Dynamic Programming | Set 16 (Floyd Warshall Algorithm)

The Floyd Warshall Algorithm is for solving the All Pairs Shortest Path problem. The problem is to find shortest distances between every pair of vertices in a given edge weighted directed Graph.

Example:

```
Input:
       graph[][] = { {0,   5,  INF, 10},
                    {INF,  0,  3,  INF},
                    {INF, INF, 0,   1},
                    {INF, INF, INF, 0} }
which represents the following graph
             10
       (0)------->(3)
```

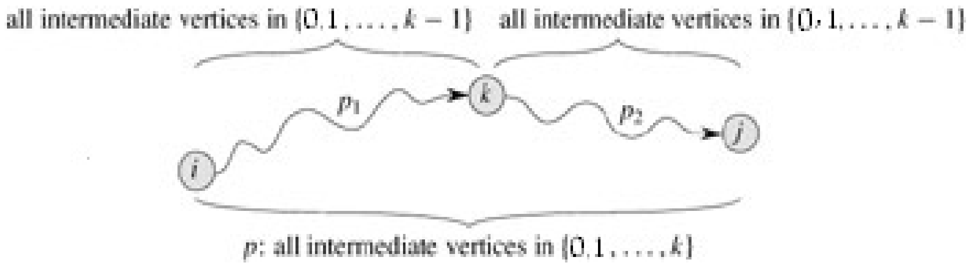
```
            |         /|\
          5 |          |
            |          | 1
           \|/         |
           (1)------->(2)
                3
Note that the value of graph[i][j] is 0 if i is equal to j
And graph[i][j] is INF (infinite) if there is no edge from vertex i to j.

Output:
Shortest distance matrix
      0      5      8      9
    INF      0      3      4
    INF    INF      0      1
    INF    INF    INF      0
```

Floyd Warshall Algorithm

We initialize the solution matrix same as the input graph matrix as a first step. Then we update the solution matrix by considering all vertices as an intermediate vertex. The idea is to one by one pick all vertices and update all shortest paths which include the picked vertex as an intermediate vertex in the shortest path. When we pick vertex number k as an intermediate vertex, we already have considered vertices {0, 1, 2, .. k-1} as intermediate vertices. For every pair (i, j) of source and destination vertices respectively, there are two possible cases.
1) k is not an intermediate vertex in shortest path from i to j. We keep the value of dist[i][j] as it is.
2) k is an intermediate vertex in shortest path from i to j. We update the value of dist[i][j] as dist[i][k] + dist[k][j].

The following figure is taken from the Cormen book. It shows the above optimal substructure property in the all-pairs shortest path problem.


all intermediate vertices in {0, 1, ..., k − 1}
all intermediate vertices in {0, 1, ..., k − 1}
p1
k
p2
i
j
p: all intermediate vertices in {0, 1, ..., k}

Following is C implementation of the Floyd Warshall algorithm.

```
// Program for Floyd Warshall Algorithm
#include<stdio.h>

// Number of vertices in the graph
#define V 4

/* Define Infinite as a large enough value. This value will be used
  for vertices not connected to each other */
#define INF 99999

// A function to print the solution matrix
void printSolution(int dist[][V]);

// Solves the all-pairs shortest path problem using Floyd Warshall algorithm
```

```
void floydWarshell (int graph[][V])
{
    /* dist[][] will be the output matrix that will finally have the shortest
      distances between every pair of vertices */
    int dist[V][V], i, j, k;

    /* Initialize the solution matrix same as input graph matrix. Or
       we can say the initial values of shortest distances are based
       on shortest paths considering no intermediate vertex. */
    for (i = 0; i < V; i++)
        for (j = 0; j < V; j++)
            dist[i][j] = graph[i][j];

    /* Add all vertices one by one to the set of intermediate vertices.
      ---> Before start of a iteration, we have shortest distances between al
      pairs of vertices such that the shortest distances consider only the
      vertices in set {0, 1, 2, .. k-1} as intermediate vertices.
      ----> After the end of a iteration, vertex no. k is added to the set of
      intermediate vertices and the set becomes {0, 1, 2, .. k} */
    for (k = 0; k < V; k++)
    {
        // Pick all vertices as source one by one
        for (i = 0; i < V; i++)
        {
            // Pick all vertices as destination for the
            // above picked source
            for (j = 0; j < V; j++)
            {
                // If vertex k is on the shortest path from
                // i to j, then update the value of dist[i][j]
                if (dist[i][k] + dist[k][j] < dist[i][j])
                    dist[i][j] = dist[i][k] + dist[k][j];
            }
        }
    }

    // Print the shortest distance matrix
    printSolution(dist);
}

/* A utility function to print solution */
void printSolution(int dist[][V])
{
    printf ("Following matrix shows the shortest distances"
            " between every pair of vertices \n");
    for (int i = 0; i < V; i++)
    {
        for (int j = 0; j < V; j++)
        {
            if (dist[i][j] == INF)
                printf("%7s", "INF");
            else
                printf ("%7d", dist[i][j]);
        }
        printf("\n");
    }
}

// driver program to test above function
int main()
{
    /* Let us create the following weighted graph
            10
       (0)------->(3)
        |         /|\
      5 |          |
        |          | 1
       \|/         |
       (1)------->(2)
            3           */
    int graph[V][V] = { {0,   5,  INF, 10},
                        {INF, 0,   3, INF},
                        {INF, INF, 0,   1},
                        {INF, INF, INF, 0}
                      };

    // Print the solution
    floydWarshell(graph);
    return 0;
}
```

Output:

```
Following matrix shows the shortest distances between every pair of vertices
      0      5      8      9
    INF      0      3      4
    INF    INF      0      1
    INF    INF    INF      0
```

Time Complexity: O(V^3)

The above program only prints the shortest distances. We can modify the solution to print the shortest paths also by storing the predecessor information in a separate 2D matrix.
Also, the value of INF can be taken as INT_MAX from limits.h to make sure that we handle maximum possible value. When we take INF as INT_MAX, we need to change the if condition in the above program to avoid arithmatic overflow.

```
#include<limits.h>

#define INF INT_MAX
..........................
if (dist[i][k] != INF && dist[k][j] != INF && dist[i][k] + dist[k][j] < dist[
    dist[i][j] = dist[i][k] + dist[k][j];
...........................
```

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

Related Topics:

- Assign directions to edges so that the directed graph remains acyclic
- K Centers Problem | Set 1 (Greedy Approximate Algorithm)
- Find the minimum cost to reach destination using a train
- Applications of Breadth First Traversal
- Optimal read list for given number of days
- Print all paths from a given source to a destination
- Minimize Cash Flow among a given set of friends who have borrowed money from each other
- Boggle (Find all possible words in a board of characters)

Tags: Dynamic Programming, Graph

Tweet 0 g+1 1

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

41 Comments **GeeksforGeeks** 1 **Login**

Recommend Share Sort by Newest

Join the discussion…

Andy Toh • 2 hours ago

My compile-time solution, which agrees with the run-time solution output above: http://ideone.com/AasLmF

• Reply • Share ›

Ralph Hector Pacardo Adricula • a month ago

is this algorithm applicable for this problem? "how should one to locate ambulance stations so as to best serve the needs of the community" tnx

• Reply • Share ›

Sanghwa Jung • 3 months ago

my javacod is here http://javamusician.blogspot.k...

• Reply • Share ›

kyoyo • 4 months ago

does floyd warshall for for undirected graph too?

1 • Reply • Share ›

daredadevil kyoyo • a month ago

daredadevil kyoyo • a month ago

I tried Floyd-Warshall for an undirected graph while I had to implement it for a graph problem & it worked.

Check this out - "Every undirected graph can be represented as directed graph by replacing every edge (i,j) with 2 edges (i,j);(j,i). And if you're running Floyd–Warshall algorithm on such directed graph - it would work correctly, as always." http://cs.stackexchange.com/a/...

Reply • Share ›

Sanghwa Jung kyoyo • 3 months ago

I think It would be

Reply • Share ›

Ethan Lim • 5 months ago

Can I say that the space complexity is O(|V|2) if using soln matrix and we can choose to destroy the orig graph rep and use O(1) space?

Reply • Share ›

Sanket Patel Ethan Lim • 2 months ago

No. You still need that space to make the solution matrix. Doesn't matter if you free it later on, the space complexity remains O(V^2)

Reply • Share ›

Sadat Mohammad Akash • 5 months ago

How to improve the complexity of floyd warshall algorithm ?

Reply • Share ›

prannu • 6 months ago

is there any possibility to do better than this??

Reply • Share ›

brahma • 10 months ago

what is difference between "shortest distances" and "shortest paths"?

4 • Reply • Share ›

Mr. A jat brahma • 3 months ago

distance means no matter how many vertices it has to cross the numeric edge weight computed should be less than any other combination of vertices edge weight possible, while, path is min no of vertices you encounter while reaching destination node.

1 • Reply • Share ›

Ty Nguyen brahma • 8 months ago

One means distance of path while the other is paht itself.

3 • Reply • Share ›

prashant jha • a year ago

apart from apsp floyd warshal can be used as
-bipartite checking
-maximin and minimax path
-transitive closure

1 • Reply • Share ›

ankunath • a year ago

my code for the optimal solution of this is

http://ideone.com/e.js/b44uZQ

• Reply • Share ›

Mr A jat ankunath • 3 months ago

Try dp

• Reply • Share ›

natalcoder • a year ago

"We can modify the solution to print the shortest paths also by storing the predecessor information in a separate 2D matrix". I'm a little confused here, what to store in the matrix and how to process that matrix in order to get the shortest path?
Thanks!

1 • Reply • Share ›

Ty Nguyen natalcoder • 8 months ago

You can simply create a matrix named parent[][] whose each element such as parent[u][v] stores the index of the ancestral node of v on path from u to v.

With k satisfying as a intermediate node from u to v, we update parent[u][v] = k.
Finally, we have a 2-D matrix parent[][] which stores the nearest ancestral nodes of the destination nodes.
In order to print out the shortest paths, just do as following:

```
void printPath(int (&parent)[V][V], int u, int v)
{
printf("\t %d %d", u, v);
if(v == u) return;
else printPath(path, parent[u][v]);
}
```

For more information, you can refer to my implementation on shortest path problem.
https://github.com/tyunist/alg...

• Reply • Share ›

• Reply • Share ›

Niana Ty Nguyen • 6 months ago

Thanks for your comment! I couldn't have finished my project without your help!

I edited some parts of your code though and I think the correct implementation for a 2D graph should be like this:

```
void printPath(int parent[V][V], int u, int v)
{
printf("%d ", u);
if (parent[u][v] == u)
{
printf("%d", v);
return;
}
else printPath(parent, parent[u][v] ,v);
}
```

• Reply • Share ›

prashant jha • a year ago

its a bottom up dynamaic programming ques
there is two possibility for a path to exist between vertex i and j using intermediate vertex 0,1,2,,,,,k
1- using intermediate vertices 0,1,2....k-1 ie excluding k
2-including k as intermediate vertex ie path from i to k and k to j
D[I][J]=MIN (P[I][J],P[I][K]+P[K][J]) where p one level low matrix thn d
create D0,D1,D2,,,,D(n-1) recursively and fill the entries
D(n-1) is the resultant matrix

• Reply • Share ›

prashant jha • a year ago

```
#include<iostream>
#define n 5
#define infinity 9999
#define max 20
#define temp 0
#define size 50
int adj[max][max];
int dist[max][max];
int status[max];
using namespace std;
struct queue
{
int front:
```

```
int rear;
int arr[size];
void insert(int);
int del();
queue();
```

see more

3 • Reply • Share ›

Mohamad • a year ago

Hey guys,

Here's an implementation of Floyd-Warshall algorithm in C++ which finds the minimum weights along with the actual paths

http://cs-and-design.blogspot....

9 • Reply • Share ›

ashraf Mohamad • 2 months ago

the blog has been removed, can you help me with this problem brother? I need to implement the algorithm with both the minimum weights and the actual paths.

• Reply • Share ›

nickpsar • a year ago

thanks guys!!!

now with this example i can say that i finally managed to fill up all the blanks about floyd warshall algorithm. But...........hmm

How can modify the code If i want in the solution graph to change all the distances up to 2 with 1 and all the other with 0 (sorry i forgot to say that my graph has weight 1 for each edge) ?

1 • Reply • Share ›

Sanidhya • a year ago

Can you please code one to find the actual path taken by the above algorithm to calculate the shortest path, need it to implement Chinese Postman

1 • Reply • Share ›

praveen • a year ago

can i get the path by this algorithm

• Reply • Share ›

kartik praveen • a year ago

Yes, you can get path. You need to maintain chosen k in a separate 2D array.

1 • Reply • Share ›

viki • 2 years ago

Changing the loop as below also gives the same solution:

for (i = 0; i < V; i++)

{

// Pick all vertices as source one by one

for (j = 0; j < V; j++)

{

// Pick all vertices as destination for the

// above picked source

for (k = 0; k < V; k++)

{

// If vertex k is on the shortest path from

see more

1 • Reply • Share ›

rahul • 2 years ago

"The above program only prints the shortest distances. We can modify the solution to print the shortest paths also by storing the predecessor information in a separate 2D matrix."
Is there a difference between shortest path and shortest distance here?

• Reply • Share ›

Arun D Patil • 2 years ago

Hi,

In the main function, may I know how to generate random graph? and pass the same to Floyds algorithm?

Thanks

• Reply • Share ›

prashanth h r • 2 years ago

write a c program to check whether 2 words are anagrams using dynamic programming i.e. the words TEA and EAT are anagrams

can any one solve this and post it here

• Reply • Share ›

danny prashanth h r • 10 months ago

Hashing or sorting can be used, to check anagrams

• Reply • Share ›

h prashanth h r • a year ago

that is just an LCS problem . find out LCS . if LCS== length of array. then anagram

• Reply • Share ›

brahma h • 10 months ago

For input TEA and EAT LCS is 2 so 2!=3 but they are anagrams..

• Reply • Share ›

ron007 brahma • 4 months ago

sort both the strings and do comparison
if(string a==string b)"yes"
else"no"

• Reply • Share ›

code1101 • 2 years ago

```
    public static int[][] getAllShortestPath(int[][] graph)  {
        int N = graph.length;
        for(int i=0; i<N; i++) {
            for(int j=i+1; j<N; j++) {
                // INF is a large value
                if(graph[i][j] != INF) {
                    for(int k=j+1; k<N; k++) {
                        if(graph[i][k] > graph[i][j] + graph[j][k]) {
                            graph[i][k] = graph[i][j] + graph[j][k];
                        }
                    }
                }
            }
        }
        return graph;
    }
```

• Reply • Share ›

lohith • 3 years ago

```
 import java.util.ArrayList;
import java.util.HashMap;
```

```
public class FloydWarshallsAlgorithm {

        public static int INF = 999;

        public static void main(String str[]) {

                int path[][] = { { 0, 5, INF, 10 }, { INF, 0, 3, INF },
                                { INF, INF, 0, 1 }, { INF, INF, INF, 0 } };

                ArrayList<Integer> visited = new ArrayList<Integer>();

                for(int i=0;i<4;i++){
                        for(int j=0;j<4;j++)
                                System.out.println(i+1+"  to  "+(j+1) +"  =  "+shortestPath
                }
```

see more

Reply • Share ›

lohith • 3 years ago

[sourcecode language="JAVA"]

import java.util.ArrayList;
import java.util.HashMap;

public class FloydWarshallsAlgorithm {

public static int INF = 999;

public static void main(String str[]) {

int path[][] = { { 0, 5, INF, 10 }, { INF, 0, 3, INF },
{ INF, INF, 0, 1 }, { INF, INF, INF, 0 } };

ArrayList<Integer> visited = new ArrayList<Integer>();

for(int i=0;i<4;i++){
for(int j=0;j<4;j++)
System.out.println(i+1+" to "+(j+1) +" = "+shortestPath(i,j,path,visited));
}

see more

Reply • Share ›

marti • 3 years ago

can you please explain why the k-loop is not the innermost out of the 3 loops?

• Reply • Share ›

kartik marti • 3 years ago

If we make k-loop as the innermost loop, then the code will not follow the optimal substructure property shown in the diagram. The main idea for puttink k-loop outside is following:
When we include vertex number k to the intermediate set, we must have shortest distance between every pair of vertex such that the shortest distances consider vertices from the set {0, 1, .. k-1} as intermediate vertices.
If we make the k-loop innermost, then we won't have the above mentioned values available before the start of a iteration.
Let me know if you still have doubts.

2 • Reply • Share ›

amrit kartik • 3 years ago

okay , this is how i understand it. we are considering k=1 the first time, then k=1,2 then k=1,2,3 etc , this is the optimal substructure and hence the code...am i right?

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

2 • Reply • Share ›

-
-
-
-
    - Interview Experiences
    - Advanced Data Structures
    - Dynamic Programming
    - Greedy Algorithms
    - Backtracking
    - Pattern Searching
    - Divide & Conquer
    - Mathematical Algorithms
    - Recursion
    - Geometric Algorithms
-

- ## Popular Posts

    - All permutations of a given string
    - Memory Layout of C Programs
    - Understanding “extern” keyword in C
    - Median of two sorted arrays
    - Tree traversal without recursion and without stack!
    - Structure Member Alignment, Padding and Data Packing
    - Intersection point of two Linked Lists
    - Lowest Common Ancestor in a BST.
    - Check if a binary tree is BST or not
    - Sorted Linked List to Balanced BST
- Follow @GeeksforGeeks

- ## Recent Comments

    - lt_k

      i need help for coding this function in java...

      Java Programming Language · 1 hour ago

    - Piyush

      What is the purpose of else if (recStack[*i])...

      Detect Cycle in a Directed Graph · 1 hour ago

    - Andy Toh

      My compile-time solution, which agrees with the...

Dynamic Programming | Set 16 (Floyd Warshall Algorithm) · 1 hour ago

- lucy

  because we first fill zero in first col and...

  Dynamic Programming | Set 29 (Longest Common Substring) · 2 hours ago

- lucy

  @GeeksforGeeks i don't n know what is this long...

  Dynamic Programming | Set 28 (Minimum insertions to form a palindrome) · 3 hours ago

- manish

  Because TAN is not a subsequence of RANT. ANT...

  Given two strings, find if first string is a subsequence of second · 3 hours ago

-