CSE 201: Data Structures

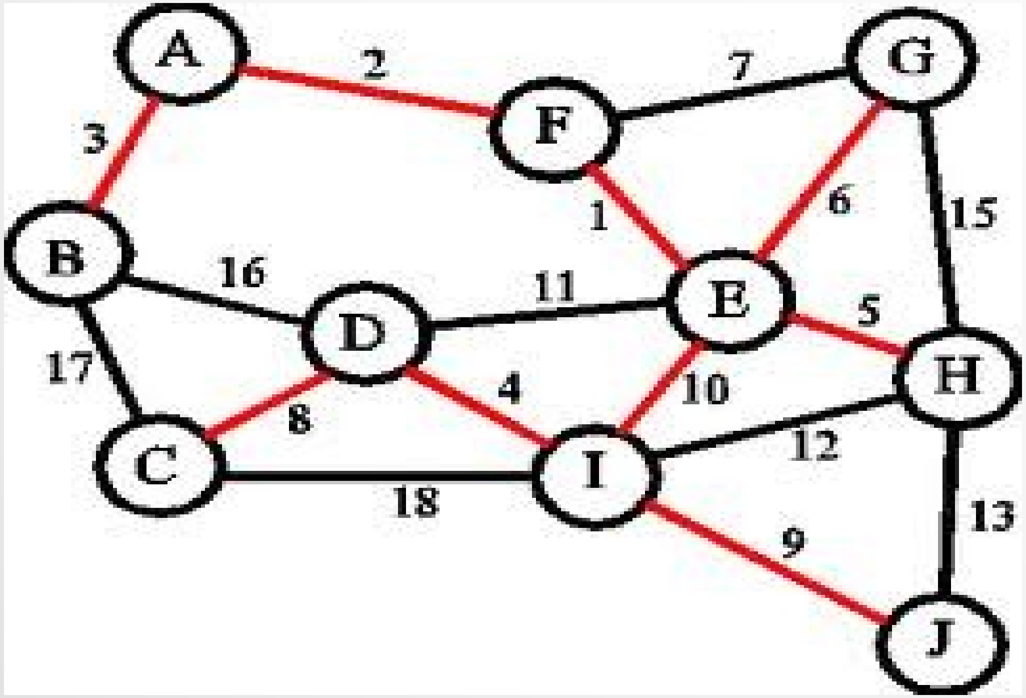
Lecture 10: Graphs

Dr. Vidhya Balasubramanian

Amrita School of Engineering
Amrita Vishwa Vidyapeetham

Minimum Spanning Tree

- Given a weighted undirected graph G, goal is to find a tree T such that
  - T contains all vertices in G
  - Sum of weights of edges in T is minimum



https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcR0WC7jxovpohGVu8VtcB_uGiCnlv3sDAl8zgGZ7NMJgO2QpkrS

Algorithms

- First algorithm to find MST was by a Czech scientist Baruvka
  - Baruvka's algorithm
  - Uses greedy approach
- Other greedy approaches
  - Prim's algorithm
  - Kruskal's algorithm

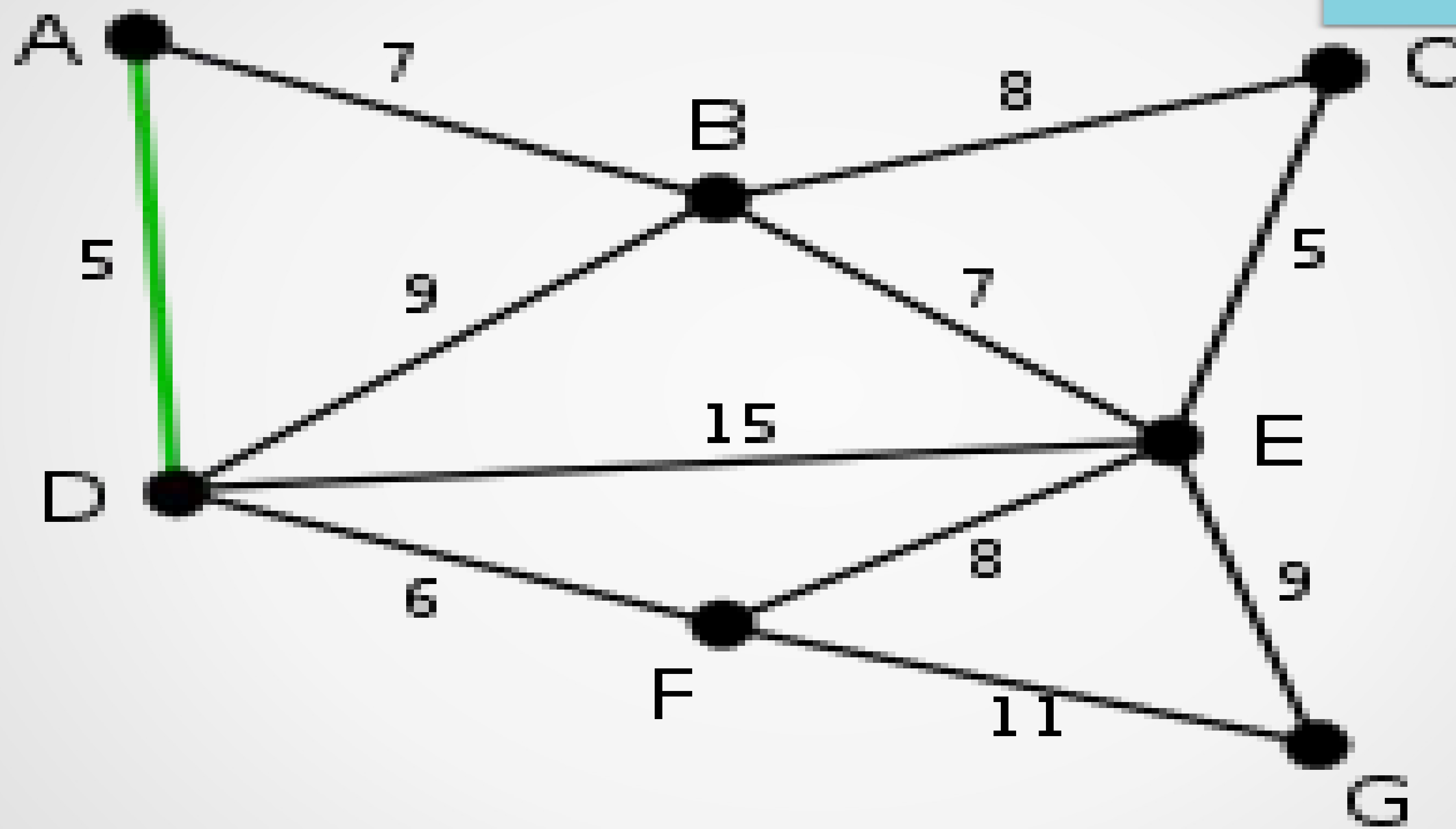
Kruskal's Algorithm

- Greedy Algorithm
- The algorithm maintains a forest of trees
- An edge is accepted it if connects distinct trees
  - The edge is chosen such that its weight is minimum amongst the edges connecting the two trees
  - The trees are also known as clusters or clouds

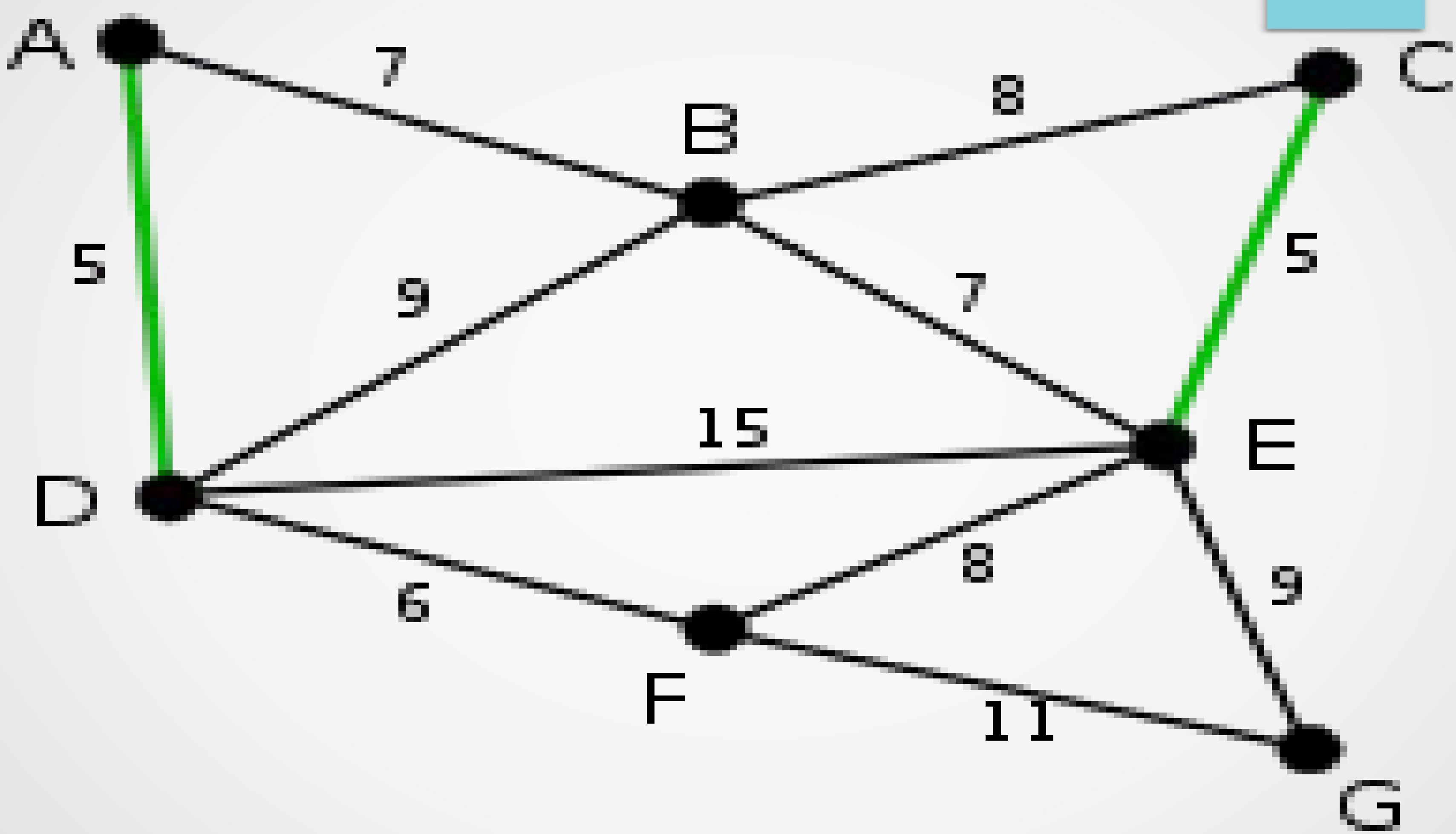
Kruskal's Algorithm

- Let every node in G be a cluster C(v)
- Initialize a priority queue Q with all edges in G using weights as keys
- Take the minimum weight edge in Q and if C(u) <> C(v)
  - add the edge to MST T
  - Merge the clusters C(u) and C(v)
- Repeat till there are no more clusters to merge

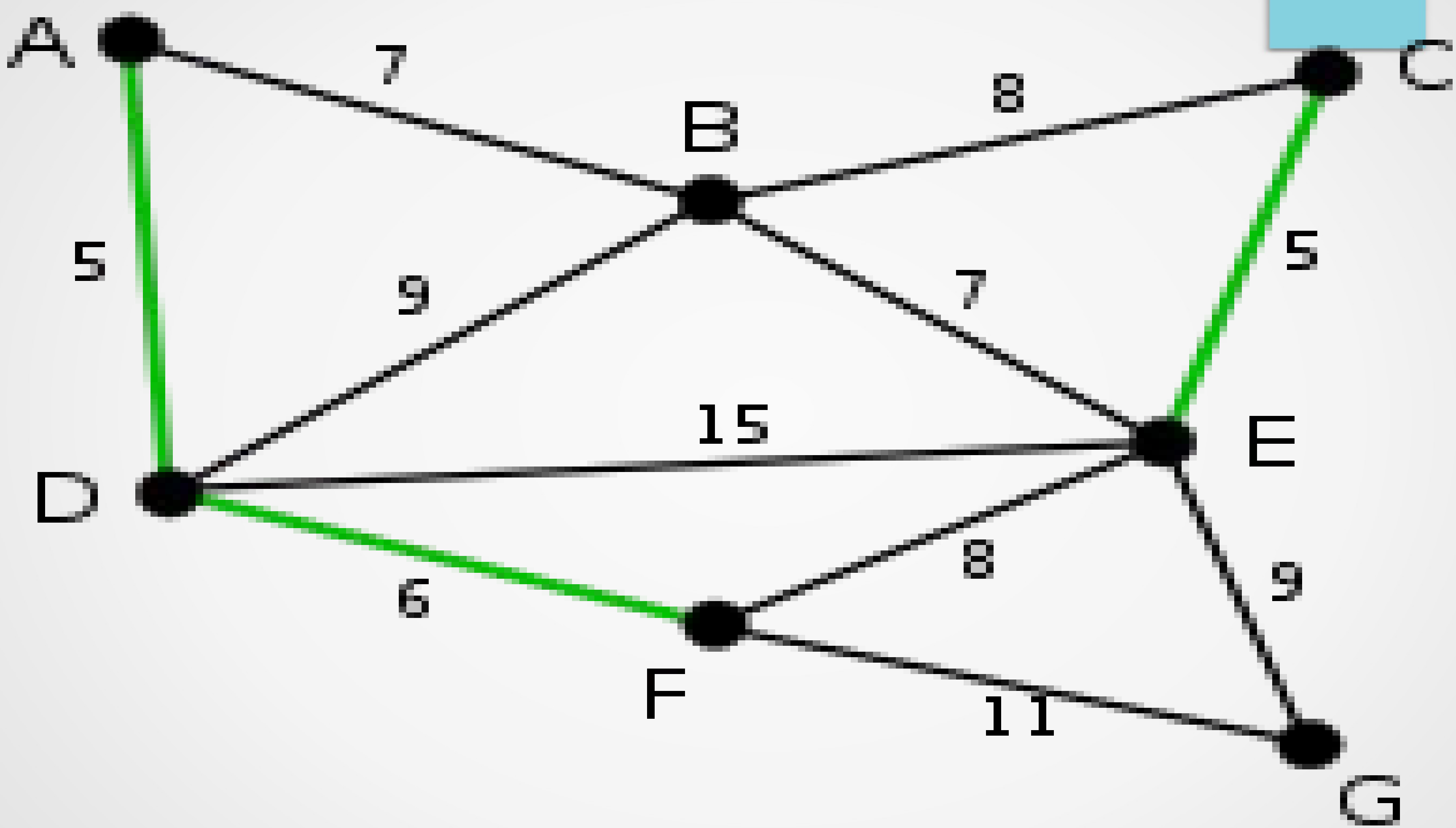
Kruskal's Algorithm: Example



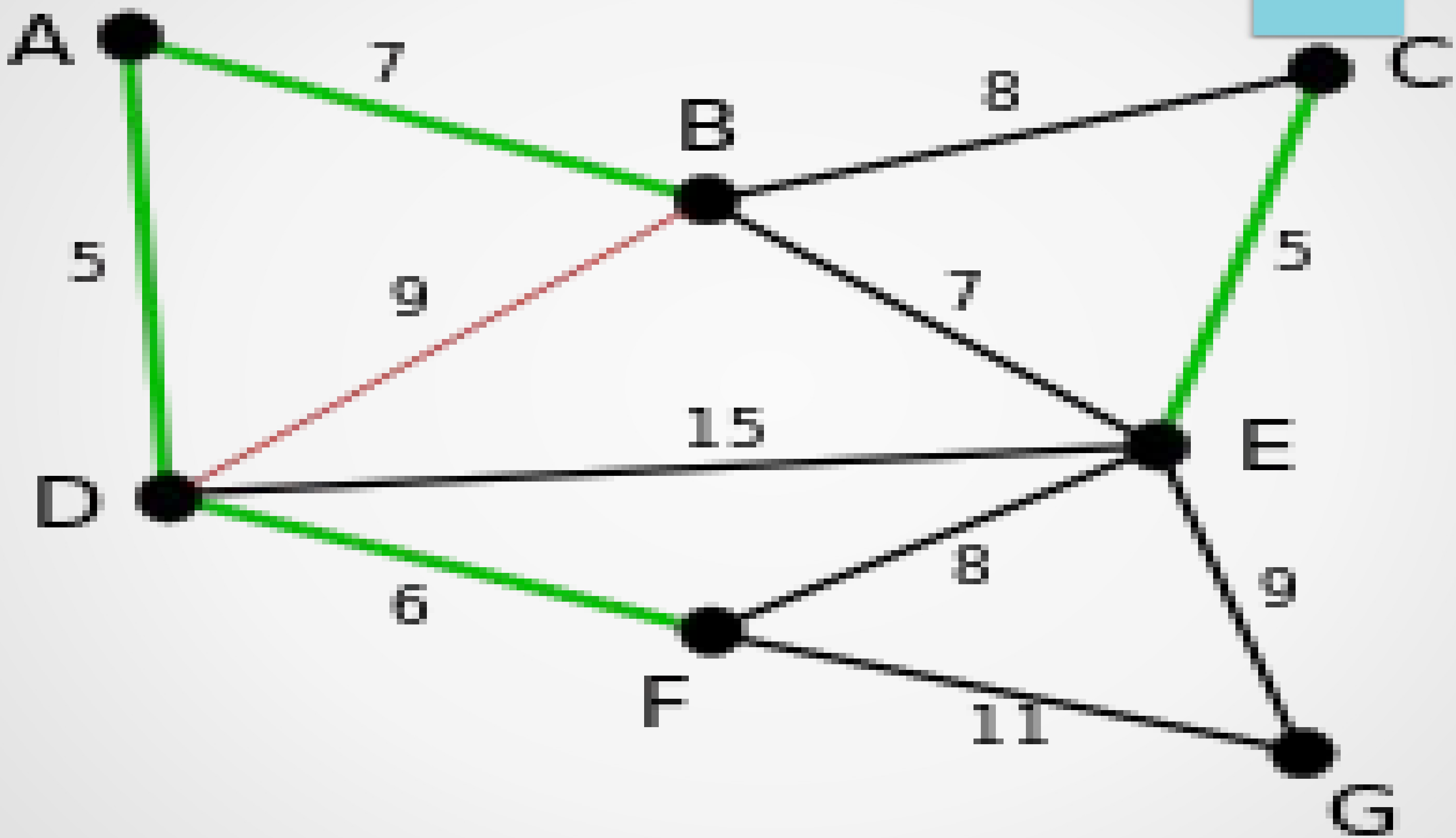
Kruskal's Algorithm: Example



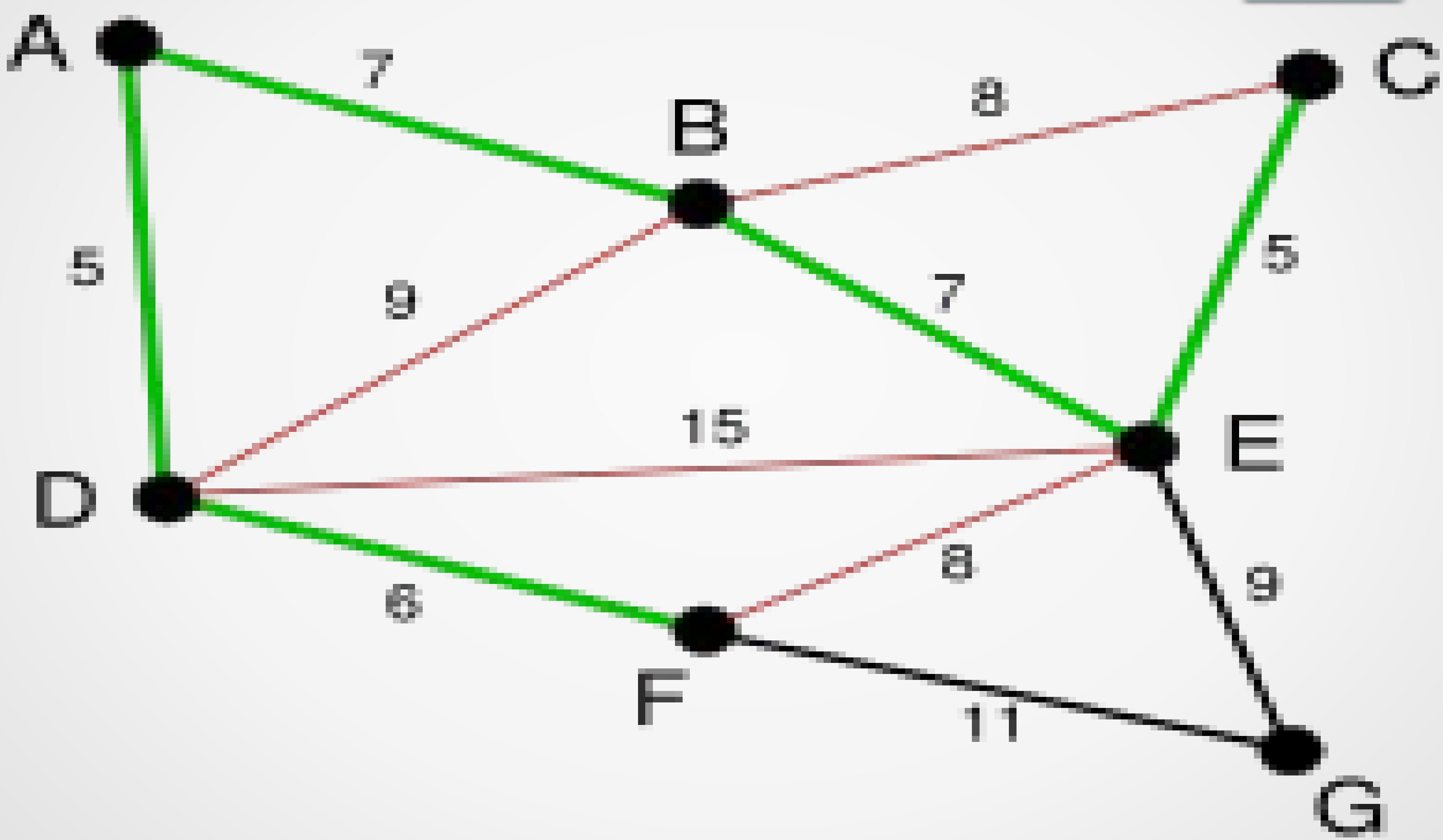
Kruskal's Algorithm: Example



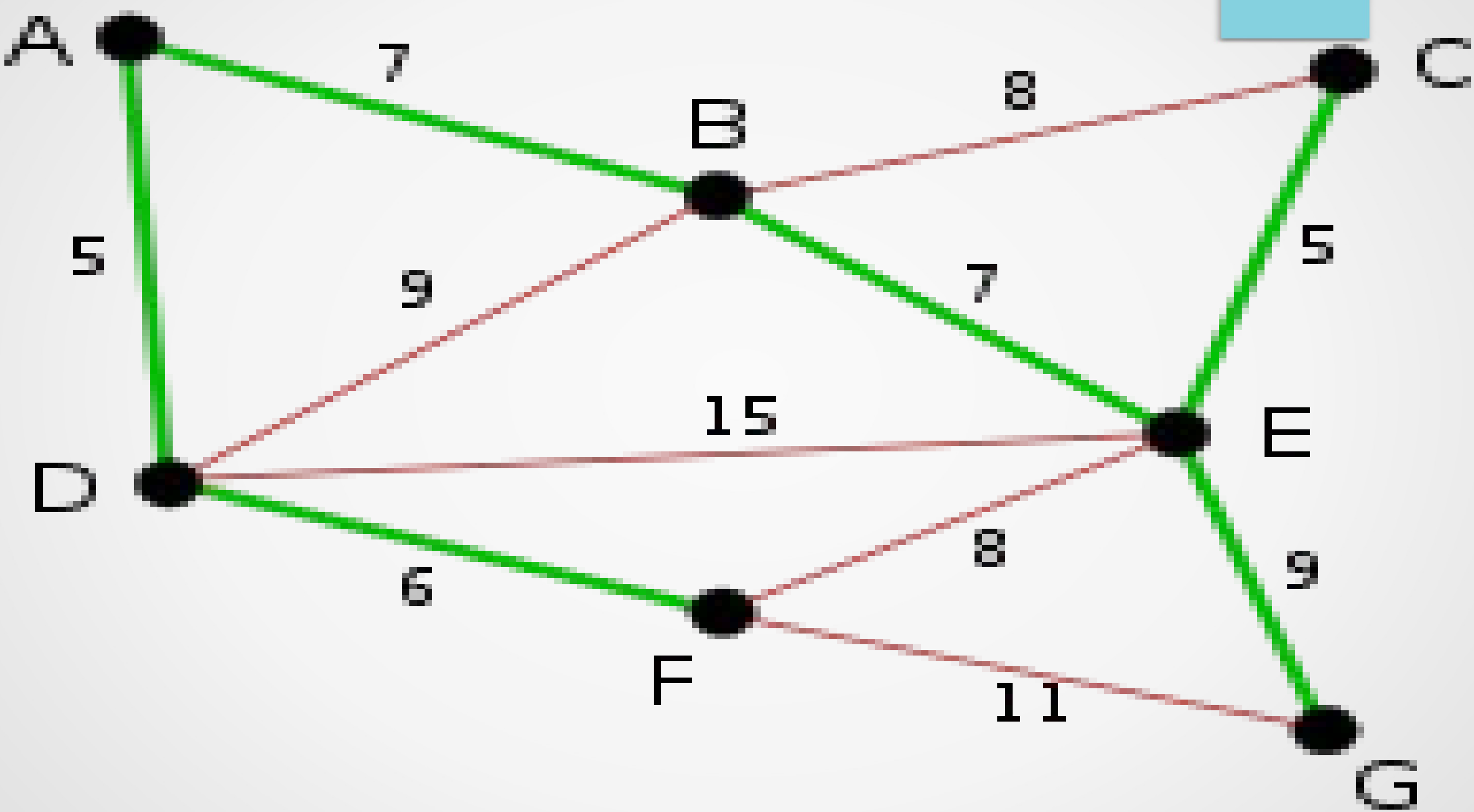
Kruskal's Algorithm: Example



Kruskal's Algorithm: Example

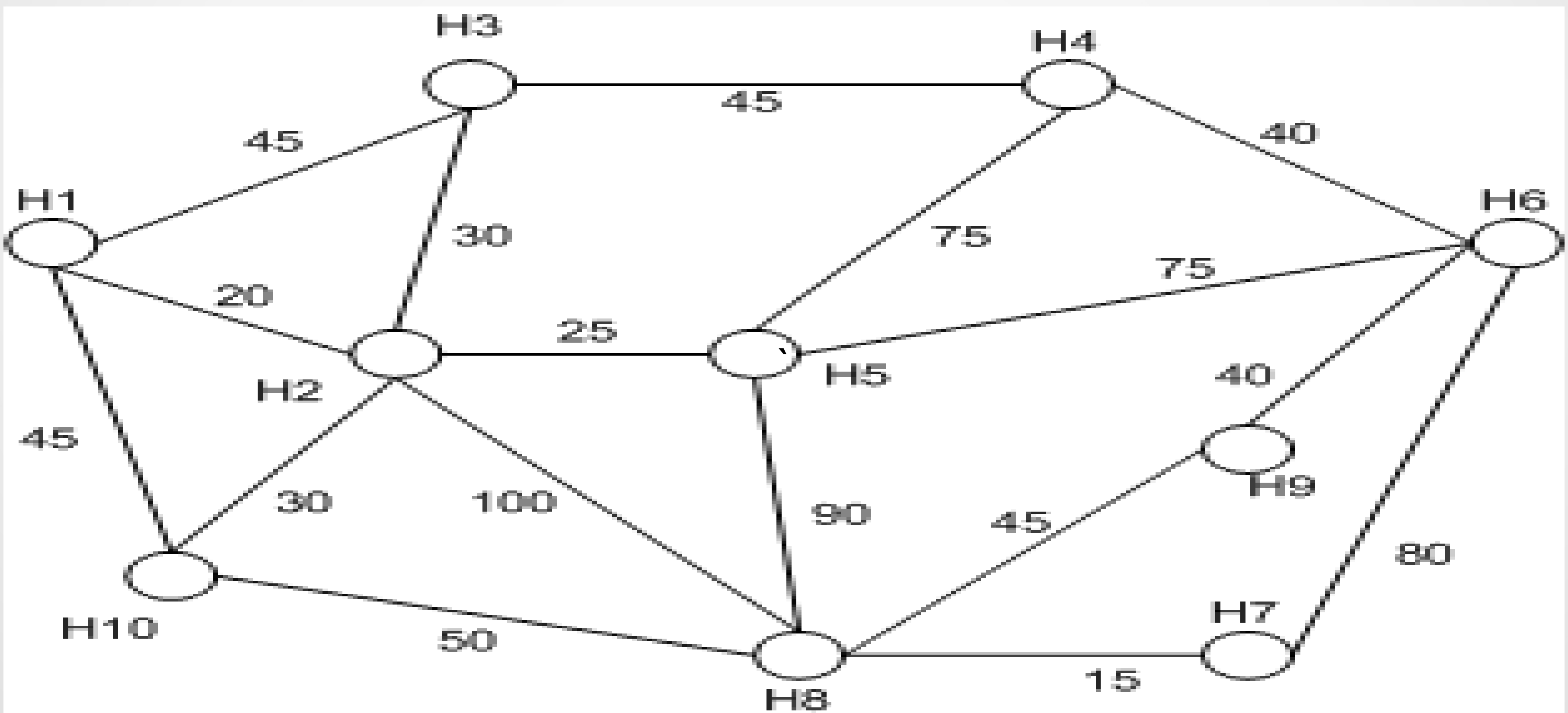


Kruskal's Algorithm: Example



Exercise

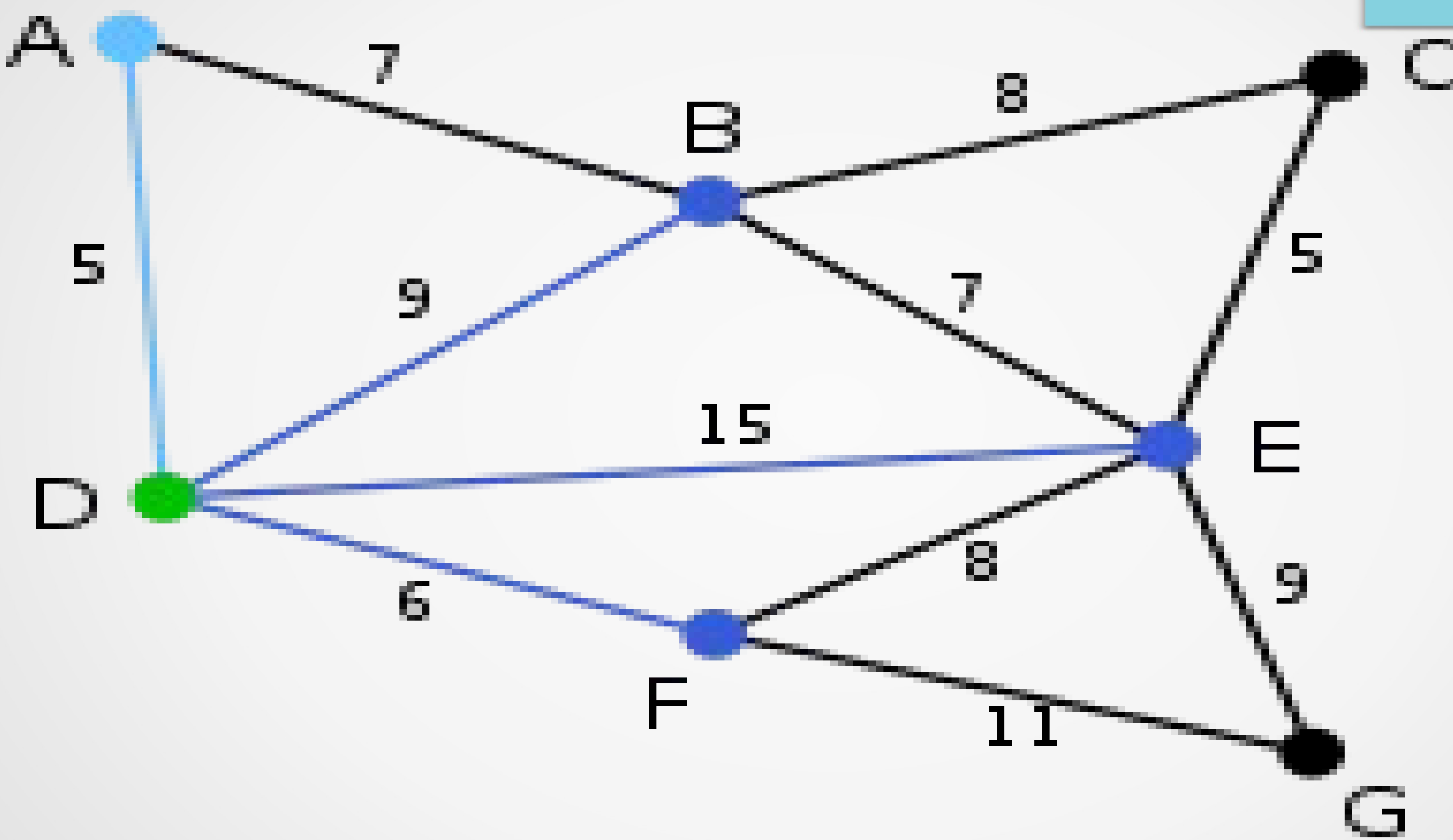
- Find the MST for the following graph using Kruskal's algorithm



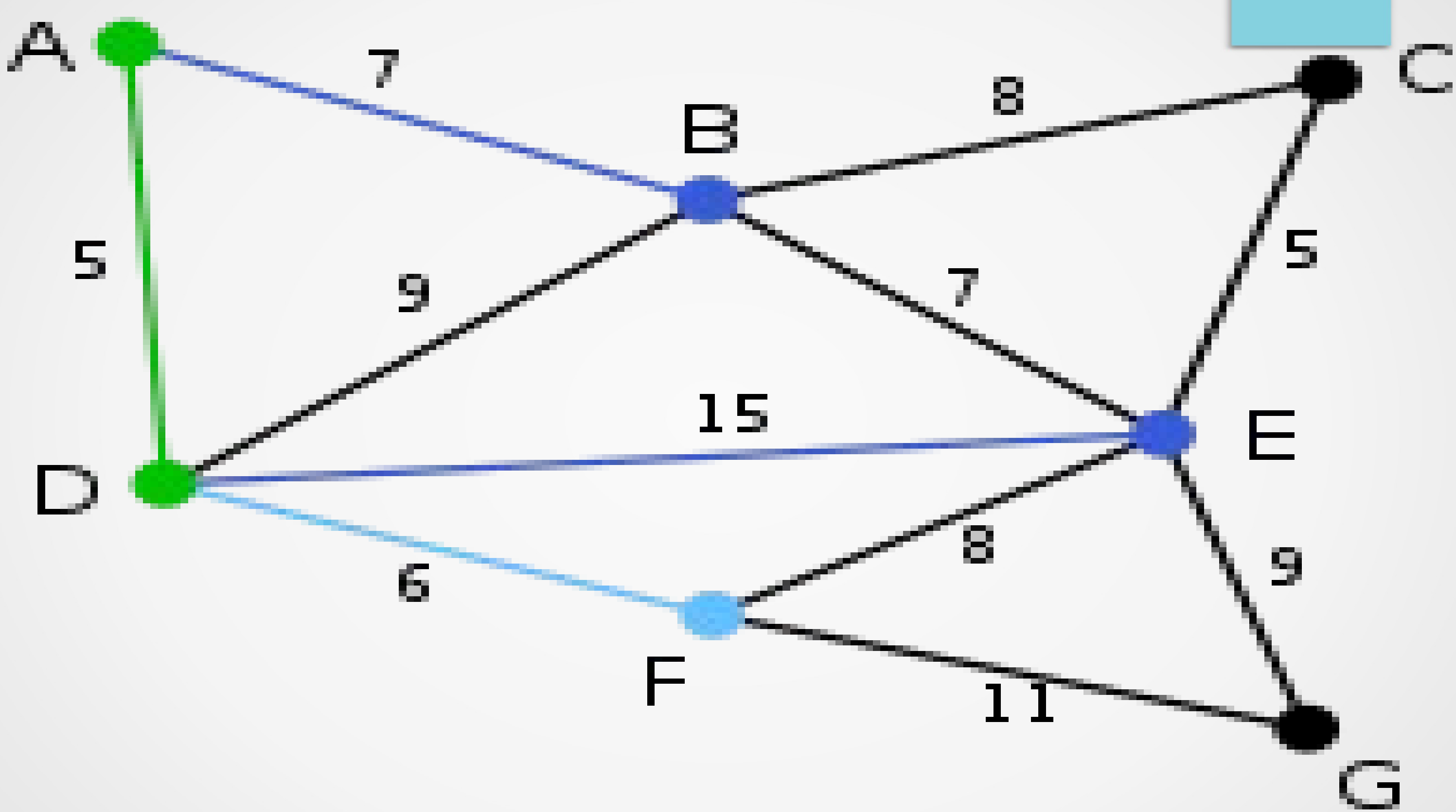
Prim's Algorithm

- Similar to Dijkstra
- Pick any vertex v of G
- Initialize for all u not v, d[u] to infinity and d[v]=0
- Remove from Q u with minimum d[u]
  - Add u and edge (u,v) to T
  - For all neighbors z of u, do relaxation by finding d[z] = w(u,z), and update Q
    - If d[z] > d(u,z), d[z] = d(u,z)

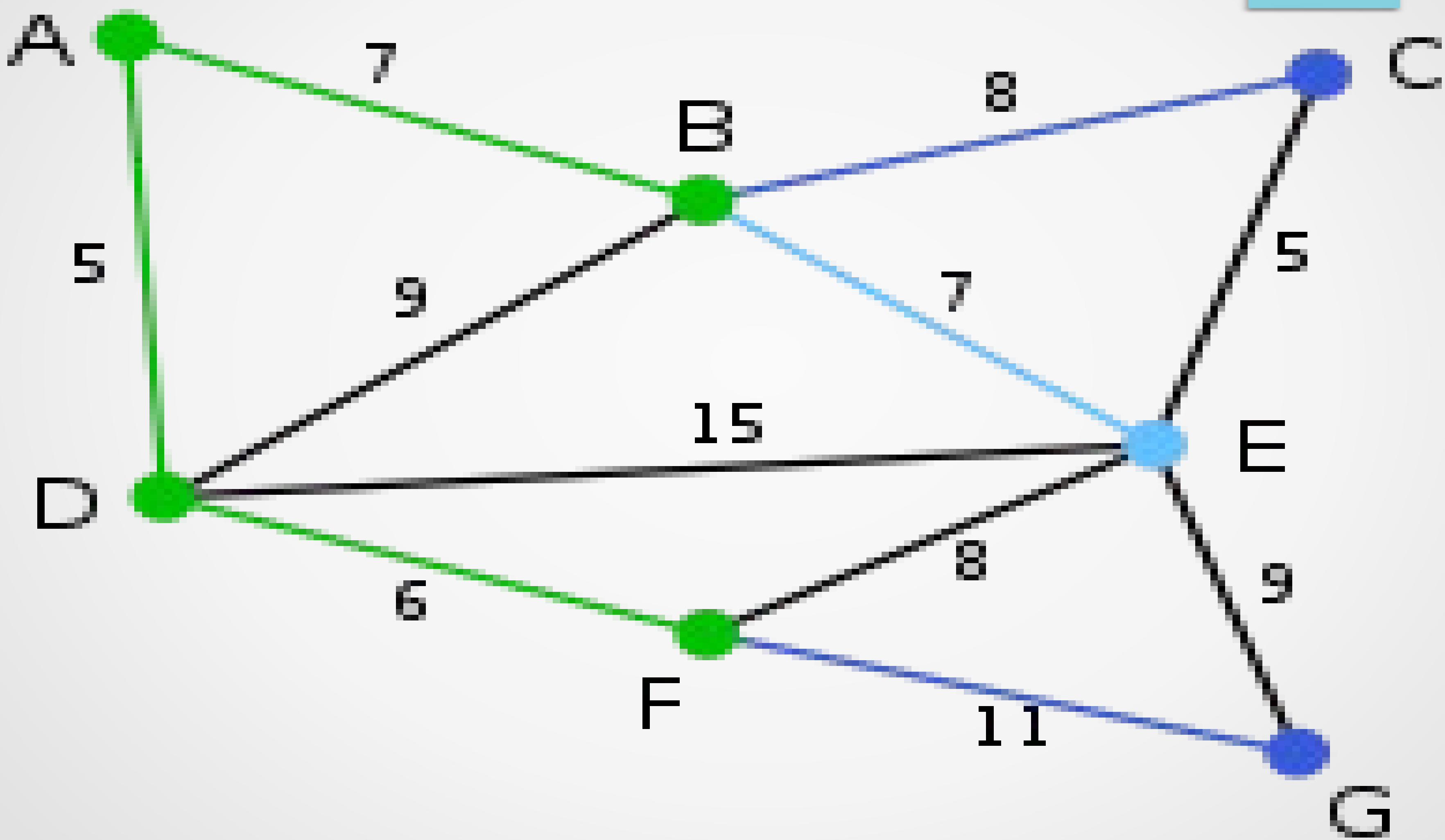
Prim's Algorithm: Example



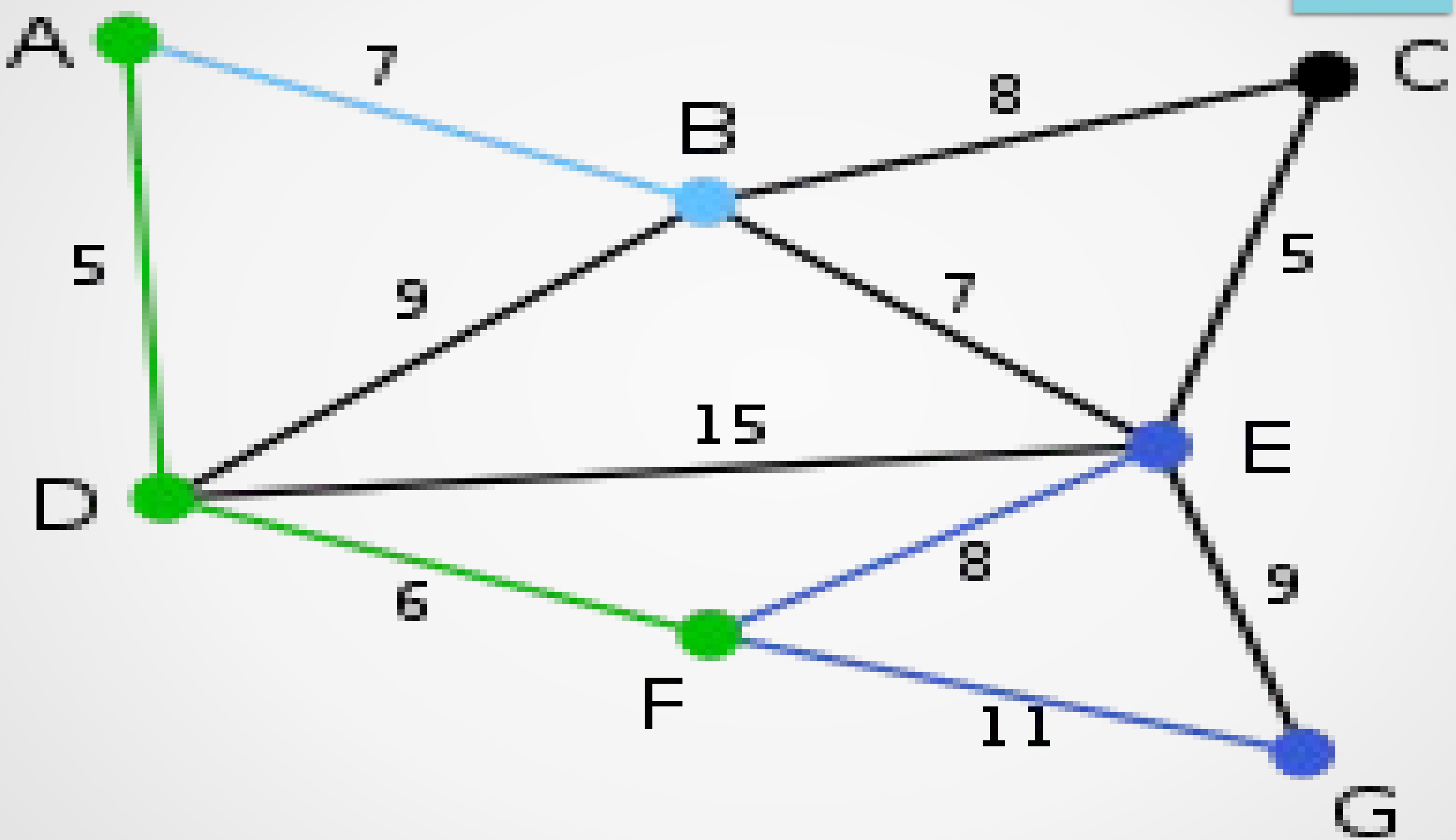
Prim's Algorithm: Example



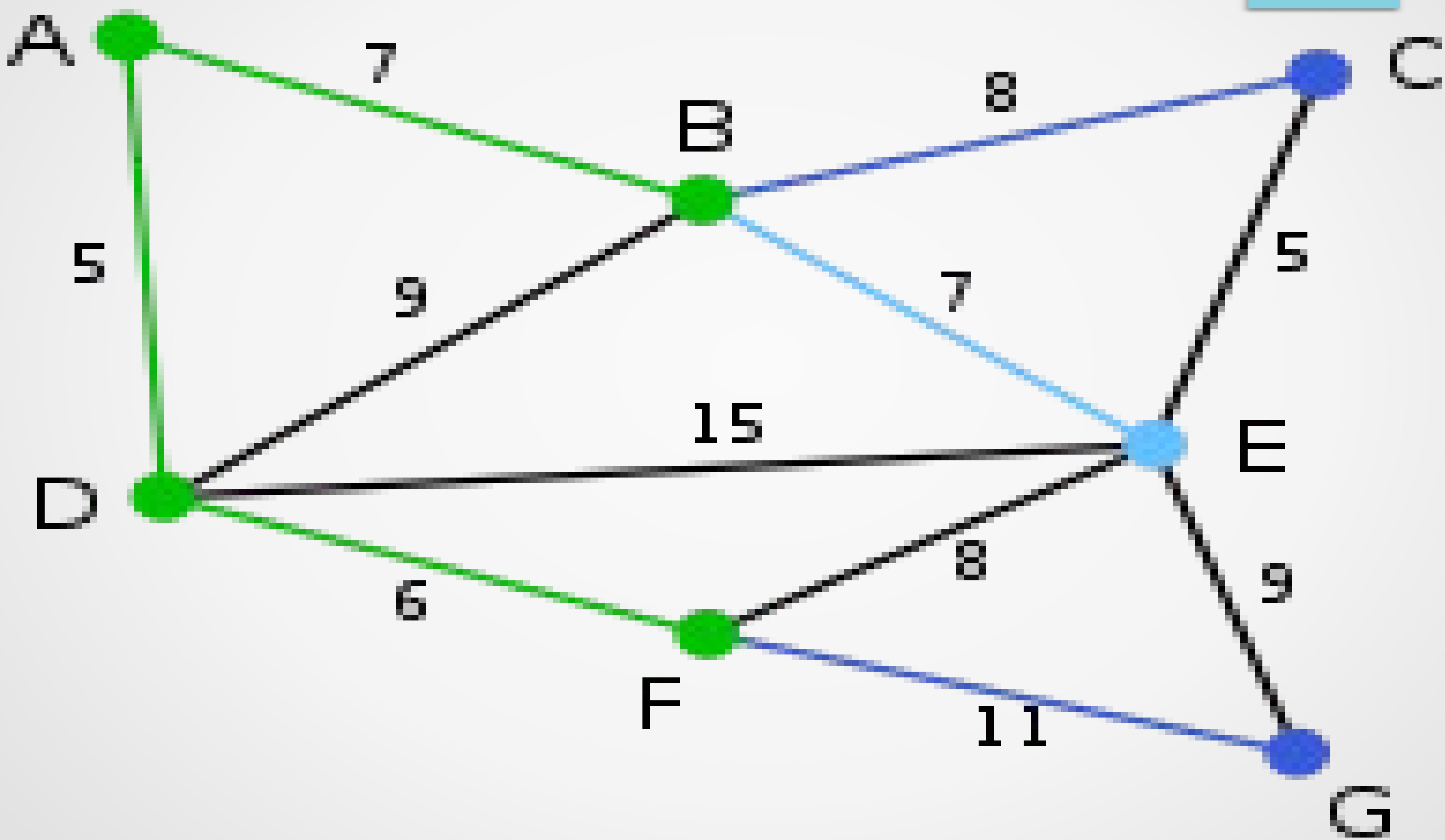
Prim's Algorithm: Example



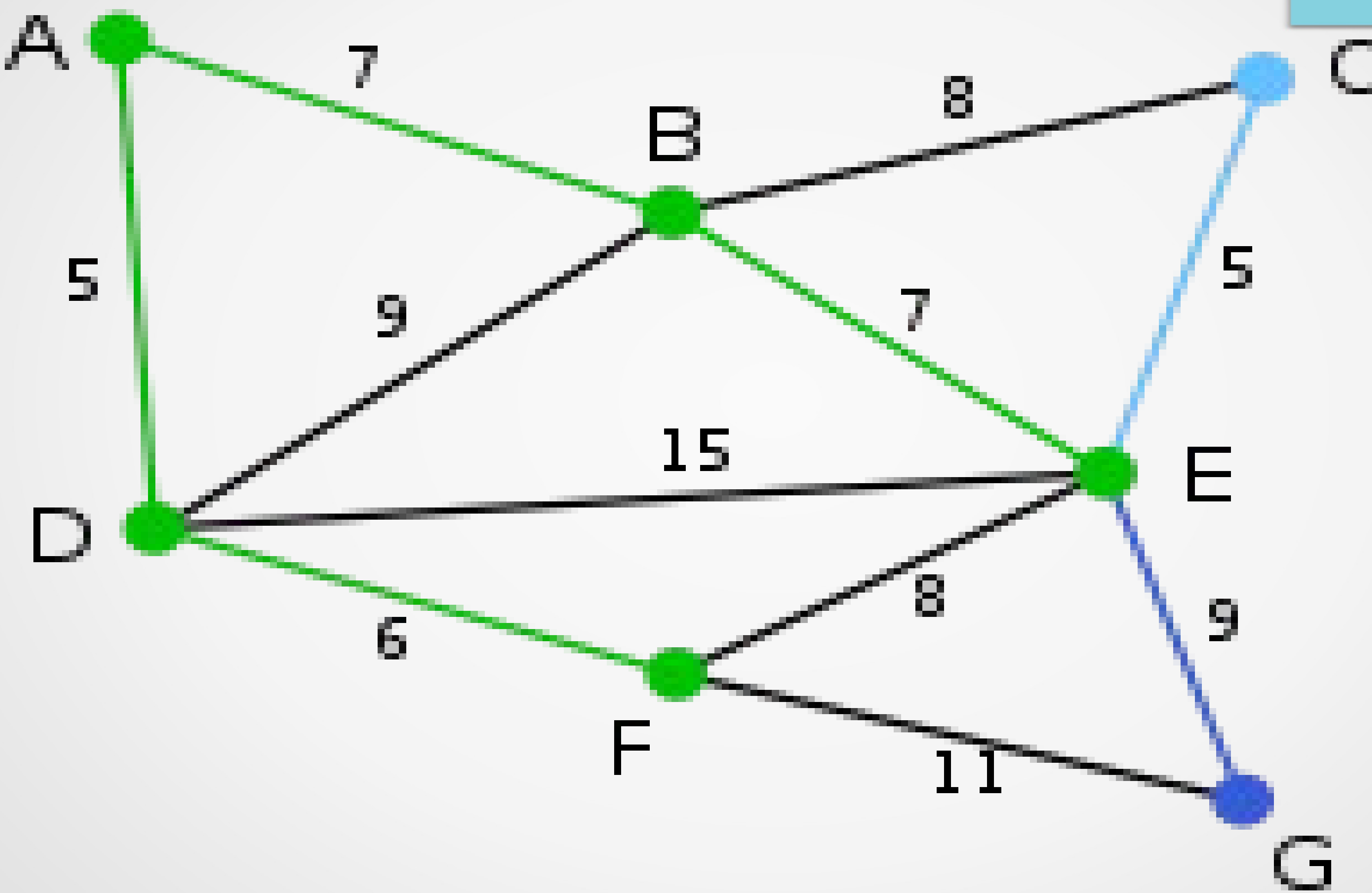
Prim's Algorithm: Example



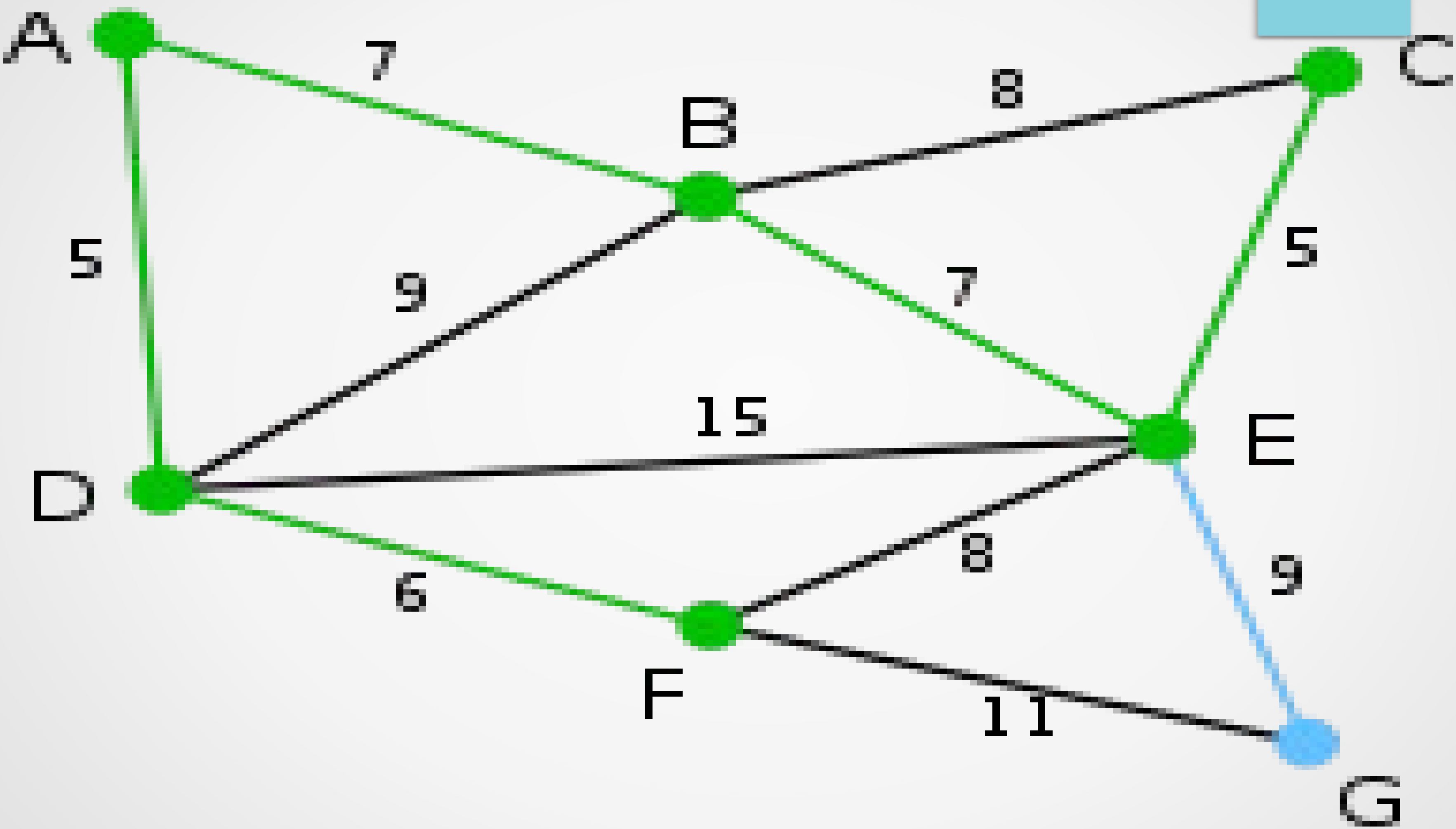
Prim's Algorithm: Example



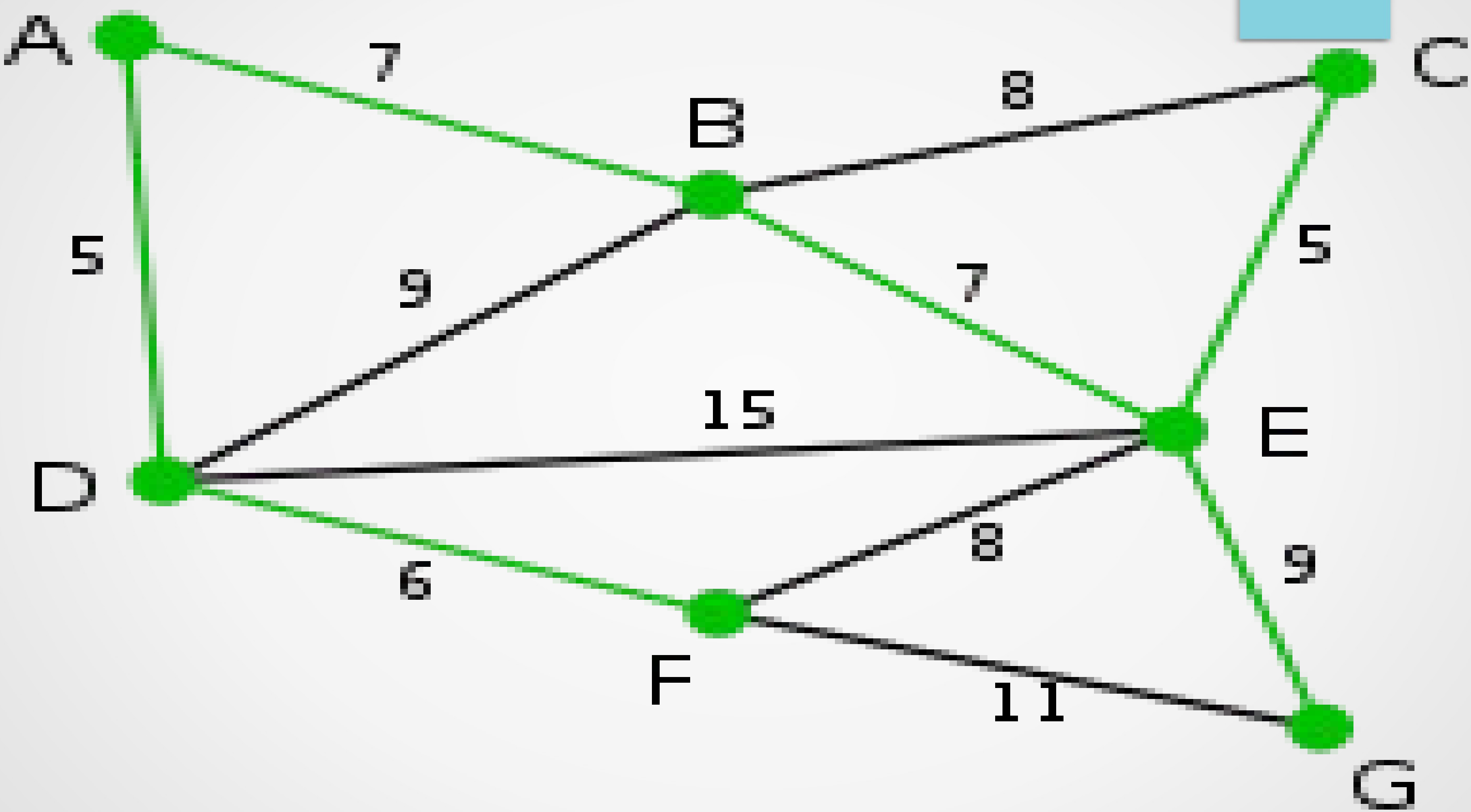
Prim's Algorithm: Example



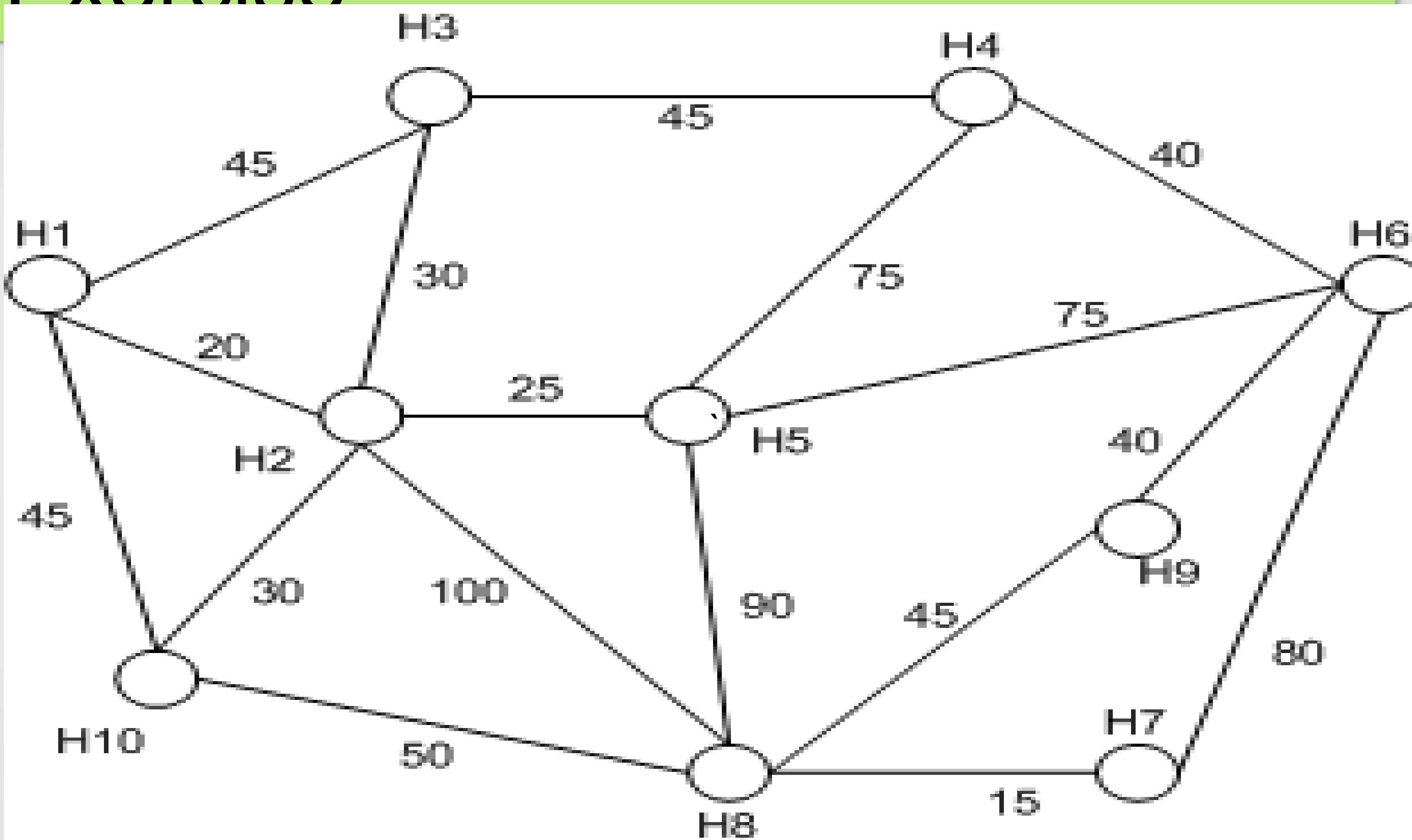
Prim's Algorithm: Example



Prim's Algorithm: Example



Exercise



H3
H4
45
45
40
H1
H6
30
75
75
20
25
H5
H2
40
45
H9
30
100
90
45
80
H7
H10
50
15
H8

Baruvka's Algorithm

- Like Kruskal's this grows many clouds at once
- Initially every vertex is a component C_i
- For each Ci find the smallest weight edge (v,u) in E and add to C_i such that
  - v is in C_i, u is not in C_i
  - Add e to T
- Each iteration reduces the number of components by half

Baruvka's Algorithm

- Algorithm BaruvkaMST(G)

  T ← V {just the vertices of G}

  while T has fewer than n-1 edges do

  for each connected component C in T do

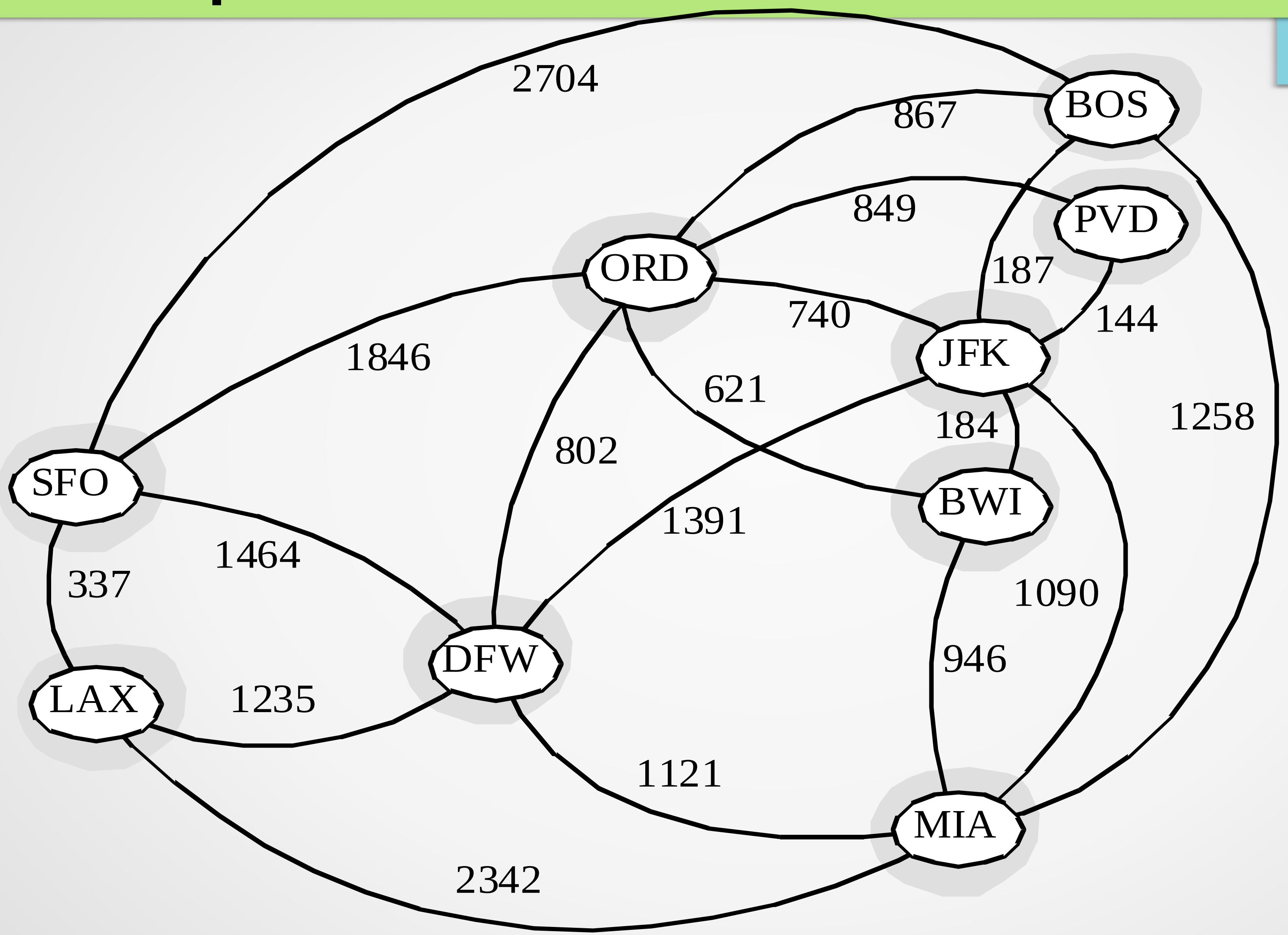
  Let edge e be the smallest-weight edge from C to another component in T.

  if e is not already in T then

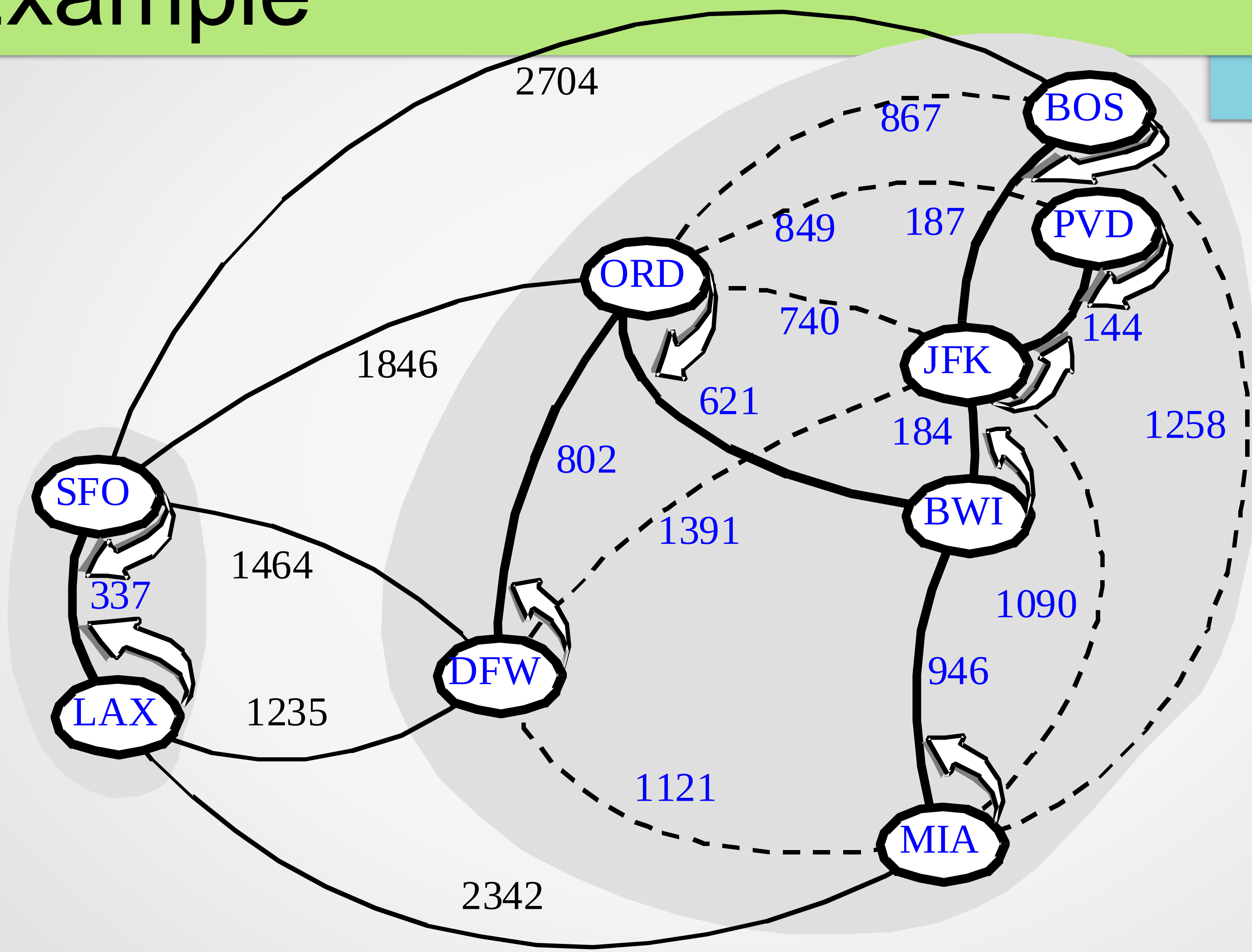
  Add edge e to T

  return T

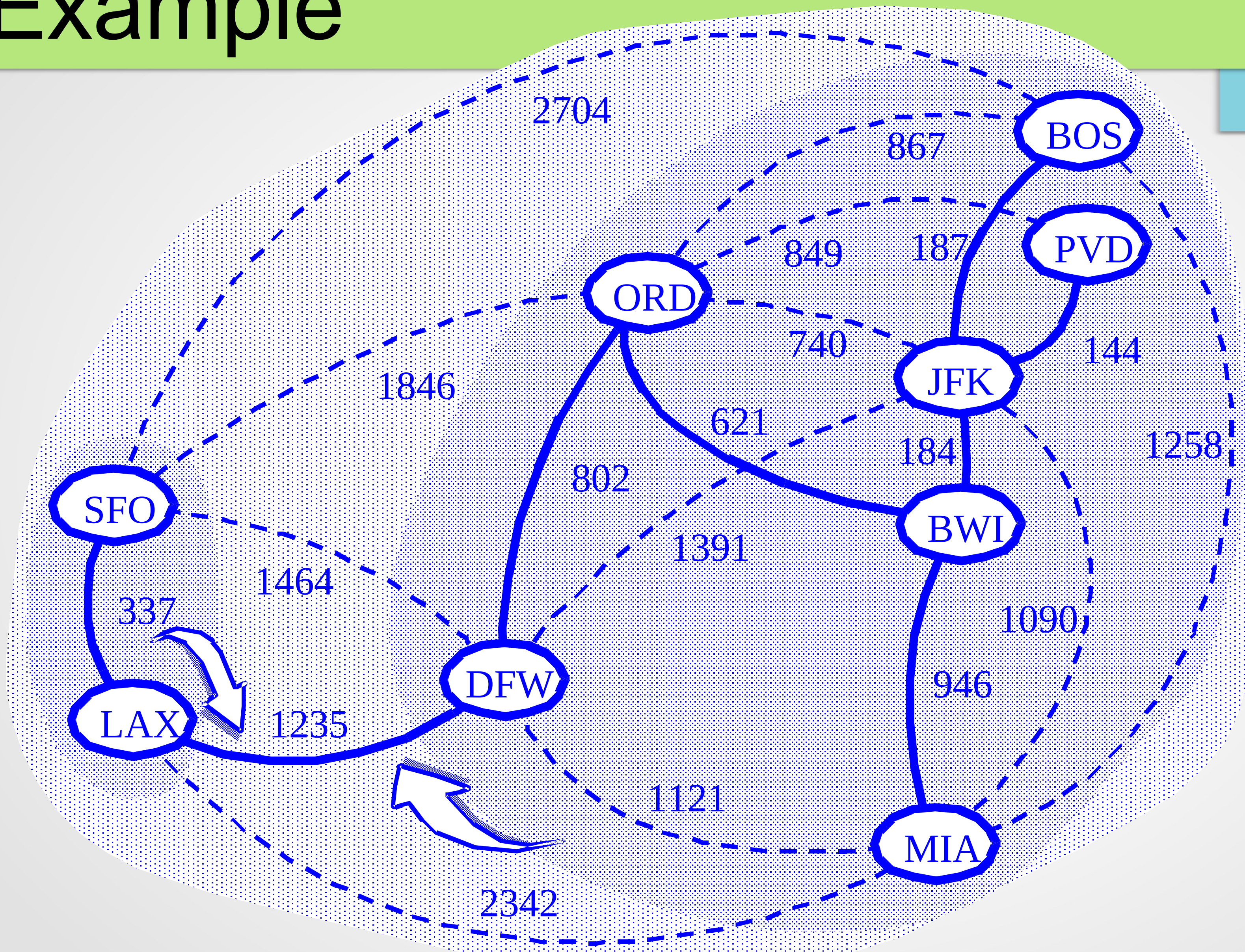
Example


2704
867
BOS
849
PVD
ORD
187
740
144
JFK
1846
621
1258
184
802
SFO
BWI
1391
1464
337
1090
DFW
946
LAX
1235
1121
MIA
2342

Example


2704
867
BOS
849
187
PVD
ORD
740
144
JFK
1846
621
1258
184
802
SFO
BWI
1391
1464
337
1090
DFW
946
LAX
1235
1121
MIA
2342

Example



2704
BOS
867
PVD
849
187
ORD
740
144
JFK
1846
621
1258
184
802
SFO
BWI
1391
1464
1090
337
DFW
946
LAX
1235
1121
MIA
2342

Properties of MSTs

- There may be multiple minimum spanning trees of same weight having minimum number of edges
- If each edge has a distinct weight then there will be only one, unique minimum spanning tree.
- For any cycle C in the graph, if the weight of an edge e of C is larger than the weights of all other edges of C, then this edge cannot belong to an MST.

Shortest Paths

- Single Source Single Destination Shortest Path
  - Given a source and destination find the path between source and destination with minimum cost
- Single Source Shortest Path
  - Find the shortest paths from a given source to all other nodes

Dijkstra's Algorithm

- Similar to Prim’s
- Let v be the source node
- Initialize for all u not v, d[u] to infinity and d[v]=0
- Remove from Q u with minimum d[u]
  - Add u and edge (u,v) to T
  - For all neighbors z of u, do relaxation by finding d[z] = d[u]+w(u,z), and update Q
    - d[z] = min(d[z],d(u,z)+d[u])
- Terminate when Q empty

Shortest Paths

- Single Source Single Destination Shortest Path
  - Given a source and destination find the path between source and destination with minimum cost
- Single Source Shortest Path
  - Find the shortest paths from a given source to all other nodes

Dijkstra’s Algorithm

- Complexity
  - $O(|E|+|V|\log|V|)$
  - Cost of maintaining priority queue plays a major role
- Does not work if there are negative weighted cycles

Exercise

- Find the shortest path from A to all other nodes