CZ2001
Graph Algorithms

Group members: Goh Wei Pin, Tan Keng Kai Luke, Jonathan Chang Ji Ming, Lynn Masillamoni

Contents

Breadth First Search (BFS)

Starts at a selected node, and explores all of the neighbor nodes at the present depth prior to moving on to the nodes at the next depth level

Parameters used:

1. startingNode
2. requiredHospitals
3. hospitalList
4. outputPath
5. adjacencyList

Why Adjacency List?

1. Assume that graph is a sparse with less edges compared to the number of nodes
2. Better average case for space complexity:
   adjacency list = $O(|V| + |E|)$ vs adjacency matrix = $O(|V|^2)$.
3. Better time complexity than adjacency matrix: $O(|V|^2)$ vs $O(|V|^3)$

Breadth First Search (BFS)

```
def BFS(adjacencyList, startingNode, requiredHospitals, hospitalList, outputPath):
    queue = [[startingNode]] #declare an empty list 'queue'
    visited = [] #declare an empty list to store nodes that have been visited

    while (queue):
        path = queue.pop(0)
        currentNode = path[-1]

        if (currentNode not in visited):
            try:
                neighbours = adjacencyList[currentNode]
                for neighbour in neighbours:
                    newPath = list(path)
                    newPath.append(neighbour)
                    queue.append(newPath)
            except:
                pass  # currentNode has no neighbours!

        visited.append(currentNode)

        if (currentNode in hospitalList):
            Print the output into the outputPath
            Reduce the number of required hospitals by 1
            Remove currentNode from hospitalList

        break out of loop when all hospitals have been found
```

Input

Graph

containing 10 nodes

```
FROM TO:
0 2
4 0
8 0
9 0
6 1
1 8
2 5
5 2
3 7
3 8
8 3
5 6
5 9
6 7
7 6
```

Number and position of hospitals

3 hospitals, at positions 2, 5 and 6

```
#3

2

5

6
```

Output

Find the nearest 2 hospitals and print its distance and path for each node

```
start node: 0, shortest dist: 1, shortest path: [0,2]
start node: 0, shortest dist: 2, shortest path: [0,2,5]
start node: 4, shortest dist: 2, shortest path: [4,0,2]
start node: 4, shortest dist: 3, shortest path: [4,0,2,5]
start node: 8, shortest dist: 2, shortest path: [8,0,2]
start node: 8, shortest dist: 3, shortest path: [8,0,2,5]
start node: 9, shortest dist: 2, shortest path: [9,0,2]
start node: 9, shortest dist: 3, shortest path: [9,0,2,5]
start node: 6, shortest dist: 0, shortest path: [6]
start node: 6, shortest dist: 4, shortest path: [6,1,8,0,2]
start node: 1, shortest dist: 3, shortest path: [1,8,0,2]
start node: 1, shortest dist: 4, shortest path: [1,8,0,2,5]
start node: 2, shortest dist: 0, shortest path: [2]
start node: 5, shortest dist: 1, shortest path: [2,5]
start node: 5, shortest dist: 0, shortest path: [5]
start node: 2, shortest dist: 1, shortest path: [5,2]
start node: 3, shortest dist: 2, shortest path: [3,7,6]
start node: 3, shortest dist: 3, shortest path: [3,8,0,2]
start node: 7, shortest dist: 1, shortest path: [7,6]
start node: 7, shortest dist: 5, shortest path: [7,6,1,8,0,2]
```

Time Complexity

- Worst case complexity is $O(|V|^2)$ where $|V|$ is the number of vertices and $|E|$ is the number of edges
    - Each function call will have to go through the entire graph with time complexity $O(|V|)$
- Best case complexity is $O(|V|)$ where each node in the graph either has no neighbours or is the hospital
    - Each function call will have time complexity $O(2k-1)$ for k-hospitals
- Time complexity is only dependent on number of vertices,$|V|$, and not on the number of hospitals

Empirical Analysis

- Vary number of hospitals *h* with constant *k*-nearest hospitals to find
- Vary *k*-nearest hospitals to find with constant number of hospitals h
- Compare time taken to find shortest distance on graphs of different density
- All experiments were ran 100 times with a constant 1000 nodes to display the mean and median time taken to find the shortest path

Varying h

- k is a constant 2 and h is varied, taking values 4, 6 and 10
- Time taken to find shortest path decreases when h increases because there is a higher probability that a visited node is a hospital

h=4, k=2	h=6, k=2	h=10, k=2
Mean: 7.8113890600204465 Median: 7.219055414199829 SD: 2.53452765172542	Mean: 4.84510908126831 Median: 4.567678213119507 SD: 1.3419949465033418	Mean: 3.1465510058403017 Median: 3.0554988384246826 SD: 0.4887073474514526

Varying k

- h is a constant 6, and k is varied, taking values 1, 3 and 5
- time taken increases when k increases as algorithm will be finding more paths
- Time complexity is still independent of k

h=6, k=1	h=6, k=3	h=6, k=5
Mean: 1.9108275246620179 Median: 1.8630259037017822 SD: 0.3923597614318506	Mean: 8.292898228168488 Median: 8.038352727890015 SD: 1.6379449627796014	Mean: 25.988025462627412 Median: 24.44281554222107 SD: 10.179279438258437

Varying density

- Time taken increases when graph becomes denser
  - This is because as |E| increases, we cannot assume graph is sparse such that |E| is no longer in the same order as |V|
  - Time complexity of each function call is $O(|V|(|V|+|E|))$, as graph becomes more dense such that $|E| \approx |V|(|V|-1)$, overall time complexity is $O(|V|^3)$

p=0.01	p=0.2	p=0.5
Mean: 8.84615241765976 Median: 7.908851265907288 SD: 3.172104030941215	Mean: 46.41162770032883 Median: 46.36962449550629 SD: 6.726000918072522	Mean: 115.14650383472443 Median: 115.733132600784З SD: 20.64199025464202

Real World Applications

- Differences in densities of real-world graphs
  - Road network (sparse) vs Social network (dense)
  - Diminishing advantages of using adjacency list

Thank You