Initialization:

start_node = 'A'

stop_node = 'G'

open_set = {'A'} (Nodes to be evaluated, initialized with the start node)

closed_set = {} (Nodes that have been evaluated)

g = {'A': 0} (Cost of reaching each node from the start node)

parents = {'A': 'A'} (Parent node of each node in the path)

Iteration 1:

Current Node (n): 'A'

Neighbors: 'B' (cost 2), 'E' (cost 3)

'B':

Add 'B' to open_set.

Set parent of 'B' as 'A'.

Set g(B) = g(A) + cost(A to B) = 0 + 2 = 2.

'E':

Add 'E' to open_set.

Set parent of 'E' as 'A'.

Set g(E) = g(A) + cost(A to E) = 0 + 3 = 3.

Move 'A' from open_set to closed_set.

Iteration 2:

Current Node (n): 'B'

Neighbors: 'C' (cost 1), 'G' (cost 9)

'C':

Add 'C' to open_set.

Set parent of 'C' as 'B'.

Set g(C) = g(B) + cost(B to C) = 2 + 1 = 3.

'G':

Add 'G' to open_set.

Set parent of 'G' as 'B'.

Set g(G) = g(B) + cost(B to G) = 2 + 9 = 11.

Move 'B' from open_set to closed_set.

Iteration 3:

Current Node (n): 'E'

Neighbors: 'D' (cost 6)

'D':

Add 'D' to open_set.

Set parent of 'D' as 'E'.

Set g(D) = g(E) + cost(E to D) = 3 + 6 = 9.

Move 'E' from open_set to closed_set.

Iteration 4:

Current Node (n): 'D'

Neighbors: 'G' (cost 1)

'G':

Add 'G' to open_set.

Set parent of 'G' as 'D'.

Set g(G) = g(D) + cost(D to G) = 9 + 1 = 10.

Move 'D' from open_set to closed_set.

Iteration 5:

Current Node (n): 'G' (Goal Node)

Reconstruct the path from 'G' to 'A'.

Path:

Reverse the path: ['A', 'E', 'D', 'G']

Final path: ['A', 'E', 'D', 'G']

The shortest path from node 'A' to node 'G' is ['A', 'E', 'D', 'G'].