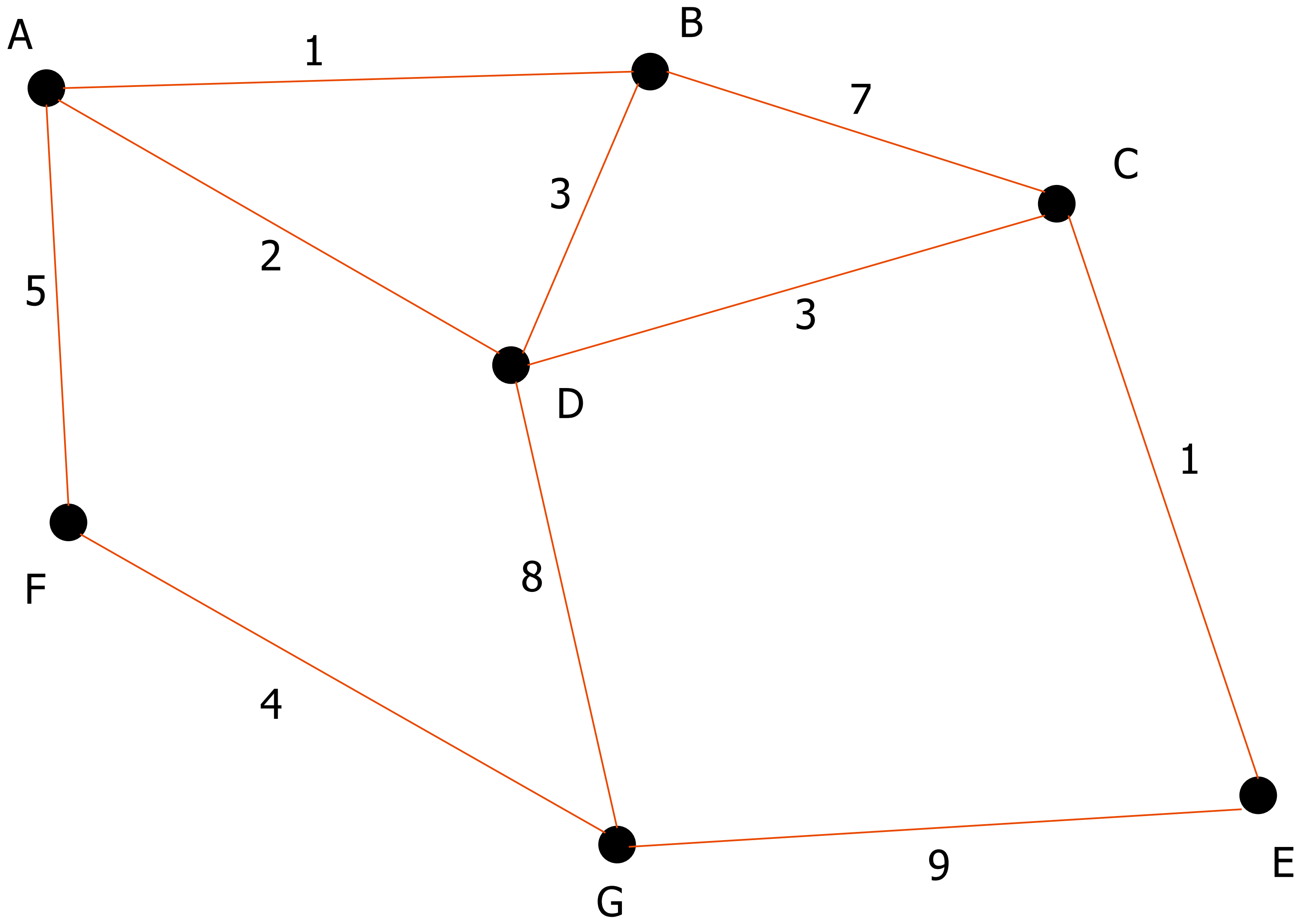
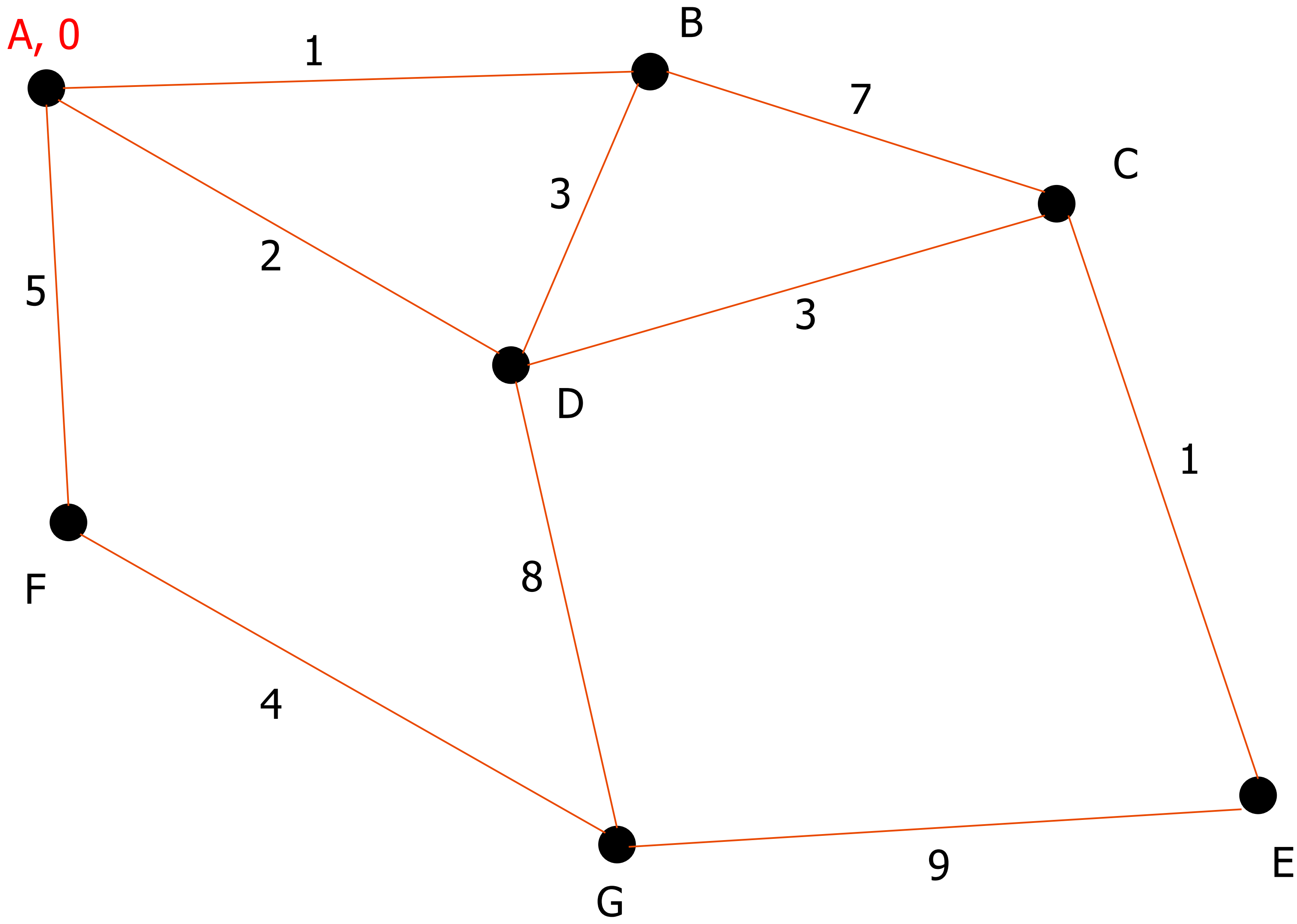
Dijkstra's Algorithm

SECTION 1.3

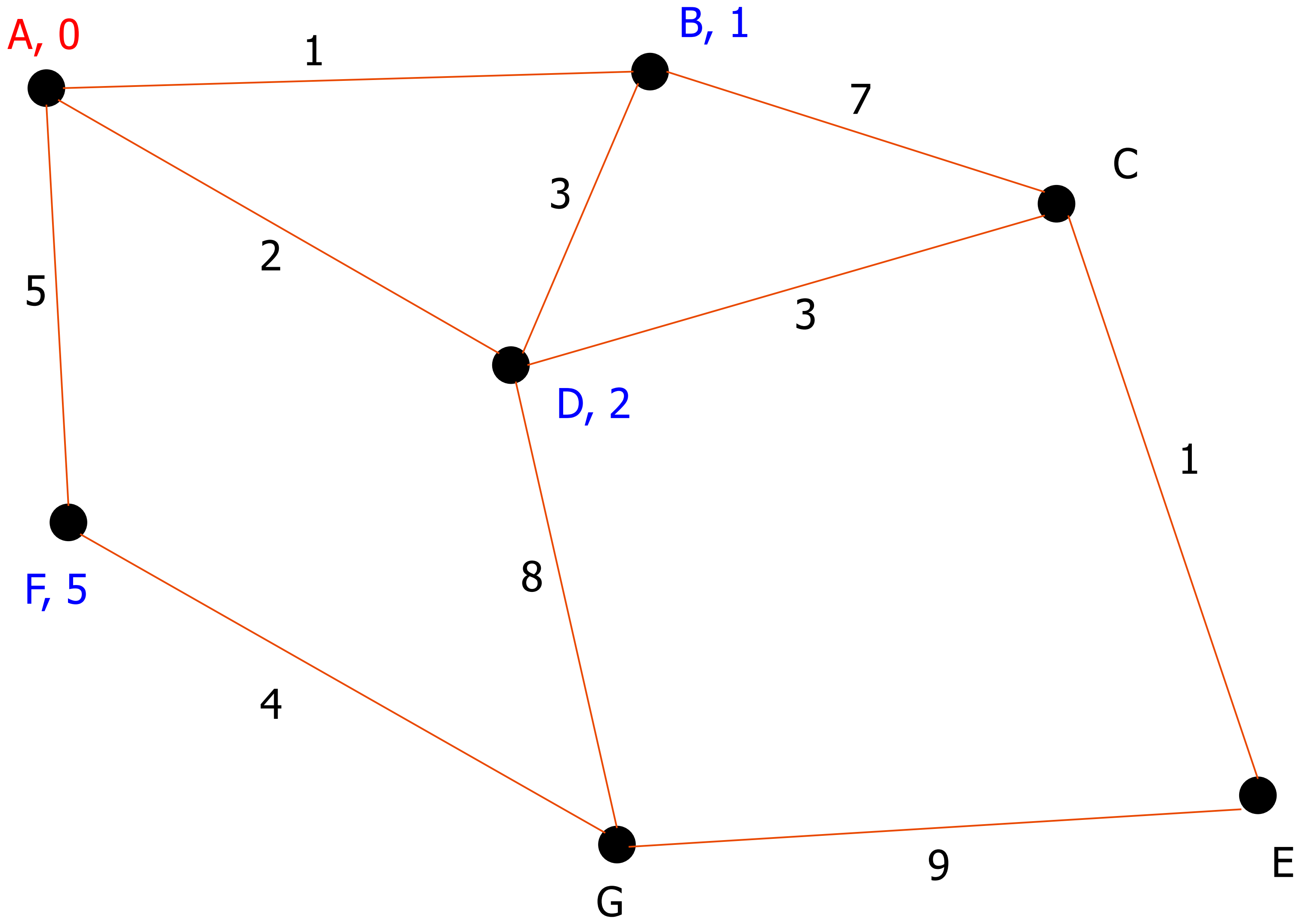
Planning shortest paths – Dijkstra's Algorithm



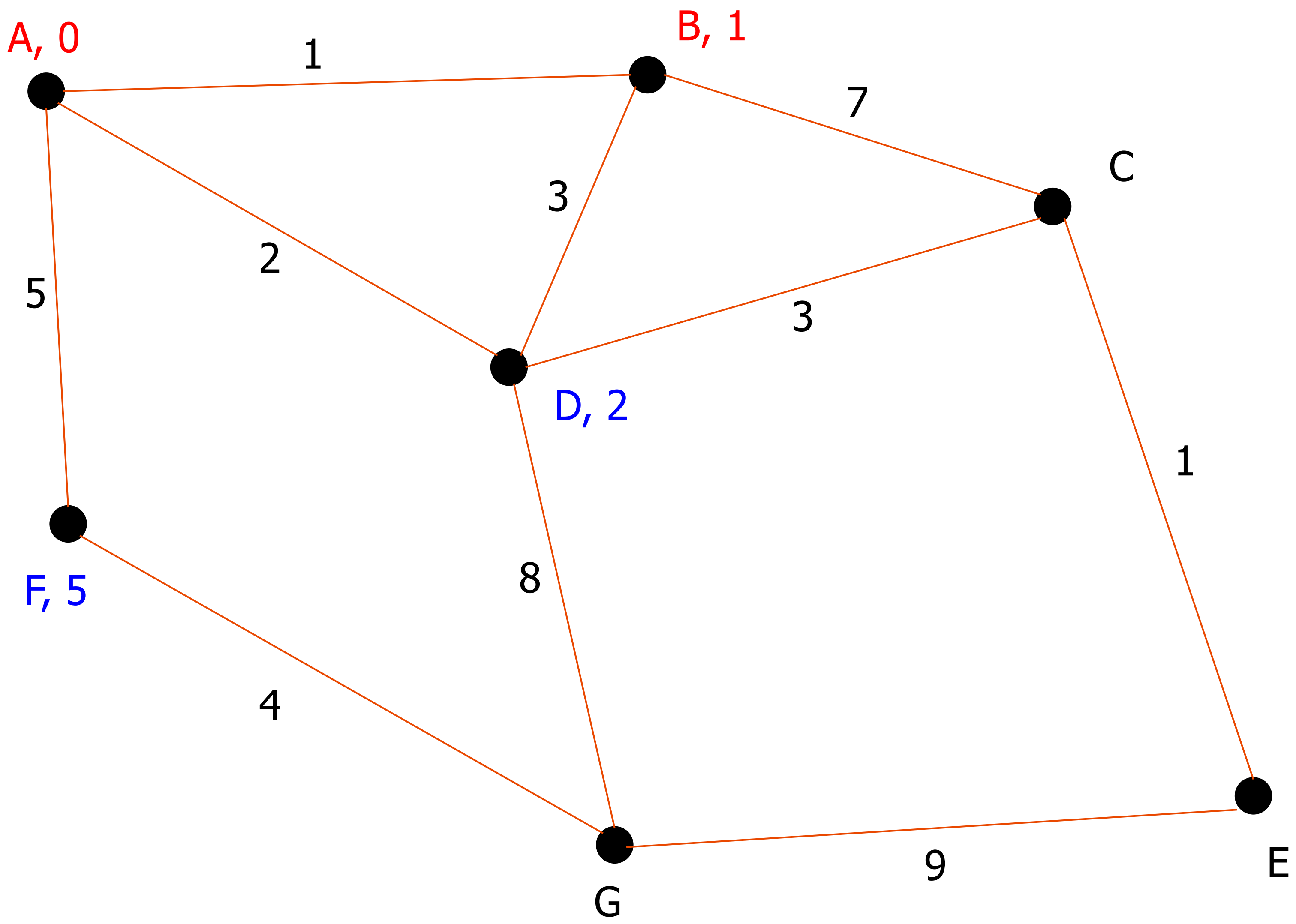
Planning shortest paths – Dijkstra's Algorithm



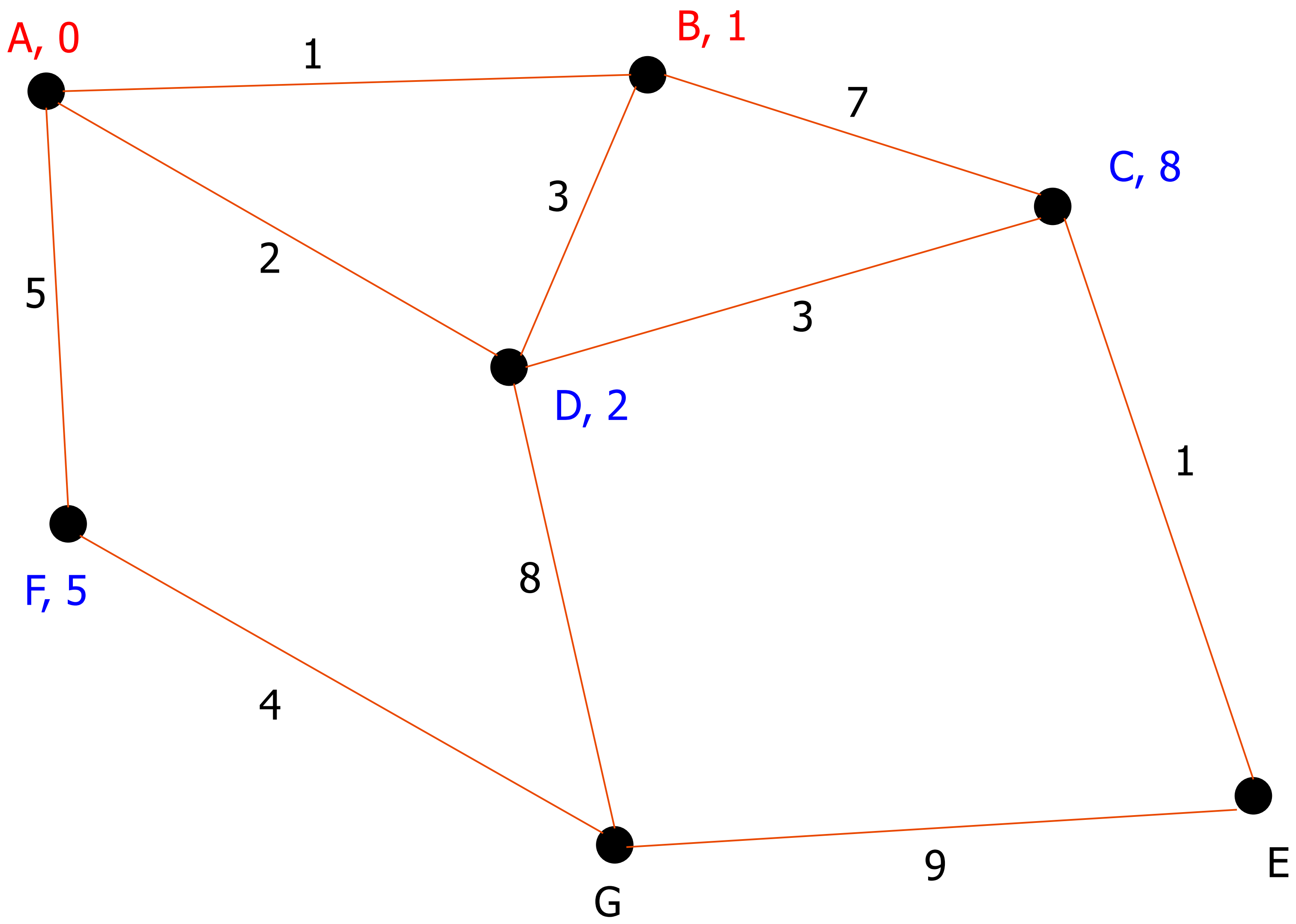
Planning shortest paths – Dijkstra's Algorithm



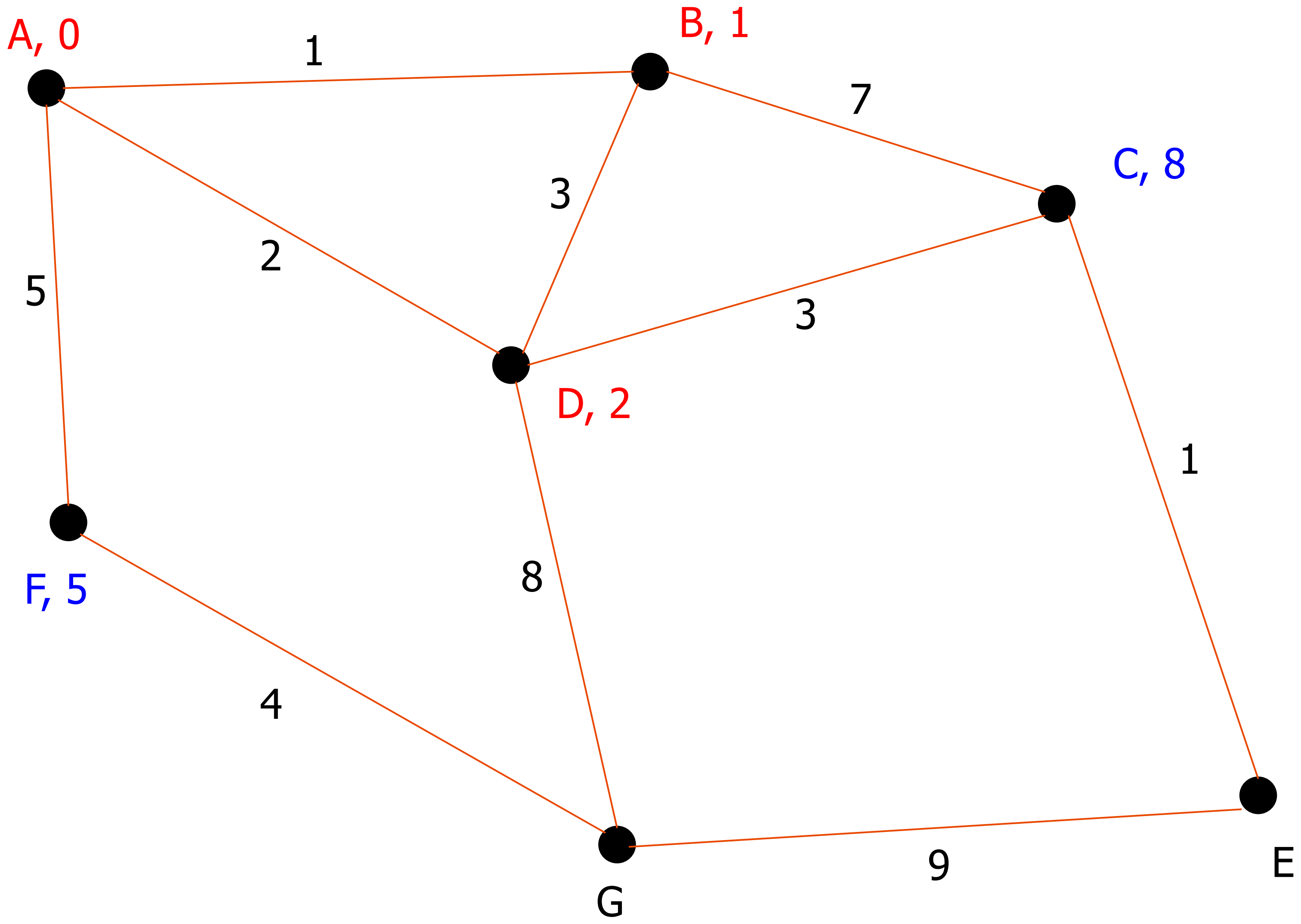
Planning shortest paths – Dijkstra's Algorithm



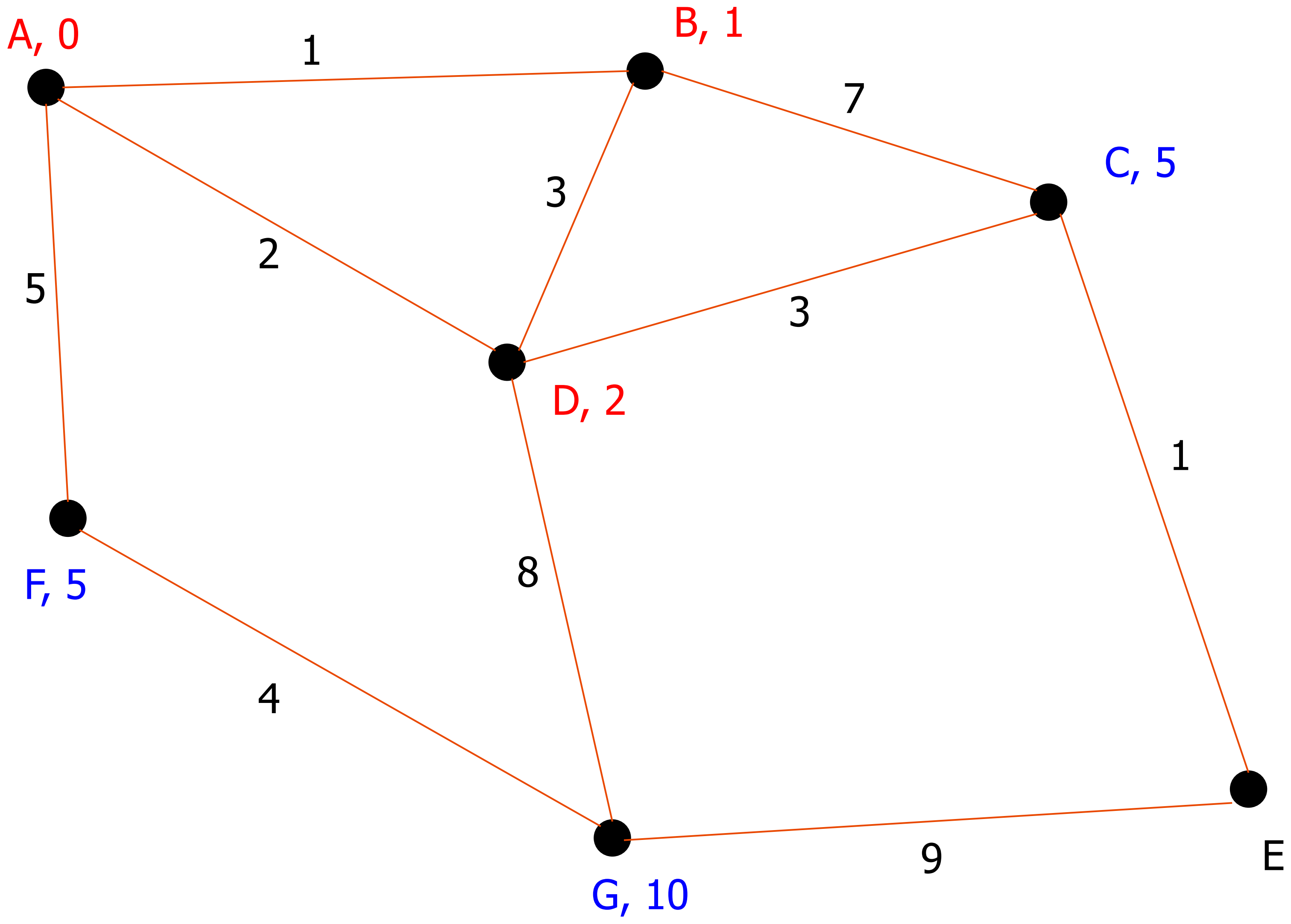
Planning shortest paths – Dijkstra’s Algorithm



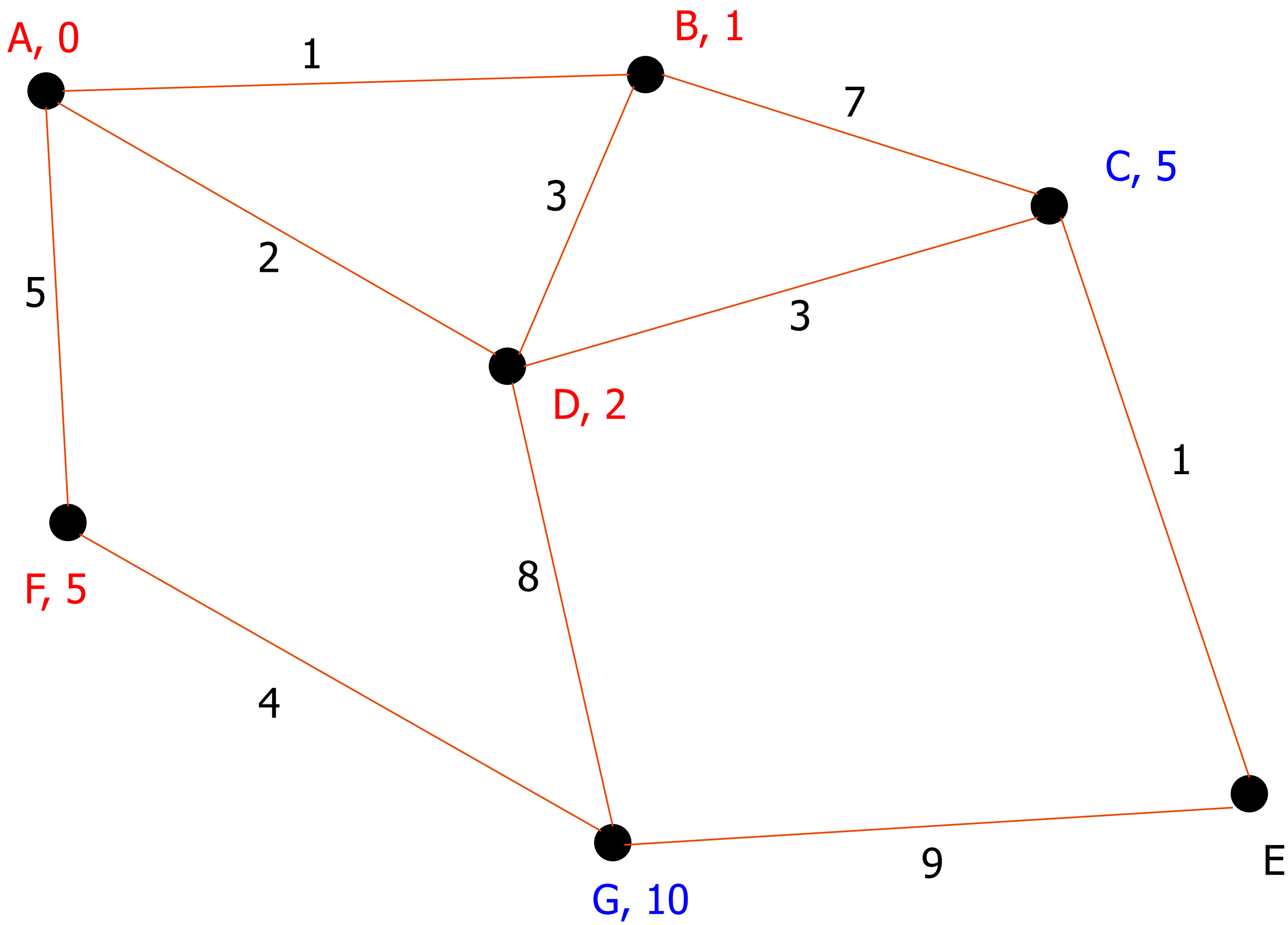
Planning shortest paths – Dijkstra's Algorithm



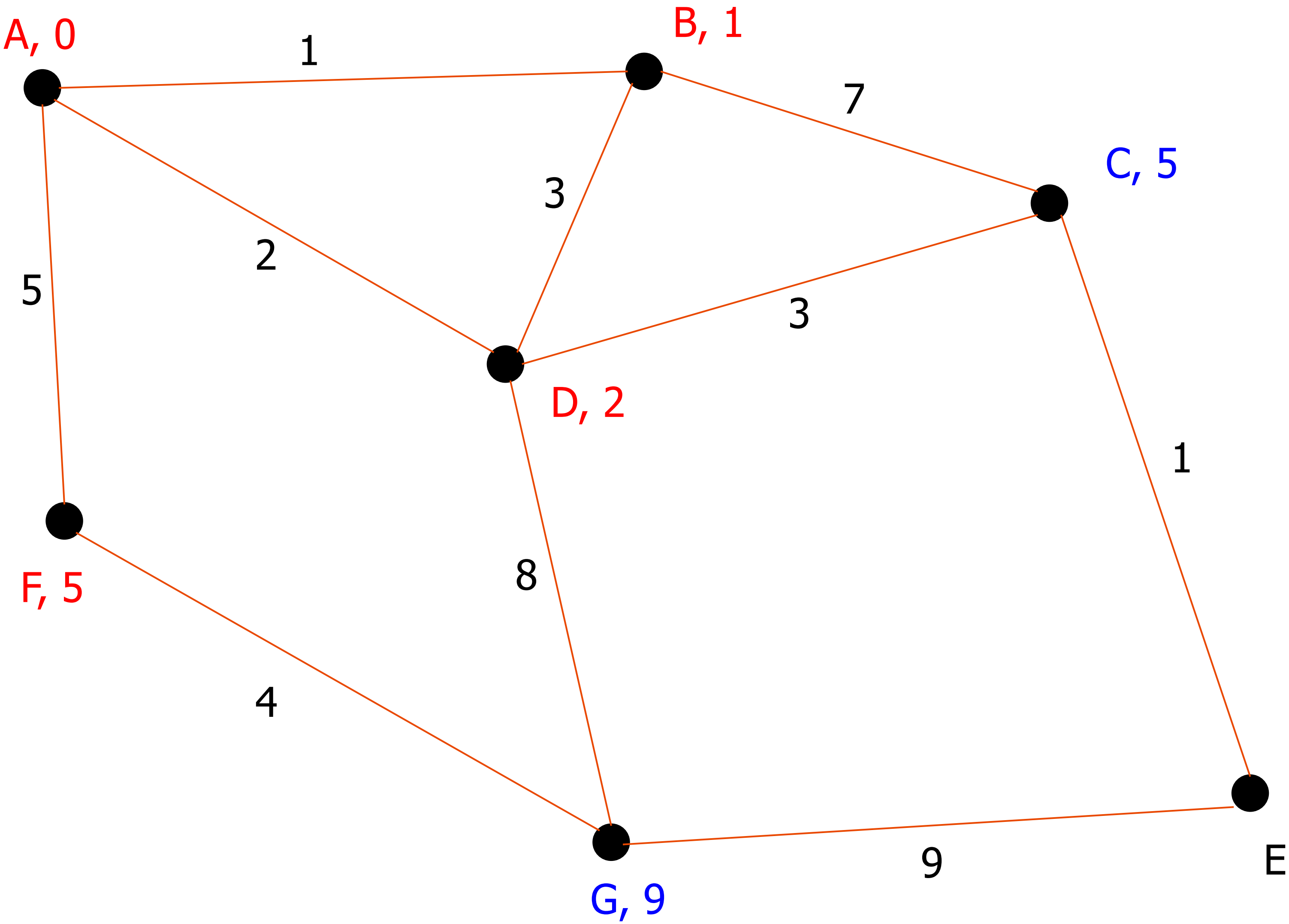
Planning shortest paths – Dijkstra's Algorithm



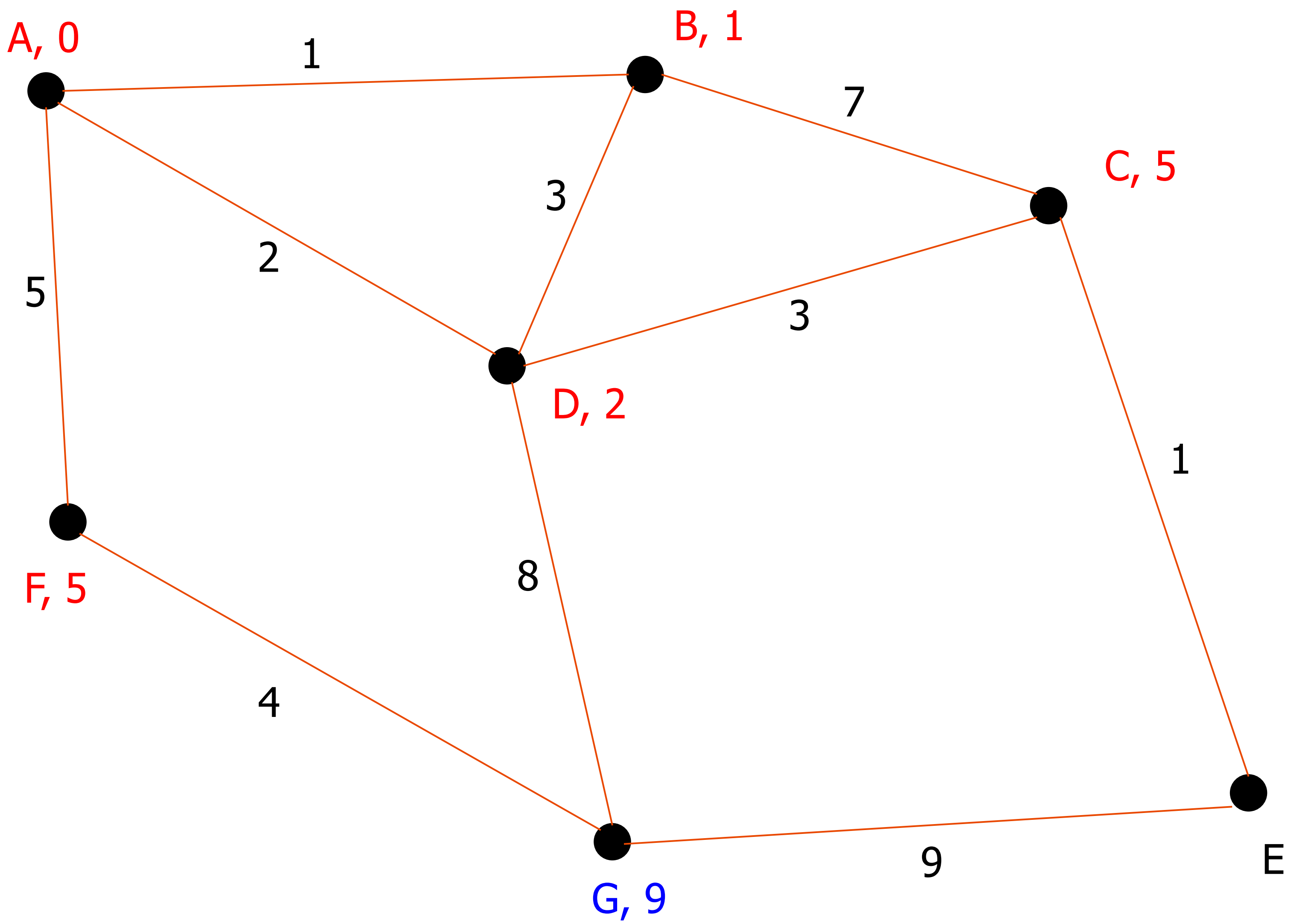
Planning shortest paths – Dijkstra's Algorithm



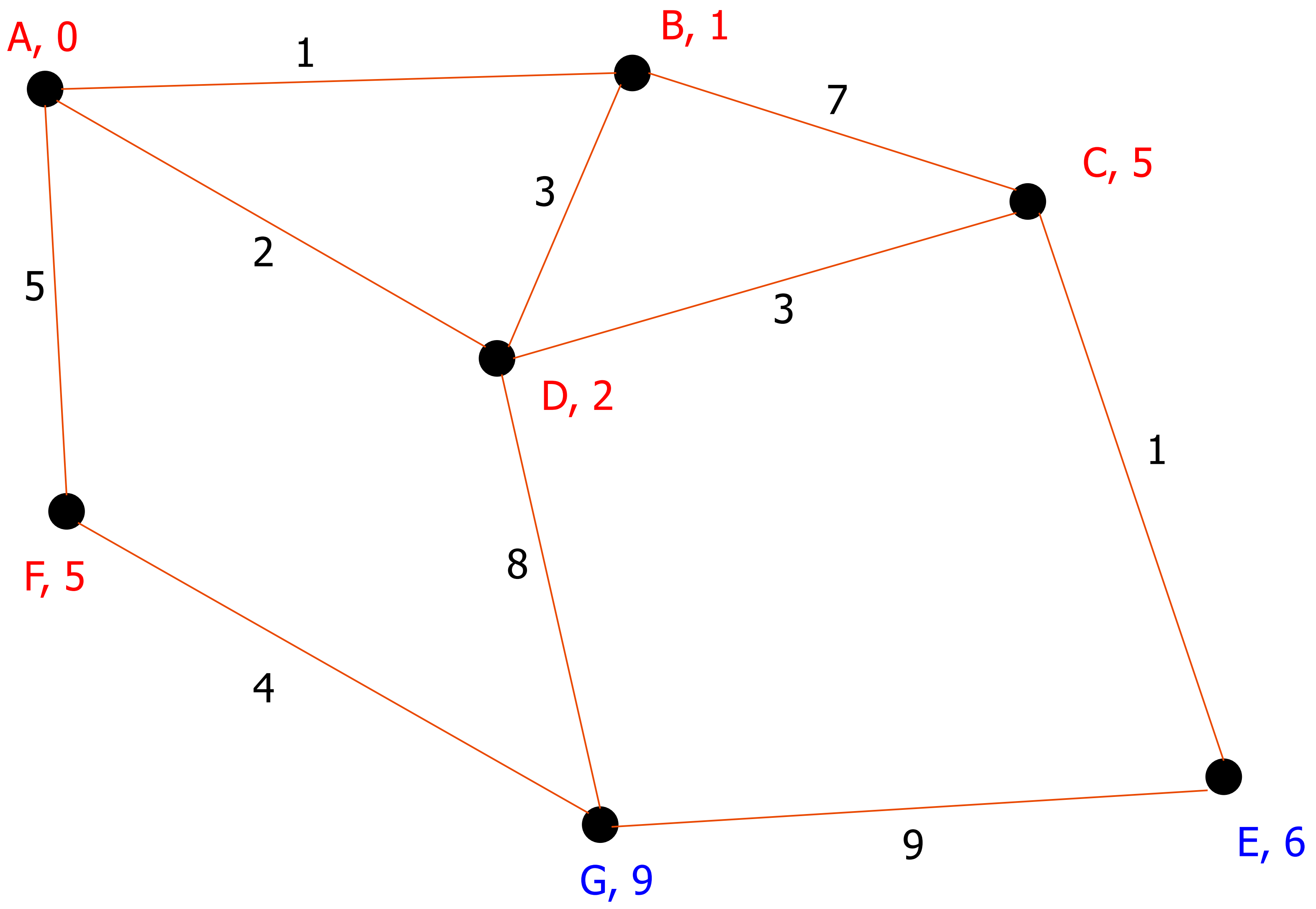
Planning shortest paths – Dijkstra's Algorithm



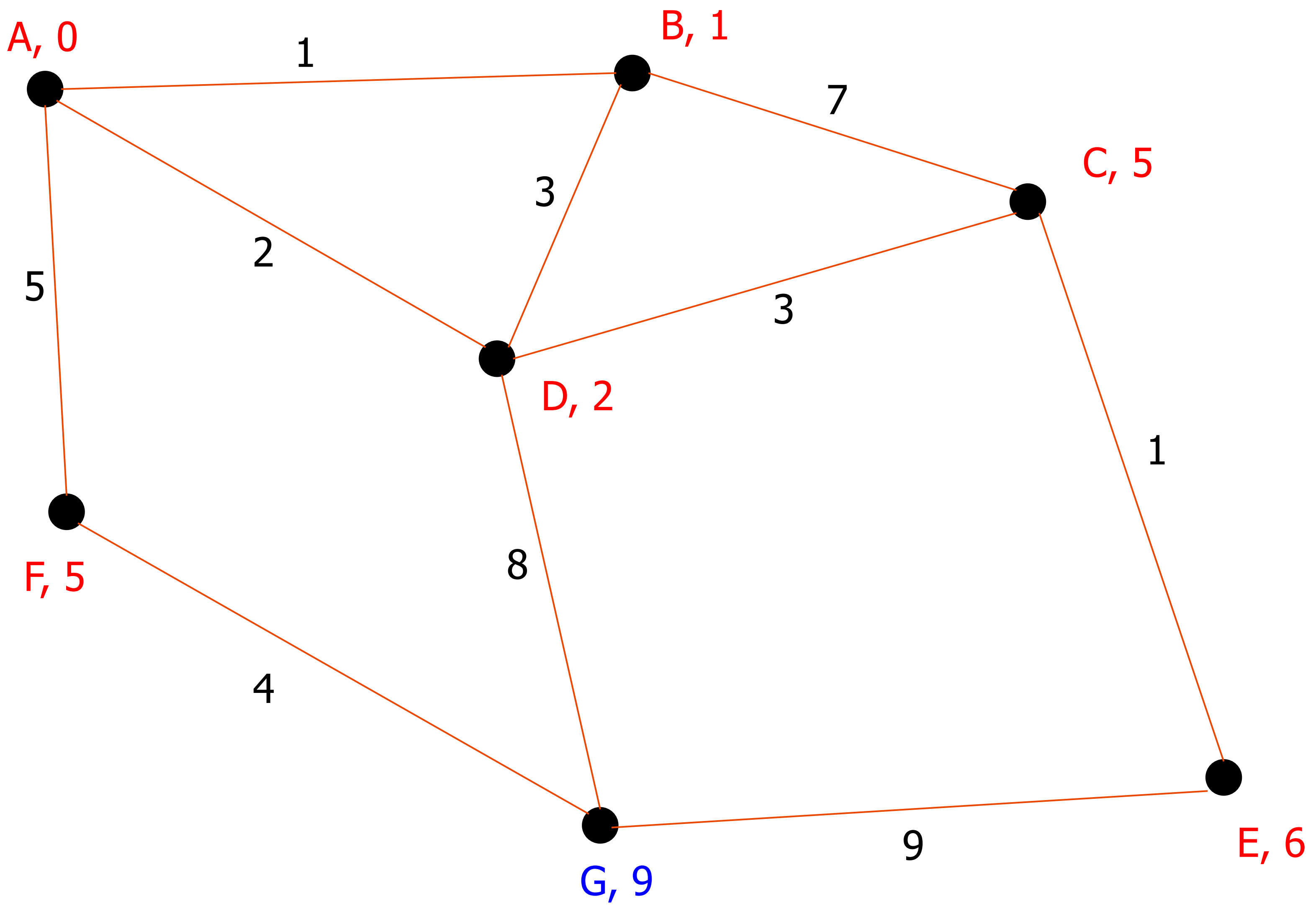
Planning shortest paths – Dijkstra's Algorithm



Planning shortest paths – Dijkstra’s Algorithm



Planning shortest paths – Dijkstra's Algorithm



Dijkstra's algorithm – pseudo code

- For each node n in the graph
  - n.distance = Infinity
- Create an empty list.
- start.distance = 0, add start to list.
- While list not empty
  - Let current = node in the list with the smallest distance, remove current from list
  - For each node, n that is adjacent to current

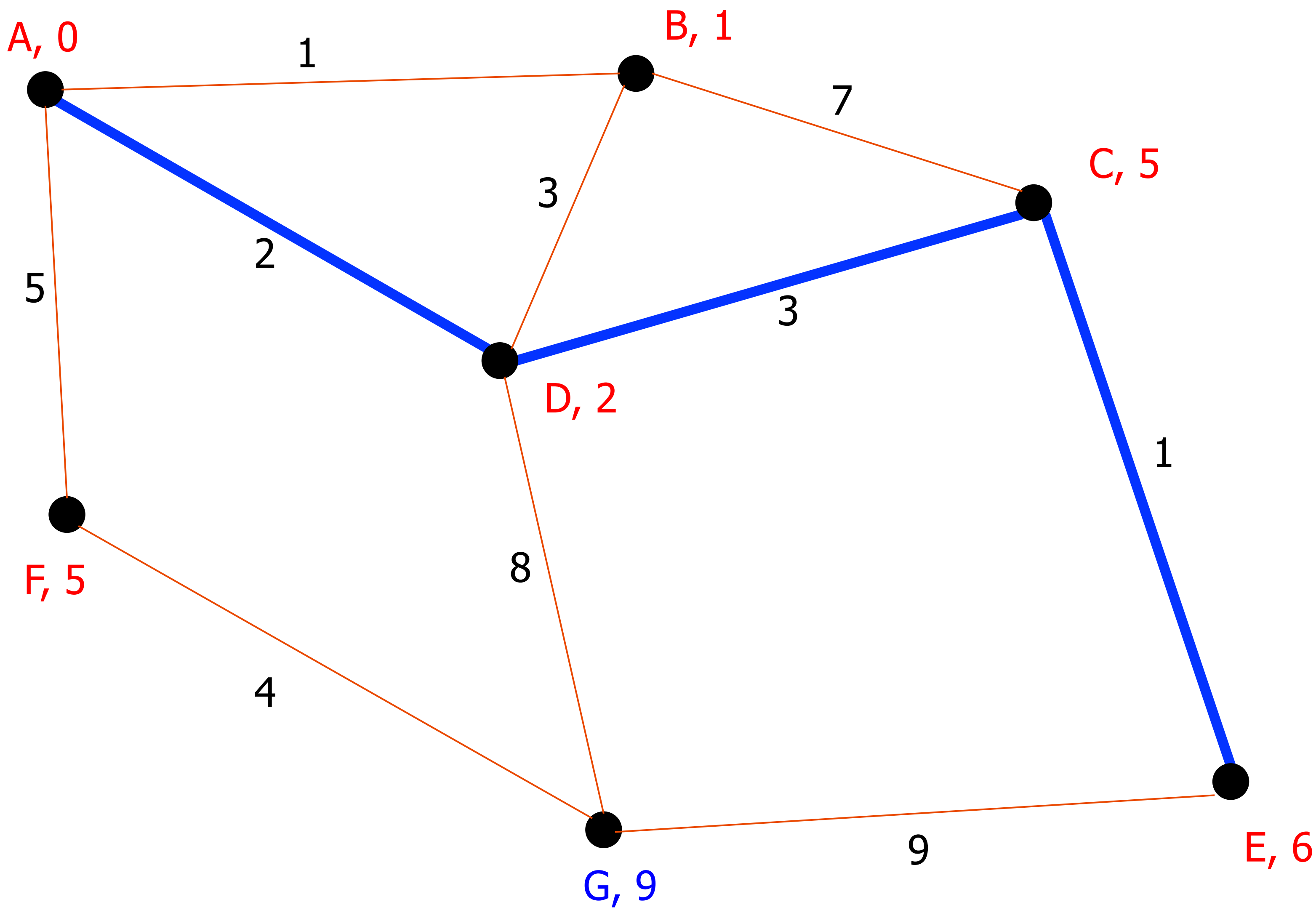
    If n.distance > current.distance + length of edge from n to current

      n.distance = current.distance + length of edge from n to current

      n.parent = current

      add n to list if it isn't there already

Planning shortest paths – Dijkstra's Algorithm



Computational Complexity of Dijkstra's algorithm

- A naive version of Dijkstra's algorithm can be implemented with a computational complexity that grows quadratically with the number of nodes.

$$\mathcal{O}(|\mathbf{V}|^{\mathbf{2}}) \tag{1}$$

- By keeping the list of nodes sorted using a clever data structure known as a priority queue the computational complexity can be reduced to something that grows more slowly

$$\mathcal{O}((|\mathbf{E}| + |\mathbf{V}|)\log(|\mathbf{V}|)) \tag{2}$$

- $|\mathbf{V}|$ denotes the number of nodes in the graph and $|\mathbf{E}|$ denotes the number of edges