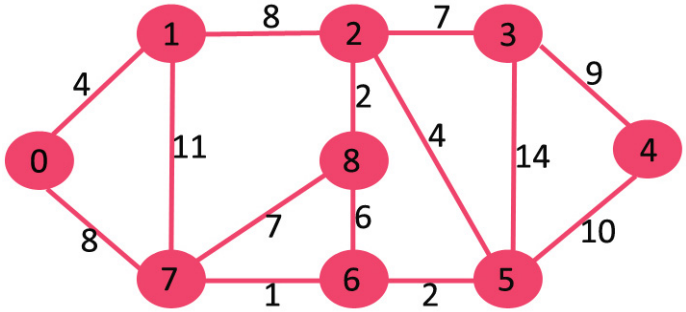
Greedy Algorithms | Set 7 (Dijkstra's shortest path algorithm)

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

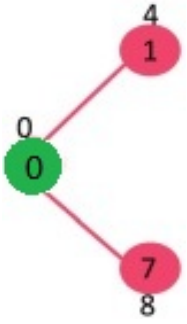
Dijkstra's algorithm is very similar to Prim's algorithm for minimum spanning tree. Like Prim's MST, we generate a *SPT (shortest path tree)* with given source as root. We maintain two sets, one set contains vertices included in shortest path tree, other set includes vertices not yet included in shortest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

Below are the 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 shortest path tree, i.e., whose minimum distance from 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* doesn't include all vertices
….**a)** Pick a vertex u which is not there in *sptSet*and has 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 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.

Let us understand with the following example:

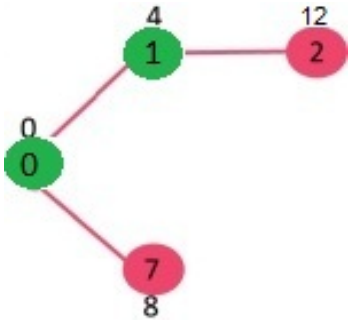


The set *sptSet*is initially empty and distances assigned to vertices are {0, INF, INF, INF, INF, INF, INF, INF} where INF indicates infinite. Now pick the vertex with minimum distance value. The vertex 0 is picked, include it in *sptSet*. So *sptSet* becomes {0}. After including 0 to *sptSet*, update 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. 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 color.

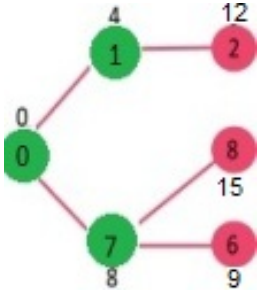


Pick the vertex with minimum distance value and not already included in SPT (not in sptSET). The vertex 1 is picked and added to sptSet. So sptSet now becomes {0, 1}. Update the distance values of adjacent vertices of 1. The distance value of vertex 2 becomes 12.

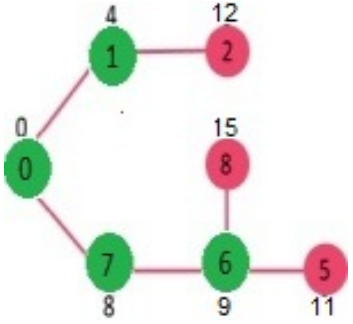
Popular Posts



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 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 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 are updated.

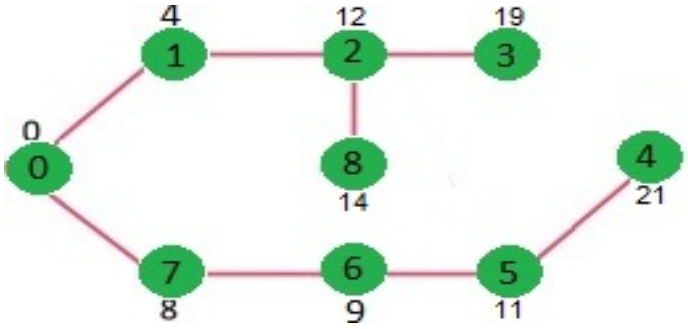


We repeat the above steps until *sptSet* doesn't include all vertices of given graph. Finally, we get the following Shortest Path Tree (SPT).

Lowest Common Ancestor in a BST.

Check if a binary tree is BST or not

Sorted Linked List to Balanced BST



How to implement the above algorithm?

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 shortest distance values of all vertices.

```
// A C / C++ program for Dijkstra's single source shortest path algori
// The program is for adjacency matrix representation of the graph

#include <stdio.h>
#include <limits.h>

// Number of vertices in the graph
#define V 9

// A utility function to find the vertex with minimum distance value,
// 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
int printSolution(int dist[], int n)
{
   printf("Vertex   Distance from Source\n");
   for (int i = 0; i < V; i++)
      printf("%d \t\t %d\n", i, dist[i]);
}
```


Recent Comments

```
// Funtion that implements Dijkstra's single source shortest path algo
// 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 shor
                      // distance from src to i

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

     // 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 no
       // yet processed. u is always equal to src in 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 ver
       for (int v = 0; v < V; v++)

         // Update dist[v] only if is not in sptSet, there is an edge
         // u to v, and total weight of path from src to  v through u
         // 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, V);
}

// driver program to test above function
int main()
{
   /* Let us create the example graph discussed above */
```

```
    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, 0, 10, 0, 2, 0, 0},
                       {0, 0, 0, 14, 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;
}
```

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 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 show the shortest path from source to different vertices.

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

3) The code finds shortest distances from source to all vertices. If we are interested only in shortest distance from source to a single target, we can break the for loop when the picked minimum distance vertex is equal to target (Step 3.a of 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 binary heap. We will soon be discussing O(E Log V) algorithm as a separate post.

5) Dijkstra's algorithm doesn't work for graphs with negative weight edges. For graphs with negative weight edges, Bellman–Ford algorithm can be used, we will soon be discussing it as a separate post.
Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Related Tpoics:

- Some interesting shortest path questions | Set 1
- Graph Coloring | Set 2 (Greedy Algorithm)
- Graph Coloring | Set 1 (Introduction and Applications)
- Johnson's algorithm for All-pairs shortest paths
- Travelling Salesman Problem | Set 2 (Approximate using MST)
- Travelling Salesman Problem | Set 1 (Naive and Dynamic Programming)
- Detect cycle in an undirected graph
- Find maximum number of edge disjoint paths between two vertices

Writing code in comment? Please use **ideone.com** and share the link here.

30 Comments GeeksforGeeks

Sort by Newest

Join the discussion…

prashant jha • 2 days ago

u can use min heap or rmq for reducing complexity
http://ideone.com/2gJjz2
not work for negative edges because here at each step of algorithm we make found shortest path upto dat vertex but if it has negative edges there may exist greedy algorithhm dont work there

andrei • 8 days ago

how should one modify Dijkstra to list all simple shortest paths to target?

1

James Sutton • 2 months ago

Can anyone convert this for use with Arduino? My attempts have failed so far. grasped what each part of the code is doing and therefore can't define them in would be much appreciated!

zak • 3 months ago

I am trying to compile the code by MVS 2012 but it does not work

tan • 4 months ago

can it work for directed graph?

2 ^ | v •

yossi • 5 months ago

hy i want to have the detail about the path if you can help me

3 ^ | v •

Tomislav • 5 months ago

N 0 1 2 3 4 5 6 7 8
0 {{0, 4, 0, 0, 0, 0, 0, 8, 0},
1 {4, 0, 8, 0, 0, 0, 0, 11, 0},
2 {0, 8, 0, 7, 0, 4, 0, 0, 2},
3 {0, 0, 7, 0, 9, 14, 0, 0, 0},
4 {0, 0, 0, 9, 0, 10, 0, 0, 0},
5 {0, 0, 4, 14 instead of 0, 10, 0, 2, 0, 0},
6 {0, 0, 0, 14, 0, 2, 0, 1, 6},
7 {8, 11, 0, 0, 0, 0, 1, 0, 7},
8 {0, 0, 2, 0, 0, 0, 6, 7, 0}
};

If 3 is connected with 5 with weight of 14, 5 is therefore connected with 3 with

2 ^ | v •

Mate ↱ Tomislav • 3 months ago

0 {{0, 4, 0, 0, 0, 0, 0, 8, 0},
1 {4, 0, 8, 0, 0, 0, 0, 11, 0},
2 {0, 8, 0, 7, 0, 4, 0, 0, 2},
3 {0, 0, 7, 0, 9, 14, 0, 0, 0},
4 {0, 0, 0, 9, 0, 10, 0, 0, 0},
5 {0, 0, 4, 14 instead of 0, 10, 0, 2, 0, 0},
6 {0, 0, 0, 14 (here is not 14), 0, 2, 0, 1, 6},

7 {8, 11, 0, 0, 0, 0, 1, 0, 7},
8 {0, 0, 2, 0, 0, 0, 6, 7, 0}
};

Because 6 is not connected with 3

Tomislav • 5 months ago

N 0 1 2 3 4 5 6 7 8
0 {{0, 4, 0, 0, 0, 0, 0, 8, 0},
1 {4, 0, 8, 0, 0, 0, 0, 11, 0},
2 {0, 8, 0, 7, 0, 4, 0, 0, 2},
3 {0, 0, 7, 0, 9, 14, 0, 0, 0},
4 {0, 0, 0, 9, 0, 10, 0, 0, 0},
5 {0, 0, 4, 0, 10, 0, 2, 0, 0},
6 {0, 0, 0, 14, 0, 2, 0, 1, 6},
7 {8, 11, 0, 0, 0, 0, 1, 0, 7},
8 {0, 0, 2, 0, 0, 0, 6, 7, 0}
};

Ankur • 7 months ago

dist[u]!=INF is a redundant check since dist[u]==INF will only be true if it was a
values of graph[u][v] will be 0 for this node and the condition

if (!sptSet[v] && graph[u][v] && dist[u] != INT_MAX && dist[u]+graph[u][v] < dis

will never be true. But since this situation is tackled by checking graph[u][v] , s
?

Nikhil Choudhary • 7 months ago

Problem Statement:-

Consider a data communication
network that must route data packets (email or MP3 files, for

example). Such a network
consists of routers connected by physical cables or links. A router can act as
a source, a destination, or a forwarder of data packets. We can model a netw
as a graph with each router corresponding to a vertex and the link or physical
between two routers corresponding to a pair of directed edges between the
vertices.

A network that follows the OSPF
(Open Shortest Path First) protocol routes packets using

Dijkstra's shortest path
algorithm. The criteria used to compute the weight corresponding to a

link can include the time taken

see more

Nikhil Choudhary • 7 months ago
can anyone plz upload the code for dijstraks and bellman algorithm in a single
have positive and negative weight edges)

javaDude • 7 months ago
could you fix the formatting T_T

javaDude • 7 months ago

The article was very helpful, thank you. I changed the code so that it is clearer

```
int minDistance(int dist[], bool sptSet[])
{
   int min = INT_MAX, min_index = -1;

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

   return min_index;
}

void dijkstra(int graph[V][V], int src)
{

   //init code
```

see more

viki • 7 months ago

I think "&& dist[u] != INT_MAX" is redundant check.

Ashish Tilokani viki • 4 months ago

Its used because the next condition is that of dist[u]+graph[u][v] < dist[v
cannot store more than INT_MAX
and the LHS can be INT_MAX + any positive integer

Should not we break the outer for loop when [sourcecode]dist[u] == INT_MAX which can be reached from source node

Tan Syh Ren • 11 months ago

the graph matrix seems to mismatch with the illustration

GeeksforGeeks • 11 months ago

GCC

Tan Syh Ren • 11 months ago

may i know what&#039s your compiler? I tried to use dev c++ but lots of erros make it compile though

Ray Garner • a year ago

Useful thanks

Sandeep Jain • a year ago

Ankit Paharia Thanks for pointing this out. We have updated the code.

Sarthak Mall 'shanky' • a year ago

Ankit bro hum to sirf padte hain is site se,tu to comment bhi karne laga ...tod b

Ankit Paharia • a year ago

There is a small error in the input graph[V][V].... if I change the source vertex t

from 1 is coming 19... as the graph input is undirected this should not happen graph[1][0]=4 , not 0.... then the output will be correct....

2 ^ | v •

atul • a year ago

Below code may return garbage value i.e min_index.
test case : suppose a node is an isolated node in a graph and does not have a value will not get updated.

```
 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;
}
```

1 ^ | v •

GeeksforGeeks ➚ atul • a year ago

Thanks for pointing this out. Replacing 'smaller than' sign with 'smaller for all cases. We will update the post soon.

```
 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;
}
```

Ankit • a year ago

Please explain the presence of condition (!sptSet[v]) in

```
if (!sptSet[v] && graph[u][v] && dist[u] + graph[u][v] < dist[v])
            dist[v] = dist[u] + graph[u][v];
    }
```

According to Cormen, we are supposed to relax all the edges leaving v and is condition doesn't affect the result, please explain how ?

Ankur ↗ Ankit • 7 months ago

This condition only ensures that the neighbour's that you are going to a because if they already are in SPtset than shortest path to them have a need for them to go through the condition : dist[u] + graph[u][v] < dist[v

Dinesh • a year ago

awesome was really helpful in understanding dijkstra algorithm...thanks a lot fo

```
/* Paste your code here (You may delete these lines if not writing co
```

yossi Dinesh • 5 months ago

hy i want to have the detail about the path if you can help me

3

Subscribe Add Disqus to your site