Shortest Path Problem

Paniz Abedin, Assistant Professor of Computer Science,
Florida Polytechnic University

Shortest Path Problem

Input: a weighted graph, a source node and a goal node

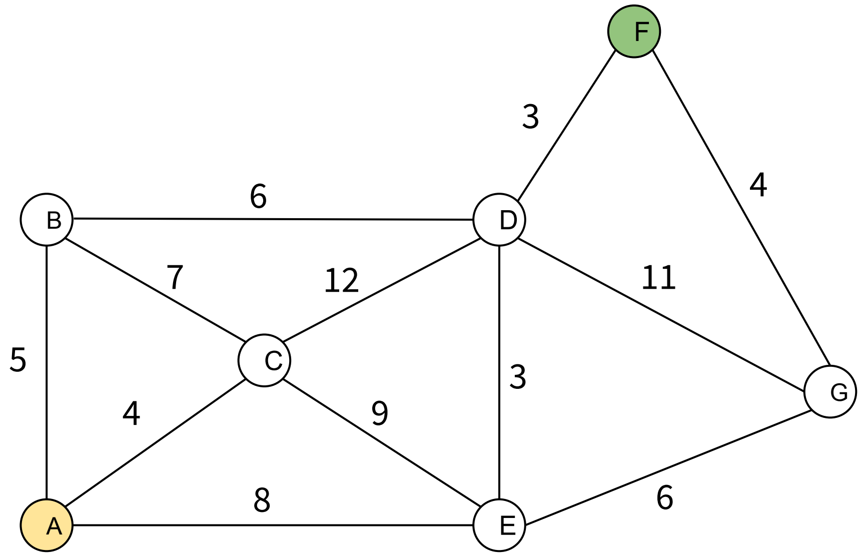
Output: The least cost path from the source to the goal

Dijkstra's Algorithm

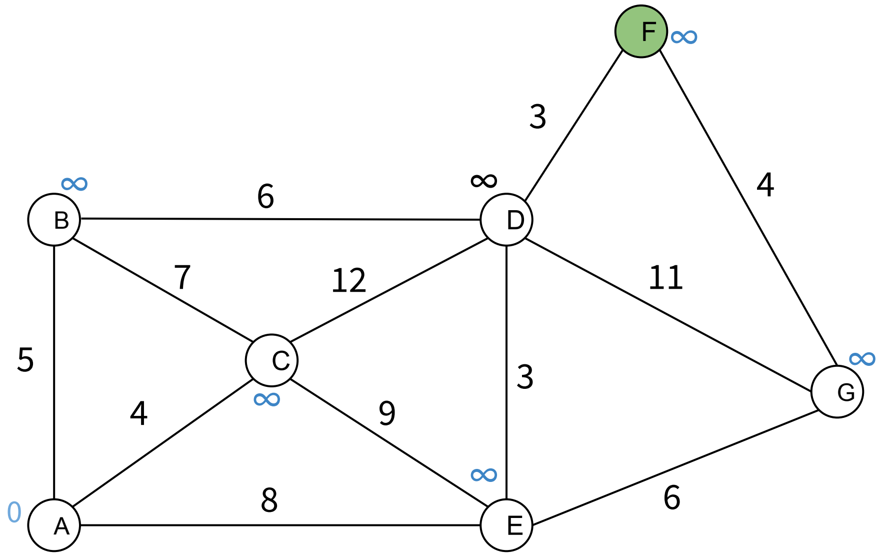
A greedy algorithm to solve shortest path problem

1. Assign 0 to source and infinity to all other nodes
2. Keep a set of visited nodes
3. For the current node consider all of its unvisited neighbors and calculate "distance to the current node" + " distance from current node to the neighbor". If it is better, update the value (relaxation step)
4. When we are done considering all neighbors of the current node, mark current node as visited
5. If the goal node is visited, we are done
6. Set the unvisited node marked with the smallest value as the next current node and repeat step 3

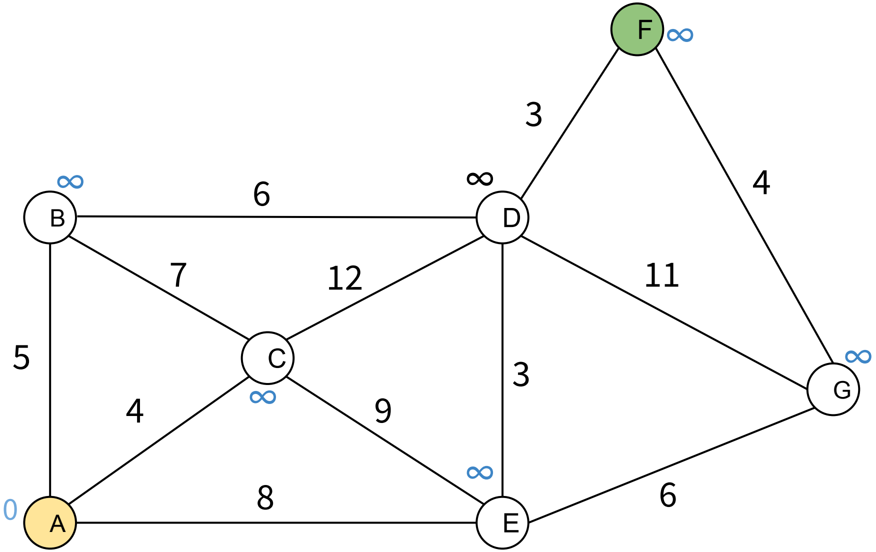
Dijkstra's Algorithm



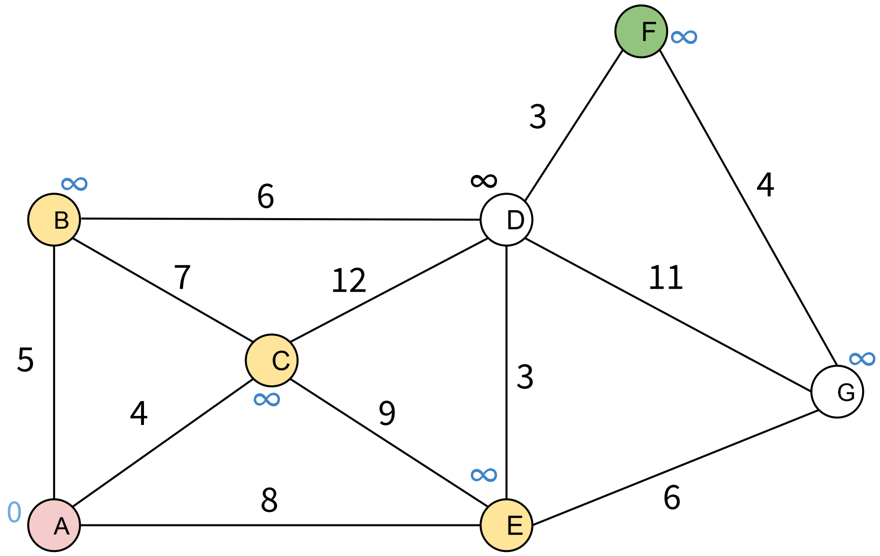
Dijkstra's Algorithm



Dijkstra’s Algorithm

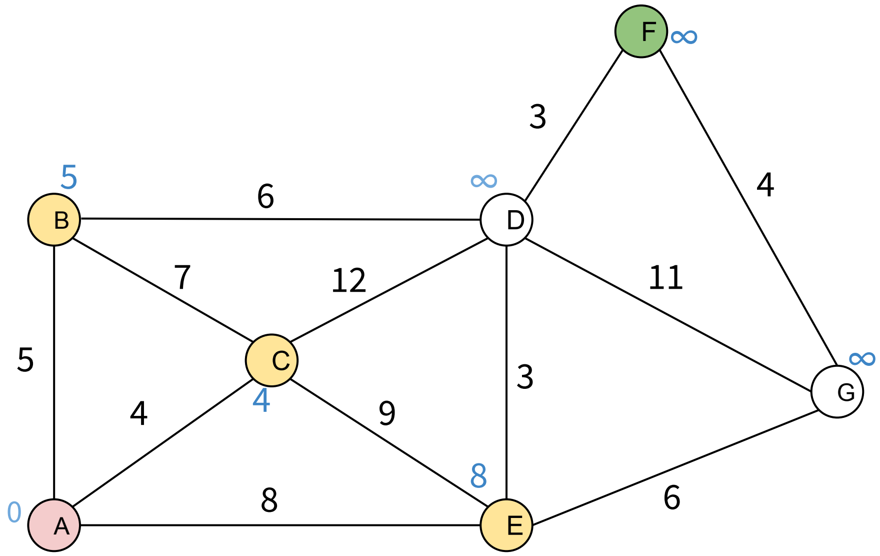


Dijkstra's Algorithm



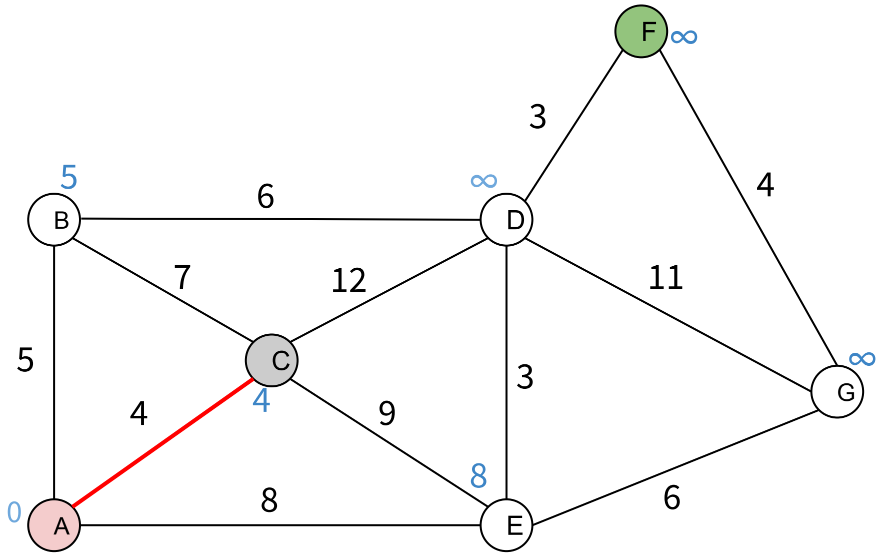
Visited: {A}

Dijkstra's Algorithm



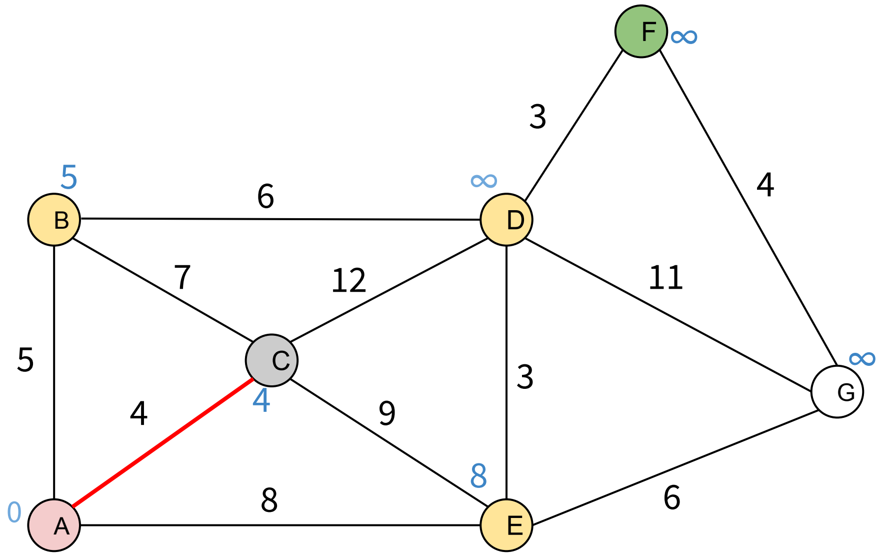
Visited: {A}

Dijkstra's Algorithm



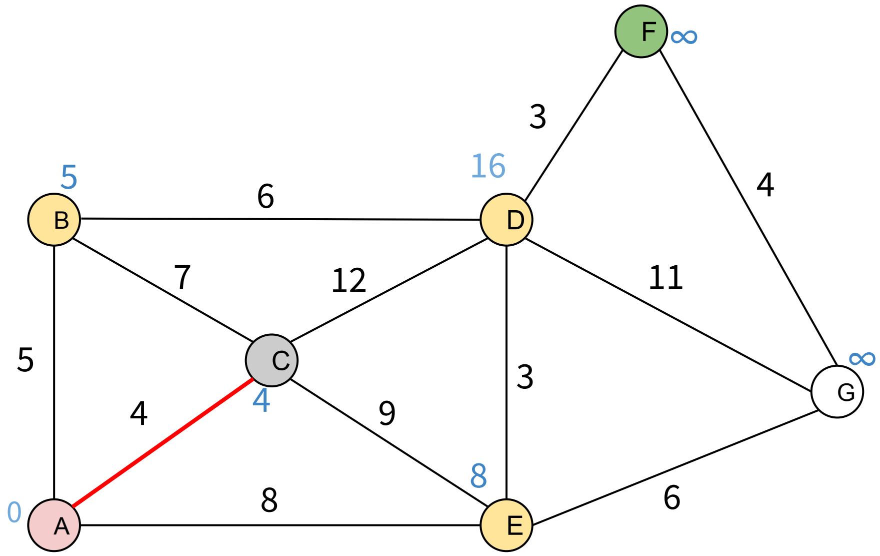
Visited: {A}

Dijkstra's Algorithm



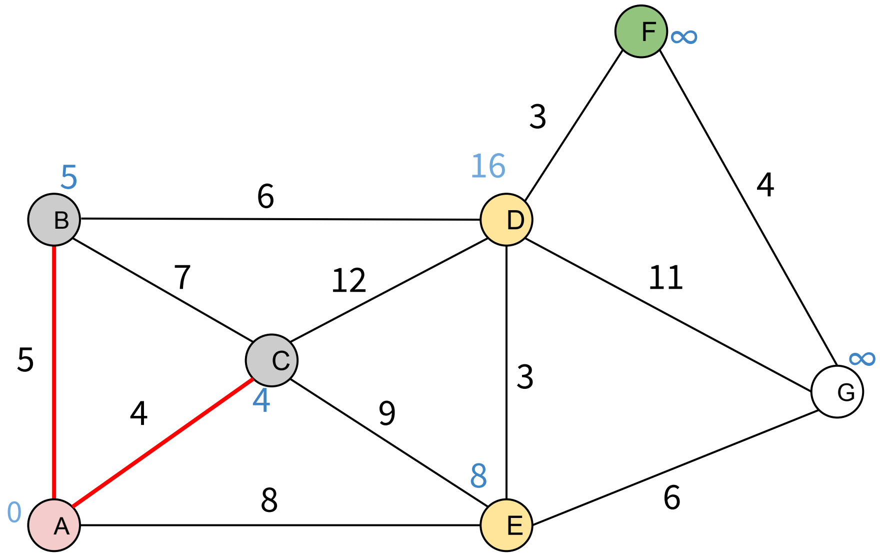
Visited: {A, C}

Dijkstra's Algorithm



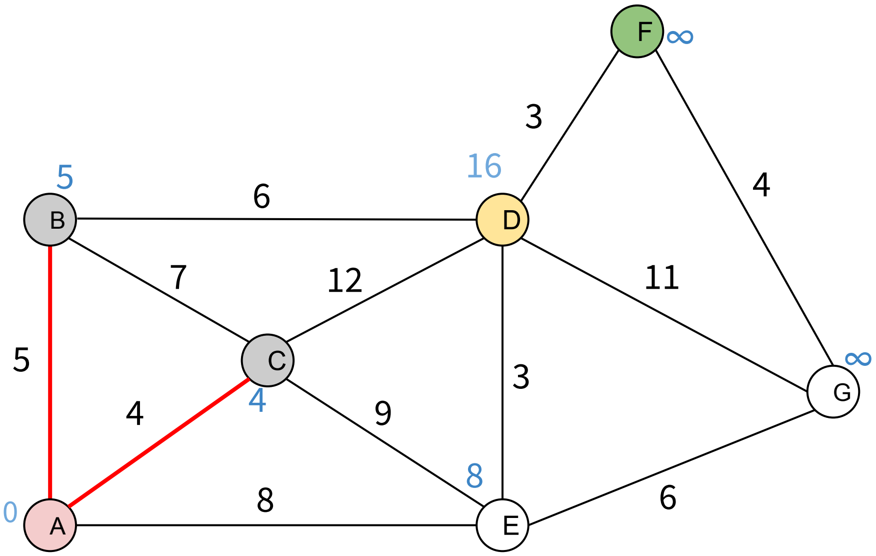
Visited: {A, C}

Dijkstra's Algorithm



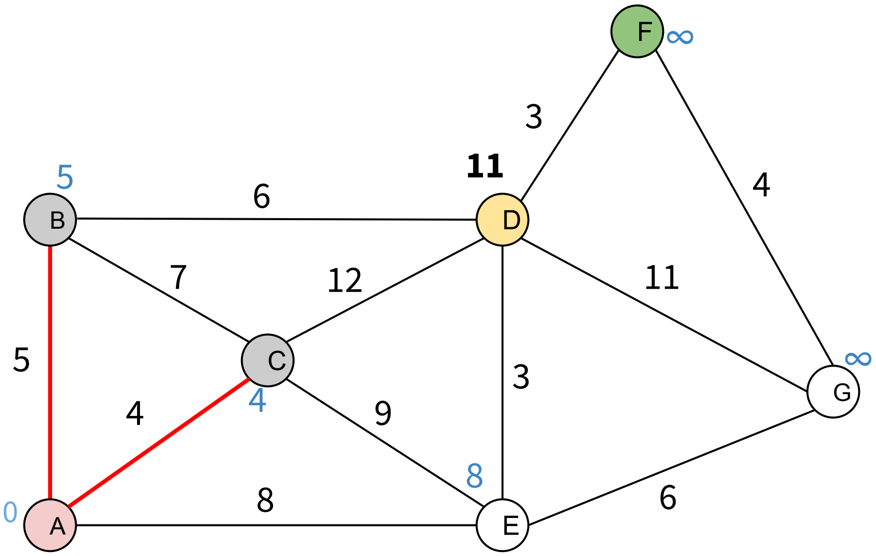
Visited: {A, C, B}

Dijkstra's Algorithm



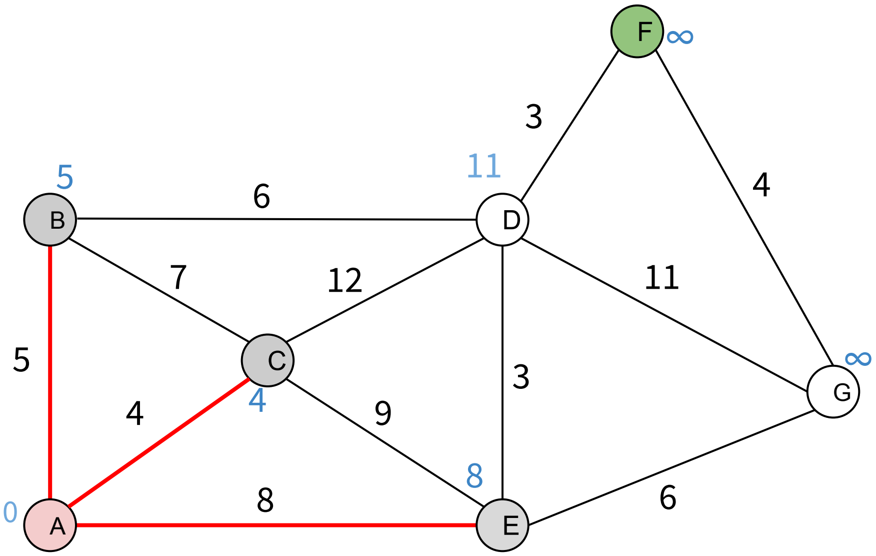
Visited: {A, C, B}

Dijkstra's Algorithm



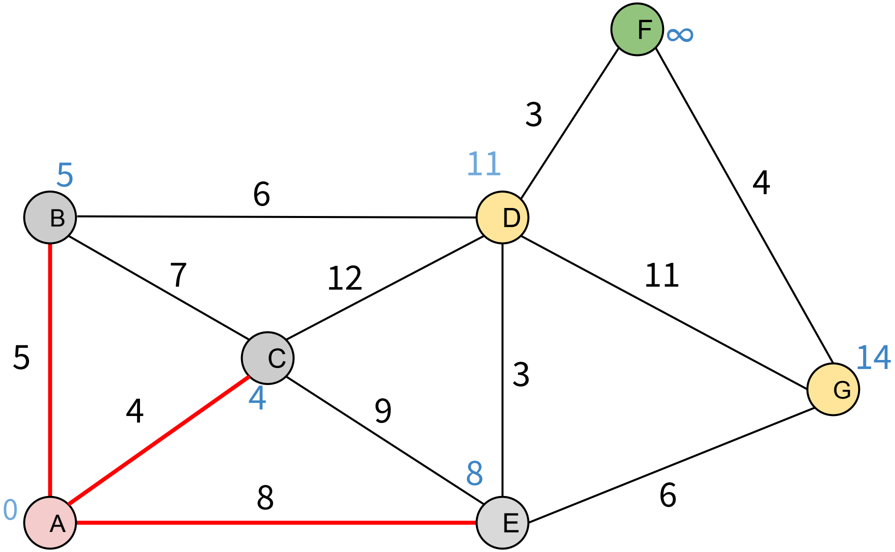
Visited: {A, C, B}

Dijkstra's Algorithm



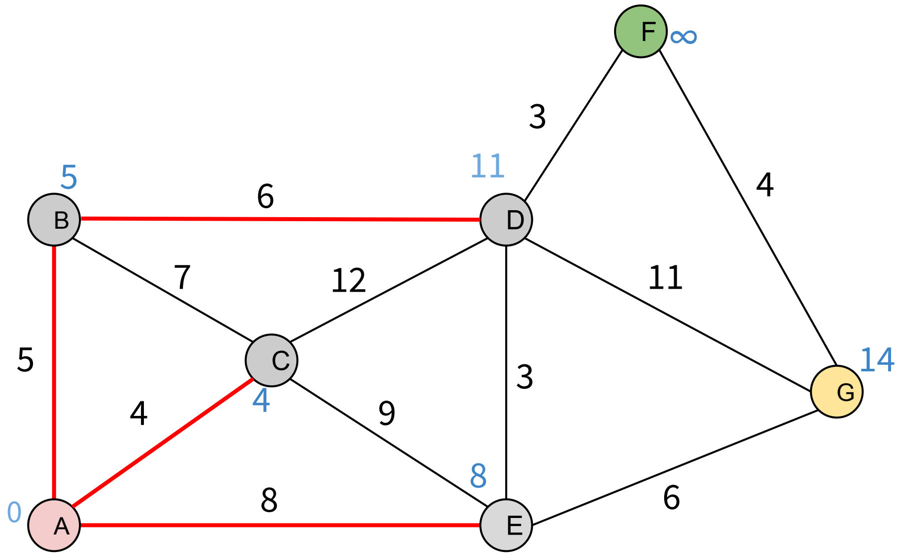
Visited: {A, C, B, E}

Dijkstra's Algorithm



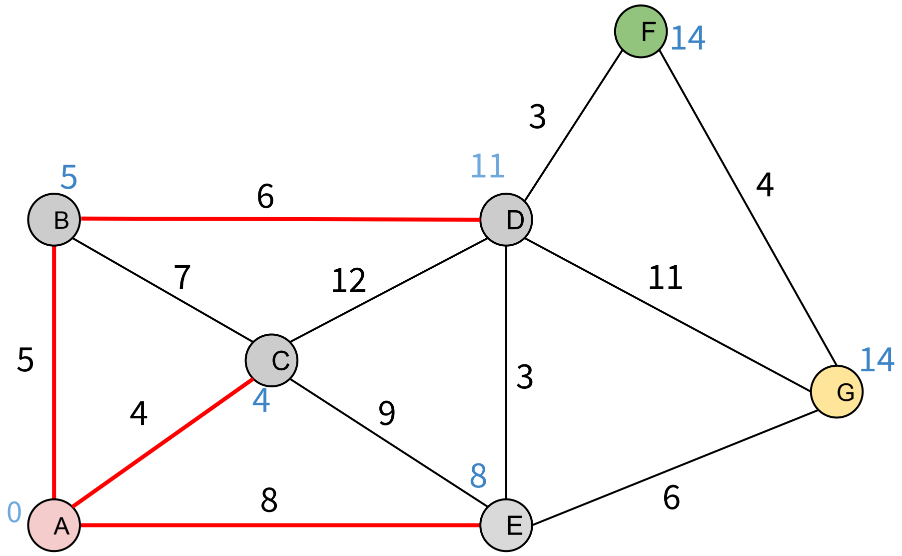
Visited: {A, C, B, E}

Dijkstra's Algorithm



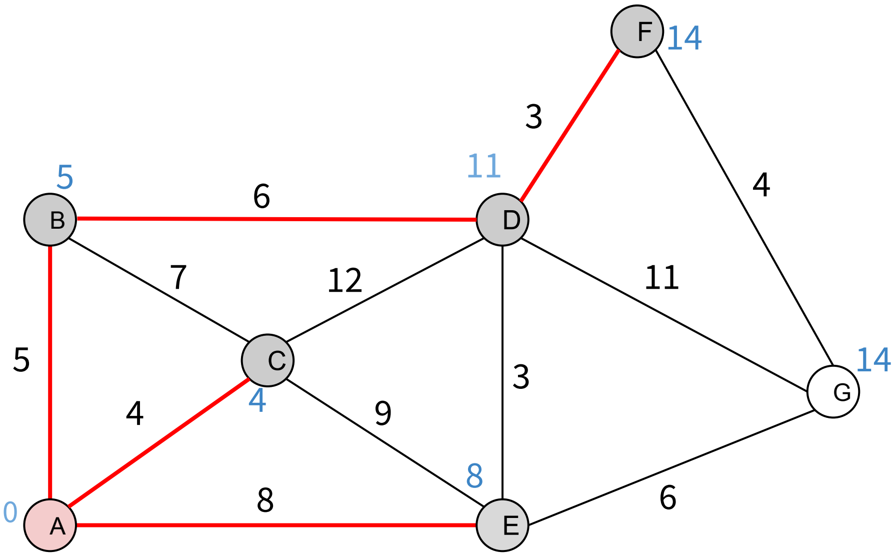
Visited: {A, C, B, E, D}

Dijkstra’s Algorithm



Visited: {A, C, B, E, D}

Dijkstra’s Algorithm

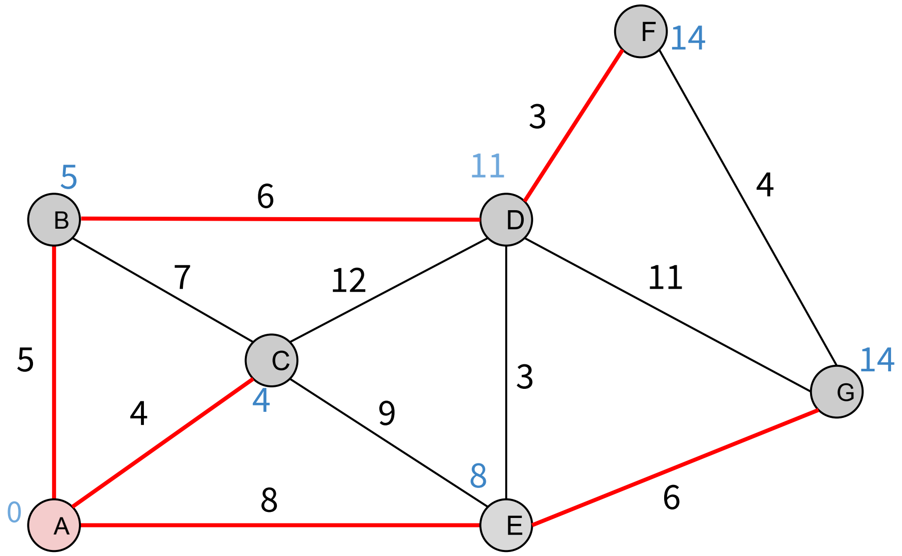


Visited: {A, C, B, E, D, F}

We just found the shortest path from the source to the goal (F) since F is visited now.

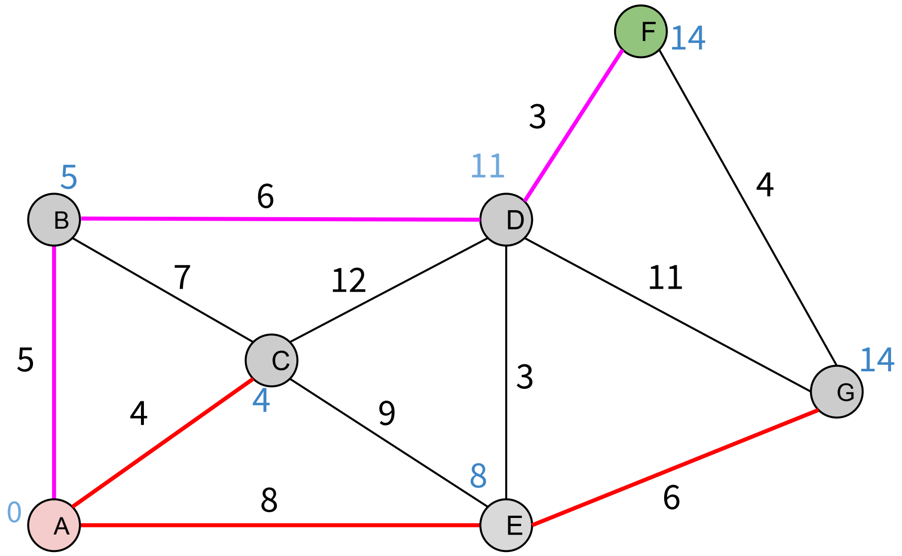
But, let’s visit all vertices in the graph

Dijkstra's Algorithm



Visited: {A, C, B, E, D, F, G}

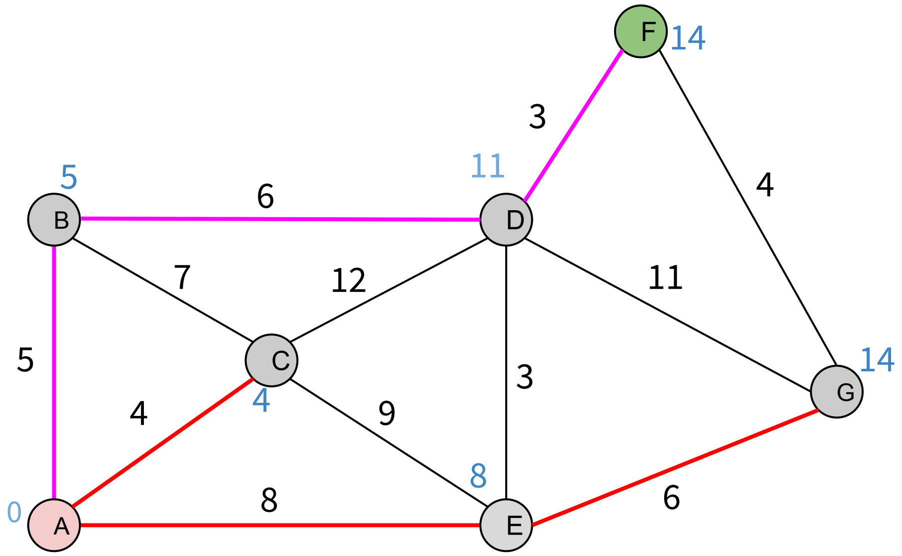
Dijkstra's Algorithm



Visited: {A, C, B, E, D, F}

COST: 14

Dijkstra’s Algorithm



Visited: {A, C, B, E, D, F}

Time
Complexity?

Dijkstra's Algorithm

```
for all  u ∈ V:
        dist(u) = infinity
        prev(u) = NULL
dist(s) = 0

H= makeheap(V)(using dist values as key)
While H is not empty:
        u= deletmin(H)
        for all edges (u,v) ∈ E:
                if dist(v) > dist(u) + l(u, v):
                        dist(v) = dist(u) + l(u, v)
                        prev(v) = u
                        decreasekey(H,v)
```

Time Complexity: O((IEI +IVI) log v)