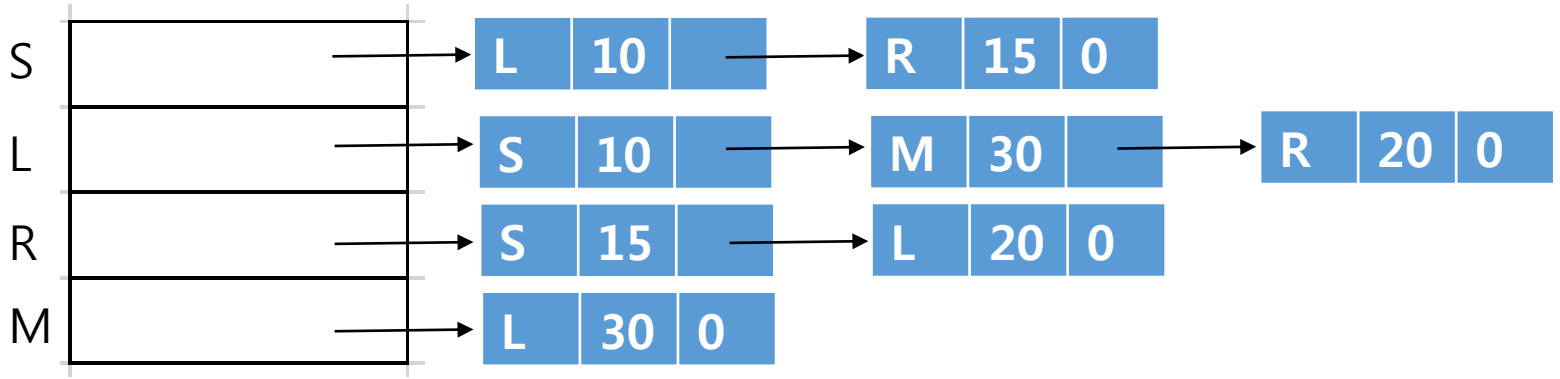


Representations of a weighted graph

- Adjacency matrix
- Adjacency list

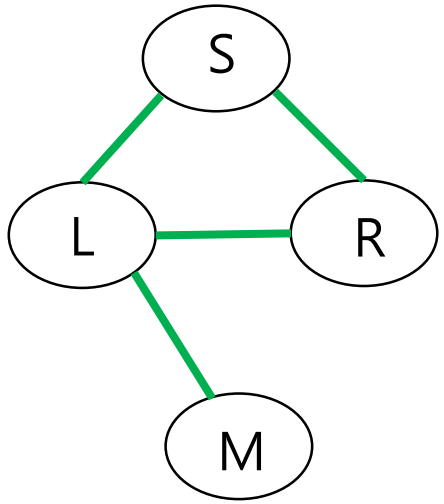


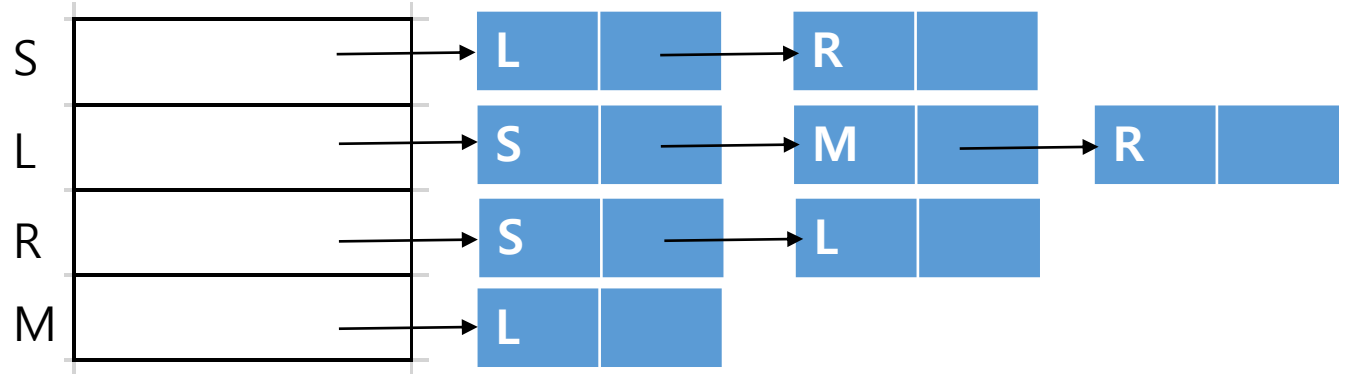




Traversal of a graph

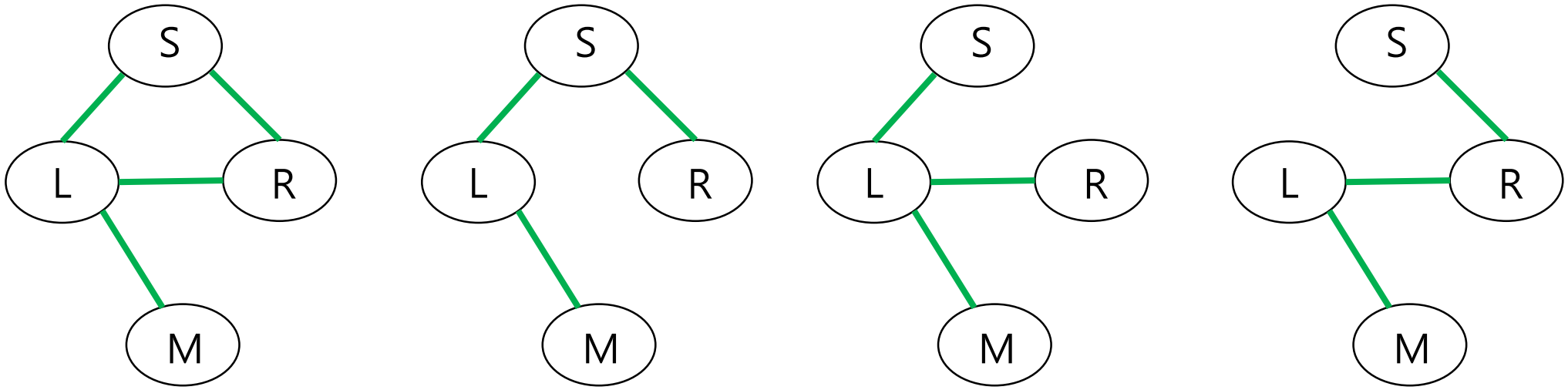
- Depth first search
- Breadth first search
  - Use a queue
- Example
  - dfs(S): S, L, M, R
  - bfs(S): S, L, R, M



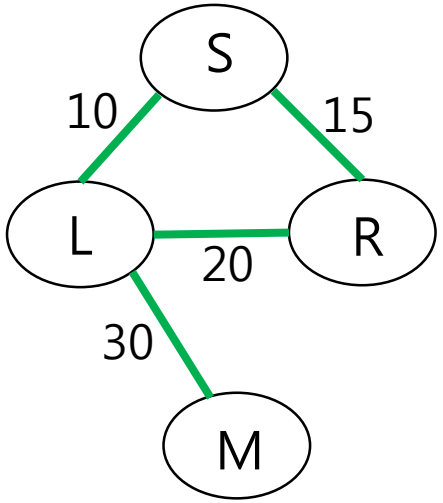


Spanning tree

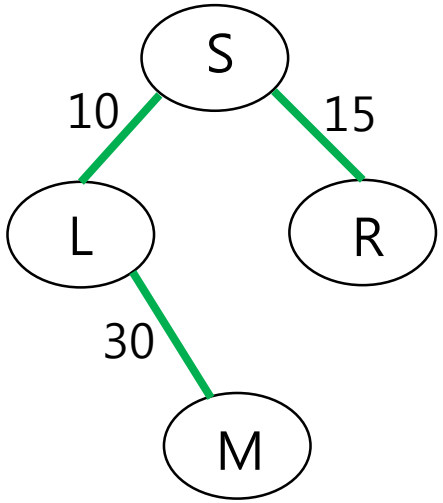
- Spanning tree of graph G
  - Acyclic subgraph of G including all the nodes of G



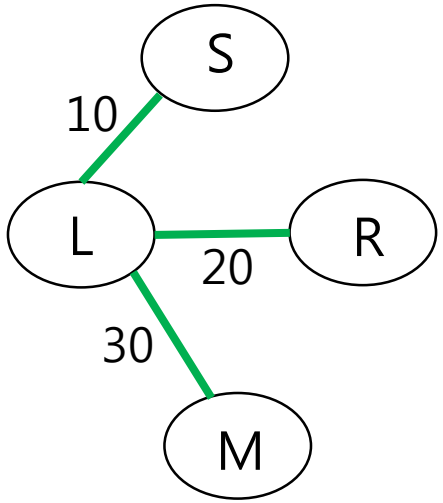
Minimum cost spanning tree



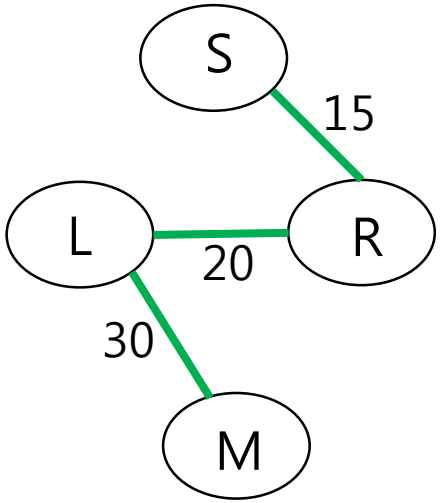
- Weighted undirected graph
- Cost = sum of the weights
- Minimum cost spanning tree
  - The spanning tree whose cost is minimum
- Greedy methods
  - Kruskal's algorithm
  - Prim's algorithm
  - Sollin's algorithm



cost = 55



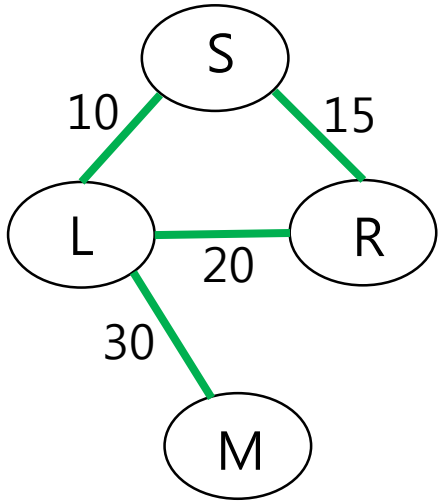
cost = 60

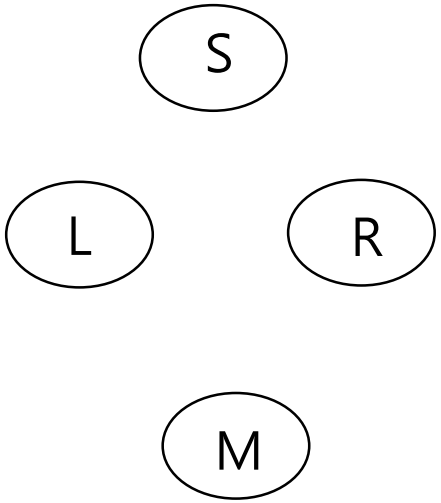


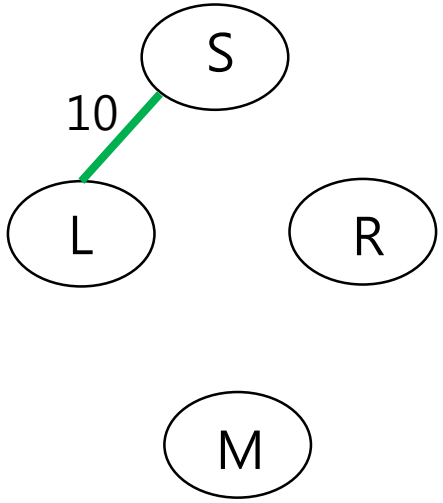
cost = 65

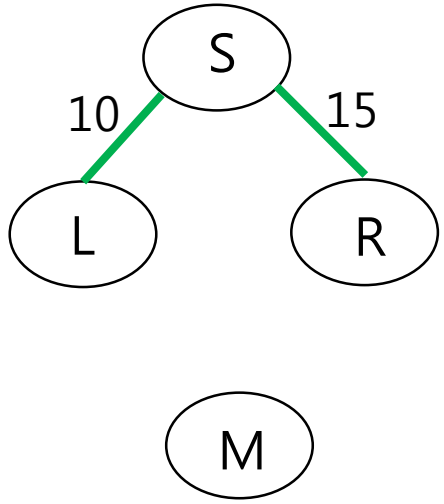
Kruskal's algorithm

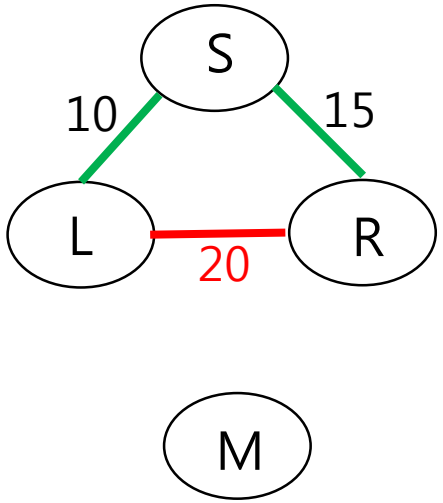
- Start with all the nodes w/o an edge
- Add min cost edge unless it creates a cycle
- Repeat until a tree is obtained

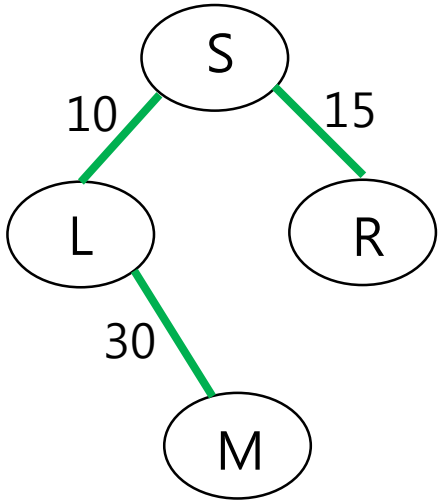






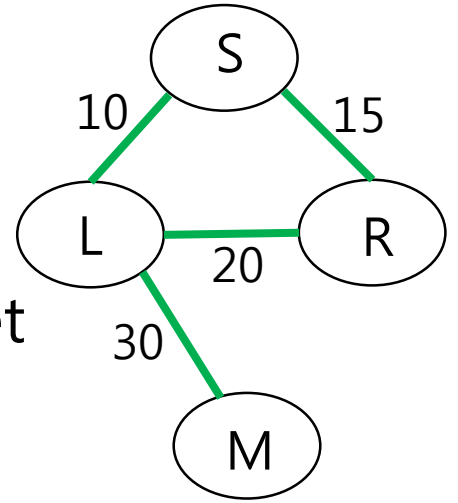


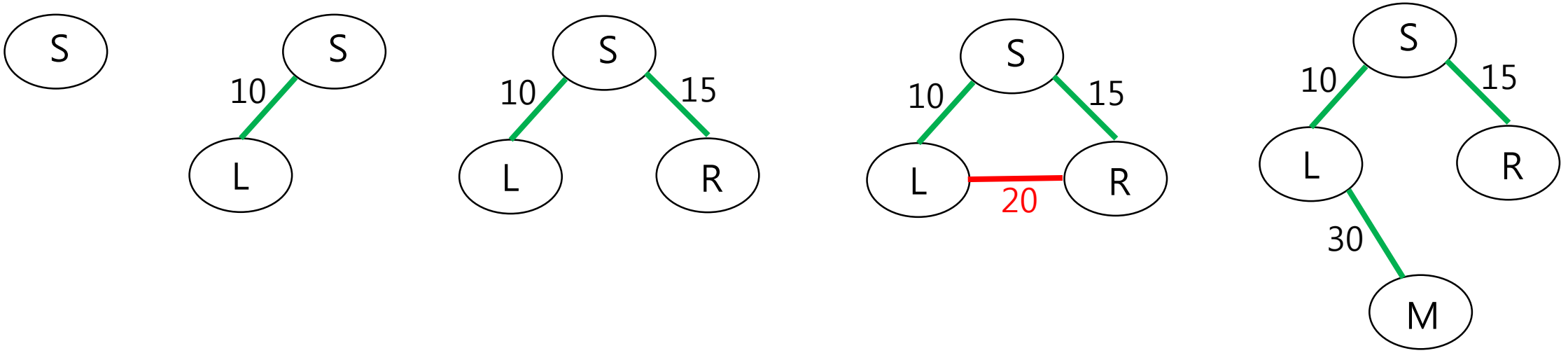




Prim's algorithm

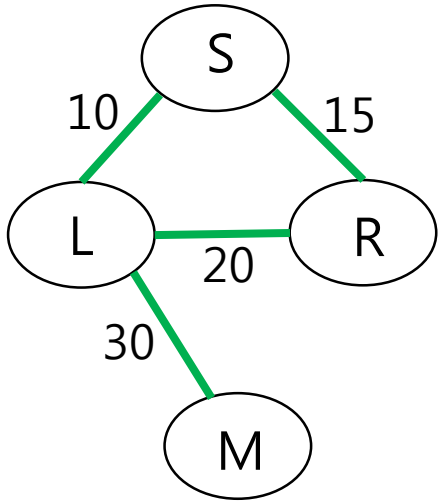
- Strat with any node x, & initializae NodeSet = { x }
- Add min cost edge (a,b) s.t. a ∈ NodeSet & b ∉ NodeSet
- Update NodeSet by adding b
- Repeat until a tree is obtained

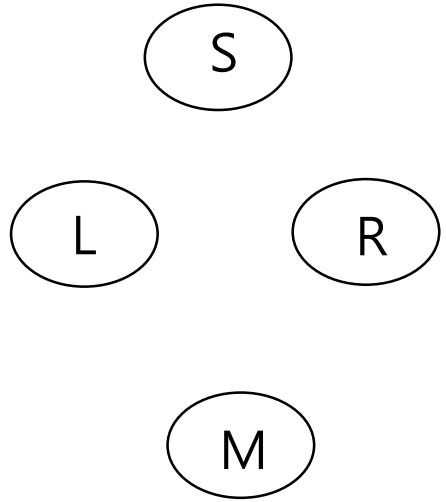


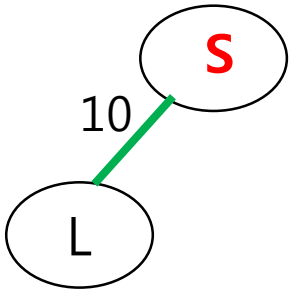


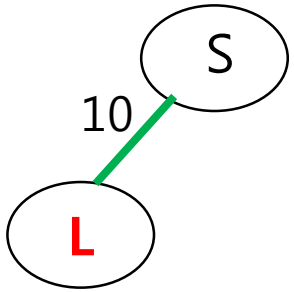
Sollin's algorithm

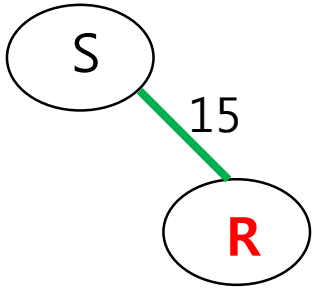
- Start with a forest of trees, each of which consists of a single node
- For each tree T, add min cost edge (a,b) s.t. a is in T & b is not in T
- Remove duplicate edges
- Repeat until a tree is obtained

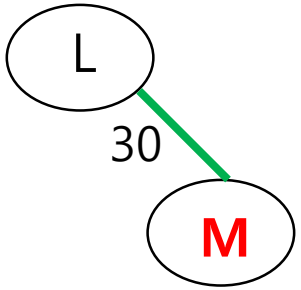


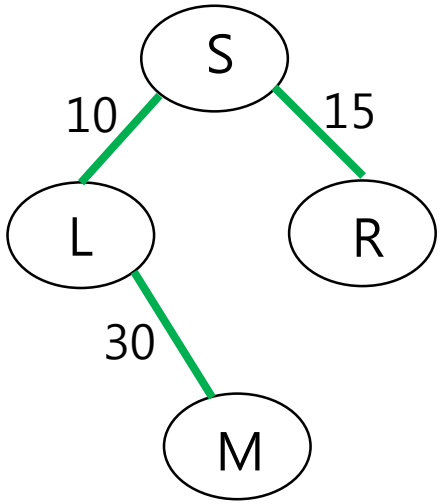






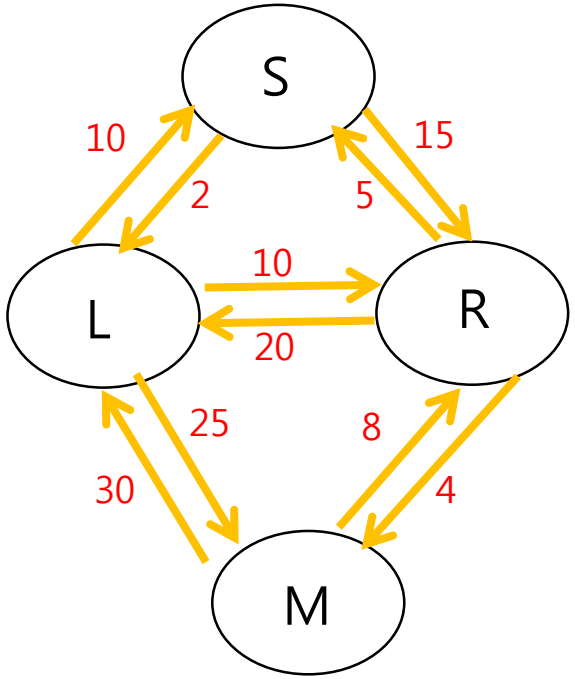






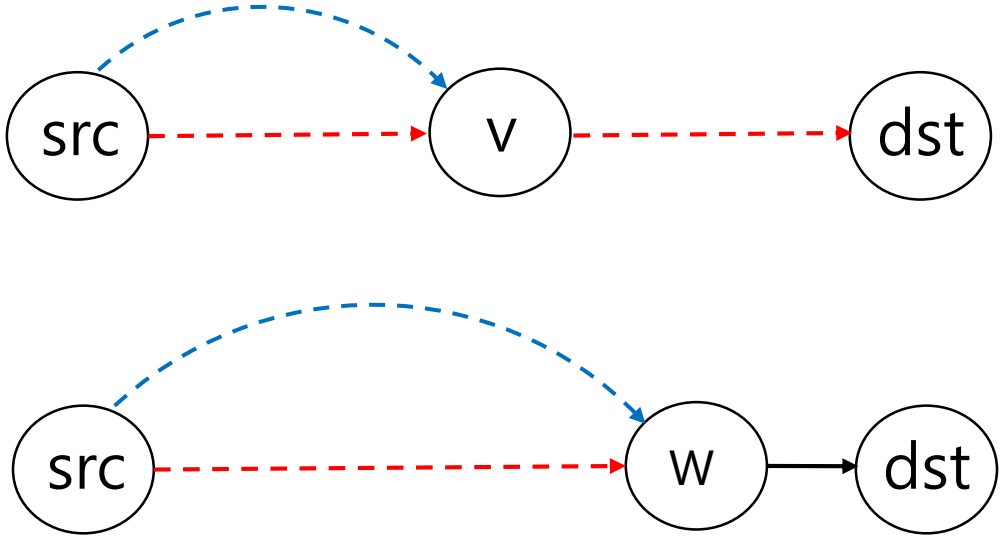
Shortest paths

- Weighted directed graph
- Path length = sum of weights in the path
- Example
  - Source = S, destination = M
  - Paths(length): SLM(27), SRM(19), SLRM(16), SRLM(60)
  - Shortest path: SLRM
- Dijkstra's algorithm
- Single source to all destinations
- Example: src = S
  - S (0)
  - S → L (2)
  - S → L → R (12)
  - S → L → R → M (16)



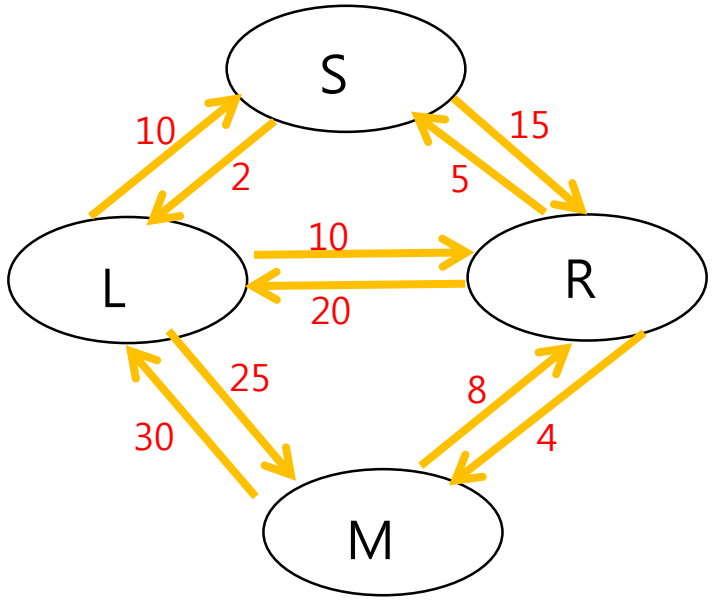
Shortest paths(2)

- Observations
  - Principle of optimality
- Shortest paths are obtained one by one in increasing order of lengths
  - The path from src to src (length 0)
  - ……
  - The shortest path from src to the “farthest” destination
- Arrays used
  - found[i]: True if shortest path from src to destination node i has been found
  - distance[i]: recording shortest distance from src to destination node i
  - predecessor[i]: recording the predecessor node of node i



Shortest paths(3)

- Example: src = S
  1. S (0)
  2. S → L (2)
  3. S → L → R (12)
  4. S → L → R → M (16)



	S	L	R	M
found[ ]	TRUE	FALSE	FALSE	FALSE
dist[ ]	0	2	15	∞
pred[ ]	undefined	S	S	undefined

	S	L	R	M
found[ ]	TRUE	TRUE	FALSE	FALSE
dist[ ]	0	2	12	27
pred[ ]	undefined	S	L	L

	S	L	R	M
found[ ]	TRUE	TRUE	TRUE	FALSE
dist[ ]	0	2	12	16
pred[ ]	undefined	S	L	R