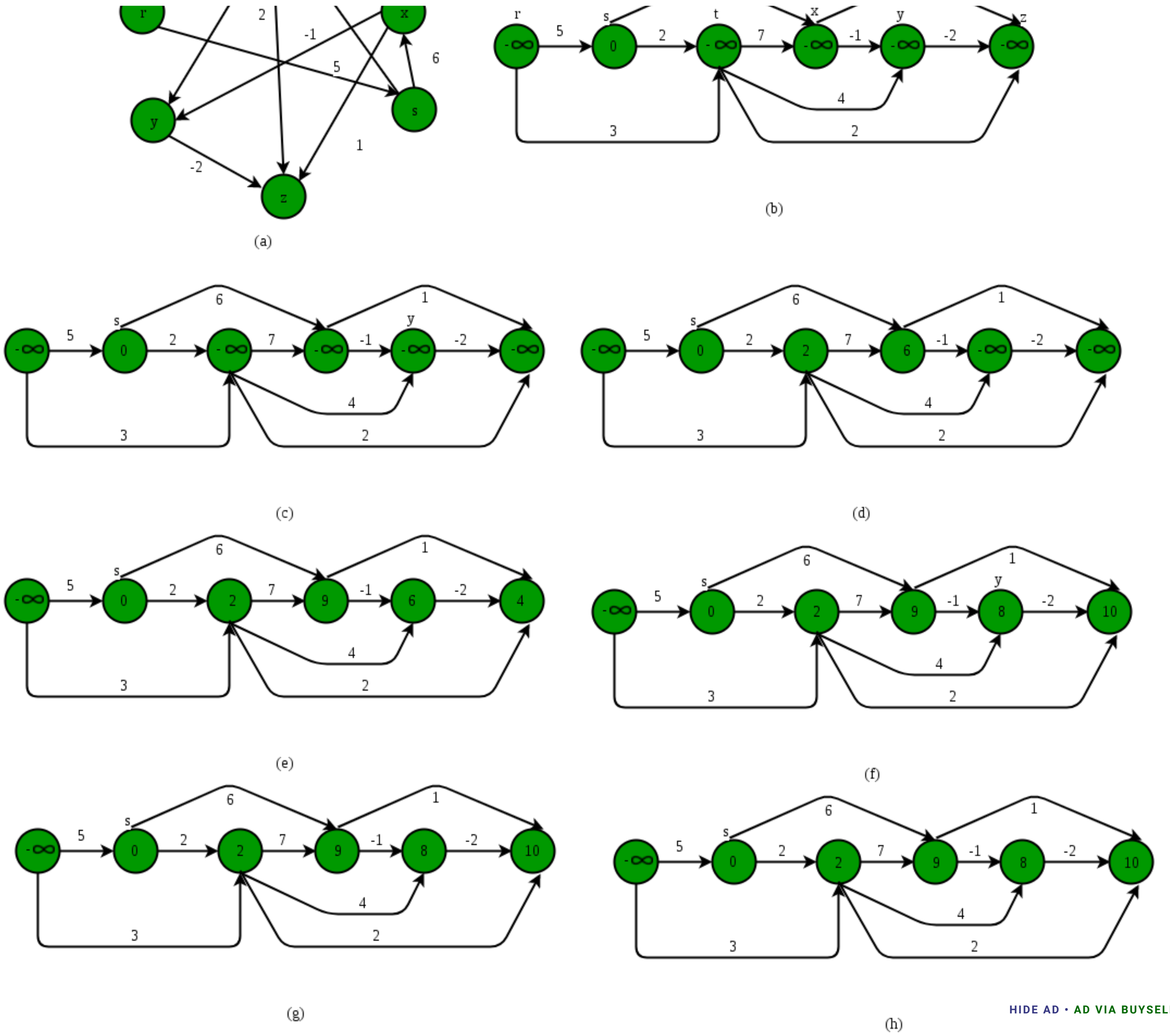
Longest Path in a Directed Acyclic Graph

Given a Weighted **D**irected **A**cyclic **G**raph (DAG) and a source vertex s in it, find the longest distances from s to all other vertices in the given graph.

The longest path problem for a general graph is not as easy as the shortest path problem because the longest path problem doesn't have optimal substructure property. In fact, the Longest Path problem is NP-Hard for a general graph. However, the longest path problem has a linear time solution for directed acyclic graphs. The idea is similar to linear time solution for shortest path in a directed acyclic graph., we use Topological Sorting.

We initialize distances to all vertices as minus infinite and distance to source as 0, then we find a topological sorting of the graph. Topological Sorting of a graph represents a linear ordering of the graph (See below, figure (b) is a linear representation of figure (a)). Once we have topological order (or linear representation), we one by one process all vertices in topological order. For every vertex being processed, we update distances of its adjacent using distance of current vertex.

Following figure shows step by step process of finding longest paths.



(a) (b) (c) (d) (e) (f) (g) (h)

Following is complete algorithm for finding longest distances.

1) Initialize dist[] = {NINF, NINF, ….} and dist[s] = 0 where s is the source vertex. Here NINF means negative infinite.

2) Create a toplogical order of all vertices.

3) Do following for every vertex u in topological order.

………..Do following for every adjacent vertex v of u

………………if (dist[v] < dist[u] + weight(u, v))

………………………dist[v] = dist[u] + weight(u, v)

Following is C++ implementation of the above algorithm.

```
// A C++ program to find single source longest distances
// in a DAG
#include <iostream>
#include <limits.h>
#include <list>
#include <stack>
#define NINF INT_MIN
```

```
// the vertex to which edge connects. It also
// contains weight of the edge
class AdjListNode {
    int v;
    int weight;

public:
    AdjListNode(int _v, int _w)
    {
        v = _v;
        weight = _w;
    }
    int getV() { return v; }
    int getWeight() { return weight; }
};

// Class to represent a graph using adjacency list
// representation
class Graph {
    int V; // No. of vertices'

    // Pointer to an array containing adjacency lists
    list<AdjListNode>* adj;

    // A function used by longestPath
    void topologicalSortUtil(int v, bool visited[],
                             stack<int>& Stack);

public:
    Graph(int V); // Constructor

    // function to add an edge to graph
    void addEdge(int u, int v, int weight);

    // Finds longest distances from given source vertex
    void longestPath(int s);
};

Graph::Graph(int V) // Constructor
{
    this->V = V;
    adj = new list<AdjListNode>[V];
}

void Graph::addEdge(int u, int v, int weight)
{
    AdjListNode node(v, weight);
    adj[u].push_back(node); // Add v to u's list
}

// A recursive function used by longestPath. See below
// link for details
// https:// www.geeksforgeeks.org/topological-sorting/
void Graph::topologicalSortUtil(int v, bool visited[],
                                stack<int>& Stack)
{
    // Mark the current node as visited
    visited[v] = true;

    // Recur for all the vertices adjacent to this vertex
    list<AdjListNode>::iterator i;
    for (i = adj[v].begin(); i != adj[v].end(); ++i) {
        AdjListNode node = *i;
        if (!visited[node.getV()])
            topologicalSortUtil(node.getV(), visited, Stack);
    }
```

```
}

// The function to find longest distances from a given vertex.
// It uses recursive topologicalSortUtil() to get topological
// sorting.
void Graph::longestPath(int s)
{
    stack<int> Stack;
    int dist[V];

    // Mark all the vertices as not visited
    bool* visited = new bool[V];
    for (int i = 0; i < V; i++)
        visited[i] = false;

    // Call the recursive helper function to store Topological
    // Sort starting from all vertices one by one
    for (int i = 0; i < V; i++)
        if (visited[i] == false)
            topologicalSortUtil(i, visited, Stack);

    // Initialize distances to all vertices as infinite and
    // distance to source as 0
    for (int i = 0; i < V; i++)
        dist[i] = NINF;
    dist[s] = 0;

    // Process vertices in topological order
    while (Stack.empty() == false) {
        // Get the next vertex from topological order
        int u = Stack.top();
        Stack.pop();

        // Update distances of all adjacent vertices
        list<AdjListNode>::iterator i;
        if (dist[u] != NINF) {
            for (i = adj[u].begin(); i != adj[u].end(); ++i)
                if (dist[i->getV()] < dist[u] + i->getWeight())
                    dist[i->getV()] = dist[u] + i->getWeight();
        }
    }

    // Print the calculated longest distances
    for (int i = 0; i < V; i++)
        (dist[i] == NINF) ? cout << "INF " : cout << dist[i] << " ";
}

// Driver program to test above functions
int main()
{
    // Create a graph given in the above diagram.
    // Here vertex numbers are 0, 1, 2, 3, 4, 5 with
    // following mappings:
    // 0=r, 1=s, 2=t, 3=x, 4=y, 5=z
    Graph g(6);
    g.addEdge(0, 1, 5);
    g.addEdge(0, 2, 3);
    g.addEdge(1, 3, 6);
    g.addEdge(1, 2, 2);
    g.addEdge(2, 4, 4);
    g.addEdge(2, 5, 2);
    g.addEdge(2, 3, 7);
    g.addEdge(3, 5, 1);
    g.addEdge(3, 4, -1);
    g.addEdge(4, 5, -2);
    g.longestPath(s);

    return 0;
}
```

Output:

```
Following are longest distances from source vertex 1
INF 0 2 9 8 10
```

Time Complexity: Time complexity of topological sorting is O(V+E). After finding topological order, the algorithm process all vertices and for every vertex, it runs a loop for all adjacent vertices. Total adjacent vertices in a graph is O(E). So the inner loop runs O(V+E) times. Therefore, overall time complexity of this algorithm is O(V+E).

Exercise: The above solution print longest distances, extend the code to print paths also.

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

Recommended Posts:

Longest Path in a Directed Acyclic Graph | Set 2
Longest path in a directed Acyclic graph | Dynamic Programming
Shortest Path in Directed Acyclic Graph
Clone a Directed Acyclic Graph
All Topological Sorts of a Directed Acyclic Graph
Assign directions to edges so that the directed graph remains acyclic
Number of paths from source to destination in a directed acyclic graph
Convert the undirected graph into directed graph such that there is no path of length greater than 1
Find if there is a path between two vertices in a directed graph
Shortest path with exactly k edges in a directed and weighted graph
DFS for a n-ary tree (acyclic graph) represented as adjacency list
Calculate number of nodes between two vertices in an acyclic Graph by Disjoint Union method
Hierholzer's Algorithm for directed graph
Check if a directed graph is connected or not
Euler Circuit in a Directed Graph

Improved By : Rohit0803, pranith maddineni

Article Tags : Graph Microfocus Shortest Path Topological Sorting

Practice Tags : Graph Shortest Path

□ To-do □ Done

3.7

Based on **126** vote(s)

Feedback/ Suggest Improvement | Add Notes | Improve Article

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

94 Comments **GeeksforGeeks** 1 **Login**

♡ **Recommend** 11 Tweet Share **Sort by Newest**

Join the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Utkarsh Behre • 18 days ago

Here is my working java solution with explanation in comments for finding the longest path and printing the path chosen to get the longest distance as well.

Reply • Share ›

Jose Silva • 7 months ago

From Brazil.
Can you help me to fine the number of edges of the longest path in a directed unwaited graph?
Thank you very much.

Reply • Share ›

Rohith ASRK • 9 months ago

Wouldn't initialising the dist array to NINF and using a simple DFS recursively to update the max value work? Like the one at https://www.ideone.com/Sd5Qjh.

Reply • Share ›

Rohit Maurya • 10 months ago

How do you know dist will have the same values in order (topological)? I can't see any checks which assigns dist based on topological order.

Reply • Share ›

Rohit Maurya ➦ Rohit Maurya • 10 months ago

i think, i got it.

Reply • Share ›

Mani Kanta • a year ago

what are some real time applications of this?

Reply • Share ›

⌃ | ⌄ • Reply • Share ›

shreyas singh ➦ last-minute preps • a year ago

Finding shortest path using Bellman Ford would work.

⌃ | ⌄ • Reply • Share ›

Abhinav Bajpai ➦ last-minute preps • a year ago

yup it does

1 ⌃ | ⌄ • Reply • Share ›

Dave Smith • a year ago

This solution has a memory leak. Every time longestPath() is called, it leaks the "visited" array of V booleans.

⌃ | ⌄ • Reply • Share ›

Dave Smith ➦ Dave Smith • a year ago

It also leaks the "adj" list every time a new Graph instance is created.

⌃ | ⌄ • Reply • Share ›

Mohammad Arsalan • a year ago

Solution in c# -> https://github.com/arsalanc...

⌃ | ⌄ • Reply • Share ›

Vivek • a year ago

Java Implementation:
https://ideone.com/V9pJMZ

⌃ | ⌄ • Reply • Share ›

Bahruz Balabayov • a year ago • edited

thank you very much for your implementation. I would say it would be better if you follow SRP(Single responsibility principle) in the range of function(instead of big functions). It would be easier to understand the code

⌃ | ⌄ • Reply • Share ›

deku • a year ago

https://ide.geeksforgeeks.o...
Using kahn's algorithm

1 ⌃ | ⌄ • Reply • Share ›

MIRIYALA JEEVAN KUMAR • 2 years ago

Can we apply dynamic programming strategy here..(an solution possible ?)

1 ⌃ | ⌄ • Reply • Share ›

Oscar Deits • 2 years ago

Typo in second paragraph in the link

The idea is similar to linear time solution for shortest path in a directed acyclic graph., we use Tological Sorting.

Should be:

The idea is similar to linear time solution for shortest path in a directed acyclic graph., we use Topological Sorting.

⌃ | ⌄ • Reply • Share ›

hulk hogan • 2 years ago

http://ide.geeksforgeeks.or...

⌃ | ⌄ • Reply • Share ›

Pabitra Padhy • 2 years ago

As the overall time complexity will remain the same, it's just extra operation.

• Reply • Share ›

Dhruvil Bhuptani • 2 years ago

why we are doing topological sort first ?

• Reply • Share ›

Ekaansh Khosla ➦ Dhruvil Bhuptani • 2 years ago • edited

Because Topological Sorting Order depicts that the only vertices that can be visited after reaching that given vertex will come after that given vertex in the Topological Sorted Order and this is the requirement to solve this problem

1 • Reply • Share ›

Dhruvil Bhuptani ➦ Ekaansh Khosla • 2 years ago

in every directed graph question it's necessary?

• Reply • Share ›

SRB • 2 years ago

I modified the longestpath() function to print the longest path between source and the last vertex(sink).

```
void Graph::longestPath(int s)
{
stack<int> Stack;
int dist[V];

// Mark all the vertices as not visited
bool *visited = new bool[V];
for (int i = 0; i < V; i++)
visited[i] = false;

// Call the recursive helper function to store Topological Sort
// starting from all vertices one by one
for (int i = 0; i < V; i++)
if (visited[i] == false)
topologicalSortUtil(i, visited, Stack);

// Initialize distances to all vertices as infinite and distance
```

see more

• Reply • Share ›

Jose Silva ➦ SRB • 7 months ago

It is not printing the resolt.

• Reply • Share ›

Luca • 2 years ago

Is there a similar algorithm for an acyclic directed MULTIGRAPH?

• Reply • Share ›

Franziska Funck • 2 years ago

Does anybody know a book in which the algorithm is explained? Thanks!

• Reply • Share ›

kousik vinjam • 2 years ago

implementation of above code in java

http://code.geeksforgeeks.o...

• Reply • Share ›

If you make INF into INT_MAX, and changed condition less to greater,
You can get a shortest path algo. in a DAG.

Reply • Share ›

Yasharyan Gaikwad • 2 years ago

There's an easier way of doing this. Replace the weights by their negatives and then find the shortest path.

Reply • Share ›

Pabitra Padhy → Yasharyan Gaikwad • 2 years ago

That would give a maximum weight spanning tree, by transposing the original graph.
Hence, instead of printing the shortest path, the shortest path of the transpose graph would be the longest path.

But, I have seen that approach with undirected graphs, would that be valid for directed graphs as well ?

Reply • Share ›

Yasharyan Gaikwad → Pabitra Padhy • 2 years ago

Not entirely sure

Reply • Share ›

Girish Humnabade • 3 years ago

This is just the code written in a proper way........There is some problrm in viewing the above code

```
#include <iostream>
#include <list>
#include <stack>
#include <limits.h>
#define NINF INT_MIN
using namespace std;

// Graph is represented using adjacency list. Every node of adjacency list
// contains vertex number of the vertex to which edge connects. It also // contains weight of the edge

class AdjListNode {
int v;
int weight;
public:
AdjListNode(int _v, int _w) {
v = _v;
weight = _w;
```

see more

Reply • Share ›

stillTravelling • 3 years ago • edited

Something went wrong
I can't see the code

1 • Reply • Share ›

sunny • 3 years ago

Simple BFS without coloring should work:

http://code.geeksforgeeks.o...

Reply • Share ›

Mayuri → sunny • 3 years ago

BFS would work only in combination to TSort.

Reply • Share ›

Qiannan ➦ deepak gupta • 3 years ago

I think we can do it by DFS without visited[] if we are given a start node or we are asked to compute longest path between a start node and an end node.

⌃ | ⌄ • Reply • Share ›

aka ➦ Qiannan • 2 years ago

Then it is not dfs algorithm my friend! DFS NEVER visits all the paths.If you remove visited array and explore all paths complexity will be O(n!).
You can also refer to the first answer given here:
http://stackoverflow.com/qu...

1 ⌃ | ⌄ • Reply • Share ›

Sboy ➦ deepak gupta • 3 years ago

No it can't be done using dfs.We are using topological sort simply because if we have not visited vertices before a given vertex and hence not have the maximum distance to all nodes before this node how will we able to update the values? So to summarise we r using topological sort coz if we have 3 vertices a->b->c connected i can find max cost to reach c we MUST have max cost to reach b..u can visuailze it this way...hope it makes sense:)

3 ⌃ | ⌄ • Reply • Share ›

Mahak ➦ Sboy • 3 years ago • edited

I think we can do it by DFS by not considering isvisited[]. Lets say b doesn't has max cost yet and it updated c, then c also won't have max cost, u r right in that. But later on when by another path b will get its max cost, it again traverse all its adjacent node (c here) as we are not using isvisited [] in this DFS, and will update its value.
Drawback of this method is complexity as e.g. here b has to visit its adjacent nodes twice or more , whereas in case of topological sort, traversal to its adjacent nodes will be only once.
Please correct me if I'm wrong.

1 ⌃ | ⌄ • Reply • Share ›

Show more replies

bhavik gujarati • 3 years ago

Java implementation which also prints paths:

http://ideone.com/CA50CG

1 ⌃ | ⌄ • Reply • Share ›

Vishwas Garg • 3 years ago

someone please explain the code...

⌃ | ⌄ • Reply • Share ›

Avishek Dutta ➦ Vishwas Garg • 2 years ago

Here we are doing the opposite of "Finding shortest path in a DAG."
I would suggest you to go through shortest path in DAG first. Then do a dry run of this code. You will get the logic.
Hope this helps.

1 ⌃ | ⌄ • Reply • Share ›

geeks13 • 3 years ago

Is "dist[u]!=INF" necessary? we are moving topological order and every vertex gets its distance updated to some value. So is it necessary?

⌃ | ⌄ • Reply • Share ›

Karan Saxena ➦ geeks13 • 2 years ago

Yes, to avoid overflow.

⌃ | ⌄ • Reply • Share ›

^ | v • Reply • Share ›

ZZZZzzzz • 3 years ago

What if we want to find the longest path without an indicated source? can we iterate the same thing for every vertex as a source? Any better suggestion?

^ | v • Reply • Share ›

Teja Vardhan Reddy → ZZZZzzzz • 3 years ago

this will give longest path to all vertices since there is no cycle .intialize first one to zero and then do this process .so now we have array d[v];if we want between u and v then d[v]-d[u] will give it

1 ^ | v • Reply • Share ›

Leo Martinez • 3 years ago

Please I need it in C implementation

^ | v • Reply • Share ›

Nipun • 3 years ago • edited

hey i want to find the longest path with minimum weight
suppose
u v w
1 2 5
1 3 2
1 4 3
ans should be for node 1
2 {distance is same but weight is minimum(2,3,5)}

^ | v • Reply • Share ›

Load more comments

Subscribe Add Disqus to your siteAdd DisqusAdd Disqus' Privacy PolicyPrivacy PolicyPrivacy

A computer science portal for geeks

5th Floor, A-118,
Sector-136, Noida, Uttar Pradesh - 201305
feedback@geeksforgeeks.org

COMPANY
About Us
Careers
Privacy Policy
Contact Us

LEARN
Algorithms
Data Structures
Languages
CS Subjects
Video Tutorials

PRACTICE
Courses
Company-wise
Topic-wise
How to begin?

CONTRIBUTE
Write an Article
Write Interview Experience
Internships
Videos