

CMSC 132: Object-Oriented Programming II



Graph Implementations & Single Source Shortest Path Algorithm

Department of Computer Science
University of Maryland, College Park

1

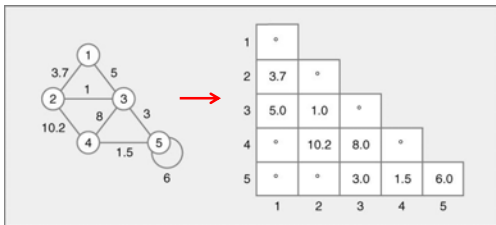
Graph Implementation

- How do we represent edges?
 - Adjacency matrix
 - 2D array of neighbors
 - Adjacency list
 - List of neighbors
 - Adjacency set / map
 - Set / map of neighbors
- Important for very large graphs
 - Affects efficiency / storage

2

Adjacency Matrix

- Representation
 - 2D array
 - Position $j, k \Rightarrow$ edge between nodes n_j, n_k
- Example



3

Adjacency Matrix

- Representation (cont.)
 - Single array for entire graph
 - Undirected graph
 - Only upper / lower triangle matrix needed
 - Since n_j, n_k implies n_k, n_j
 - Unweighted graph
 - Matrix elements \Rightarrow boolean
 - Weighted graph
 - Matrix elements \Rightarrow weight

4

Adjacency List

- Representation
 - For each node, store
 - List of neighbors / successors
 - Linked list
 - Array list
 - For weighted graph
 - Also store weight for each edge
 - For undirected graph with edge ($a \leftrightarrow b$)
 - Nodes a & b need to store each other as neighbor
 - For directed graph with edge ($a \rightarrow b$)
 - Node a needs to store node b as neighbor

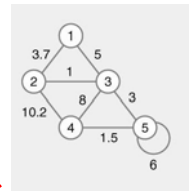
5

Adjacency List

Example

Unweighted graph

Node	Neighbor List
1	2 \rightarrow 3
2	1 \rightarrow 3 \rightarrow 4
3	1 \rightarrow 2 \rightarrow 4 \rightarrow 5
4	2 \rightarrow 3 \rightarrow 5
5	3 \rightarrow 4 \rightarrow 5



Weighted graph

Node	Neighbor List
1	(2, 3.7) \rightarrow (3, 5.0)
2	(1, 3.7) \rightarrow (3, 1.0) \rightarrow (4, 10.2)
3	(1, 5.0) \rightarrow (2, 1.0) \rightarrow (4, 8.0) \rightarrow (5, 3.0)
4	(2, 10.2) \rightarrow (3, 8.0) \rightarrow (5, 1.5)
5	(3, 3.0) \rightarrow (4, 1.5) \rightarrow (5, 6.0)

6

Adjacency Set / Map

- Representation
 - For each node, store
 - Set or map of neighbors / successors
 - For unweighted graph
 - Use **set** of neighbors
 - For weighted graph
 - Use **map** of neighbors, w/ value = weight of edge
 - For undirected graph with edge (a↔b)
 - Nodes a & b need to store each other as neighbor
 - For directed graph with edge (a→b)
 - Node a needs to store node b as neighbor

7

Graph Space Requirements

- Adjacency matrix
 - $\frac{1}{2} N^2$ entries (for graph with N nodes, E edges)
 - Many empty entries for large, sparse graphs
- Adjacency list
 - $2 \times E$ entries
- Adjacency set / map
 - $2 \times E$ entries
 - Space overhead per entry
 - Higher than for adjacency list

8

Graph Time Requirements

- Adjacency matrix
 - Can find individual edge (a,b) quickly
 - Examine entry in array Edge[a,b]
 - Constant time operation
- Adjacency list / set / map
 - Can find all edges for node (a) quickly
 - Iterate through collection of edges for a
 - On average E / N edges per node

9

Graph Time Requirements

- Average Complexity of operations
 - For graph with N nodes, E edges

Operation	Adj Matrix	Adj List	Adj Set/Map
Find edge	$O(1)$	$O(E/N)$	$O(1)$
Insert edge	$O(1)$	$O(E/N)$	$O(1)$
Delete edge	$O(1)$	$O(E/N)$	$O(1)$
Enumerate edges for node	$O(N)$	$O(E/N)$	$O(E/N)$

10

Choosing Graph Implementations

- Graph density
 - Ratio edges to nodes (dense vs. sparse)
- Graph algorithm
 - Neighbor based
 - For each node X in graph
 - For each neighbor Y of X // adj list faster if sparse
 - doWork()
 - Connection based
 - For each node X in ...
 - For each node Y in ...
 - if (X,Y) is an edge // adj matrix faster if dense
 - doWork()

11

Single Source Shortest Path

- Common graph problem
 1. Find path from X to Y with lowest edge weight
 2. 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
- Dijkstra's algorithm solves problem 2
 - Can also be used to solve problem 1
 - Would use different algorithm if only interested in a single destination

12

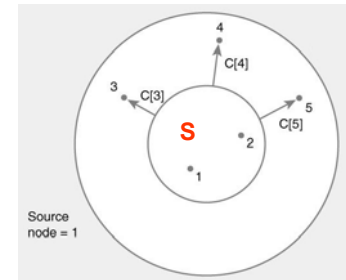
Shortest Path – Dijkstra’s Algorithm

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

13

Shortest Path – Intuition for Dijkstra’s

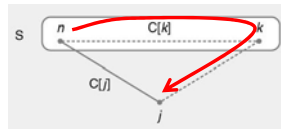
- **At each step in the algorithm**
 - Shortest paths are known for nodes in S
 - Store in $C[K]$ length of shortest path to node K (for all paths through nodes in { S })
 - Add to { S } next closest node



14

Shortest Path – Intuition for Dijkstra’s

- **Update distance to J after adding node K**
 - Previous shortest path to K already in $C[K]$
 - Possibly shorter path to J by going through node K
 - Compare $C[J]$ with $C[K] + \text{weight of } (K,J)$, update $C[J]$ if needed



15

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

```

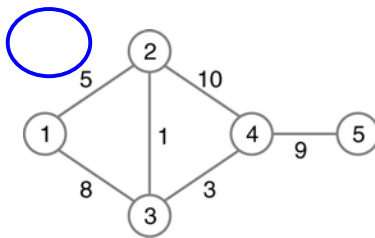
*Optimal solution computed with greedy algorithm*

16

## Dijkstra’s Shortest Path Example

- **Initial state**
- $S = \emptyset$

|   | C        | P    |
|---|----------|------|
| 1 | 0        | none |
| 2 | $\infty$ | none |
| 3 | $\infty$ | none |
| 4 | $\infty$ | none |
| 5 | $\infty$ | none |

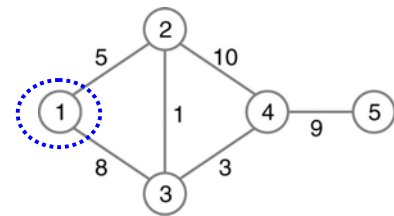


17

## Dijkstra’s Shortest Path Example

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

|   | C        | P    |
|---|----------|------|
| 1 | 0        | none |
| 2 | $\infty$ | none |
| 3 | $\infty$ | none |
| 4 | $\infty$ | none |
| 5 | $\infty$ | none |

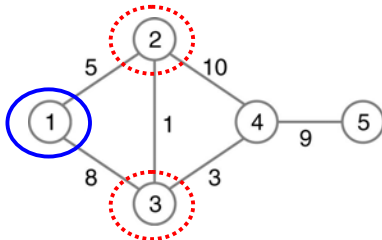


18

### Dijkstra'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 |



$$C[2] = \min(\infty, C[1] + (1,2)) = \min(\infty, 0 + 5) = 5$$

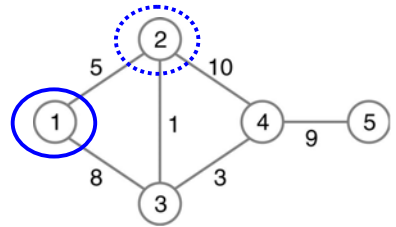
$$C[3] = \min(\infty, C[1] + (1,3)) = \min(\infty, 0 + 8) = 8$$

19

### Dijkstra'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 |

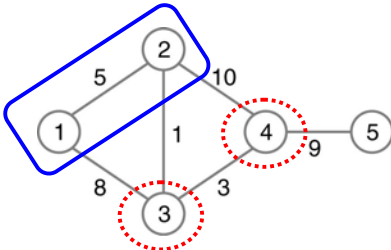


20

### 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 |



$$C[3] = \min(8, C[2] + (2,3)) = \min(8, 5 + 1) = 6$$

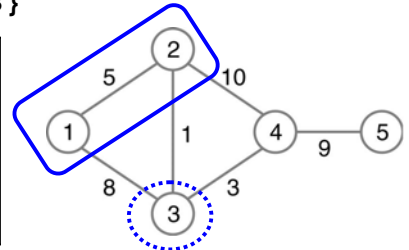
$$C[4] = \min(\infty, C[2] + (2,4)) = \min(\infty, 5 + 10) = 15$$

21

### 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 |

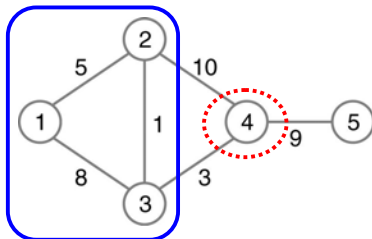


22

### 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 |



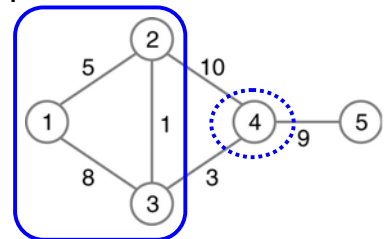
$$C[4] = \min(15, C[3] + (3,4)) = \min(15, 6 + 3) = 9$$

23

### 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 |

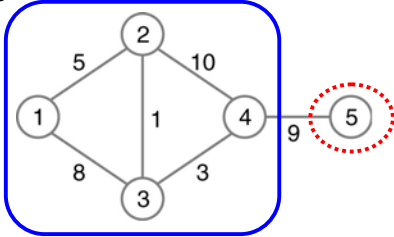


24

### 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    |



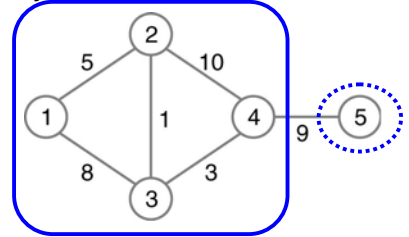
$$C[5] = \min(\infty, C[4] + (4,5)) = \min(\infty, 9 + 9) = 18$$

25

### 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    |

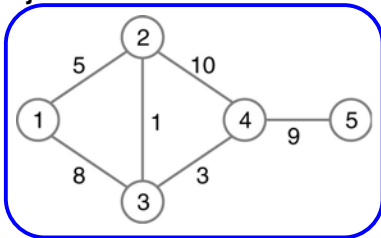


26

### 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    |

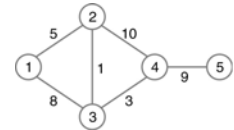


27

### 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    |



28