# GeeksforGeeks

A computer science portal for geeks

GeeksQuiz

- Home
- Algorithms
- DS
- GATE
- Interview Corner
- Q&A
- C
- C++
- Java
- Books
- Contribute
- Ask a Q
- About

Array
Bit Magic
C/C++
Articles
GFacts
Linked List
MCQ
Misc
Output
String
Tree
Graph

## Find the minimum cost to reach destination using a train

There are N stations on route of a train. The train goes from station 0 to N-1. The ticket cost for all pair of stations (i, j) is given where j is greater than i. Find the minimum cost to reach the destination.

Consider the following example:

```
Input:
cost[N][N] = { {0, 15, 80, 90},
               {INF, 0, 40, 50},
               {INF, INF, 0, 70},
               {INF, INF, INF, 0}
             };
There are 4 stations and cost[i][j] indicates cost to reach j
from i. The entries where j < i are meaningless.
```

```
Output:
The minimum cost is 65
The minimum cost can be obtained by first going to station 1
from 0. Then from station 1 to station 3.
```

**We strongly recommend to minimize your browser and try this yourself first.**

The minimum cost to reach N-1 from 0 can be recursively written as following:

```
minCost(0, N-1) = MIN { cost[0][n-1],
                        cost[0][1] + minCost(1, N-1),
                        minCost(0, 2) + minCost(2, N-1),
                        ........,
                        minCost(0, N-2) + cost[N-2][n-1] }
```

The following is C++ implementation of above recursive formula.

```
// A naive recursive solution to find min cost path from station 0
// to station N-1
#include<iostream>
#include<climits>
using namespace std;

// infinite value
#define INF INT_MAX

// Number of stations
#define N 4

// A recursive function to find the shortest path from
// source 's' to destination 'd'.
int minCostRec(int cost[][N], int s, int d)
{
    // If source is same as destination
    // or destination is next to source
    if (s == d || s+1 == d)
      return cost[s][d];

    // Initialize min cost as direct ticket from
    // source 's' to destination 'd'.
    int min = cost[s][d];

    // Try every intermediate vertex to find minimum
    for (int i = s+1; i<d; i++)
    {
        int c = minCostRec(cost, s, i) +
                minCostRec(cost, i, d);
        if (c < min)
           min = c;
    }
    return min;
}

// This function returns the smallest possible cost to
```

```
// reach station N-1 from station 0. This function mainly
// uses minCostRec().
int minCost(int cost[][N])
{
    return minCostRec(cost, 0, N-1);
}

// Driver program to test above function
int main()
{
    int cost[N][N] = { {0, 15, 80, 90},
                       {INF, 0, 40, 50},
                       {INF, INF, 0, 70},
                       {INF, INF, INF, 0}
                     };
    cout << "The Minimum cost to reach station "
         << N << " is " << minCost(cost);
    return 0;
}
```

Output:

```
The Minimum cost to reach station 4 is 65
```

Time complexity of the above implementation is exponential as it tries every possible path from 0 to N-1. The above solution solves same subrpoblems multiple times (it can be seen by drawing recursion tree for minCostPathRec(0, 5).
Since this problem has both properties of dynamic programming problems ((see this and this). Like other typical Dynamic Programming(DP) problems, re-computations of same subproblems can be avoided by storing the solutions to subproblems and solving problems in bottom up manner.

One dynamic programming solution is to create a 2D table and fill the table using above given recursive formula. The extra space required in this solution would be $O(N^2)$ and time complexity would be $O(N^3)$

We can solve this problem using O(N) extra space and $O(N^2)$ time. The idea is based on the fact that given input matrix is a Directed Acyclic Graph (DAG). The shortest path in DAG can be calculated using the approach discussed in below post.

Shortest Path in Directed Acyclic Graph

We need to do less work here compared to above mentioned post as we know topological sorting of the graph. The topological sorting of vertices here is 0, 1, ..., N-1. Following is the idea once topological sorting is known.

The idea in below code is to first calculate min cost for station 1, then for station 2, and so on. These costs are stored in an array dist[0...N-1].

1) The min cost for station 0 is 0, i.e., dist[0] = 0

2) The min cost for station 1 is cost[0][1], i.e., dist[1] = cost[0][1]

3) The min cost for station 2 is minimum of following two.

a) dist[0] + cost[0][2]
b) dist[1] + cost[1][2]

3) The min cost for station 3 is minimum of following three.
a) dist[0] + cost[0][3]
b) dist[1] + cost[1][3]
c) dist[2] + cost[2][3]

Similarly, dist[4], dist[5], ... dist[N-1] are calculated.

Below is C++ implementation of above idea.

```
// A Dynamic Programming based solution to find min cost
// to reach station N-1 from station 0.
#include<iostream>
#include<climits>
using namespace std;

#define INF INT_MAX
#define N 4

// This function returns the smallest possible cost to
// reach station N-1 from station 0.
int minCost(int cost[][N])
{
    // dist[i] stores minimum cost to reach station i
    // from station 0.
    int dist[N];
    for (int i=0; i<N; i++)
       dist[i] = INF;
    dist[0] = 0;

    // Go through every station and check if using it
    // as an intermediate station gives better path
    for (int i=0; i<N; i++)
       for (int j=i+1; j<N; j++)
          if (dist[j] > dist[i] + cost[i][j])
             dist[j] = dist[i] + cost[i][j];

    return dist[N-1];
}

// Driver program to test above function
int main()
{
    int cost[N][N] = { {0, 15, 80, 90},
                       {INF, 0, 40, 50},
                       {INF, INF, 0, 70},
                       {INF, INF, INF, 0}
                     };
    cout << "The Minimum cost to reach station "
         << N << " is " << minCost(cost);
    return 0;
```

```
}
```

Output:

```
The Minimum cost to reach station 4 is 65
```

This article is contributed by **Udit Gupta**. 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)
- 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)
- Biconnected Components

Tags: Dynamic Programming



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

**prashant jha** • 2 months ago

since it is a DAG ,apply topologial sorting and then it is a legitimate dp problem

• Reply • Share ›

**creeping_death** • 2 months ago

Python solution: O(N^2) dp

https://gist.githubusercontent...

• Reply • Share ›

**Gaurav** • 2 months ago

How about this approach. O(n) space and O(n) time.
https://ideone.com/S8gEF1

• Reply • Share ›

**Manish** ➚ Gaurav • a month ago

Your O(n) solution is wrong, although approach is right. At first look, i dont think O(n2) DP solution is possible. But if some one come up with linear solution pls do share.

• Reply • Share ›

**Priyal Rathi** • 2 months ago

Another O(n^2) DP approach:
let DP[i]= min cost to reach from 0th city to ith city.
DP[j]= MIN (for all k= 1 to j-1 cities, min cost to reach from 0th to Kth city + cost to reach from Kth city to jth city)

Time complexity: O(n^2)
Space complexty: O(n)
Link of code: http://ideone.com/GmATrO

• Reply • Share ›

**NoobInHell** • 2 months ago

This can be solved using Djikstra

• Reply • Share ›

**Itachi Uchiha** ➚ NoobInHell • a month ago

There is no need to use Dijkstra cuz the graph is DAG so only topological sorting followed by relaxation of edges is sufficient to get the shortest path. We don't even need to apply topological sort as its itself given in the question, simply did relaxation to get the answer.

• Reply • Share ›

**NoobInHell** ➚ Itachi Uchiha • 25 days ago

yeah should have seen the whole problem, immediately jumped on the

conclusion.

• Reply • Share ›

**Sudhir Kumar** • 2 months ago

One simpler approach will be using DFS traveral. Find all possible paths between source and destination using DFS traversal and calculate its cost as sum of the path. then select the minium one.
Will be solved in linear time O(v+E)

• Reply • Share ›

**Ankit Chaudhary** Sudhir Kumar • a month ago

There are O(N^2) edges. So in E = N^2.
Hence not a linear solution

• Reply • Share ›

**rahul singh** • 2 months ago

One simpler method is to just traverse and construct a difference matrix .And then take the minimum from each column and then take their sum , we will get shortest path., for the above problem, the difference matrix will be:-

{ { 0,15,65,10},
{INF,0,40,50},
{INF,INF,0,70},
{INF,INF,INF,0}};

Take minimum from each column other then '0', except from the first column 0+15+40+10=65

• Reply • Share ›

**Paridhi** • 2 months ago

the second way of doing it is like Dijkstra's shortest path Algorithm only..

2 • Reply • Share ›

**Debanjan Chanda** • 2 months ago

Problem source?

• 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 · 2 hours ago

- Piyush

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

  Detect Cycle in a Directed Graph · 2 hours ago

- Andy Toh

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

Dynamic Programming | Set 16 (Floyd Warshall Algorithm) · 2 hours 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