|  |
| --- |
| 1. **Purpose of experiment**   The purpose of this experiment is to use Python to implement two routing algorithms: Dijkstra algorithm and Distance vector algorithm, and calculate the shortest Distance and path from the specified node to other nodes by simulating the network topology.   1. **Experimental principle**   Dijkstra algorithm is a greedy algorithm, which is able to find the shortest path from the source node to all other nodes in a directed or undirected graph. The basic idea of the algorithm is to maintain a set of visited nodes and a set of unvisited nodes, as well as a distance vector that records the current shortest distance from the source node to each node. Initially, only the source node is added to the set of visited nodes, and its distance is 0, and the distance of other nodes is infinite. Then, each time, a node with the minimum distance is selected from the set of never visited nodes, and it is added to the set of visited nodes, and the distance of its neighbor nodes is updated if a shorter path can be obtained through the node. This process is repeated until either all nodes are visited or the destination node is visited. But Dijkstra needs weights of all nodes are positive.  Distance-Vector algorithm is a distributed algorithm, it can let each router update its routing table according to the information of neighbor routers in a network, so as to achieve the global optimal routing. The basic idea of the algorithm is that each router maintains a distance vector that records the shortest distance to each destination and the next hop router. Initially, each router only knows its distance to its directly connected neighbors. Then, each router periodically sends its distance vector to its neighbors and receives the distance vector of its neighbors. Based on the received information, each router updates its own distance vector if a shorter path to a certain destination is found. This process iterates until the distance vectors of all routers are stable.   1. **Content**   (1)Dijkstra  Implement the main logic of Dijkstra algorithm. The script should define a function called dijkstra that takes two arguments: a graph and a source. graph is a dictionary representing a directed or undirected graph, where the keys are the names of the nodes and the values are a list representing the weights of the neighbors and edges of the node.  The graph each nodes’ weights to their neighbours.  graph = {'0': [('1', 1), ('3', 6)],  '1': [('0', 1), ('2', 3), ('3', 4)],  '2': [('1', 3), ('3', 2), ('4', 6)],  '3': [('0', 6), ('1', 4), ('2', 2), ('4', 9), ('5', 2)],  '4': [('2', 6), ('3', 9)],  '5': [('3', 3)]}  The function should return two dictionaries: distance and path. distance is a dictionary representing the shortest distance from the source node to each node. For example, distance = {' 0 ': 0,' 1 ': 1,' 2 ': 4,' 3 ': 5,' 4 ': 10, ‘ 5 ’: 7} means that the shortest distance from 0 to 1 is 1, the shortest distance from 0 to 2 is 4, and so on. path is a dictionary representing the shortest path from the source node to each node. For example,  The detailed node path from the least cost path from node 0 to 5 is:  ['0', '1', '3', '5'] says the shortest path from 0 to 5 is 0 - > 1 - > 3 - > 5.  (2)Distance vector  he Distance-Vector algorithm  Implement the main logic of the Distance-Vector algorithm. The script should define a function called distance\_vector that takes three arguments: graph and source. A graph is a dictionary representing a network topology, where the keys are the names of the nodes and the values are a list representing the weights of the neighbors and edges of the node. The format of this parameter is the same as in Experiment I. source is a string that represents the name of the source node. The function should return two dictionaries: distance and path. distance is a dictionary representing the shortest distance from the source node to each node. path is a dictionary representing the shortest path from the source node to each node. The format of this parameter is the same as in Experiment I.  At the end of the script, add a test to verify the correctness of the distance\_vector function. Using the graph above to test the codes. If the result is same as the first experiment, the codes are correctly written. Then add the loop to print all the path and distance vector of all nodes. |
| 1. **Conclusion and discussion**   Through this experiment, we have mastered the principle and implementation method of two commonly used routing algorithms, and have written the corresponding scripts using Python language, simulated the situation of network topology, and calculated the shortest distance and path from the specified node to other nodes. We find that both the Dijkstra algorithm and the distation-vector algorithm can effectively solve the shortest path problem, but they also have their own advantages and disadvantages. The advantage of Dijkstra algorithm is that it can quickly find the optimal solution, but the disadvantage is that it needs global information and can not deal with negative weight edges. The advantage of the Distance-Vector algorithm is that it only needs local information and can deal with edges with negative weights, but the disadvantage is that the convergence speed is slow and the counting to infinity problem may occur. Therefore, in the actual network, it is necessary to choose the appropriate routing algorithm according to different scenarios and requirements |