# Shortest Paths Algorithms Exercises ANS

Lecture 5.1, Spring 2026

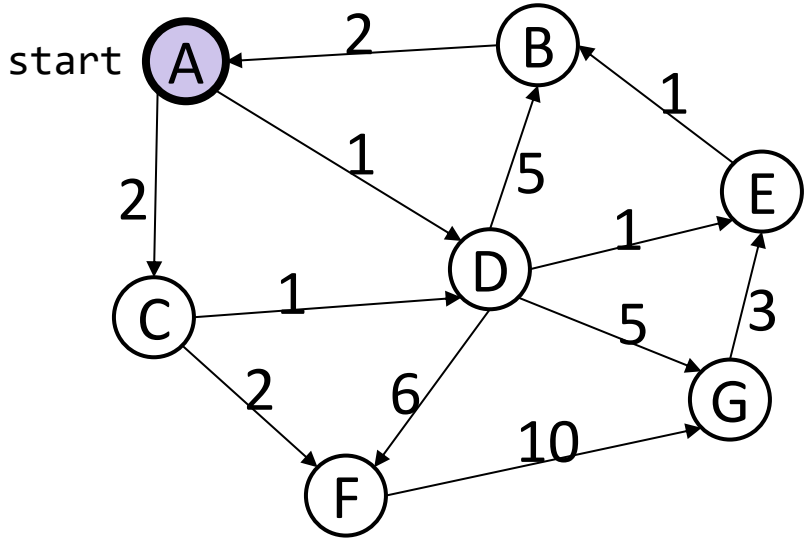
BFS

Dijkstra's Algorithm

Bellman-Ford Algorithm

# Q. Dijkstra's Algorithm

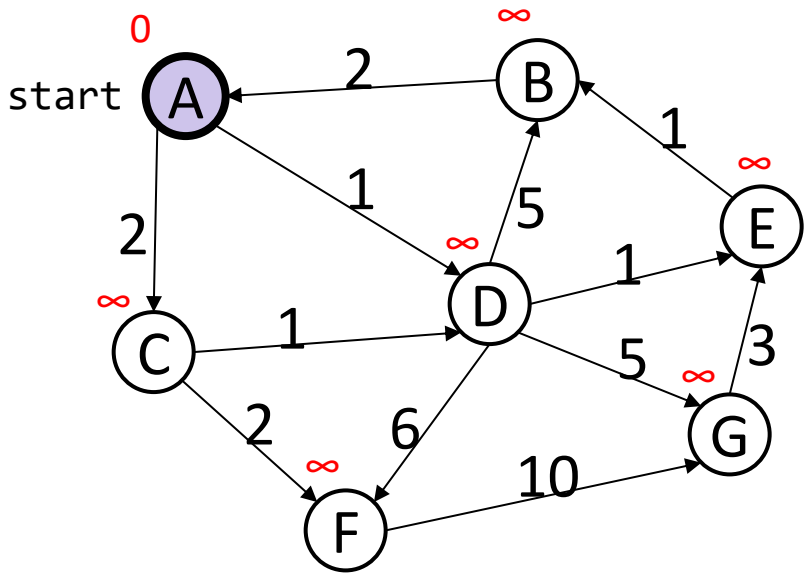
Given this directed graph, run Dijkstra's Algo to find shortest paths starting from source node A. Give the node visit order, and fill in this table of SN (Shortest Distance) and PN (Previous Node), crossing out old SD and PN as you find a shortcut path with smaller SD




Visit Order

| Node | SD | PN |
|---|---|---|
| A | | |
| B | | |
| C | | |
| D | | |
| E | | |
| F | | |
| G | | |

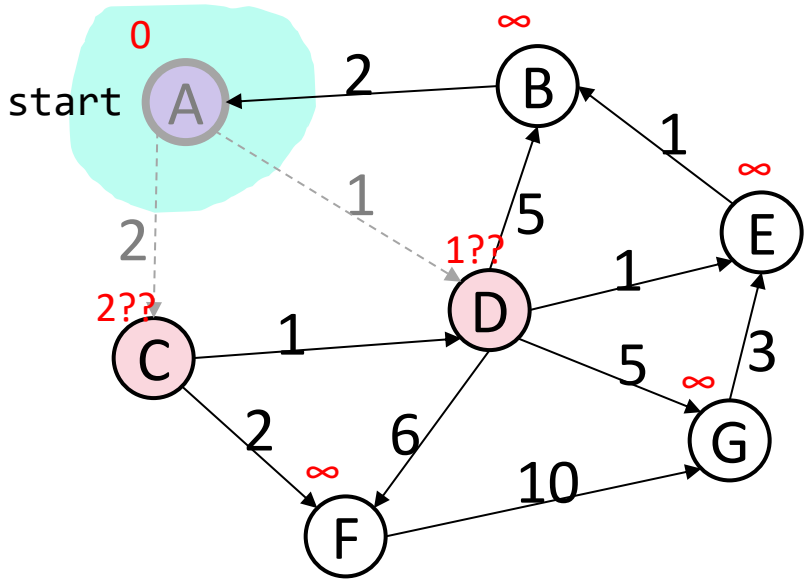
# Q. Dijkstra's Algorithm




Visit Order

| Node | SD | PN |
|---|---|---|
| A | ∞ | |
| B | ∞ | |
| C | ∞ | |
| D | ∞ | |
| E | ∞ | |
| F | ∞ | |
| G | ∞ | |

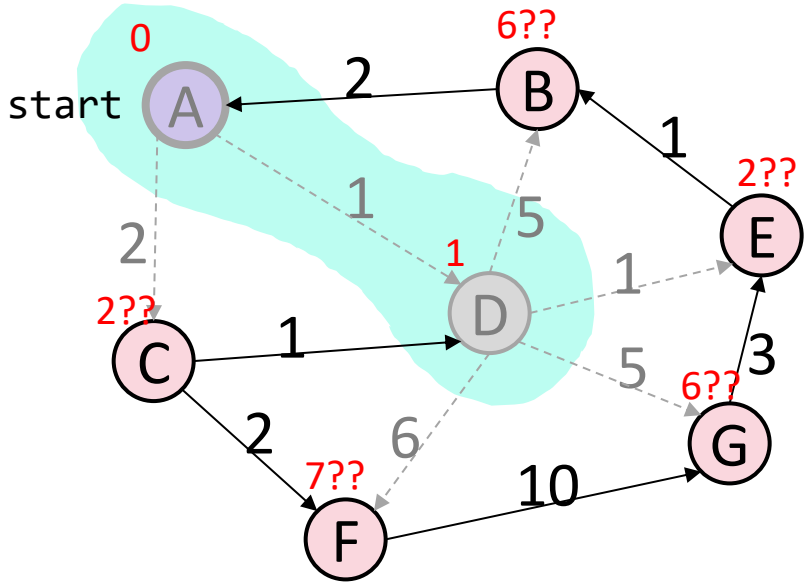
# Q. Dijkstra's Algorithm




Visit Order
A

| Node | SD | PN |
|---|---|---|
| A | 0 | / |
| B | ∞ | |
| C | 2 | A |
| D | 1 | A |
| E | ∞ | |
| F | ∞ | |
| G | ∞ | |

# Q. Dijkstra's Algorithm

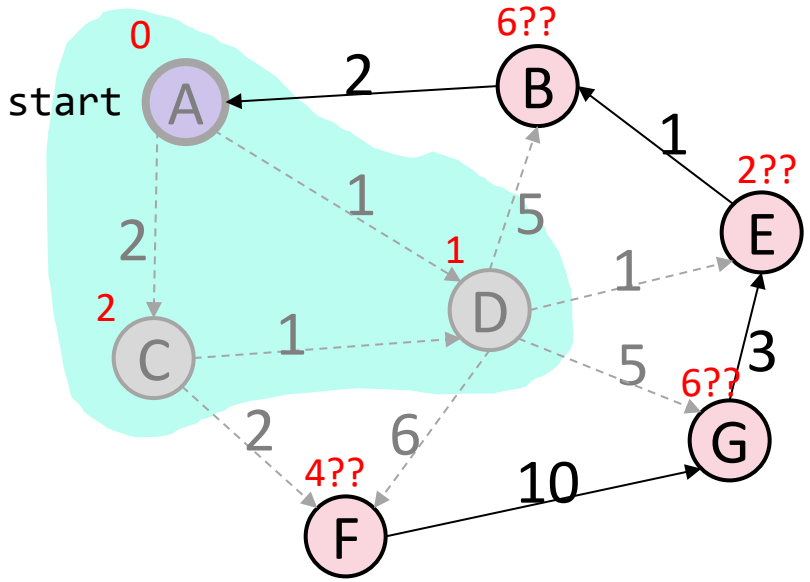



| Node | SD | PN |
|---|---|---|
| A | 0 | / |
| B | **6** | **D** |
| C | 2 | A |
| D | 1 | A |
| E | **2** | **D** |
| F | **7** | **D** |
| G | **6** | **D** |

Visit Order
A, D

We can choose to visit either C or E next, since they have equal smallest SD of 2 among all unvisited nodes. Let's visit C in alphabetical order

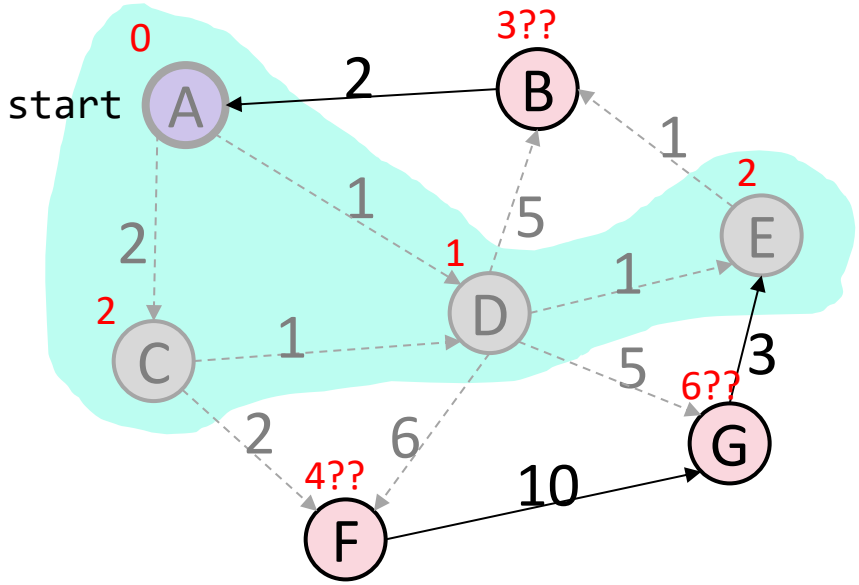
# Q. Dijkstra's Algorithm




| Node | SD | PN |
|---|---|---|
| A | 0 | / |
| B | 6 | D |
| C | 2 | A |
| D | 1 | A |
| E | 2 | D |
| F | ~~7~~ 4 | ~~D~~ C |
| G | 6 | D |

Visit Order
A, D, C

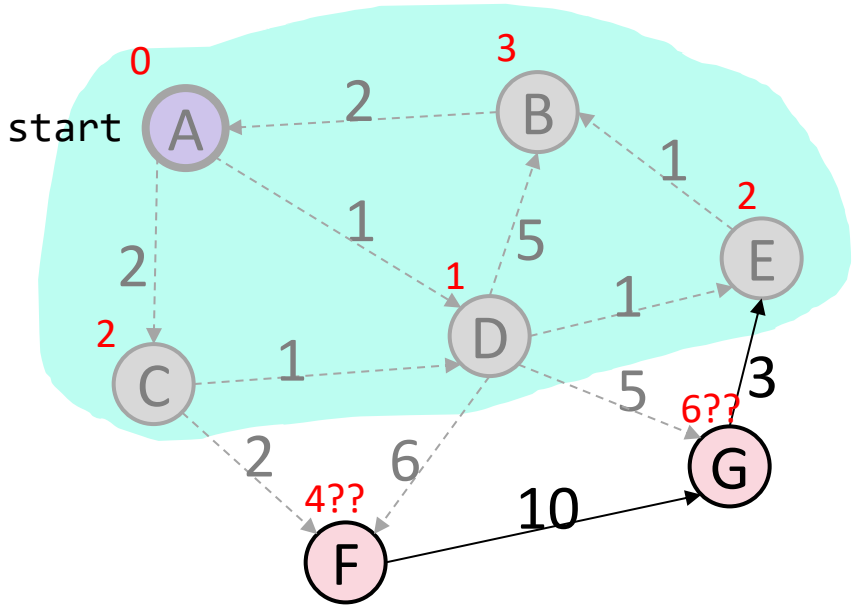
# Q. Dijkstra's Algorithm




| Node | SD | PN |
|---|---|---|
| A | 0 | / |
| B | ~~6~~ **3** | ~~D~~ **E** |
| C | 2 | A |
| D | 1 | A |
| E | 2 | D |
| F | ~~7~~ 4 | ~~D~~ C |
| G | 6 | D |

**Visit Order**
A, D, C, E

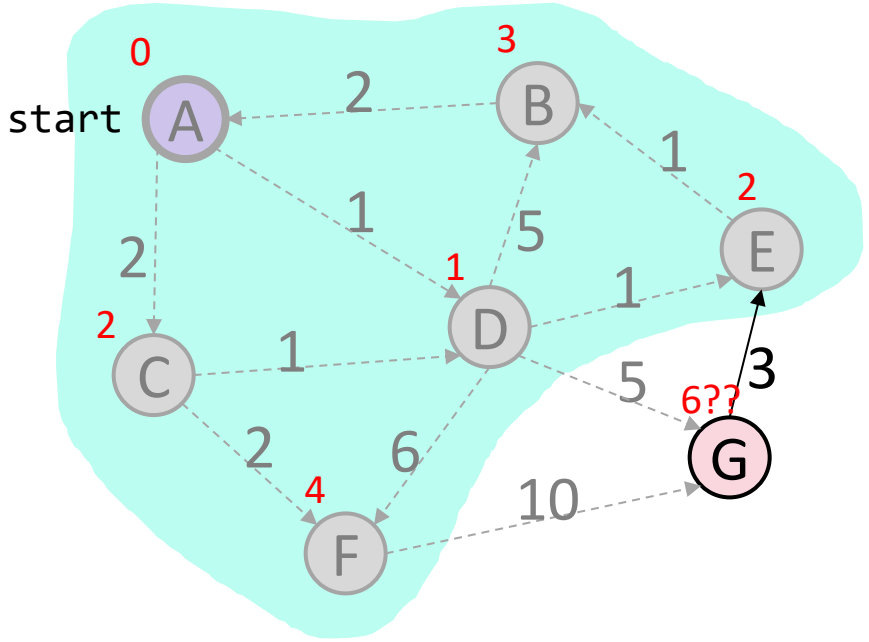
# Q. Dijkstra's Algorithm




Visit Order
A, D, C, E, B

| Node | SD | PN |
|---|---|---|
| A | 0 | / |
| B | ~~6~~ 3 | ~~D~~ E |
| C | 2 | A |
| D | 1 | A |
| E | 2 | D |
| F | ~~7~~ 4 | ~~D~~ C |
| G | 6 | D |

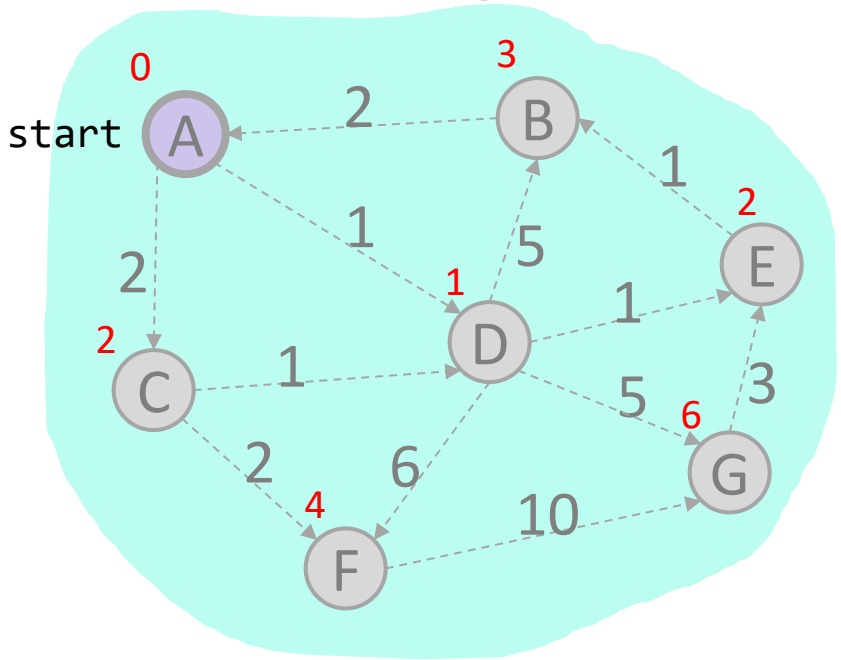
# Q. Dijkstra's Algorithm




Visit Order
A, D, C, E, B, F

| Node | SD | PN |
|---|---|---|
| A | 0 | / |
| B | ~~6~~ 3 | ~~D~~ E |
| C | 2 | A |
| D | 1 | A |
| E | 2 | D |
| F | ~~7~~ 4 | ~~D~~ C |
| G | 6 | D |

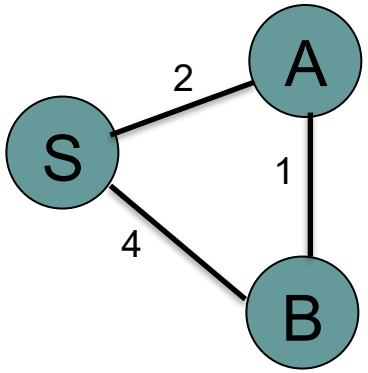
# Q. Dijkstra's Algorithm ANS




| Node | SD | PN |
|---|---|---|
| A | 0 | / |
| B | ~~6~~ 3 | ~~D~~ E |
| C | 2 | A |
| D | 1 | A |
| E | 2 | D |
| F | ~~7~~ 4 | ~~D~~ C |
| G | 6 | D |

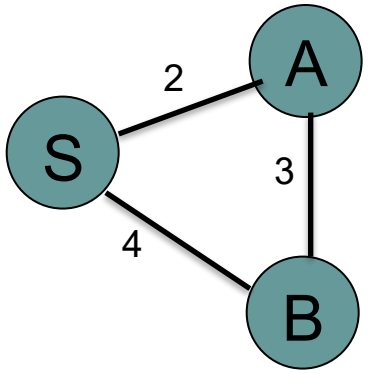
Visit Order
A, D, C, E, B, F, G

# Q. Dijkstra's Algorithm (Source Node S)



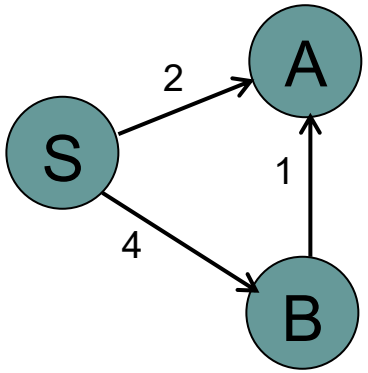

ANS

| Node | SD | PN |
|---|---|---|
| S | 0 | / |
| A | | |
| B | | |



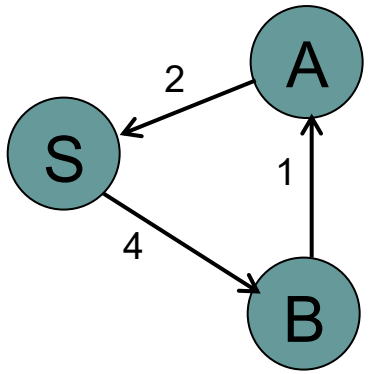

ANS

| Node | SD | PN |
|---|---|---|
| S | 0 | / |
| A | | |
| B | | |




ANS

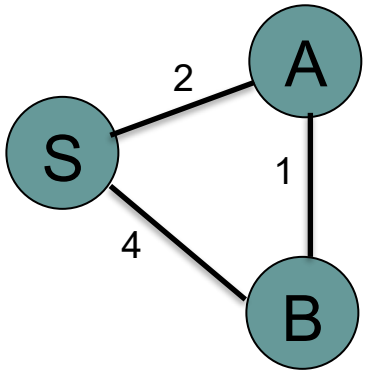
| Node | SD | PN |
|---|---|---|
| S | 0 | / |
| A | | |
| B | | |




ANS

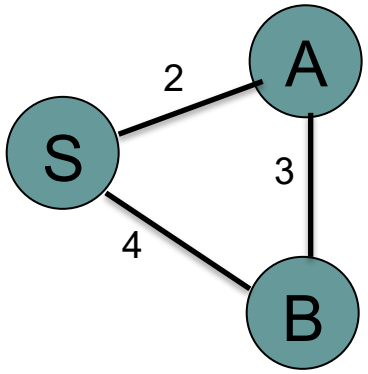
| Node | SD | PN |
|---|---|---|
| S | 0 | / |
| A | | |
| B | | |

# Q. Dijkstra's Algorithm (Source Node S) ANS



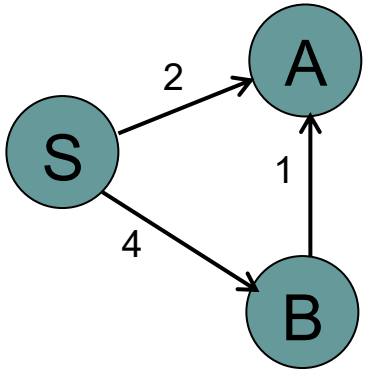

ANS

| Node | SD | PN |
|---|---|---|
| S | 0 | / |
| A | 2 | S |
| B | 4 3 | ~~S~~ A |



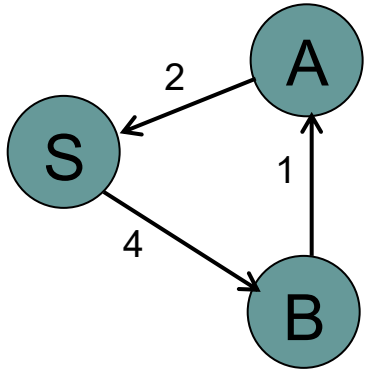

ANS

| Node | SD | PN |
|---|---|---|
| S | 0 | / |
| A | 2 | S |
| B | 4 | S |




ANS

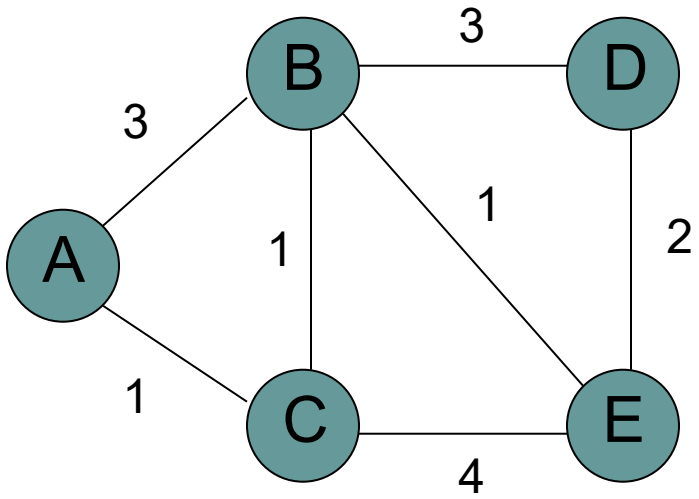
| Node | SD | PN |
|---|---|---|
| S | 0 | / |
| A | 2 | S |
| B | 4 | S |




ANS

| Node | SD | PN |
|---|---|---|
| S | 0 | / |
| A | 5 | B |
| B | 4 | S |

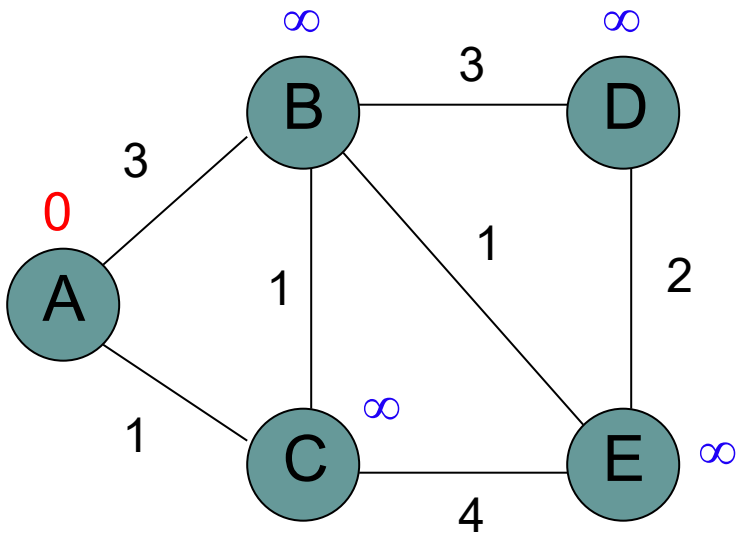
# Q. Dijkstra's Algorithm (Source Node A, Undirected Graph)




Visit Order

| Node | SD | PN |
|---|---|---|
| A | | |
| B | | |
| C | | |
| D | | |
| E | | |

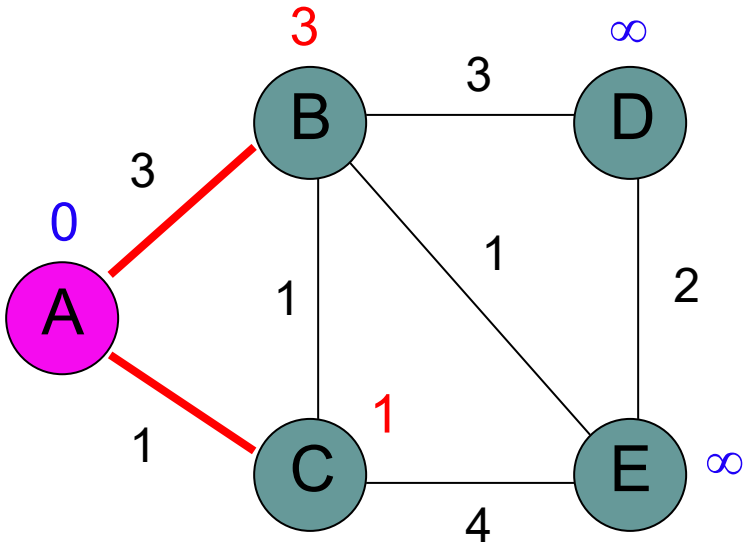
# Initialize




Visit Order

| Node | SD | PN |
|---|---|---|
| A | 0 | / |
| B | ∞ | |
| C | ∞ | |
| D | ∞ | |
| E | ∞ | |

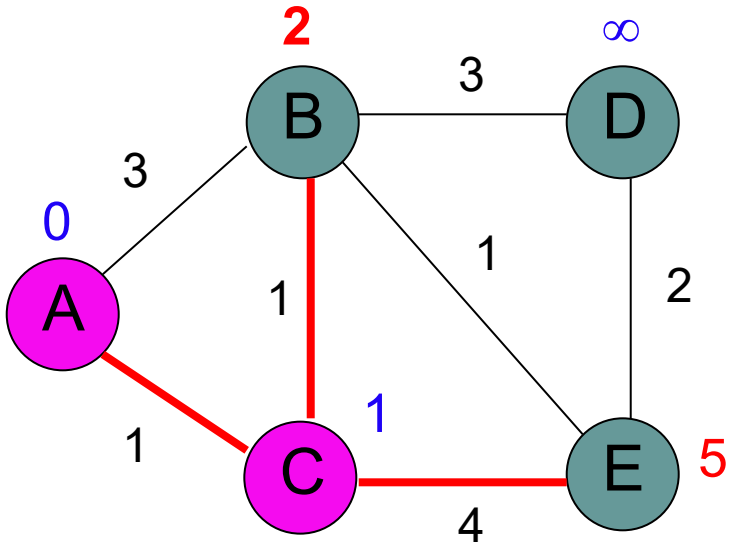
# Visit Node A




Visit Order
A

| Node | SD | PN |
|---|---|---|
| A | 0 | / |
| B | 3 | A |
| C | 1 | A |
| D | ∞ | |
| E | ∞ | |

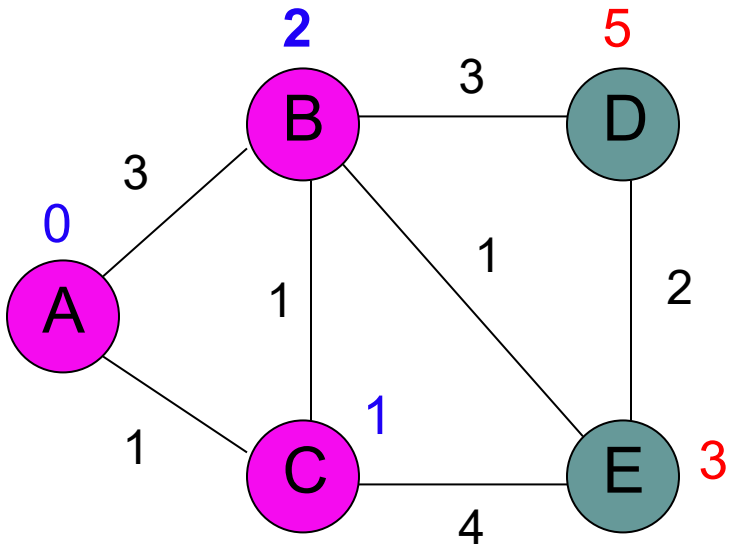
# Visit Node C




Visit Order
A, C

| Node | SD | PN |
|---|---|---|
| A | 0 | / |
| B | ~~3~~ 2 | ~~A~~ C |
| C | 1 | A |
| D | ∞ | |
| E | 5 | C |

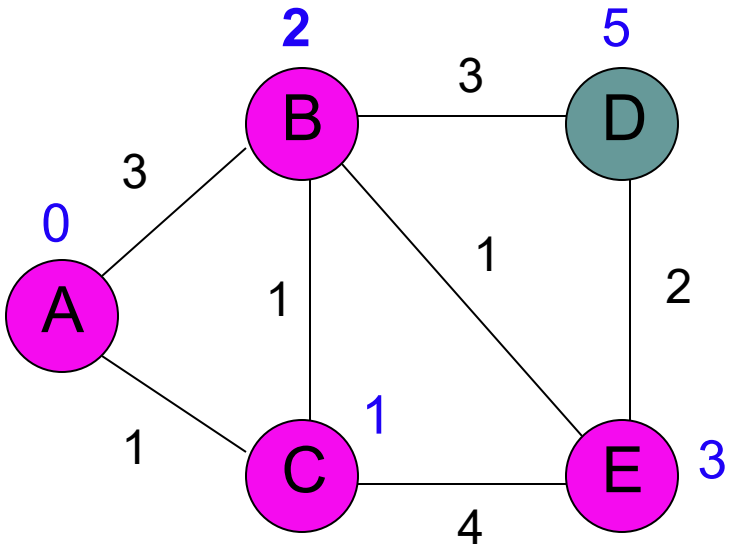
# Visit Node B




| Node | SD | PN |
|---|---|---|
| A | 0 | / |
| B | ~~3~~ 2 | ~~A~~ C |
| C | 1 | A |
| D | 5 | B |
| E | ~~5~~ 3 | ~~C~~ B |

Visit Order
A, C, B

# Visit Node E

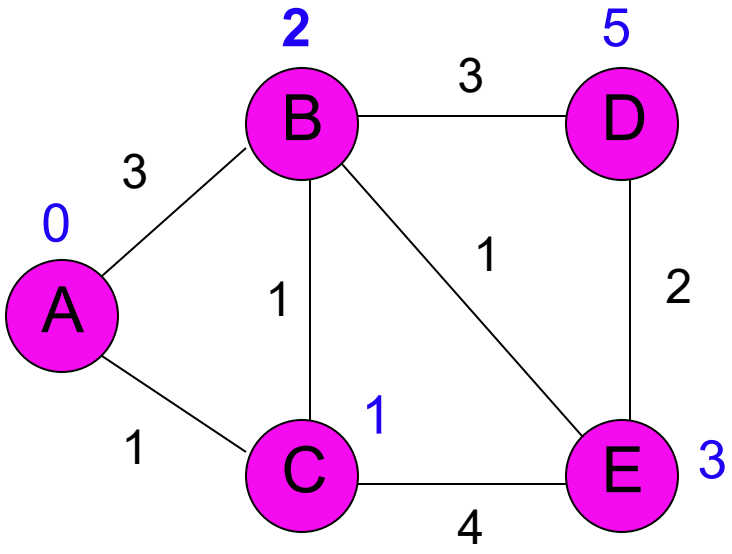



Visit Order
A, C, B, E

| Node | SD | PN |
|---|---|---|
| A | 0 | / |
| B | ~~3~~ 2 | ~~A~~ C |
| C | 1 | A |
| D | 5 | B |
| E | ~~5~~ 3 | ~~C~~ B |

Nothing changes

# Visit Node D

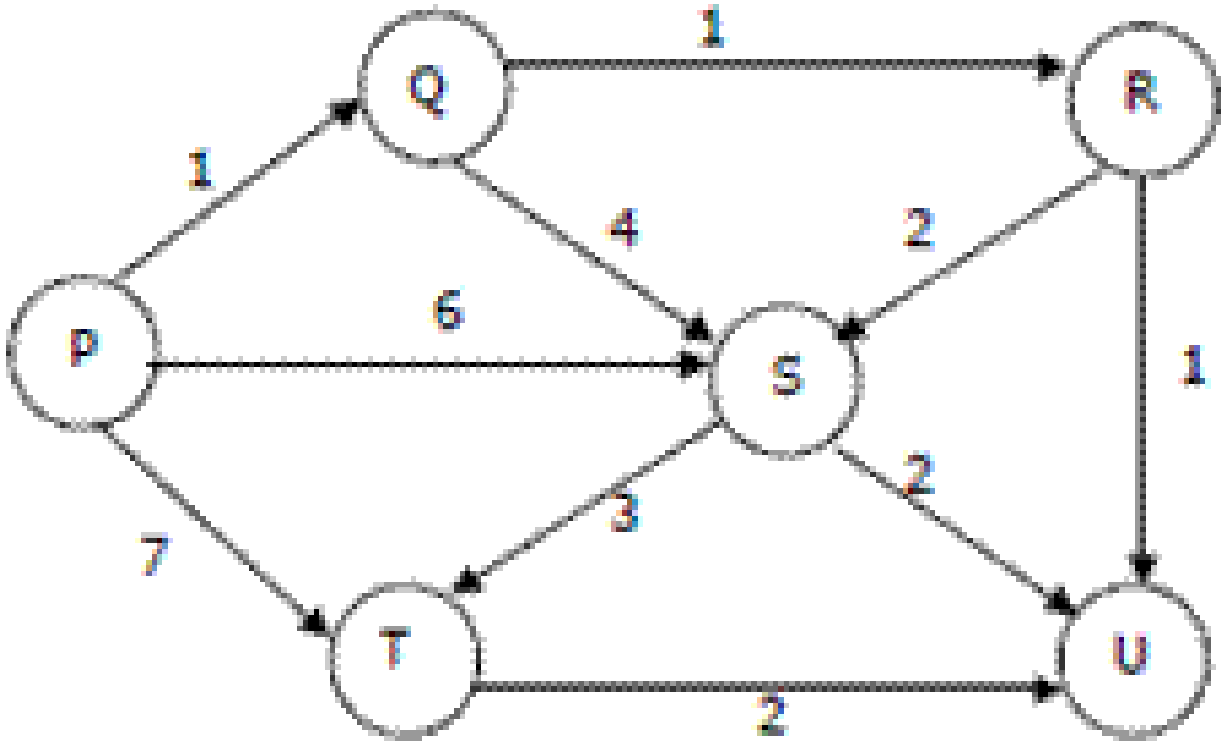



Visit Order
A, C, B, E, D

| Node | SD | PN |
|---|---|---|
| A | 0 | / |
| B | ~~3~~ 2 | ~~A~~ C |
| C | 1 | A |
| D | 5 | B |
| E | ~~5~~ 3 | ~~C~~ B |

Nothing changes

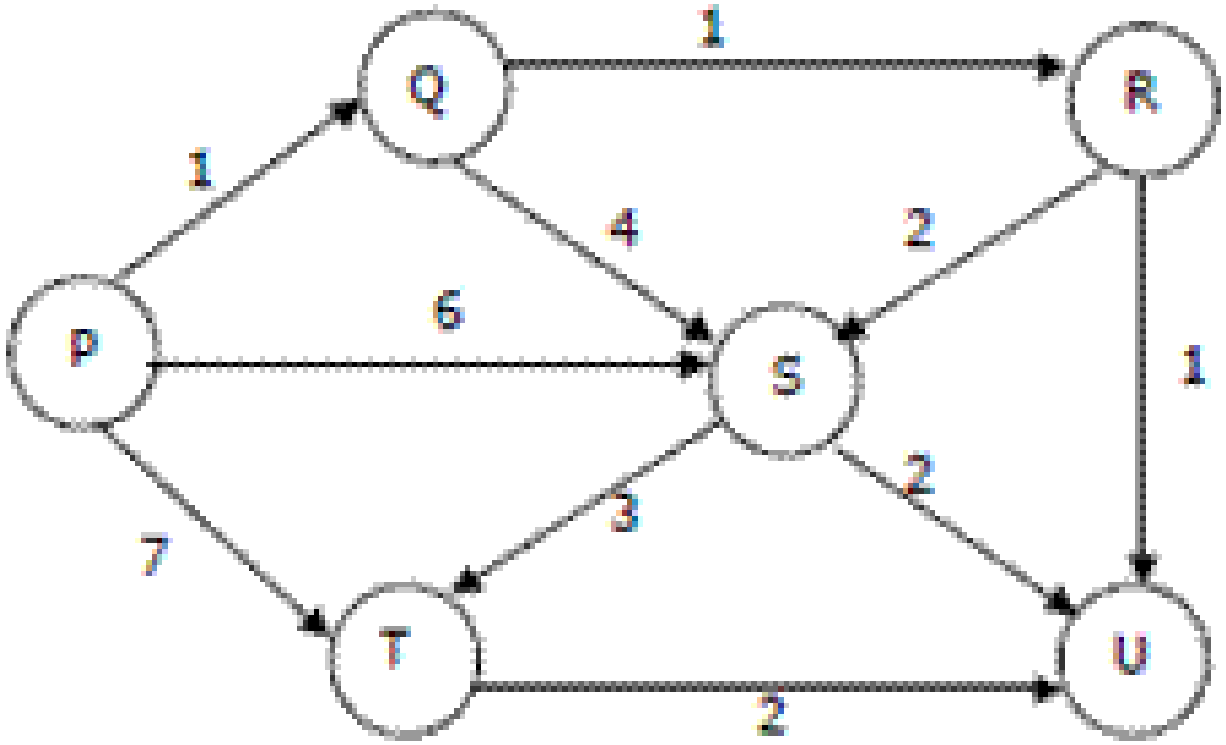
# Q. Dijkstra's Algorithm (Source Node P, Directed Graph)




Visit order:

| Node | SD | PN |
|---|---|---|
| P | 0 | |
| Q | | |
| R | | |
| S | | |
| T | | |
| U | | |

# Q. Dijkstra's Algorithm (Source Node P, Directed Graph) ANS

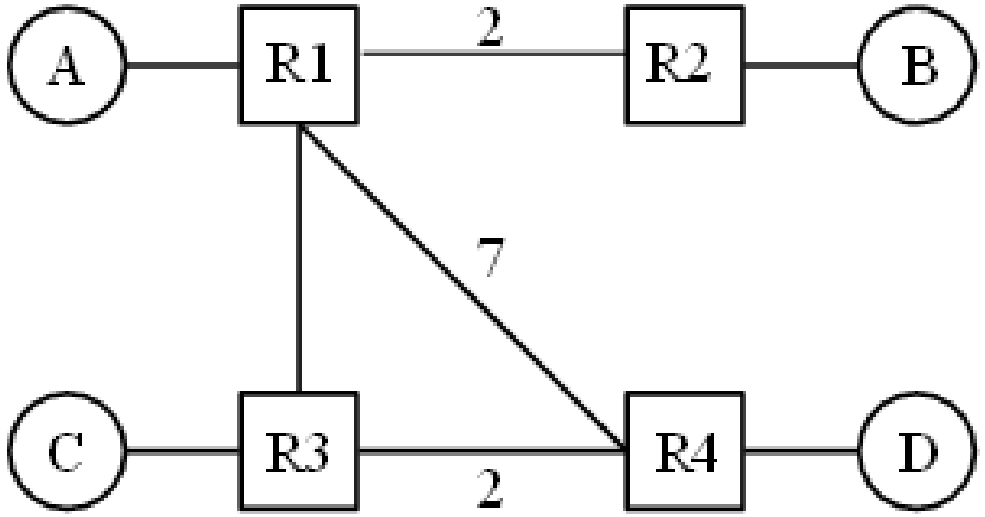



Visit order: P, Q, R, U, S, T

| Node | SD | PN |
|---|---|---|
| P | 0 | |
| Q | 1 | P |
| R | 2 | Q |
| S | ~~6~~ ~~5~~ 4 | ~~P~~ ~~Q~~ R |
| T | 7 | P |
| U | 3 | R |

# Q. Distance-Vector

Consider running the distance-vector protocol on the topology below. Unlabeled links have cost 1. Fill in the routing tables based on Distance-Vector algorithm. (refer to the handout for details.)




R1's table

| Dest. | Hop, Dist. |
|---|---|
| A | Direct, 1 |
| | |
| | |
| | |

R2's table

| Dest. | Hop, Dist. |
|---|---|
| B | Direct, 1 |
| | |
| | |
| | |

R3's table

| Dest. | Hop, Dist. |
|---|---|
| C | Direct, 1 |
| | |
| | |
| | |

R4's table

| Dest. | Hop, Dist. |
|---|---|
| D | Direct, 1 |
| | |
| | |
| | |