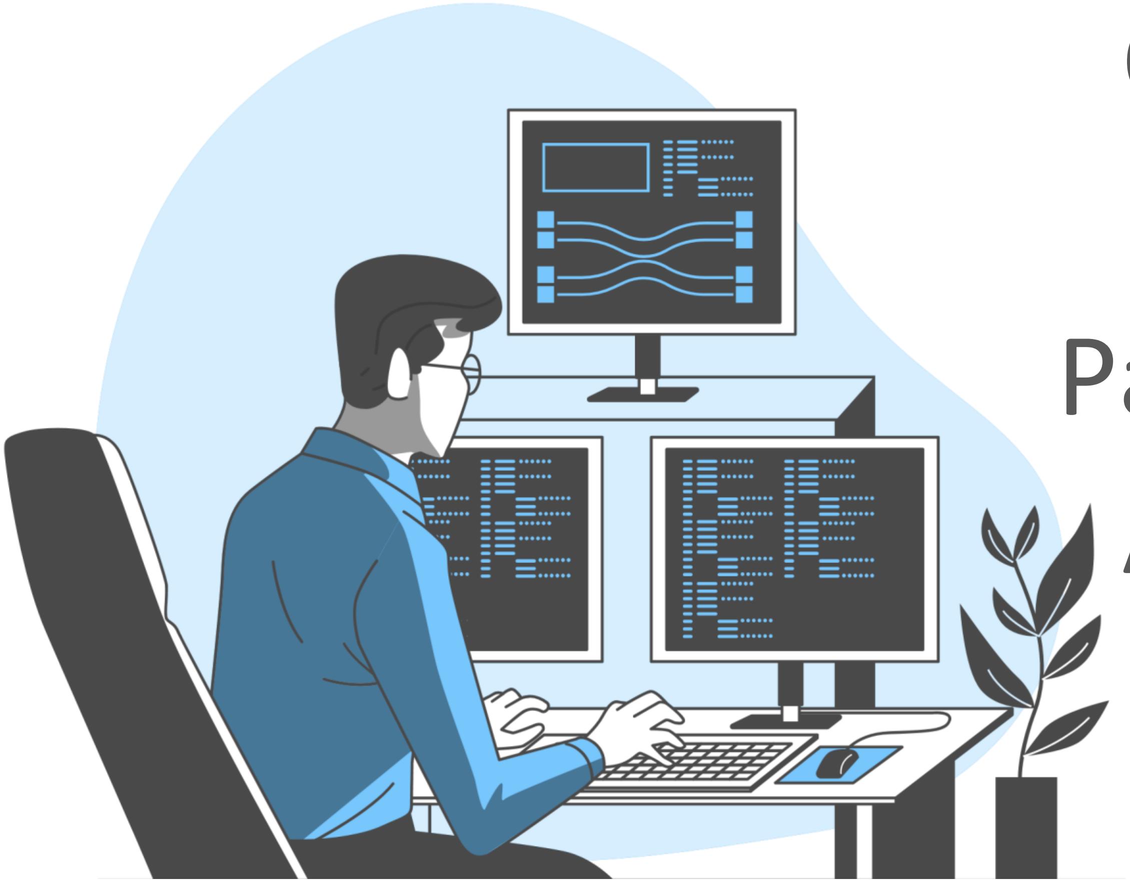


CZ2001 Lab Project 2- Path Finding Algorithms

Objectives

To propose algorithms to find:

1. **Shortest path** from each node to any hospital nodes (a & b)
2. **k nearest** hospital nodes for each node (c & d)

Table of Contents

1. Algorithm 1

(Dependent on number of hospitals)

- Design and Execution
- Complexity Analysis

2. Algorithm2

(Not dependent on number of hospitals)

- Design and Execution
- Complexity Analysis

3. Comparison & Conclusion

- Empirical analysis in different cases
- Conclusion: Recommended algorithm in different situations

01

Algorithm 1

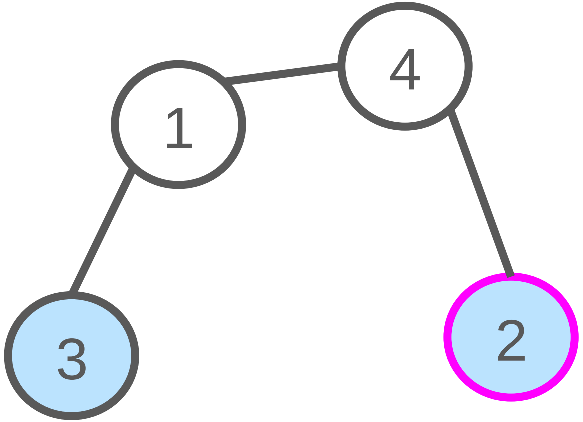
Dependent on number of hospitals

Concept

Hospital Node

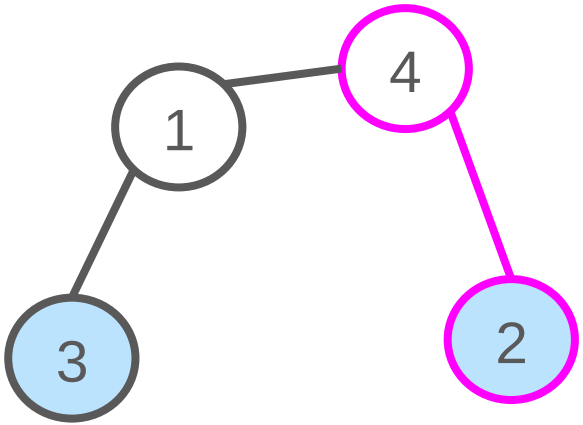
- When a node is encountered during BFS **starting from a hospital node**
- Shortest path from the normal node to the hospital node is found
- But, this is not the shortest path to **any hospital node**

Normal Node



Concept

- When a node is encountered during BFS **starting from a hospital node**
- Shortest path from the normal node to the hospital node is found



Hospital Node

Normal Node

Paths:
4->2

- But, this is not the shortest path to **any hospital node**

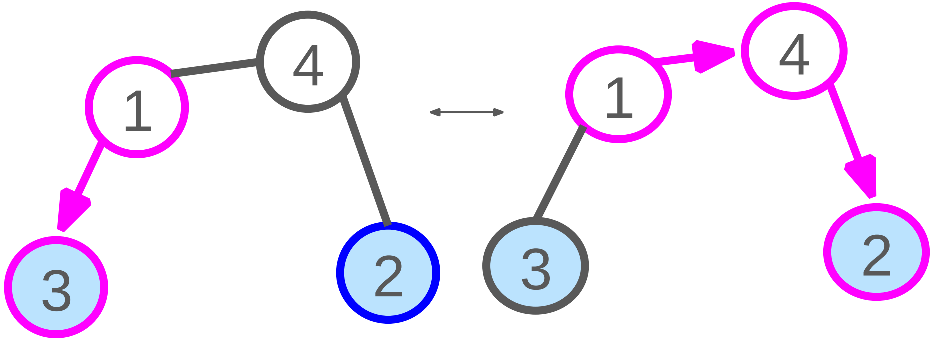
Hospital Node

Concept

- To get the shortest path to **any**

Normal Node

hospital:
  - Carry out BFS for **each hospital node**
  - Record down path if there are **no previous paths**



- Replace the previous path if the **new path is shorter**

New path

Distance: 1

BFS from 3

Previous path

Distance: 2

BFS from 2

...

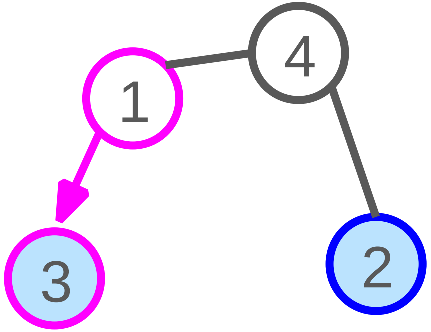
...

Hospital Node

Concept

- To get the shortest path to **any** Normal Node hospital:

- Carry out BFS for **each hospital node**
- Record down path if there are **no previous paths**
- Replace the previous path if the **new path is shorter**

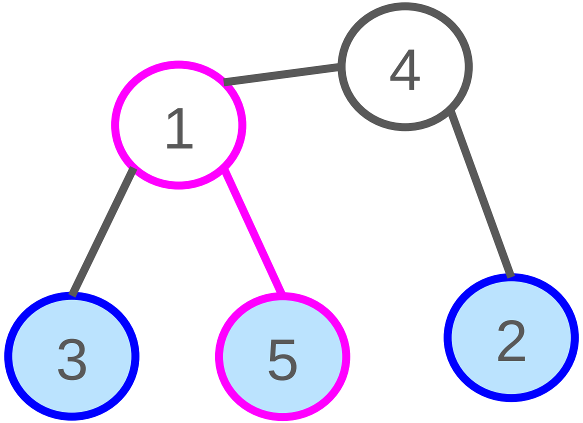


New path

Distance: 1

BFS from 3

Concept (k nearest hospitals)



Hospital Node

- Similar concept
- Only store **hospital node** &

Normal Node **distance**

- Maintain a list of hospital nodes and distances ordered in **increasing distance** ○ Linear Search

Paths from node 1:

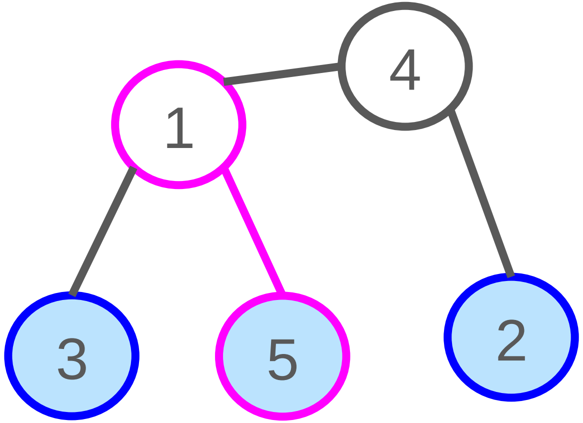
(Node3, 1)	(Node2, 2)
	(Node5, 1)

Concept (k nearest hospitals)

Hospital Node

- Similar concept
- Only store **hospital node** & **distance**

Normal Node



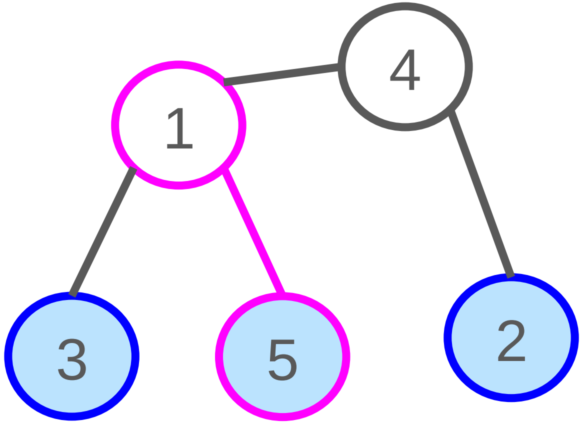
- Maintain a list of hospital nodes and distances ordered in **increasing distance**
  - Linear Search

Paths from node 1:

(Node3, 1)	(Node2, 2)
(Node5, 1)	

Concept (k nearest hospitals)

- Similar concept
- Only store **hospital node** & Normal Node **distance**
- Maintain a list of hospital nodes
and distances ordered in k=2 **increasing distance** ○ Linear Search

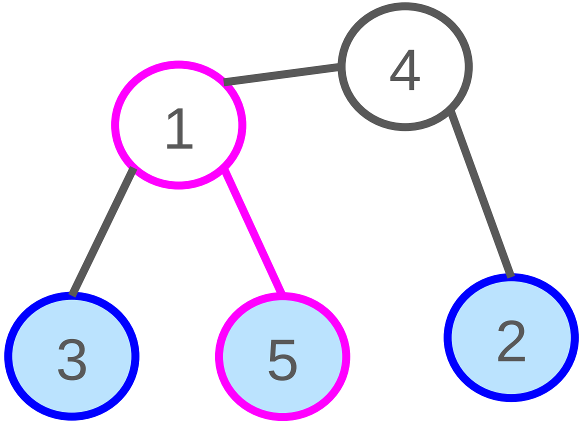


Paths from node 1:

(Node3, 1)	(Node5, 1)	(Node2, 2)

Concept (k nearest hospitals)

Hospital Node



- Similar concept
- Only store **hospital node** &

  Normal Node **distance**

- Maintain a list of hospital nodes

  and distances ordered in k=2 **increasing distance** ○ Linear Search

Paths from node 1:

(Node3, 1)	(Node5, 1)

Time Complexity

- Since we are using an adjacency list:
  - BFS for each hospital: O(|V|+|E|)
- (a) Path finding: O(h[|V|+|E|])
- (c) & (d) Path finding: O(h[k|V|+|E|])

h=Number of Hospitals

|E|=Number of Edges

|V|=Number of Vertices

k=Number of Nearest Hospitals

- (a): Output path: O(|V|)
  - Iterate through all nodes to output path
- (c) & (d): Output path: O(k|V|)
  - For each node output k hospitals and distances

Space Complexity

- Adjacency list: O(|V|+|E|)

- (a): Output dictionary: $O(|V|^2)$
  - {node1: [2,3,5], node2: [4,3,1]}
  - Each node has a list of nodes (path)

- (c) & (d): Output dictionary: O(k|V|)

h=Number of Hospitals |E|=Number of Edges |V|=Number of Vertices k=Number of Nearest Hospitals

- {node1: [(hospital1, dist1), (hospital2, dist2)]}
- Each node has a list of k hospital nodes and distances

02

Algorithm 2

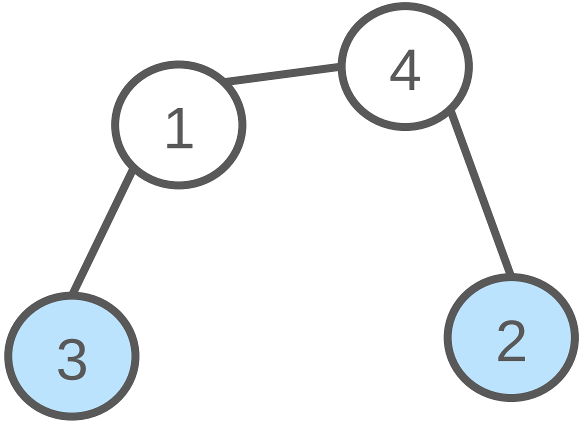
Independent on number of hospitals

Concept

Hospital Node

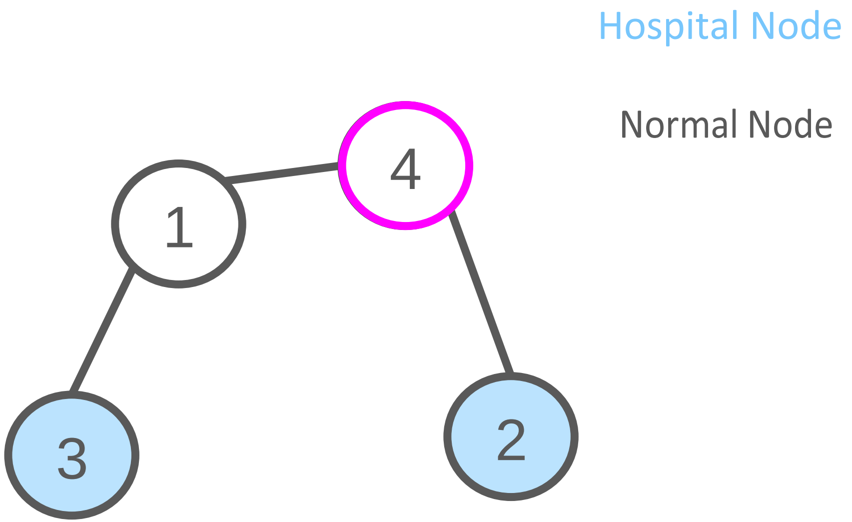
Normal Node

- Breadth-first searching for all nodes in the graphs (excluding hospital nodes)
- Stop searching when a hospital node is marked visited
- A shortest path from a source node to a hospital is found



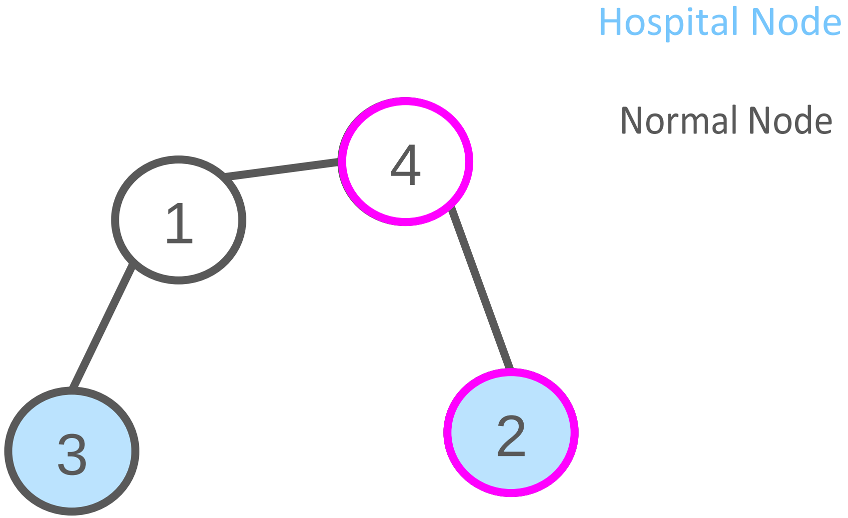
Concept

- Breadth-first searching for all nodes in the graphs (excluding hospital nodes)
- Stop searching when a hospital node is marked visited
- A shortest path from a source node to a hospital is found



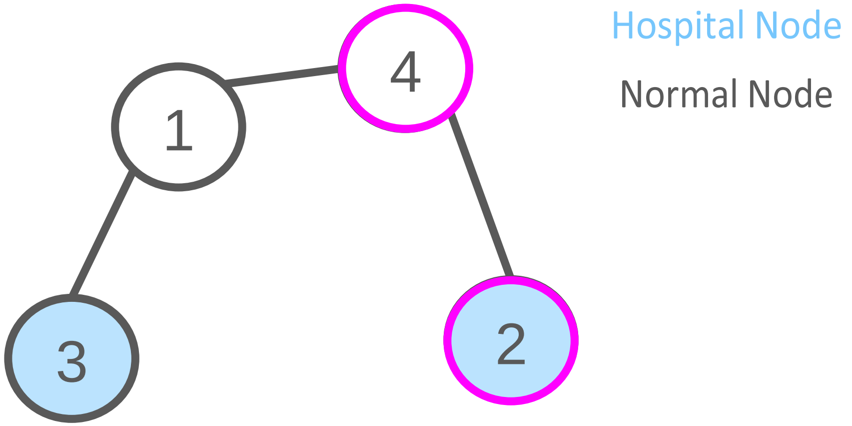
Concept

- Breadth-first searching for all nodes in the graphs (excluding hospital nodes)
- Stop searching when a hospital node is marked visited
- A shortest path from a source node to a hospital is found



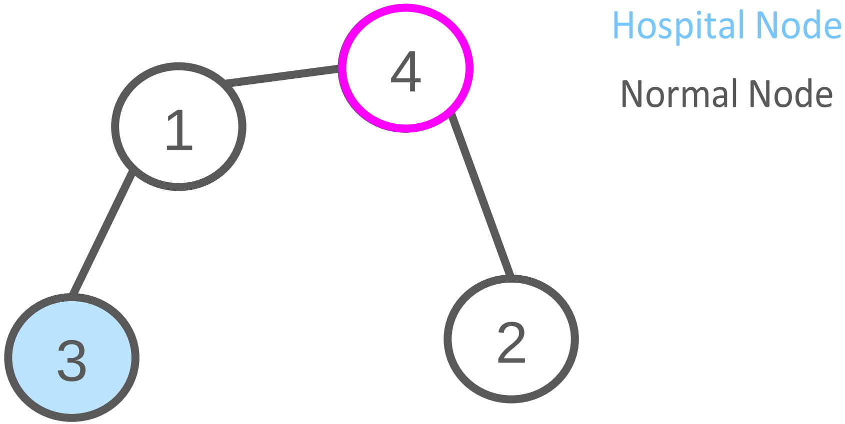
Concept (k nearest hospitals)

- Similar concept
- Update list of hospitals by removing the reached hospital from the list
- Iterate k times for top k nearest hospitals



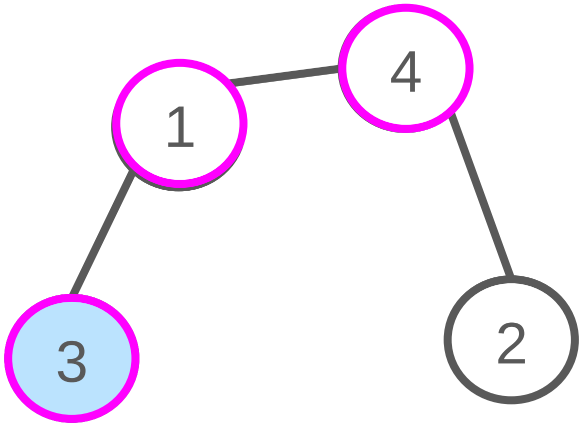
Concept (k nearest hospitals)

- Similar concept
- Update list of hospitals by removing the reached hospital from the list
- Iterate k times for top k nearest hospitals



Concept (k nearest hospitals)

- Similar concept
- Update list of hospitals by removing the reached hospital from the list
- Iterate k times for top k nearest hospitals



Hospital Node

Normal Node

Time Complexity

- Since we are using an adjacency list:
  - BFS for each source vertex: $O(|V|+|E|)$

- (a) & (b) Path finding: $\mathbf{O(|V|[|V|+|E|])}$
- (c) & (d) Path finding: $\mathbf{O(k|V|[|V|+|E|])}$

Space Complexity

- Adjacency list: $O(|V|+|E|)$
- During execution of algorithm, no additional space is needed to store information.

03

Comparison and Conclusion

Changing k value for top-k nearest hospitals

Searching Time VS k Value

Changing number of nodes

Search Time VS Number of Nodes

Changing number of hospital nodes

Searching Time VS Number of Hospitals

Conclusion

Algorithm 1

Use when:

Algorithm 2

Use when:

- Has large amount of nodes in total
- Has high number of k
(reduce fluctuations)
- Has large amount of hospital nodes (high hopital:total nodes ratio)

- Has limited space to run algorithm