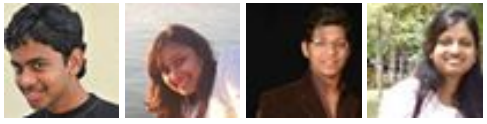
Dynamic Programming | Set 16 (Floyd Warshall Algorithm)

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
```

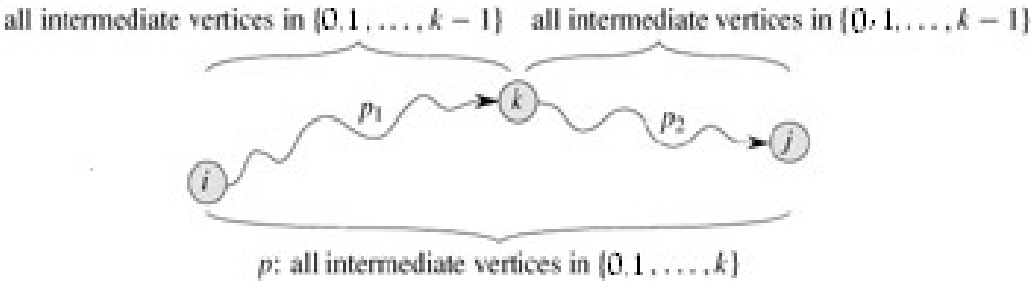
Floyd Warshall Algorithm

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.



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
```

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

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

// Solves the all-pairs shortest path problem using Floyd Warshall alg
void floydWarshell (int graph[][V])
{
    /* dist[][] will be the output matrix that will finally have the s
      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 bet
      pairs of vertices such that the shortest distances consider only
      vertices in set {0, 1, 2, .. k-1} as intermediate vertices.
      ----> After the end of a iteration, vertex no. k is added to the
      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
```


Recent Comments

```
    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[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

▶ Java Algorithm

▶ Algorithm
▶ Graph Java
▶ Graph C++

▶ Java Array
▶ Java C++
▶ Graph Theory

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.

18 Comments GeeksforGeeks

Sort by Newest

Join the discussion…

prashant jha • 2 days ago

its a bottom up dynamaic programming ques
there is two possibility for a path to exist between vertex i and j using intermed
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

prashant jha • 2 days 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

Mohamad • a month ago

Hey guys,

Here's an implementation of Floyd-Warshall algorithm in C++ which finds the
paths

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

5

nickpsar • 3 months ago

thanks guys!!!

now with this example i can say that i finally managed to fill up all the blanks at
But.....hmm

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

Sanidhya • 5 months ago

Can you please code one to find the actual path taken by the above algorithm
implement Chinese Postman

1

praveen • 6 months ago

can i get the path by this algorithm

kartik ➚ praveen • 6 months ago

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

viki • 6 months 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
```

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

{

// If vertex k is on the shortest path from

see more

rahul • 8 months ago

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

Arun D Patil • 11 months ago

Hi,

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

Thanks

prashanth h r • a year ago

write a c program to check whether 2 words are anagrams using dynamic prc
are anagrams

can any one solve this and post it here

h ➦ prashanth h r • 2 months ago

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

code1101 • a year 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;
    }
```


lohith • 2 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, (

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)
}
```

see more

lohith • 2 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

marti • 2 years ago

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

kartik ➦ marti • 2 years ago

If we make k-loop as the innermost loop, then the code will not follow th
shown in the diagram. The main idea for puttink k-loop outside is follow
When we include vertex number k to the intermediate set, we must hav
pair of vertex such that the shortest distances consider vertices from th
vertices.
If we make the k-loop innermost, then we won't have the above mentio
a iteration.
Let me know if you still have doubts.

1

amrit ➦ kartik • 2 years ago

okay , this is how i understand it. we are considering k=1 the fir
this is the optimal substructure and hence the code...am i right'

```
/* Paste your code here (You may delete these lines if
```

Subscribe Add Disqus to your site