深 圳 大 学 实 验 报 告

课程名称：计算机网络（Computer Networks）

实验名称：Network Layer Assignment

学院：电子与信息工程学院

专业：电子信息工程

指导教师：毕宿志

报告人：贾苏健 班级：06 学号：2022280485

实验时间：2023.12.08

实验报告提交时间：2023.12.15

教务部制

1. Purpose of experiment

(1) Implementing Algorithms:Learn to implement Dijkstra's algorithm and Distance Vector algorithm using Python.

(2) Understanding Network Routing:Understand the application of Dijkstra's algorithm and Distance Vector algorithm in calculating the shortest paths and distances between network nodes.

(3) Solving Network Problems:Be capable of handling input for a given network topology and specified node, and outputting the distance list and paths from that node to other nodes.

2. Experimental principle

(1) Dijkstra algorithm.

1) Initialization:

Maintain two pieces of information for each node: the current known shortest path length (distance) from the starting node to that node and the previous node on that shortest path.

Set the distance from the starting node to itself to 0, and distances to other nodes are initially set to infinity (or a very large value).

2) Greedy Selection:

At each step, select an unmarked node whose currently known shortest path has the minimum value.

Begin with the starting node as the initial choice.

3) Relaxation Operation:

For the selected node, calculate the path lengths from the starting node, passing through that node, to other adjacent nodes.

If the path through the selected node is shorter than the known path, update the distance and path information for the adjacent nodes.

This step progressively extends the known shortest paths in a greedy manner.

4) Marking Nodes:

Mark the selected node as visited to ensure it won't be reconsidered.

5) Repeat Steps:

Repeat the above steps until all nodes are marked as visited or the distances of all marked nodes are infinity.

(2) Distance-Vector algorithm.

1) Initialization:

Each node initializes its distance vector to other nodes, typically setting the distance to itself as 0 and the distance to other nodes as infinity.

Each node also keeps track of the next-hop node information for each destination, indicating the next node on the shortest path to the target.

2) Information Exchange:

Nodes periodically send their distance vectors to neighboring nodes.

Upon receiving a distance vector, a node updates its own distance vector and propagates the updated information to its neighbors.

3) Distance Update:

When a node receives a distance vector from a neighbor, it compares the received distance vector with its existing one.

If the received distance vector provides a shorter path, the node updates its own distance vector and sets the next-hop node to be the sending node.

4) Iteration:

The algorithm iterates through the process of information exchange and distance updates until the distance vectors of all nodes in the network converge to their minimum values.

As the distance vectors of nodes continuously update, each node gradually gains knowledge of the shortest paths to reach other nodes in the network.

5) Path Retrieval:

Once the algorithm converges, nodes can use the distance vector table to determine the shortest paths to any other node in the network.

Utilizing the next-hop information in the distance vector table, nodes can forward packets along the shortest paths.

3. Content

(1) Dijkstra algorithm.

1) Initialization:

Utilize a priority queue (min-heap) to expedite the search for the shortest path. Initialize distance dictionary and path list.

Here I choose to use the heapq library.

The main benefit of using import heapq is its support for the heap data structure, providing convenience for implementing the priority queue (min-heap) in the Dijkstra algorithm. Through heapq, heap operations become simpler and more intuitive, making the code for the Dijkstra algorithm clearer and more readable. Additionally, heapq supports a min-heap, aiding in the quick selection of the node with the current minimum distance. Its implementation based on a binary heap offers higher performance optimization, speeding up operations such as insertion and extraction of the minimum value, thereby enhancing the overall efficiency of the algorithm.

```
import heapq
```

2) Dijkstra Algorithm:

Employ a greedy approach by selecting nodes using a priority queue. Conduct relaxation operations on neighbors, updating distance and path. The heap structure ensures the selection of nodes with the smallest distance.

```
# 使用优先队列（最小堆）来加速查找最短路径
priority_queue = [(0, start)]

while priority_queue:
    current_distance, current_node = heapq.heappop(priority_queue)
```

3) Path Retrieval:

Employ recursive methods to obtain paths, outputting the shortest path and distance from each node to other nodes.

4) Application:

Execute the Dijkstra algorithm on the specified network structure and node. Output the distance list and path from the designated node to other nodes.

(2) Distance-Vector algorithm.

1) Initialization:

Initialize the distance table and next-hop table. Utilize the network topology to initialize distance and next-hop information.

2) Relaxation Operations:

Iteratively compute and update the distance table and next-hop table. Check for the existence of shorter paths and update information accordingly.

3) Distance Vector Algorithm:

Iterate a number of times equal to the number of nodes, performing relaxation operations on all nodes in each iteration.

```
def distance_vector_algorithm(self):
    # 距离矢量算法的主循环
    for _ in range(self.num_nodes):  # 迭代次数
        for source in range(self.num_nodes):
            for destination in range(self.num_nodes):
                for intermediate in range(self.num_nodes):
                    self.relax(source, destination, intermediate)
```

4) Path Retrieval:

Obtain paths recursively, outputting the shortest path and distance from each node to other nodes.

5) Application:

Execute the distance vector algorithm on the specified network topology and node. Output the distance list and path from the designated node to other nodes.

4. Conclusion and discussion

(1) Dijkstra algorithm.

```
E:\software\Python_all\Python\python.exe F:\Projec
Distance from node 0 to other nodes:
{0: 0, 1: 1.0, 2: 4.0, 3: 5.0, 4: 10.0, 5: 7.0}

Shortest paths:
No path from node 0 to node 0
Path from node 0 to node 1: 0 -> 1
Path from node 0 to node 2: 0 -> 1 -> 2
Path from node 0 to node 3: 0 -> 1 -> 3
Path from node 0 to node 4: 0 -> 1 -> 2 -> 4
Path from node 0 to node 5: 0 -> 1 -> 3 -> 5
Distance from node 0 to node 1: 1.0
Distance from node 0 to node 2: 4.0
Distance from node 0 to node 3: 5.0
Distance from node 0 to node 4: 10.0
Distance from node 0 to node 5: 7.0

 ==============================
==============================
Distance from node 2 to other nodes:
{0: 4.0, 1: 3.0, 2: 0, 3: 2.0, 4: 6.0, 5: 4.0}

Shortest paths:
Path from node 2 to node 0: 2 -> 1 -> 0
Path from node 2 to node 1: 2 -> 1
No path from node 2 to node 2
Path from node 2 to node 3: 2 -> 3
Path from node 2 to node 4: 2 -> 4
Path from node 2 to node 5: 2 -> 3 -> 5
Distance from node 2 to node 0: 4.0
Distance from node 2 to node 1: 3.0
Distance from node 2 to node 3: 2.0
Distance from node 2 to node 4: 6.0
Distance from node 2 to node 5: 4.0

==============================
 ==============================
 Distance from node 4 to other nodes:
 {0: 10.0, 1: 9.0, 2: 6.0, 3: 8.0, 4: 0, 5: 10.0}

 Shortest paths:
 Path from node 4 to node 0: 4 -> 2 -> 1 -> 0
 Path from node 4 to node 1: 4 -> 2 -> 1
 Path from node 4 to node 2: 4 -> 2
 Path from node 4 to node 3: 4 -> 2 -> 3
 No path from node 4 to node 4
 Path from node 4 to node 5: 4 -> 2 -> 3 -> 5
 Distance from node 4 to node 0: 10.0
 Distance from node 4 to node 1: 9.0
 Distance from node 4 to node 2: 6.0
 Distance from node 4 to node 3: 8.0
 Distance from node 4 to node 5: 10.0

 ==============================
```

```
------------------------------
Distance from node 1 to other nodes:
{0: 1.0, 1: 0, 2: 3.0, 3: 4.0, 4: 9.0, 5: 6.0}

Shortest paths:
Path from node 1 to node 0: 1 -> 0
No path from node 1 to node 1
Path from node 1 to node 2: 1 -> 2
Path from node 1 to node 3: 1 -> 3
Path from node 1 to node 4: 1 -> 2 -> 4
Path from node 1 to node 5: 1 -> 3 -> 5
Distance from node 1 to node 0: 1.0
Distance from node 1 to node 2: 3.0
Distance from node 1 to node 3: 4.0
Distance from node 1 to node 4: 9.0
Distance from node 1 to node 5: 6.0

==============================
 ==============================
 Distance from node 3 to other nodes:
 {0: 5.0, 1: 4.0, 2: 2.0, 3: 0, 4: 8.0, 5: 2.0}

 Shortest paths:
 Path from node 3 to node 0: 3 -> 1 -> 0
 Path from node 3 to node 1: 3 -> 1
 Path from node 3 to node 2: 3 -> 2
 No path from node 3 to node 3
 Path from node 3 to node 4: 3 -> 2 -> 4
 Path from node 3 to node 5: 3 -> 5
 Distance from node 3 to node 0: 5.0
 Distance from node 3 to node 1: 4.0
 Distance from node 3 to node 2: 2.0
 Distance from node 3 to node 4: 8.0
 Distance from node 3 to node 5: 2.0

 ==============================
==============================
Distance from node 5 to other nodes:
{0: 7.0, 1: 6.0, 2: 4.0, 3: 2.0, 4: 10.0, 5: 0}

Shortest paths:
Path from node 5 to node 0: 5 -> 3 -> 1 -> 0
Path from node 5 to node 1: 5 -> 3 -> 1
Path from node 5 to node 2: 5 -> 3 -> 2
Path from node 5 to node 3: 5 -> 3
Path from node 5 to node 4: 5 -> 3 -> 2 -> 4
No path from node 5 to node 5
Distance from node 5 to node 0: 7.0
Distance from node 5 to node 1: 6.0
Distance from node 5 to node 2: 4.0
Distance from node 5 to node 3: 2.0
Distance from node 5 to node 4: 10.0

==============================
```

(2) Distance-Vector algorithm.

```
E:\software\Python_all\Python\python.exe F:\Proj
Distance from node 0 to other nodes:
{0: 0, 1: 1.0, 2: 4.0, 3: 5.0, 4: 10.0, 5: 7.0}

Shortest paths:
No path from node 0 to node 0
Path from node 0 to node 1: 0 -> 1
Path from node 0 to node 2: 0 -> 1 -> 2
Path from node 0 to node 3: 0 -> 1 -> 3
Path from node 0 to node 4: 0 -> 1 -> 2 -> 4
Path from node 0 to node 5: 0 -> 1 -> 3 -> 5
Distance from node 0 to node 1: 1.0
Distance from node 0 to node 2: 4.0
Distance from node 0 to node 3: 5.0
Distance from node 0 to node 4: 10.0
Distance from node 0 to node 5: 7.0

==============================
------------------------------
Distance from node 2 to other nodes:
{0: 4.0, 1: 3.0, 2: 0, 3: 2.0, 4: 6.0, 5: 4.0}

Shortest paths:
Path from node 2 to node 0: 2 -> 1 -> 0
Path from node 2 to node 1: 2 -> 1
No path from node 2 to node 2
Path from node 2 to node 3: 2 -> 3
Path from node 2 to node 4: 2 -> 4
Path from node 2 to node 5: 2 -> 3 -> 5
Distance from node 2 to node 0: 4.0
Distance from node 2 to node 1: 3.0
Distance from node 2 to node 3: 2.0
Distance from node 2 to node 4: 6.0
Distance from node 2 to node 5: 4.0

==============================

==============================
Distance from node 4 to other nodes:
{0: 10.0, 1: 9.0, 2: 6.0, 3: 8.0, 4: 0, 5: 10.0}

Shortest paths:
Path from node 4 to node 0: 4 -> 2 -> 1 -> 0
Path from node 4 to node 1: 4 -> 2 -> 1
Path from node 4 to node 2: 4 -> 2
Path from node 4 to node 3: 4 -> 2 -> 3
No path from node 4 to node 4
Path from node 4 to node 5: 4 -> 2 -> 3 -> 5
Distance from node 4 to node 0: 10.0
Distance from node 4 to node 1: 9.0
Distance from node 4 to node 2: 6.0
Distance from node 4 to node 3: 8.0
Distance from node 4 to node 5: 10.0

==============================
```

```
==============================
Distance from node 1 to other nodes:
{0: 1.0, 1: 0, 2: 3.0, 3: 4.0, 4: 9.0, 5: 6.0}

Shortest paths:
Path from node 1 to node 0: 1 -> 0
No path from node 1 to node 1
Path from node 1 to node 2: 1 -> 2
Path from node 1 to node 3: 1 -> 3
Path from node 1 to node 4: 1 -> 2 -> 4
Path from node 1 to node 5: 1 -> 3 -> 5
Distance from node 1 to node 0: 1.0
Distance from node 1 to node 2: 3.0
Distance from node 1 to node 3: 4.0
Distance from node 1 to node 4: 9.0
Distance from node 1 to node 5: 6.0

==============================
==============================
Distance from node 3 to other nodes:
{0: 5.0, 1: 4.0, 2: 2.0, 3: 0, 4: 8.0, 5: 2.0}

Shortest paths:
Path from node 3 to node 0: 3 -> 1 -> 0
Path from node 3 to node 1: 3 -> 1
Path from node 3 to node 2: 3 -> 2
No path from node 3 to node 3
Path from node 3 to node 4: 3 -> 2 -> 4
Path from node 3 to node 5: 3 -> 5
Distance from node 3 to node 0: 5.0
Distance from node 3 to node 1: 4.0
Distance from node 3 to node 2: 2.0
Distance from node 3 to node 4: 8.0
Distance from node 3 to node 5: 2.0

==============================

  ==============================
  Distance from node 5 to other nodes:
  {0: 7.0, 1: 6.0, 2: 4.0, 3: 2.0, 4: 10.0, 5: 0}

  Shortest paths:
  Path from node 5 to node 0: 5 -> 3 -> 1 -> 0
  Path from node 5 to node 1: 5 -> 3 -> 1
  Path from node 5 to node 2: 5 -> 3 -> 2
  Path from node 5 to node 3: 5 -> 3
  Path from node 5 to node 4: 5 -> 3 -> 2 -> 4
  No path from node 5 to node 5
  Distance from node 5 to node 0: 7.0
  Distance from node 5 to node 1: 6.0
  Distance from node 5 to node 2: 4.0
  Distance from node 5 to node 3: 2.0
  Distance from node 5 to node 4: 10.0

  ==============================
```

指导教师批阅意见：

成绩评定：

指导教师签字：

年　　月　　日

备注：