Dijkstra's shortest path algorithm | Greedy Algo-7

Difficulty Level : Medium ● Last Updated : 15 Sep, 2021

Given a graph and a source vertex in the graph, find the shortest paths from the source to all vertices in the given graph.

Dijkstra's algorithm is very similar to Prim's algorithm for minimum spanning tree. Like Prim's MST, we generate a SPT (shortest path tree) with a given source as root. We maintain two sets, one set contains vertices included in the shortest path tree, other set includes vertices not yet included in the shortest path tree. At every step of the algorithm, we find a vertex which is in the other set (set of not yet included) and has a minimum distance from the source.

Below are detailed steps used in Dijkstra's algorithm to find the shortest path from a single source vertex to all other vertices in the given graph.

Algorithm

1) Create a set sptSet (shortest path tree set) that keeps track of vertices included in the shortest path tree, i.e., whose minimum distance from the source is calculated and finalized. Initially, this set is empty.

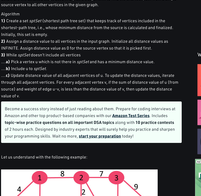
2) Assign a distance value to all vertices in the input graph. Initialize all distance values as INFINITE. Assign distance value as 0 for the source vertex so that it is picked first.

3) While sptSet does not include all vertices

- **a)** Pick a vertex u which is not there in sptSet and has a minimum distance value.
- **b)** Include u to sptSet.
- **c)** Update distance value of all adjacent vertices of u. To update the distance values, iterate through all adjacent vertices. For every adjacent vertex v, if the sum of distance value of u (from source) and weight of edge u-v, is less than the distance value of v, then update the distance value of v.

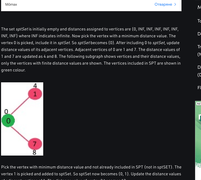
Become a success story instead of just reading about them. Prepare for coding interviews at Amazon and other top product based companies with our **Amazon Test Series**. Includes **topic-wise practice questions on all important DSA topics** along with **10 practice contests** of 2 hours each. Designed by industry experts that will surely help you practice and sharpen your programming skills. Wait no more, **start your preparation** today!

Let us understand with the following example:

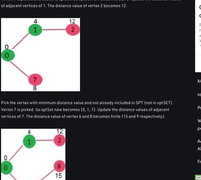


Online
Разгледай онлайн
Пакети от с клиентски настройки - Важни е 20% потъпкани цените на дрехи от Mitmar - до 50 %
Mitmar
Отворете

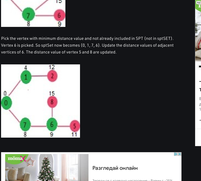
The set sptSet is initially empty and distances assigned to vertices are {0, INF, INF, INF, INF, INF, INF, INF} where INF indicates infinite. Now pick the vertex with a minimum distance value. The vertex 0 is picked, include in sptSet. So sptSet becomes {0}. After including 0 to sptSet, update distance values of its adjacent vertices. Adjacent vertices of 0 are 1 and 7. The distance values of 1 and 7 are updated as 4 and 8. The following subgraph shows vertices and their distance values, only the vertices with finite distance values are shown. The vertices included in SPT are shown in green colour.



Pick the vertex with minimum distance value and not already included in SPT (not in sptSET). The vertex 1 is picked and added to sptSet. So sptSet now becomes {0, 1}. Update the distance values of adjacent vertices of 1. The distance value of vertex 2 becomes 12.



Pick the vertex with minimum distance value and not already included in SPT (not in sptSET). Vertex 7 is picked. So sptSet now becomes {0, 1, 7}. Update the distance values of adjacent vertices of 7. The distance value of vertex 6 becomes 15 (1+7) respectively.



Pick the vertex with minimum distance value and not already included in SPT (not in sptSET). Vertex 6 is picked. So sptSet now becomes {0, 1, 7, 6}. Update the distance values of adjacent vertices of 6. The distance value of vertex 5 and 8 becomes finite (15 and 9 respectively).

Pick the vertex with minimum distance value and not already included in SPT (not in sptSET). Vertex 5 is picked. So sptSet now becomes {0, 1, 7, 6, 5}. Update the distance values of adjacent vertices of 5. The distance value of vertex 9 and 8 are updated.

Online
Разгледай онлайн
Пакети от с клиентски настройки - Важни е 20% потъпкани цените на дрехи от Mitmar - до 50 %
Mitmar
Отворете

We repeat the above steps until sptSet includes all vertices of the given graph. Finally, we get the following Shortest Path Tree (SPT).

How to implement the above algorithm?

Recommended: Please solve it on **PRACTICE** first, before moving on to the solution.

We use a boolean array sptSet[] to represent the set of vertices included in SPT. If a value sptSet[v] is true, then vertex v is included in SPT, otherwise not. Array dist[] is used to store the shortest distance values.

C++ | C | Java | Python | C# | JavaScript

```
// A utility function to find the vertex with minimum distance value, from
// the set of vertices not yet included in shortest path tree
int minDistance(int dist[], bool sptSet[])
{
    // Initialize min value
    int min = INT_MAX, min_index;

    for (int v = 0; v < V; v++)
        if (sptSet[v] == false && dist[v] <= min)
            min = dist[v], min_index = v;

    return min_index;
}

// A utility function to print the constructed distance array
void printSolution(int dist[])
{
    cout<<"Vertex \t Distance from Source" << endl;
    for (int i = 0; i < V; i++)
        cout << i << " \t " << dist[i]<< endl;
}

// Function that implements Dijkstra's single source shortest path algorithm
// for a graph represented using adjacency matrix representation
void dijkstra(int graph[V][V], int src)
{
    int dist[V];  // The output array.  dist[i] will hold the shortest
                  // distance from src to i

    bool sptSet[V];  // sptSet[i] will be true if vertex i is included in shortest
                     // path tree or shortest distance from src to i is finalized

    // Initialize all distances as INFINITE and stpSet[] as false
    for (int i = 0; i < V; i++)
        dist[i] = INT_MAX, sptSet[i] = false;

    // Distance of source vertex from itself is always 0
    dist[src] = 0;

    // Find shortest path for all vertices
    for (int count = 0; count < V - 1; count++)
    {
        // Pick the minimum distance vertex from the set of vertices not
        // yet processed. u is always equal to src in the first iteration.
        int u = minDistance(dist, sptSet);

        // Mark the picked vertex as processed
        sptSet[u] = true;

        // Update dist value of the adjacent vertices of the picked vertex.
        for (int v = 0; v < V; v++)

            // Update dist[v] only if is not in sptSet, there is an edge from
            // u to v, and total weight of path from src to  v through u is
            // smaller than current value of dist[v]
            if (!sptSet[v] && graph[u][v] && dist[u] != INT_MAX
                && dist[u] + graph[u][v] < dist[v])
                dist[v] = dist[u] + graph[u][v];
    }

    // print the constructed distance array
    printSolution(dist);
}

// driver program to test above function
int main()
{
    /* Let us create the example graph in discussion
    int graph[V][V] = { { 0, 4, 0, 0, 0, 0, 0, 8, 0},
                        { 4, 0, 8, 0, 0, 0, 0, 11, 0},
                        { 0, 8, 0, 7, 0, 4, 0, 0, 2},
                        { 0, 0, 7, 0, 9, 14, 0, 0, 0},
                        { 0, 0, 0, 9, 0, 10, 0, 0, 0},
                        { 0, 0, 4, 14, 10, 0, 2, 0, 0},
                        { 0, 0, 0, 0, 0, 2, 0, 1, 6},
                        { 8, 11, 0, 0, 0, 0, 1, 0, 7},
                        { 0, 0, 2, 0, 0, 0, 6, 7, 0 } };

    dijkstra(graph, 0);

    return 0;
}
```

This code is contributed by shivanisinghss2110

Output:

```
Vertex   Distance from Source
0        0
1        4
2        12
3        19
4        21
5        11
6        9
7        8
8        14
```

Notes:

1) The code calculates the shortest distance but doesn't calculate the path information. We can create a parent array, update the parent array when distance is updated (like prim's implementation) and use it to show the shortest path from source to different vertices.

2) The code is for undirected graphs, the same Dijkstra function can be used for directed graphs also.

3) The code finds the shortest distances from the source to all vertices. If we are interested only in the shortest distance from the source to a single target, we can break the for loop when the picked minimum distance vertex is equal to the target (Step 3 of the algorithm).

4) Time Complexity of the implementation is $O(V^2)$. If the input graph is represented using adjacency list, it can be reduced to $O(E \log V)$ with the help of a binary heap. Please see Dijkstra's Algorithm for Adjacency List Representation for more details.

5) Dijkstra's algorithm doesn't work for graphs with negative weight cycles. It may give correct results for a graph with negative edges but you must allow a vertex can be visited multiple times and that version will lose its fast time complexity. For graphs with negative weight edges and cycles, Bellman-Ford algorithm can be used, we will soon be discussing it as a separate post.

Dijkstra's Algorithm for Adjacency List Representation
Printing Paths in Dijkstra's Shortest Path Algorithm
Dijkstra's shortest path algorithm using set in STL

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Like 0

Previous
Prim's MST for Adjacency List Representation | Greedy Algo-6

Next
Dijkstra's Algorithm for Adjacency List Representation | Greedy Algo-8

RECOMMENDED ARTICLES

Page : 1 2 3

01 C / C++ Program for Dijkstra's shortest path algorithm | Greedy Algo-7
01 Mar, 2018

02 Java Program for Dijkstra's shortest path algorithm | Greedy Algo-7
01 Mar, 2018

03 C# Program for Dijkstra's shortest path algorithm | Greedy Algo-7
01 Mar, 2018

04 Python Program for Dijkstra's shortest path algorithm | Greedy Algo-7
01 Mar, 2018

05 Difference between the shortest and second shortest path in an Unweighted Bidirectional Graph
01 Jun, 2020

06 Shortest path from source to destination such that edge weights along path are alternatively increasing and decreasing
01 Jul, 2019

07 Printing Paths in Dijkstra's Shortest Path Algorithm
01 Jun, 2020

08 Probabilistic shortest path routing algorithm for optical networks
01 Jun, 2020

Article Contributed By :

GeeksforGeeks

Vote for difficulty

Easy | Normal | Medium | Hard | Expert

Improved By :

shivani, nishanjain, Chinmoy1997, SachinDhoundiyal_gfg, rajeshkumarkaushik417@yahoo.com, Shivamkhandelwal, rajgirish, abhaysingh0x29, rishabhsaxena, gaboanatol, Akashkhandelwal, pushkar2, ajit, rishabhsaxena2000, ashishkhandelwal@gmail.com

Article Tags :

Accolite, Adobe, Amazon, Cisco, DSArray, Morgan Stanley, Samsung, Shortest Path, Flipkart, Interviews Bolstering, Greedy, Greedy

Practice Tags :

Morgan Stanley, Accolite, Amazon, Samsung, Adobe, Cisco, Flipkart, Interviews Bolstering, Greedy, Greedy, Shortest Path

Improve Article | Report Issue

Write your comment! Please sign in.

Post Comment

Login/Signup to post comments from this site here.

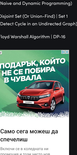
Load Comments

WHAT'S NEW

Full Stack Development with React and Node.js (Live Classes)
View Details

Competitive Programming Live Classes for Students
View Details

DSA Self Paced Course
View Details



Само сега можеш да спечелиш
Включи се в конкурса на месеца и поеми често нов автомобил всеки да бъде твой. Участвай!
~ Easy Credit

Вижте още

MOST POPULAR IN GRAPH

Topological Sorting
Detect Cycle in a Directed Graph
Travelling Salesman Problem | Set 1 (Naive and Dynamic Programming)
Disjoint Set (Or Union-Find) | Set 1 (Detect Cycle in an Undirected Graph)
Floyd Warshall Algorithm | DP-16

Само сега можеш да спечелиш
Включи се в конкурса на месеца и поеми често нов автомобил всеки да бъде твой. Участвай!
~ Easy Credit

Вижте още

MOST VISITED IN GREEDY

Huffman Coding | Greedy Algo-3
Program for array rotation
Write a program to print all permutations of a given string
Activity Selection Problem | Greedy Algo-1
Fractional Knapsack Problem

-20% ЗА ДОМАШЕН ТЕКСТИЛ
Включете се от 20% отстъпка на домашен текстил в Mitmar до 15.02. - Разгледай сега.
~ Mitmar

Отворете

geeksforgeeks, Some rights reserved