

**East West University**

**Project Report**

**CSE-246, Algorithm**

**Submitted by-**

**Nazmul Hasan Khan**

**2017-2-60-055**

**Md.Jusef**

**2017-2-60-039**

**Submitted to-**

**Jesan Ahammed Ovi,**

Senior lecturer,

Department of Computer Science & Engineering

East West University

**Problem Statement:** We have given a polygon, with having input of co-ordinate from user. After the having the input of the co-ordinates, this program needs to find the lowest cost solution for creating a triangle into that polygon. Like other dynamic programming problems, this one requires determining a structure. We want to break the problem down into smaller chunks, with solutions that can be combined to solve the larger issue. We must, as with any DP solution, find a way to break the problem down into smaller sub problems and determine a recursive formulation, which represents the best solution to each problem in terms of sub problem solutions.

**ii. System Requirements:**

**Processor:** Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz 2.71 GHz

**RAM:** 12GB

**Operating system:** Windows 10 Pro

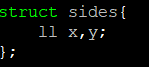
**IDE:** Code Blocks

**System Design:** In this project, to solve this problem we have taken the dynamic programming approach using Matrix Chain Multiplication technique. To begin, let us assume that all we really want to know is the minimum cost. If we are only multiplying two matrices, there is only one way to multiply them, so the minimum cost is the cost of doing this. In general, we can find the minimum cost using the following recursive algorithm:

* • Take the sequence of matrices and separate it into two subsequences.
* • Find the minimum cost of multiplying out each subsequence.
* • Add these costs together and add in the cost of multiplying the two result matrices.
* • Do this for each possible position at which the sequence of matrices can be split, and take the minimum over all of them.

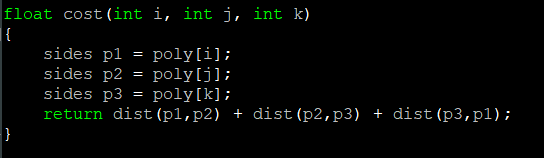
For example, if we have four matrices *ABCD*, we compute the cost required to find each of (*A*)(*BCD*), (*AB*)(*CD*), and (*ABC*)(*D*), making recursive calls to find the minimum cost to compute *ABC*, *AB*, *CD*, and *BCD*. We then

choose the best one. Better still, this yields not only the minimum cost, but also demonstrates the best way of doing the multiplication: group it the way that yields the lowest total cost, and do the same for each factor.

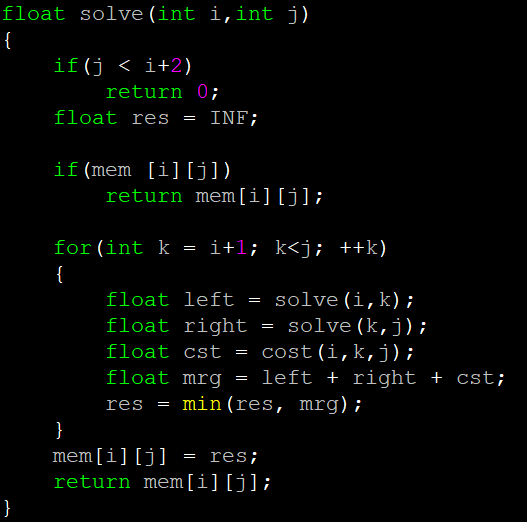
**Implementation:** To store the value of the co-ordinates we have taken a struct type variable name sides.

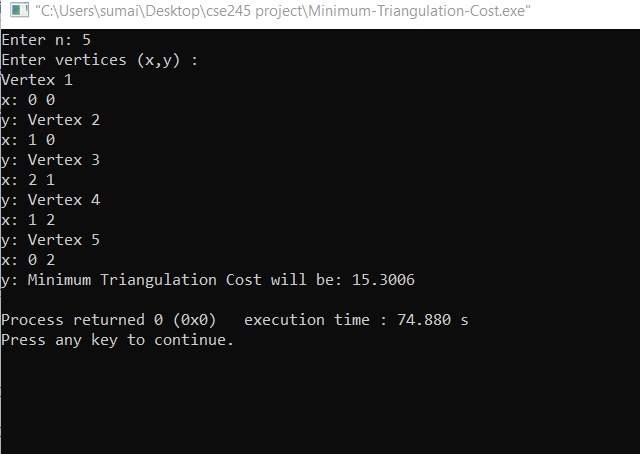
Here we store the values of the co-ordinates.

Then we have a distance function to measure the distance between the co-ordinates. We have a function named as cost , to measure the cost to make a triangle.



Then there is Solve function, where the program compares between the cost of the triangle and find out the lowest cost triangle.



**Output:**

Here the user input the co-ordinates then this program calculates the distance among them and return the lowest possible cost.

**Limitations:** One of the primary limitation of this project is its time complexity which is n3. Which means with the increasing of the co-ordinates its time complexity will increase.