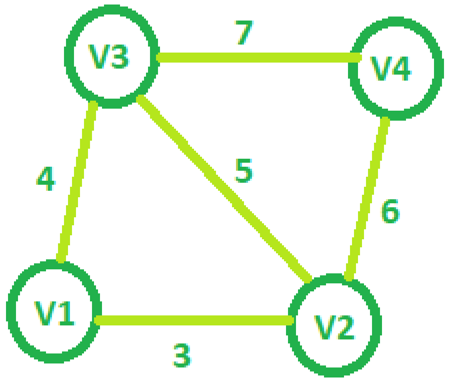
My Peer-graded Assignment 3

Aim - Minimum spanning tree cost of given Graphs

Given an undirected graph of **V** nodes (V > 2) named $V_1, V_2, V_3, \ldots, V_n$. Two nodes $\mathbf{V_i}$ and $\mathbf{V_j}$ are connected to each other if and only if $\mathbf{0 < | i - j | \leq 2}$. Each edge between any vertex pair $\mathbf{(V_i, V_j)}$ is assigned a weight **i + j**. The task is to find the cost of the minimum spanning tree of such graph with **V** nodes.

Examples:

Input: *V = 4*



Output: *13*

Input: *V = 5*

Output: *21*

Recommended: Please try your approach on *{IDE}* first, before moving on to the solution.

Approach: Starting with a graph with minimum nodes (i.e. 3 nodes), the cost of the minimum spanning tree will be 7. Now for every node **i** starting from the fourth node which can be added to this graph, $\mathbf{i^{th}}$ node can only be connected to $\mathbf{(i - 1)^{th}}$ and $\mathbf{(i - 2)^{th}}$ node and the minimum spanning tree will only include the node with the minimum weight so the newly added edge will have the weight **i + (i – 2)**.

So addition of fourth node will increase the overall weight as 7 + (4 + 2) = 13
Similarly adding fifth node, weight = 13 + (5 + 3) = 21
…
*For n^{th} node, **weight = weight + (n + (n – 2))**.*

This can be generalized as **weight = V^2 – V + 1** where **V** is the total nodes in the graph.
Below is the implementation of the above approach:

C++

filter_none
edit
play_arrow
brightness_4

```
// C++ implementation of the approach
#include <bits/stdc++.h>
using namespace std;

// Function that returns the minimum cost
// of the spanning tree for the required graph
int getMinCost(int Vertices)
{
    int cost = 0;

    // Calculating cost of MST
    cost = (Vertices * Vertices) - Vertices + 1;

    return cost;
}

// Driver code
int main()
{
    int V = 5;
    cout << getMinCost(V);

    return 0;
}
```

Java

filter_none
edit
play_arrow
brightness_4

```
// Java implementation of the approach
class GfG
{

// Function that returns the minimum cost
// of the spanning tree for the required graph
static int getMinCost(int Vertices)
```

```
{
    int cost = 0;

    // Calculating cost of MST
    cost = (Vertices * Vertices) - Vertices + 1;

    return cost;
}

// Driver code
public static void main(String[] args)
{
    int V = 5;
    System.out.println(getMinCost(V));
}
}

// This code is contributed by
// Prerna Saini.
```

C#

filter_none
edit
play_arrow
brightness_4

```
// C# implementation of the above approach
using System;

class GfG
{

    // Function that returns the minimum cost
    // of the spanning tree for the required graph
    static int getMinCost(int Vertices)
    {
        int cost = 0;

        // Calculating cost of MST
        cost = (Vertices * Vertices) - Vertices + 1;

        return cost;
    }

    // Driver code
    public static void Main()
    {
        int V = 5;
        Console.WriteLine(getMinCost(V));
    }
}

// This code is contributed by Ryuga
```

Python3

filter_none
edit
play_arrow
brightness_4

```
# python3 implementation of the approach

# Function that returns the minimum cost
# of the spanning tree for the required graph
def getMinCost( Vertices):
    cost = 0

    # Calculating cost of MST
    cost = (Vertices * Vertices) - Vertices + 1

    return cost

# Driver code
if __name__ == "__main__":

    V = 5
    print (getMinCost(V))
```

PHP

filter_none
edit
play_arrow
brightness_4

```
<?php
// PHP implementation of the approach
// Function that returns the minimum cost
// of the spanning tree for the required graph
function getMinCost($Vertices)
{
    $cost = 0;

    // Calculating cost of MST
    $cost = ($Vertices * $Vertices) - $Vertices + 1;

    return $cost;
}

// Driver code
$V = 5;
echo getMinCost($V);
```

```
#This Code is contributed by ajit..
?>
```

Output:

```
21
```