Dijkstra algorithm and its real-life use processes

Sotimboeva Zarifakhon

Senior Lecturer, Department of Higher Mathematics, Namangan State University

Algoritm

For each v peak, let's take the D[v] array, which stores the current length of the shortest path from S to V. Initially d [s] = 0, and for all other peaks, this length is equal to infinity (when done on a computer, usually a sufficiently large number is chosen as Infinity, which is greater than the possible length of the exact path):

In addition, for each v peak, we keep whether this peak was used or not, that is, we get a logical U[] array. Initially, not all peaks were used, i.e.

The Dijkstra algorithm itself consists of n iterations. In the next iteration, a peak with the smallest value d[v] is selected among the V peaks that have not yet been used, i.e.:

(It is clear that in the first iteration the initial C peak is selected)

After that the selected V peak is marked as used. In addition, in the current iteration, walks from peak v are carried out: all (v, to) edges protruding from peak v are checked, and for each such peak to, the algorithm tries to improve the value of D[to]. Let the length of the current Edge be len, then the implementation of walks in the form of a code looks like this:

Here the current iteration ends, the algorithm goes to the next iteration (again, a peak with the smallest value is selected d, from which walks are made, etc.). In this case, at the end, after N repetitions, all the ends of the graph are used, and the algorithm completes its work. It is noted that the found values \ u200b \ u200bare the required length of the shortest paths from s to v d[v].

It should be noted that if it is not possible to get from peak C to all peaks of the count, then for them the value d[v] remains infinite. It is clear that the last few iterations of the algorithm choose only these peaks, but these iterations do not bring any useful work (since infinite distance cannot free up other, even infinite distances). Therefore, as soon as a peak with an infinite distance is perceived as a selected peak, the algorithm can be stopped immediately.

Road recovery . Of course, it is usually necessary to know not only the length of the shortest roads, but also the roads themselves. Let's show you how to save the shortest path to any peak, which is enough for later recovery. To do this, it is enough to take a P[v] ancestral array, which maintains one previous peak order from that peak on the shortest path to that peak for each peak v (). This means that if we take the shortest path to any v peak, and then remove the last peak from this path, then we will have a path that ends with some kind of peak, and this path will be the shortest for the peak. Thus, if we have this ancestral line, the shortest path can be restored from it, just each time we take the ancestor from the current peak and go to the initial peak, in this way we get the shortest path we need, but the result must be written in reverse order.

It remains to understand how to build this ancestral line. However, this is done quite simply: with each successful rest, i.e. When the distance from the chosen peak V to some peak improves, we write that the ancestor of the peak to peak v is the peak:

Djektra algorithm psevdokod:

```
1  function Dijkstra(Graph, source):
2
3      for each vertex v in Graph.Vertices:
4          dist[v] ← INFINITY
5          prev[v] ← UNDEFINED
6          add v to Q
7      dist[source] ← 0
8
9      while Q is not empty:
10         u ← vertex in Q with min dist[u]
11         remove u from Q
12
13         for each neighbor v of u still in Q:
14             alt ← dist[u] + Graph.Edges(u, v)
15             if alt < dist[v]:
16                 dist[v] ← alt
```

```
17              prev[v] ← u
18
19      return dist[], prev[]
```

How the Dijkstra algorithm works

The B - > D Dijkstra algorithm is based on the fact that any part path of the shortest path between Peaks A and D is the shortest path between Peaks A - > D B and D.

the shortest path between source and address
section path between anba and address.

There are several real use cases in Dijkstra's algorithm, some of which are:

1. Digital mapping services on Google Maps: in G-Maps, the shortest path algorithm is encountered to find the distance from one city to another or from a specific location to the nearest desired location, since there are different paths connecting them, but it must show the minimum distance, so the Dijkstra algorithm is used to find the minimum distance between two places along

2.Social networking apps: in many apps, we witness that the app offers a list of friends that a specific user may know. Many social media companies implement this feature effectively, especially if the system has more than a billion users. The standard Dijkstra algorithm can be applied using the shortest path between users, measured through the links between them. When the social network graph is very small, it uses some other functions along with the standard Dijkstra algorithm to find the shortest paths, but when the graph becomes larger and larger, the standard algorithm takes a few seconds to calculate and replace the progenitors. algorithms are used.

3.Telephone network: as you know, the bandwidth of each line in the telephone network has a "b". If we imagine a city as a graph, then the vertices represent switching stations, and the edges represent transmission lines, and the weight of the edges represents "b". As you can see, it can fall into the category of the most short-range problems, for which Dijkstra can be used.

4.IP routing to find the first shortest path: the first to open the shortest path (OSPF) is a link - state routing protocol used to find the best path between source and target router using its shortest path . The Dijkstra algorithm is widely used in

routing protocols required by routers to update the routing table. The algorithm provides the shortest cost path from the source router to other routers on the network.

5.Avia flight schedule: if a person needs software to create a flight agenda for customers. The Agent has access to the database with all airports and flights. In addition to the flight number, the airport of origin and the destination, there is a departure and arrival time on flights. In particular, the agent wants to determine the time of arrival at the address at the earliest, taking into account the airport of origin and the start time. There this algorithm is used.

6.File server markup: the Dijkstra algorithm can be used to specify the file server on the LAN (local network). It may take an unlimited amount of time to transfer files from one computer to another. Using the Dijkstra algorithm to minimize the number of transitions from a file server to each other computer on the network, minimizing the shortest path between networks, resulting in the minimum number of hops.

7.Robotic road: drones and robots have now appeared, some of which are manual and some automated. Drones/robots that are automated and used to deliver packages to a specific location or for assignment are loaded with this algorithm module so that when the source and destination are known, the robot/drone moves in the following direction. the shortest way to continue delivering the package in the minimum amount of time.

III. Conclusion

The main advantage of the Dijkstra algorithm is its rather low complexity, which is almost linear. The Dijkstra algorithm solves the shortest path problem for graphs of any weight, direction and direction with non-negative weights. It can handle graphs consisting of cycles, but negative weights cause this algorithm to produce inaccurate results. This algorithm is very popular on geographical maps.

REFERENCES

1. http://e-maxx.ru/algo/dijkstra
2. https://www.geeksforgeeks.org/applications-of-dijkstras-shortest-path-algorithm/
3. Algorithms Notes for Professionals. Chapter 11. Dijkstra’s Algorithm
4. https://www.programiz.com/dsa/dijkstra-algorithm

https://www.baeldung.com/cs/dijkstra-vs-bellman-ford