Graphs & Graph Algorithms 2

CMSC 132

Department of Computer Science

University of Maryland, College Park

Overview

- Shortest path
- Djikstra's algorithm

Single Source Shortest Path

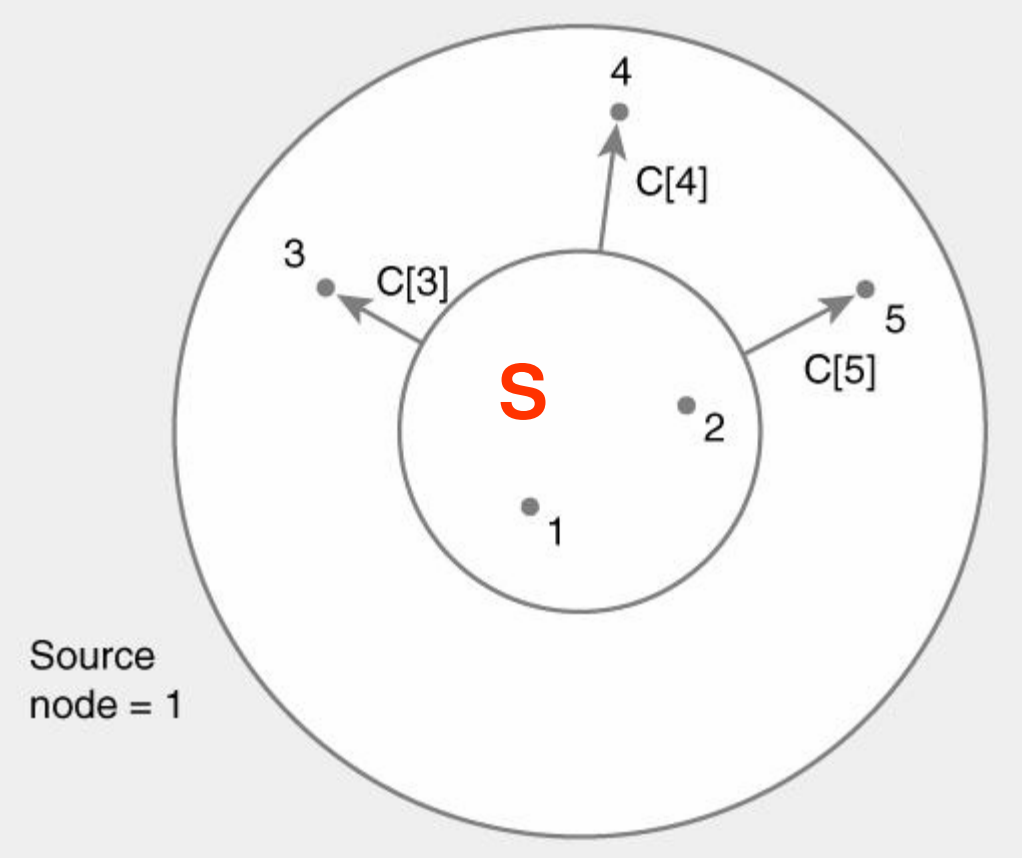
- Common graph problem
  - Find path from X to Y with lowest edge weight
  - Find path from X to any Y with lowest edge weight
- Useful for many applications
  - Shortest route in map
  - Lowest cost trip
  - Most efficient internet route
- Can solve both problems with same algorithm
  - Would actually do something slightly different if we only wanted shortest path from X to Y
  - Don't need to find shortest path from College Park to Seattle in order to find shortest path from College Park to Miami

Shortest Path – Dijkstra's Algorithm

- **Maintain**
  - **Nodes with known shortest path from start ⇒ S**
  - **Cost of shortest path to node K from start ⇒ C[K]**
    - **Only for paths through nodes in S**
  - **Predecessor to K on shortest path ⇒ P[K]**
    - **Updated whenever new (lower) C[K] discovered**
    - **Remembers actual path with lowest cost**

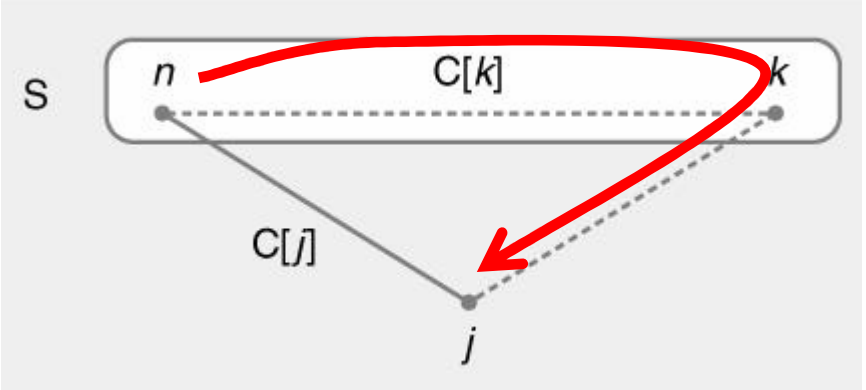
Shortest Path – Intuition for Dijkstra's

- At each step in the algorithm
  - Shortest paths are known for nodes in S
  - Store in C[K] length of shortest path to node K (for all paths through nodes in { S })
  - Add to { S } next closest node



Shortest Path – Intuition for Djikstra's

- **Update distance to J after adding node K**
  - **Previous shortest paths already in C[K]**
  - **Possibly shorter path by going through node K**
  - **Compare C[J] with C[K] + weight of (K,J)**



Shortest Path – Djikstra's Algorithm

- **Algorithm**
  - Add starting node to S
  - Repeat until all nodes in S
    - Find node K not in S with smallest C[K]
    - Add K to S
    - Examine C[J] for all neighbors J of K not in S
      - If (C[K] + weight for edge (K,J)) < C[J]
        - New shortest path by first going to K, then J
        - Update C[J] ← C[K] + weight for edge (K,J)
        - Update P[J] ← K

Shortest Path – Dijkstra's Algorithm

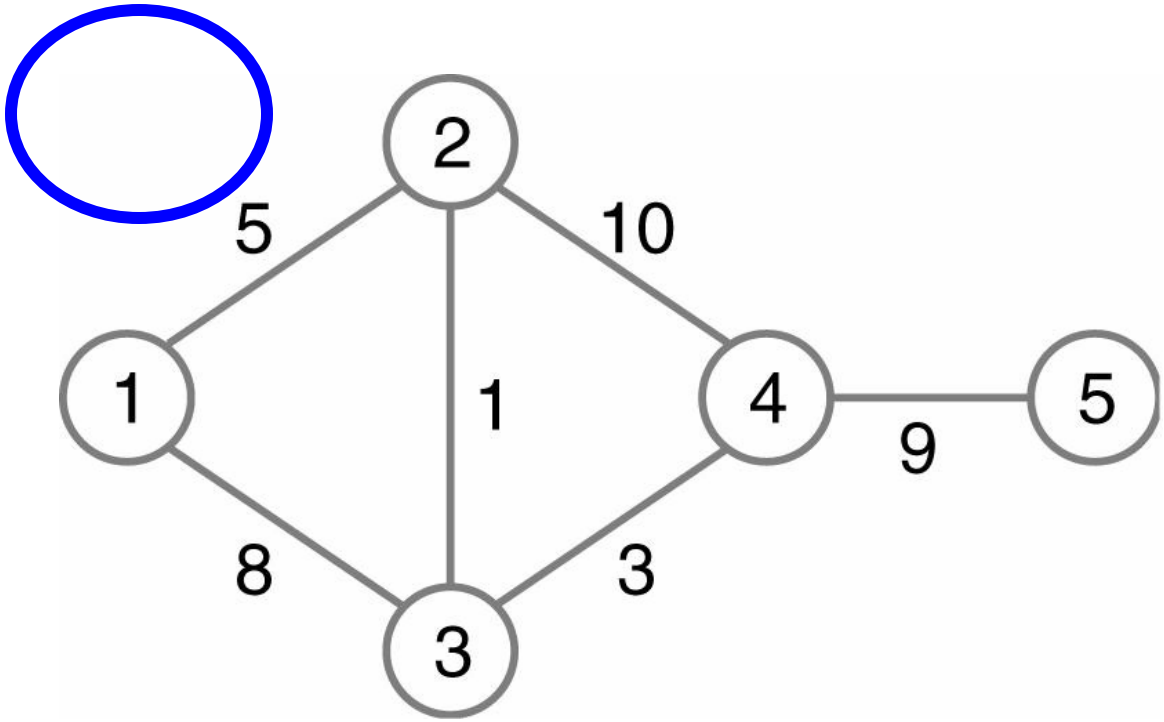
```
S = {}, P[ ] = none for all nodes
C[start] = 0, C[ ] = ∞ for all other nodes
while ( not all nodes in S )
    find node K not in S with smallest C[K]
    add K to S
    for each node J not in S  adjacent to K
        if ( C[K] + cost of (K,J) < C[J] )
            C[J] = C[K] + cost of (K,J)
            P[J] = K
```

Optimal solution computed with greedy algorithm

Dijkstra's Shortest Path Example

- **Initial state**
- **S = ∅**

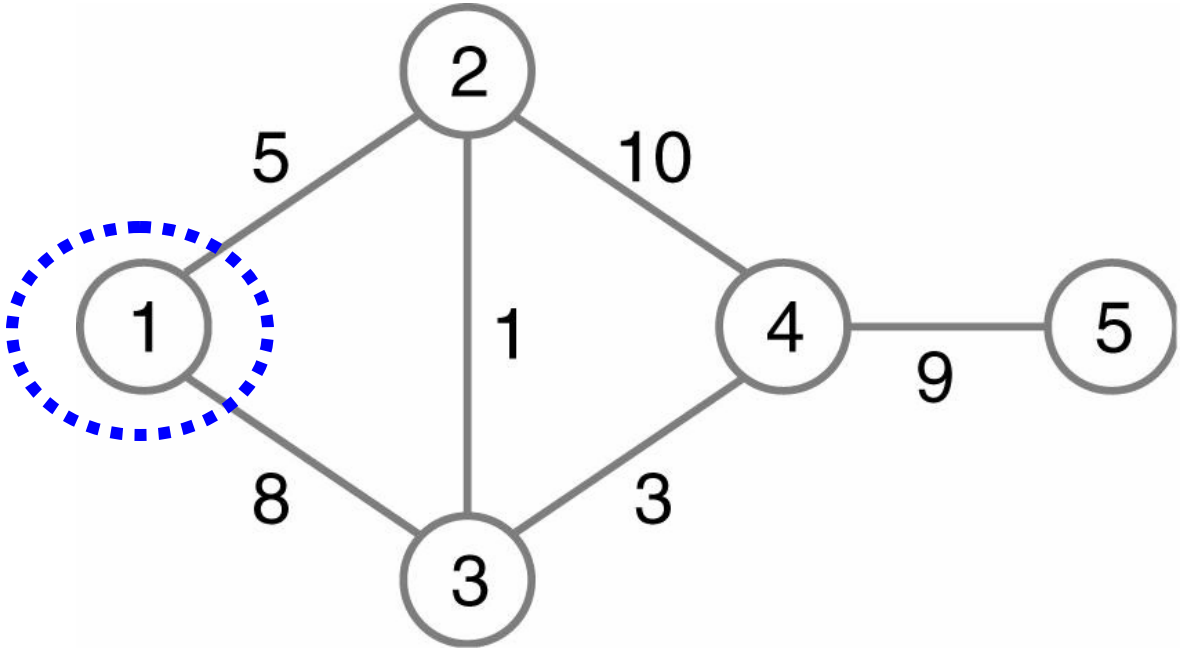
	C	P
1	0	none
2	∞	none
3	∞	none
4	∞	none
5	∞	none



Dijkstra's Shortest Path Example

- Find shortest paths starting from node 1
- S = 1

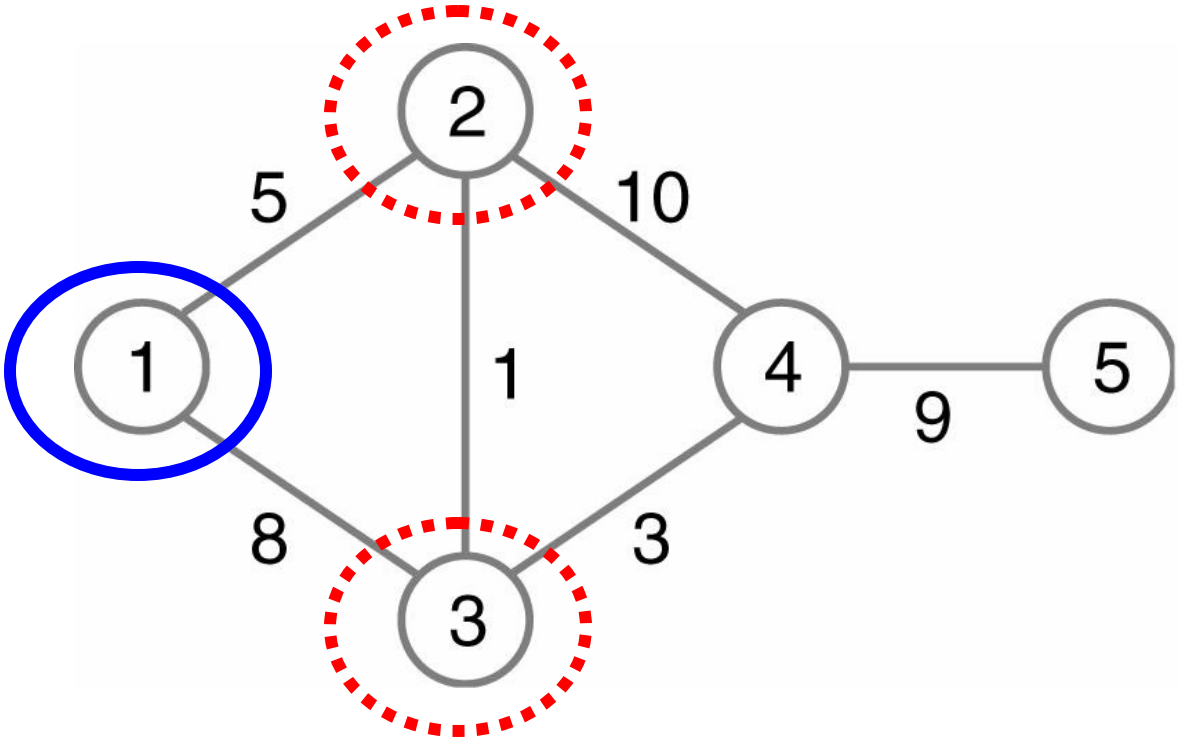
	C	P
1	0	none
2	∞	none
3	∞	none
4	∞	none
5	∞	none



Djikstra's Shortest Path Example

- Update C[K] for all neighbors of 1 not in { S }
- S = { 1 }

	C	P
1	0	none
2	5	1
3	8	1
4	∞	none
5	∞	none

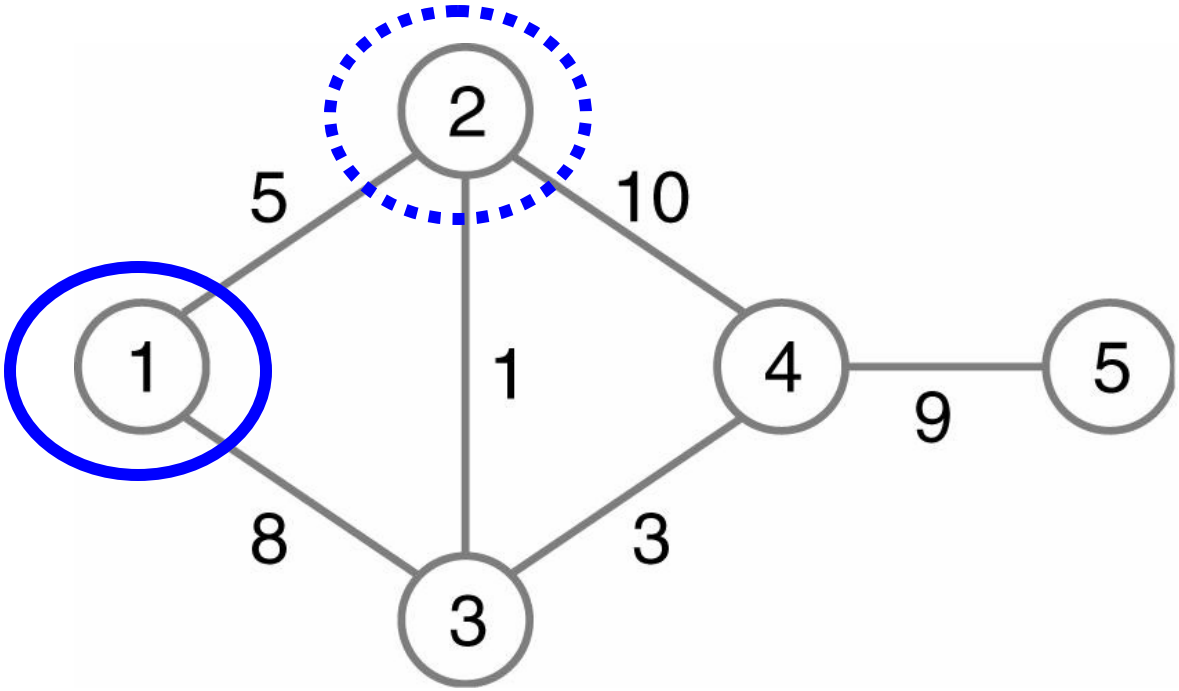


C[2] = min (∞ , C[1] + (1,2)) = min (∞ , 0 + 5) = 5
C[3] = min (∞ , C[1] + (1,3)) = min (∞ , 0 + 8) = 8

Djikstra's Shortest Path Example

- Find node K with smallest C[K] and add to S
- S = { 1, 2 }

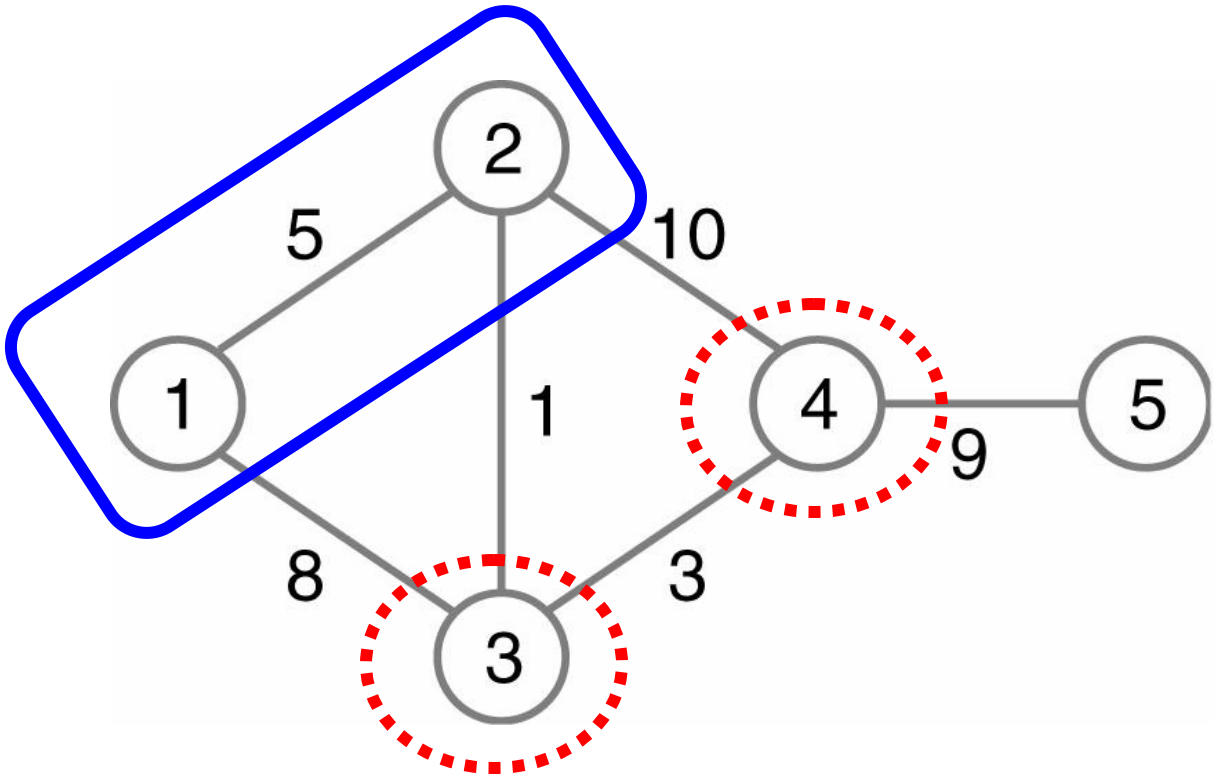
	C	P
1	0	none
2	5	1
3	8	1
4	∞	none
5	∞	none



Dijkstra's Shortest Path Example

- Update C[K] for all neighbors of 2 not in S
- S = { 1, 2 }

	C	P
1	0	none
2	5	1
3	6	2
4	15	2
5	∞	none

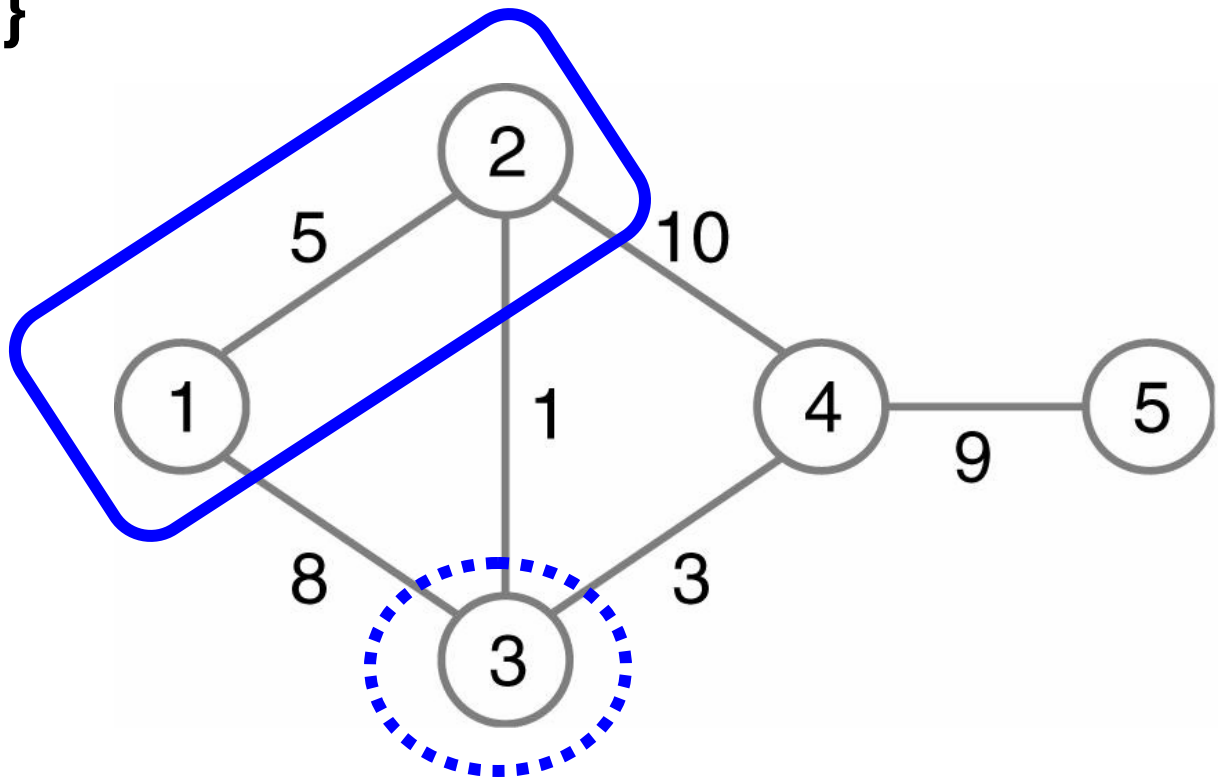


C[3] = min (8 , C[2] + (2,3)) = min (8 , 5 + 1) = 6
C[4] = min (∞ , C[2] + (2,4)) = min (∞ , 5 + 10) = 15

Dijkstra's Shortest Path Example

- Find node K with smallest C[K] and add to S
- S = { 1, 2, 3 }

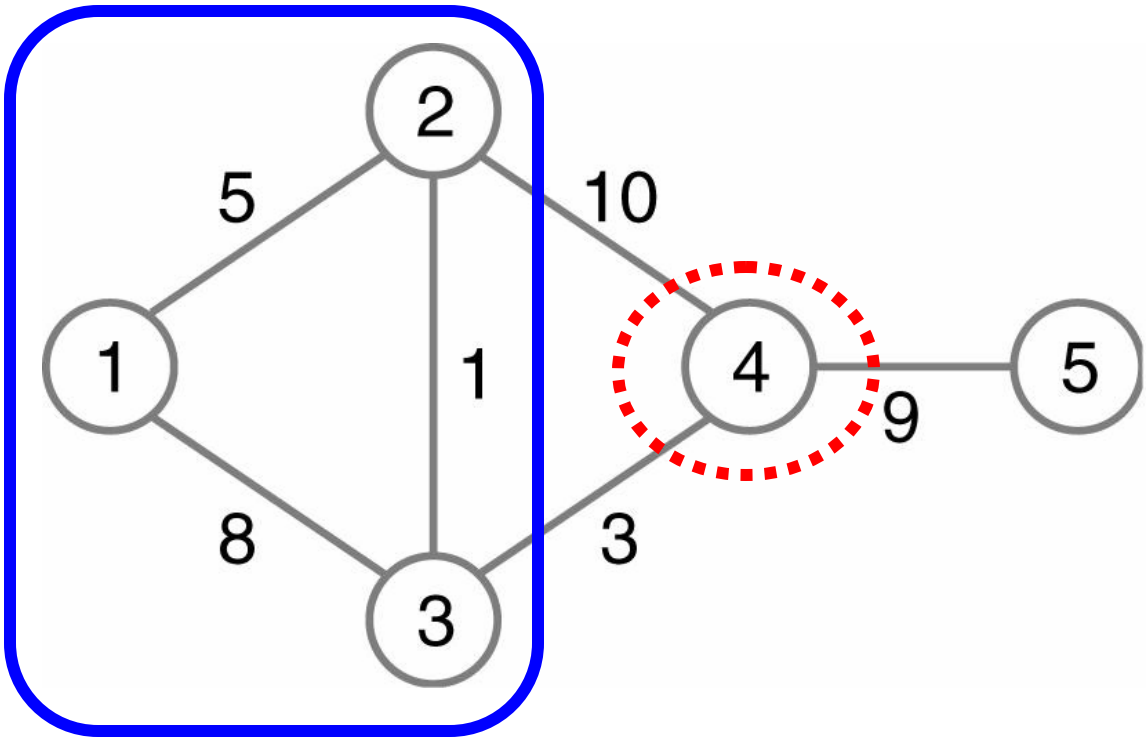
	C	P
1	0	none
2	5	1
3	6	2
4	15	2
5	∞	none



Dijkstra's Shortest Path Example

- Update C[K] for all neighbors of 3 not in S
- { S } = 1, 2, 3

	C	P
1	0	none
2	5	1
3	6	2
4	9	3
5	∞	none

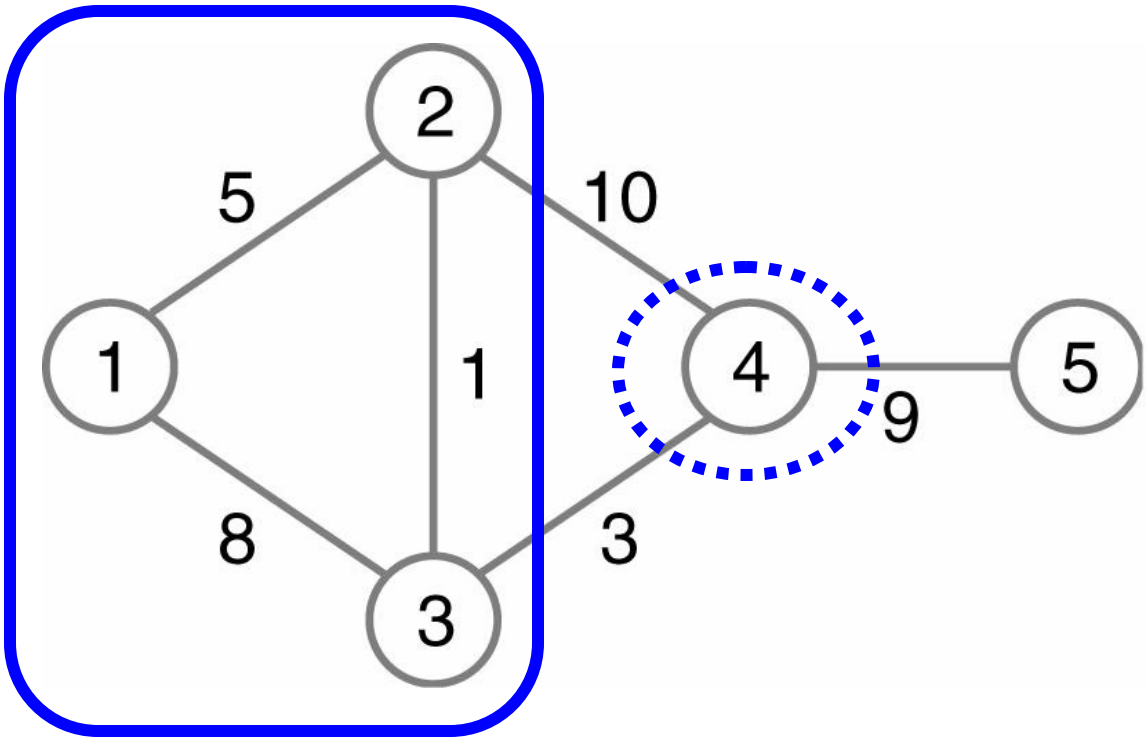


C[4] = min (15 , C[3] + (3,4)) = min (15 , 6 + 3) = 9

Dijkstra's Shortest Path Example

- Find node K with smallest C[K] and add to S
- { S } = 1, 2, 3, 4

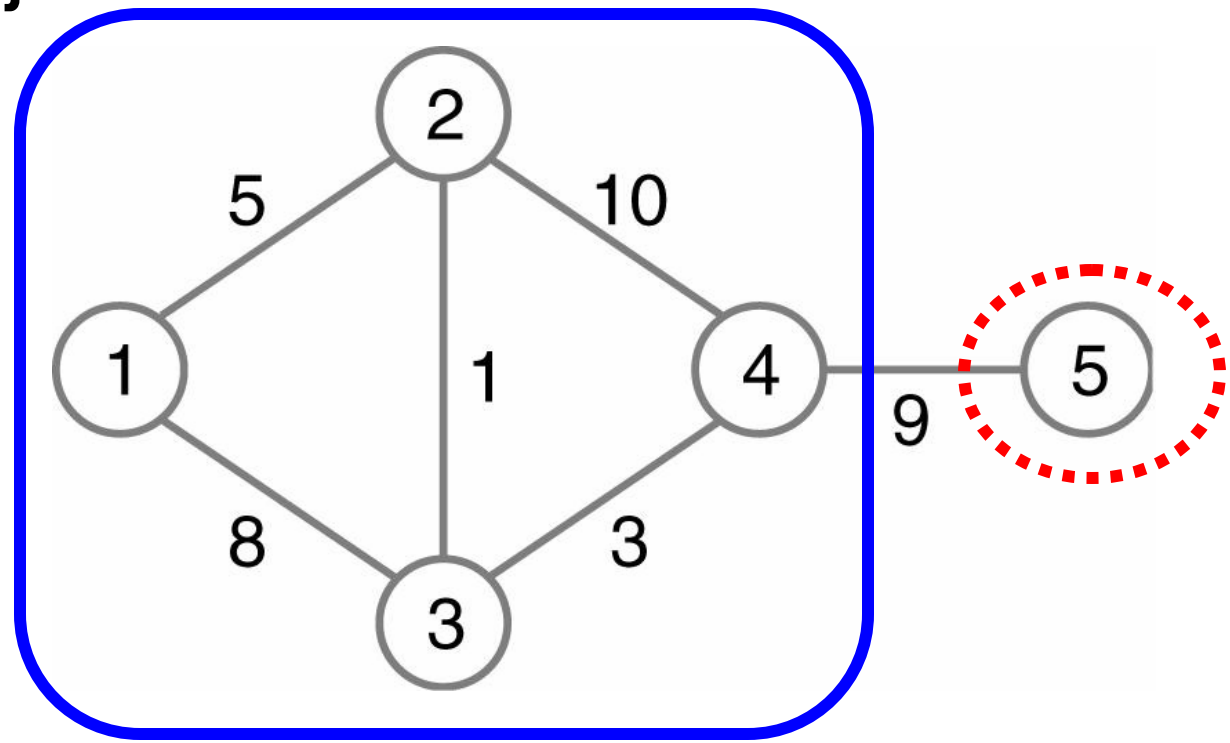
	C	P
1	0	none
2	5	1
3	6	2
4	9	3
5	∞	none



Dijkstra's Shortest Path Example

- Update C[K] for all neighbors of 4 not in S
- S = { 1, 2, 3, 4 }

	C	P
1	0	none
2	5	1
3	6	2
4	9	3
5	18	4

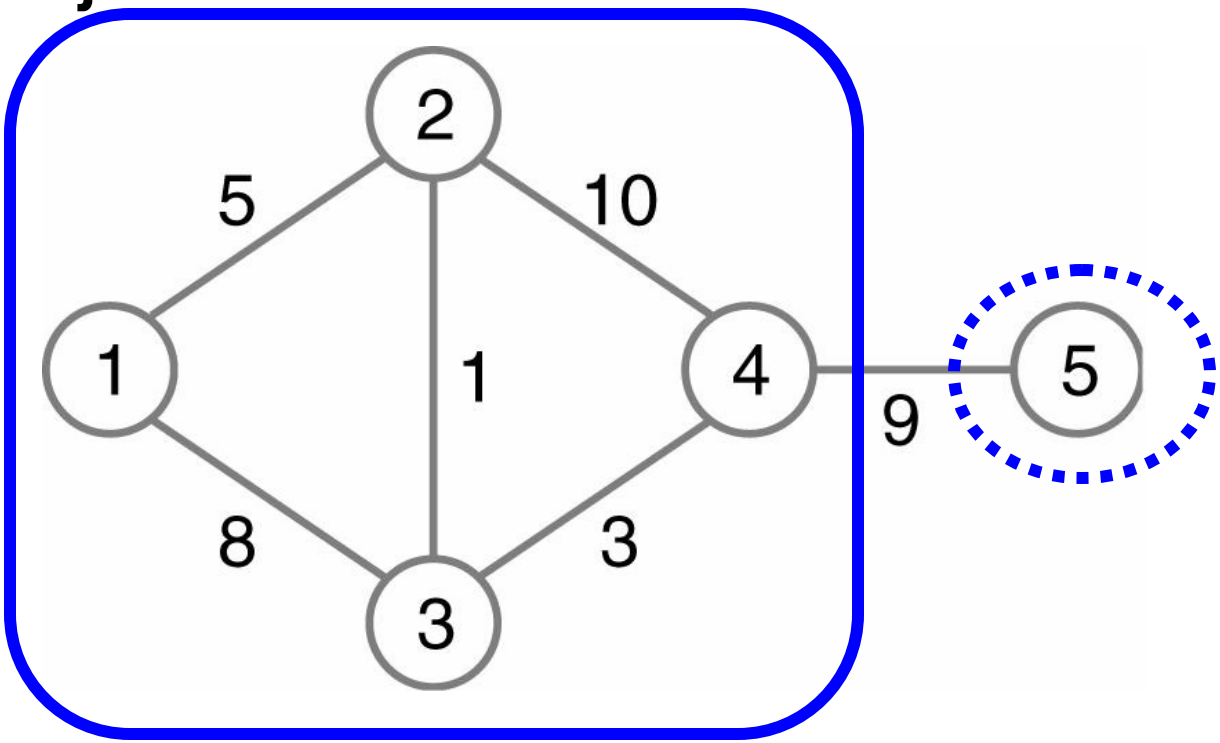


C[5] = min (∞ , C[4] + (4,5)) = min (∞ , 9 + 9) = 18

Dijkstra's Shortest Path Example

- Find node K with smallest C[K] and add to S
- S = { 1, 2, 3, 4, 5 }

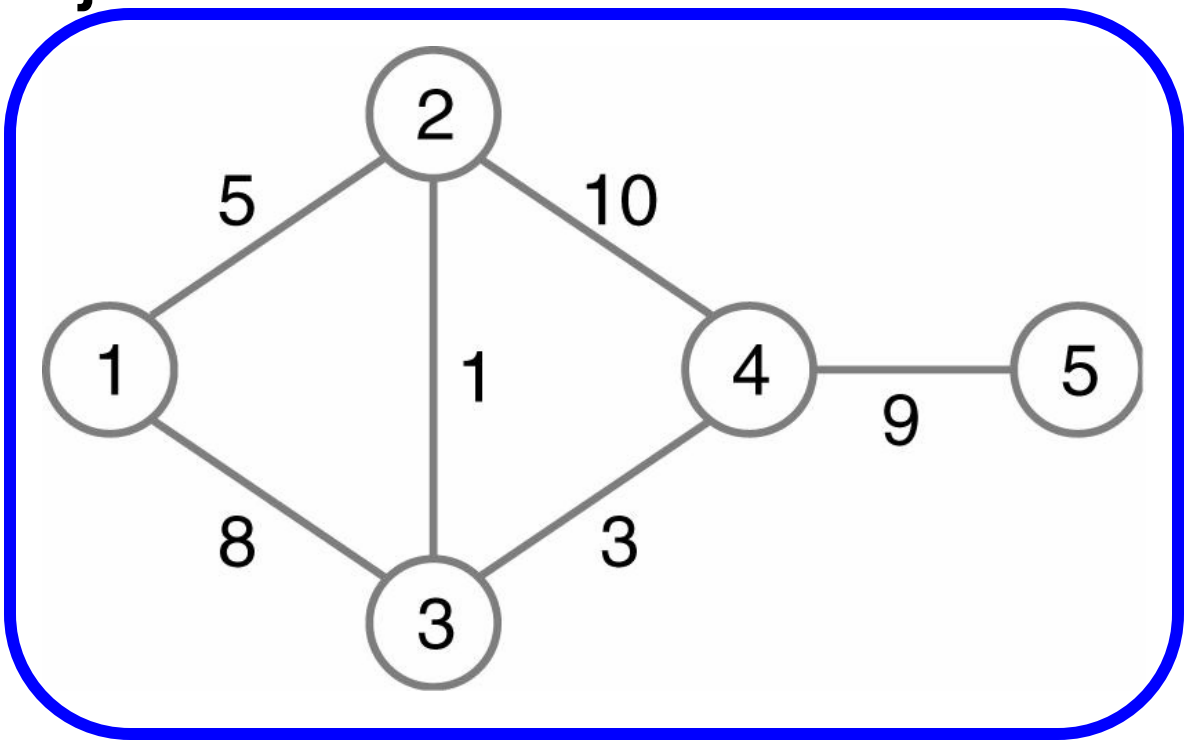
	C	P
1	0	none
2	5	1
3	6	2
4	9	3
5	18	4



Dijkstra's Shortest Path Example

- All nodes in S, algorithm is finished
- S = { 1, 2, 3, 4, 5 }

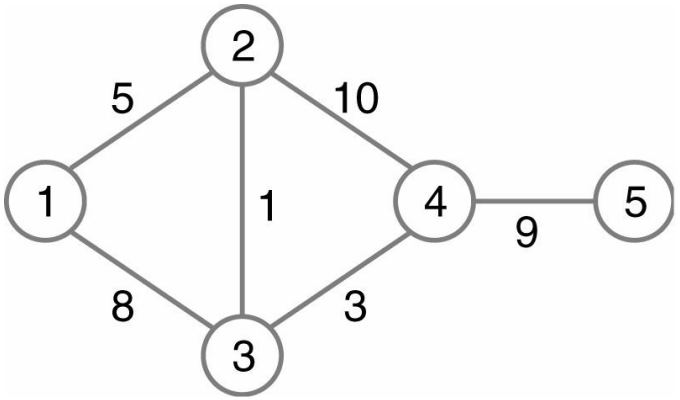
	C	P
1	0	none
2	5	1
3	6	2
4	9	3
5	18	4



Dijkstra's Shortest Path Example

- Find shortest path from start to K
  - Start at K
  - Trace back predecessors in P[ ]
- Example paths (in reverse)
  - 2 → 1
  - 3 → 2 → 1
  - 4 → 3 → 2 → 1
  - 5 → 4 → 3 → 2 → 1

	C	P
1	0	none
2	5	1
3	6	2
4	9	3
5	18	4



Algorithm animations

- **Good animation of Dijkstra's algorithm at**
  - **http://www-b2.is.tokushima-u.ac.jp/~ikeda/suuri/dijkstra/Dijkstra.shtml**
  - **Also has animations of Prim's and Kruskal's algorithms for minimal spanning trees**