Research on the Best Method of Shortest Path to Solve Network Communication Problem

Wen-rong JIANG* and Tao-zi SI
School of Computer and Information, Shanghai Polytechnic University, Shanghai, China
*Corresponding author

Keywords: Dijkstra algorithm, Shortest path, Network communication, Research, Graph theory.

Abstract. As the basic theory of shortest path calculation, Dijkstra algorithm is widely used in various aspects, such as: transportation, construction of signal station, laying of high-speed rail. Based on graph theory, this paper studies Dijkstra algorithm to analyze network communication problems, introduces the principle of Dijkstra algorithm in detail, and makes a contribution to the research of shortest path.

Introduction
The shortest path is to find the shortest path among many routes through calculation. However, the shortest path in the data structure is different from the shortest path in the geographical location in our usual sense. The shortest path in the data structure is also based on the actual needs, considering the choice of traffic routes, pipeline laying and other problems. Therefore, it is of great significance in urban planning and transportation and logistics[1]. Therefore, the research on the shortest path algorithm of data structure has become a hot topic nowadays, and how to apply it has become a problem that relevant staff need to think deeply.

Basic Tool-graph Theory
Graph is a data structure that is more complex than linear tables and trees. The difference between the graph and the structure of the linear table and the tree is shown in the relationship between nodes. In the linear table, every node has a predecessor node and a successor node (the first node is out), and there is a one-to-one correspondence between nodes. Every node in the tree has its corresponding parent and one or more child nodes (leaf nodes out), and the relationship between nodes belongs to one-to-many relationship. Different from the above two node modes, the node relationship in the figure is many-to-many. Nodes can be connected to each other, and each node can choose to connect or not to connect.

There are many kinds of graph storage structures, such as adjacency matrix, adjacency list, adjacency multiple list, and cross adjacency list. In addition to storing information about individual vertex in the graph, the relationships between vertex are also stored.

Shortest Path Analysis under Data Structure
The weighted directed graph is analyzed to obtain the shortest path from V0 to each point, as shown in the figure. As you can see from the figure, there is a path from V0 to V4:(V0,V4) with a length of 10. There's no path from V0 to V1.

An auxiliary vector D is introduced, each of its components D[i] representing the length of the shortest path of the currently found starting point V to each end point Vi. His initial state is: If there is an arc from V to Vi, then D[i] is the weight on the arc; otherwise, D[i] is set to ∞. Apparently the length is

\[ D[a] = \text{Min} \{ D[i] \mid Vi \in V \} \]

The path of D[a] is the shortest path with the shortest length from V. This path is (V, Va).
Analysis of Dijkstra Algorithm

The Dijkstra algorithm is a typical single-source shortest path algorithm used to calculate the shortest path from one node to all other nodes. The main feature is that the starting point is centered on the outer layer until it extends to the end point. The Dijkstra algorithm is a representative shortest path algorithm, which is introduced in detail in many professional courses, such as data structure, graph theory, operations research and so on. Note that the algorithm requires no negative edges in the graph. Algorithm idea: Let $G=(V, E)$ be a weighted directed graph, divide the vertex set $V$ in the graph into two groups, and the first group is the set of vertices that have obtained the shortest path (indicated by $S$, initially $S$ there is only one source point. Each time a shortest path is obtained, it will be added to the set $S$ until all the vertices are added to $S$, and the algorithm ends. The second set is the set of vertices of the remaining undetermined shortest paths. $U$ indicates) that the vertices of the second group are sequentially added to $S$ in increasing order of the shortest path length. During the joining process, the shortest path length that always maintains the vertices from the source point $v$ to $S$ is not greater than the shortest path length from the source point $v$ to any of the vertices in $U$. In addition, each vertex corresponds to a distance, the distance of the vertex in $S$ is the shortest path length from $v$ to this vertex, and the distance of the vertex in $U$ is from $v$ to this vertex including only the vertex in $S$ is the current vertex of the middle vertex. It’s the shortest path length.

Analysis of Floyd Algorithm

The shortest path between each point pair $u$ and $v$ may pass $N$ points, which are denoted as $k$. Assuming that the shortest path between $u$ and $k$ has been found, and the shortest path between $k$ and $v$ has been found, then the shortest path between $u$ and $v$ is to traverse the possible $k$ points and then ask $(u,k)$ The minimum value between $(k,v)$. So this actually divides the large-scale problem from the top down to the small-scale problem, which is the dynamic planning idea. The Floyd algorithm is a classic dynamic programming algorithm. To describe in plain language, first of all our goal is to find the shortest path from point $i$ to point $j$. From the perspective of dynamic programming, we need to re-interpret this goal (this interpretation is the essence of the most creative of dynamic programming). The shortest path from any node $i$ to any node $j$ is nothing more than two possibilities. 1 is directly from $i$ to $j$, and 2 is from $i$ through several nodes $k$ to $j$. Therefore, we assume that $Dis(i,j)$ is the distance of the shortest path from node $u$ to node $v$. For each node $k$, we check $Dis(i,k) + Dis(k,j) < Dis(i,j)$ Whether it is established, if it is established, prove that the path from $i$ to $k$ to $j$ is shorter than the path from $i$ to $j$, set $Dis(i,j) = Dis(i,k) + Dis(k,j)$, thus When traversing all nodes $k$, $Dis(i, j)$ records the distance of the shortest path from $i$ to $j$. 

![Figure 1. Weighted directed graph.](image)
Analysis of Network Communication Problems

Network Communication Problems

(1) The input will describe the topology of the network to which n processors are connected. The first line of input will be n, the number of processors, such that 1 <= n <= 100. The remaining inputs define an adjacency matrix, which is the square and the size n x n. Each entry of it will be an integer or the character x. The value of (i, j) represents the cost of sending a message directly from node to node j. A value of x (i, j) indicates that a message cannot be sent directly from node to node j. Note that for a node to send a message to itself does not require network communication, so for 1 <= i <= n, then a(i, i) = 0. Therefore, only the items on the lower triangular portion of A are provided (strictly). The input to the program will be the lower triangle of A, that is, the second line input will contain an entry A(2,1). The next line will contain two entries, A(3,1) and A(3,2), and so on.

(2) Topic analysis:

Dijkstra algorithm Angle analysis: open container v to store child nodes, distance and cost; Open array dis to record the distance from the starting point to each point; Perform n-1 relaxation operation (first find the nearest point to the starting point in the unmarked point, mark the point, and then calculate the shortest distance (priority) and shortest cost from the node to the starting point); The shortest distance to the destination;

Perspective analysis of Floyd algorithm: at the beginning, only 1 vertex is allowed to be transferred, and then only 1 and 2 vertex are allowed to be transferred. It allows the shortest distance between any two points to be dynamically updated through all vertex of 1-n. Find the shortest distance from vertex I to vertex j that only goes through the first k.

Algorithm Design and Code Analysis

(1) (Dijkstra algorithm) program code as follows:

```
... (code snippet) ...
```
Figure 3. The program code of Dijkstra algorithm.

(2) (Floyd algorithm) program code is as follows:

```c
#define MAXN 102
int map[MAXN][MAXN];

int main() {
    int n;
    cin >> n;
    for (int i=1;i<=n+1;i++) {
        string s;
        cin>>s;
        for (int j=1;j<=n;j++) {
            if(s[0]==x) {
                tmp=stringToNum<int>(s);
                map[i][j]=map[i][j]+tmp;
            }
        }
    }
    for (int k=1;k<=n+1;k++) {
        for (int j=1;j<=n+1;j++) {
            map[j][k]=min(map[j][k],map[j][i]+map[i][k]);
        }
    }
    int time=-1;
    for (int i=1;i<=n+1;i++) {
        time=time>map[i][i]?time : map[i][i];
    }
    cout<<time<<endl;
    return 0;
}
```

Figure 4. The program code of Floyd algorithm.

Compare the two algorithms (1) and (2):

1) Analysis of time complexity: By analyzing the code, the time complexity of the Dijkstra algorithm on this problem is $O(N^2)$, and the time complexity of the Floyd algorithm is $O(N^3)$, then the relative It is said that the calculation time of the Floyd algorithm is relatively long.

2) analysis of code running time:

By running two algorithms, the following three figures can be obtained:
According to the screenshot of the operation results of the two algorithms, Dijkstra algorithm is relatively short in operation time, and Dijkstra algorithm is better for the shortest path calculation of this problem.

**The Advantages and Disadvantages of the Two Algorithms are Compared**

Floyd algorithm is $O(N^3)$ in time, but with only five lines of code, it's pretty easy to implement. Because it is so easy to implement, it is possible to use Floyd to specify the shortest path between two points or the shortest path from one point to each other if the time complexity is not too high. Dijkstra algorithm has relatively low time complexity and is more suitable for the operation of a large number of nodes.

**Summary**

Dijkstra algorithm can quickly and conveniently calculate the shortest path of a specific problem. This algorithm is based on graph theory, which is convenient for programmers to use Dijkstra algorithm to solve practical problems in the most concise and fast way. It is not only relatively low complexity, but also has a visual solution to practical problems. Floyd algorithm is relatively easy to understand, but relatively high complexity, not suitable for algorithm optimization.

**Acknowledgment**

This work is supported by the Key Disciplines of Computer Science and Technology of Shanghai Polytechnic University under Grant No. XXKZD1604.
References


