Q5. Implement Optimal Binary Search Tree. Analyze its time complexity.

TIME COMPLEXITY: O(n ^ 3)

SPACE COMPLEXITY: O(n ^ 2)

```
#include <iostream>
#include <queue>
#include <vector>
#include <string>
#include <map>
#include <cmath>

using namespace std;

vector<int> getPreSum(vector<int> & frequencies)
{
    vector<int> sum(frequencies.size());
    sum[0] = frequencies[0];
    for(int i = 1; i < frequencies.size(); ++i)
        sum[i] = sum[i - 1] + frequencies[i];
    return sum;
}

int sumInterval(vector<int> & sum, int i, int j)
{
    return sum[j] - ((i == 0) ? 0 : sum[i - 1]);
}

int optimalSearchTree(vector<int> & keys, vector<int> & freq)
{
    vector<int> sum = getPreSum(freq);
    int n = int(keys.size());
    int cost[n][n];
    for (int i = 0; i < n; i++)
        cost[i][i] = freq[i];
    for (int L = 2; L <= n; L++)
    {
        for (int i = 0; i <= n - L + 1; i++)
        {
            int j = i + L - 1;
            cost[i][j] = INT_MAX;
            for (int r = i; r <= j; r++)
            {
                int c = ((r > i) ? cost[i][r - 1] : 0) + ((r < j) ? cost[r
                 + 1][j] : 0) + sumInterval(sum, i, j);
                cost[i][j] = min(c, cost[i][j]);
            }
        }
    }
    return cost[0][n-1];
}
```

```
int main()
{
    vector<int> keys = {10, 12, 20, 24, 30};
    vector<int> frequencies = {34, 8, 50, 12, 9};
    cout << "KEYS: ";
    for(int & v : keys)
        cout << v << " ";
    cout << endl;
    cout << "FREQUENCIES: ";
    for(int & v : keys)
        cout << v << " ";
    cout << endl;
    cout << "Cost of Optimal BST: " << optimalSearchTree(keys, frequencies)
     << endl;
    return 0;
}


//OUTPUT:
//
//KEYS: 10 12 20 24 30
//FREQUENCIES: 10 12 20 24 30
//Cost of Optimal BST: 193
//
//Program ended with exit code: 0
```