**OPTIMAL BINARY SEARCH TREE**

#include <bits/stdc++.h>

using namespace std;

int sum(int frequency[], int i, int j)

{

int sum = 0;

for (int x = i; x <= j; x++)

sum += frequency[x];

return sum;

}

int optimalCost(int frequency[], int i, int j)

{

if (j < i)

return 0;

if (j == i)

return frequency[i];

int frequencySum = sum(frequency, i, j);

int min = INT\_MAX;

for (int r = i; r <= j; ++r)

{

int cost = optimalCost(frequency, i, r - 1) + optimalCost(frequency, r + 1, j);

if (cost < min)

min = cost;

}

return min + frequencySum;

}

int optimalSearchTree(int keys[], int frequency[], int n)

{

return optimalCost(frequency, 0, n - 1);

}

int main()

{

int keys[] = {10, 12, 20};

int frequency[] = {34, 8, 50};

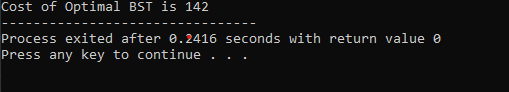
int n = sizeof(keys) / sizeof(keys[0]);

cout << "Cost of Optimal BST is " << optimalSearchTree(keys, frequency, n);

return 0;

}

**OUTPUT**

****