Session 9: Reachability Matrix & All Pair Shortest Path

Mahesh Bharadwaj K - 185001089

February 27, 2020

1 Reachabilty Matrix Generation

```
import numpy as np

v = int(input())
e = int(input())

adjacency = np.array([[False for j in range(v)] for i in range(v)])

for edge in range(e):
    ip = tuple(map(int,input().split(' ')))
    adjacency[ip[0] - 1][ip[1] - 1] = True

for k in range(v):
    for i in range(v):
        for j in range(v):
            adjacency[i][j] = (adjacency[i][j]) or (adjacency[i][k] and adjacency[k][j])

print(adjacency)
```

Output Generated

- Graph_ip1.txt is a graph with **100 vertices and 1000 edges**
- Graph_ip2.txt is a graph with **4 vertices and 4 edges**

```
python3 Reachability.py < graph_ip1.txt
[[False  True False ...  True  True  True]
 [False False False ...  True  True  True]
 [False False False ...  True  True  True]
 ...
 [False False False ... False  True False]
 [False False False ... False False False]
 [False False False ... False False False]]

python3 Reachability.py < graph_ip2.txt
[[ True  True  True  True]
 [ True  True  True  True]
 [ True  True  True  True]
 [False False False False]]
```

---

2 All Pair Shortest Path Code

```
import math
import numpy as np

v = int(input())
e = int(input())

distance = np.array([[math.inf for j in range(v)] for i in range(v)])
previous = np.array([[None for j in range(v)] for i in range(v)])

for edge in range(e):
    ip = tuple(map(int,input().split(' ')))
    distance[ip[0] - 1][ip[1] - 1] = ip[2]
    previous[ip[0] - 1][ip[1] - 1] = ip[0] - 1

for k in range(v):
    for i in range(v):
        for j in range(v):
            if distance[i][j] > (distance[i][k] + distance[k][j]):
                distance[i][j] = distance[i][k] + distance[k][j]
                previous[i][j] = k

print('The Distance Matrix Is:\n')
print(distance)

print('\n\nThe Previous Matrix Is: \n')
print(previous)


def path_display(source,dest):
    global previous

    if dest == source:
        return str(source+1)

    if previous[source][dest] is None:
        return 'No path from %d to %d' % (source + 1,dest + 1)

    return path_display(source,previous[source][dest]) + ' --> ' + str(dest + 1)

print('\nThe paths are:')

for i in range(v):
    for j in range(v):
        print(path_display(i,j))
    print('------------------------------')
```

Output Generated

- Graph_ip1.txt is a graph with **100 vertices and 1000 edges**
- Graph_ip2.txt is a graph with **4 vertices and 4 edges**

```
python3 AllPairShortest.py < graph_ip1.txt | head -40
The Distance Matrix Is:

[[  inf 1985.   inf ... 2046. 2705. 5659.]
 [  inf   inf   inf ...   61. 3143. 4154.]
 [  inf   inf   inf ... 2496. 3546. 2648.]
 ...
 [  inf   inf   inf ...   inf 3082.   inf]
 [  inf   inf   inf ...   inf   inf   inf]
 [  inf   inf   inf ...   inf   inf   inf]]


The Previous Matrix Is:

[[None 0 None ... 1 81 88]
 [None None None ... 1 97 63]
 [None None None ... 63 81 88]
 ...
```

```
 [None None None ... None 97 None]
 [None None None ... None None None]
 [None None None ... None None None]]

The paths are:
1
1 --> 2
No path from 1 to 3
No path from 1 to 4
1 --> 2 --> 5
1 --> 2 --> 6
1 --> 2 --> 5 --> 7
1 --> 2 --> 6 --> 8
1 --> 2 --> 5 --> 9
1 --> 2 --> 6 --> 10
1 --> 2 --> 6 --> 10 --> 11
1 --> 2 --> 12
1 --> 2 --> 6 --> 13
No path from 1 to 14
1 --> 2 --> 6 --> 15
1 --> 2 --> 16
1 --> 2 --> 5 --> 17
1 --> 18


The Distance Matrix Is:

[[11.  4.  6. 15.]
 [ 7. 11.  2. 11.]
 [ 5.  9. 11.  9.]
 [inf inf inf inf]]
....


The Previous Matrix Is:

[[2 0 1 2]
 [2 2 1 2]
 [2 0 1 2]
 [None None None None]]

The paths are:
1
1 --> 2
1 --> 2 --> 3
1 --> 2 --> 3 --> 4
--------------------------------
2 --> 3 --> 1
2
2 --> 3
2 --> 3 --> 4
--------------------------------
3 --> 1
3 --> 1 --> 2
3
3 --> 4
--------------------------------
No path from 4 to 1
No path from 4 to 2
No path from 4 to 3
4
--------------------------------
```