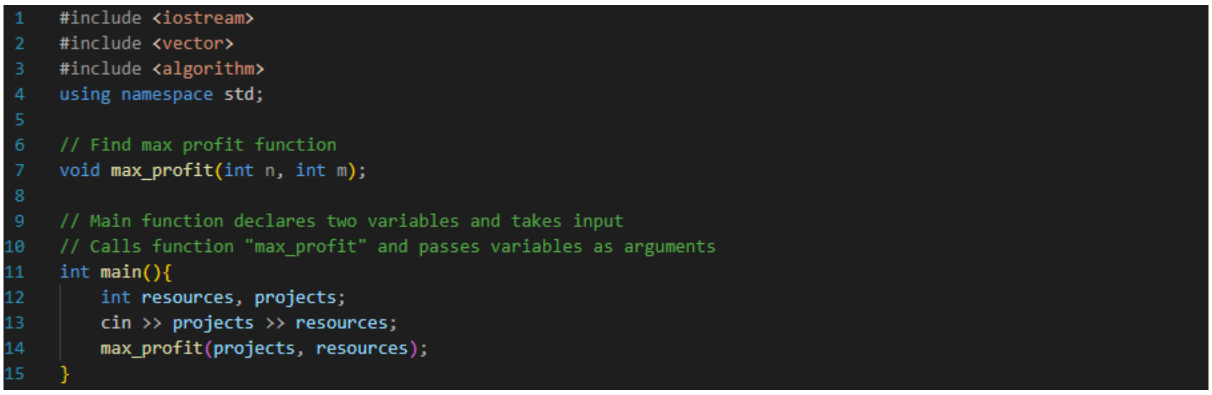
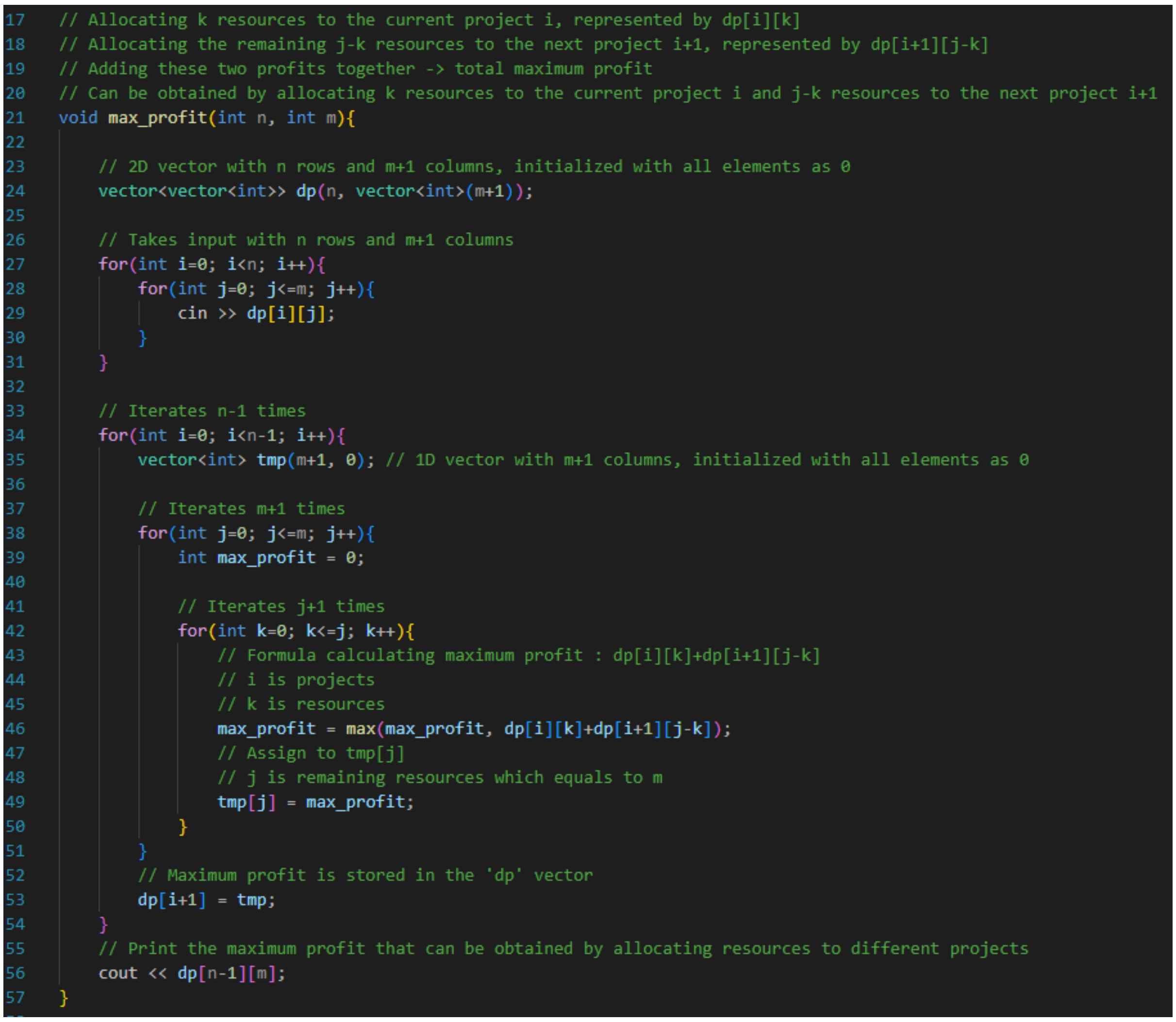
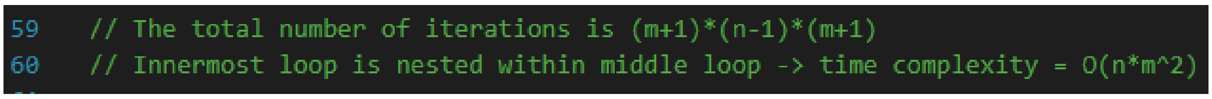
1.vscode/g++
2.Methods and solution (details in comments), time complexity



```cpp
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

// Find max profit function
void max_profit(int n, int m);

// Main function declares two variables and takes input
// Calls function "max_profit" and passes variables as arguments
int main(){
    int resources, projects;
    cin >> projects >> resources;
    max_profit(projects, resources);
}
```



```cpp
// Allocating k resources to the current project i, represented by dp[i][k]
// Allocating the remaining j-k resources to the next project i+1, represented by dp[i+1][j-k]
// Adding these two profits together -> total maximum profit
// Can be obtained by allocating k resources to the current project i and j-k resources to the next project i+1
void max_profit(int n, int m){

    // 2D vector with n rows and m+1 columns, initialized with all elements as 0
    vector<vector<int>> dp(n, vector<int>(m+1));

    // Takes input with n rows and m+1 columns
    for(int i=0; i<n; i++){
        for(int j=0; j<=m; j++){
            cin >> dp[i][j];
        }
    }

    // Iterates n-1 times
    for(int i=0; i<n-1; i++){
        vector<int> tmp(m+1, 0); // 1D vector with m+1 columns, initialized with all elements as 0

        // Iterates m+1 times
        for(int j=0; j<=m; j++){
            int max_profit = 0;

            // Iterates j+1 times
            for(int k=0; k<=j; k++){
                // Formula calculating maximum profit : dp[i][k]+dp[i+1][j-k]
                // i is projects
                // k is resources
                max_profit = max(max_profit, dp[i][k]+dp[i+1][j-k]);
                // Assign to tmp[j]
                // j is remaining resources which equals to m
                tmp[j] = max_profit;
            }
        }
        // Maximum profit is stored in the 'dp' vector
        dp[i+1] = tmp;
    }
    // Print the maximum profit that can be obtained by allocating resources to different projects
    cout << dp[n-1][m];
}
```



```cpp
// The total number of iterations is (m+1)*(n-1)*(m+1)
// Innermost loop is nested within middle loop -> time complexity = O(n*m^2)
```