1. **Optimal binary search using dynamic programming**

**Code:**

#include <stdio.h>

#include <stdlib.h>

#include <limits.h>

// A utility function to get sum of array elements freq[i] to freq[j]

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

{

int s = 0;

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

s += freq[k];

return s;

}

// A utility function to print the constructed BST

void printOptimalBST(int root[], int i, int j, int level, int n)

{

if (i > j) {

printf("Level %d: Null\n", level);

return;

}

int mid = root[i \* n + j];

printf("Level %d: %d\n", level, mid);

printOptimalBST(root, i, mid - 1, level + 1, n);

printOptimalBST(root, mid + 1, j, level + 1, n);

}

// Function to construct optimal BST and return its cost

int optimalBST(int keys[], int freq[], int n)

{

// Create an auxiliary 2D matrix to store results of subproblems

int cost[n][n];

// root[i][j] will store the index of the root of the optimal BST that includes keys[i] to keys[j]

int root[n \* n];

// Initialize cost and root arrays as INT\_MAX

for (int i = 0; i < n; i++) {

cost[i][i] = freq[i];

root[i \* n + i] = i;

}

// Build the table cost[][] in a bottom-up manner

for (int L = 2; L <= n; L++) {

for (int i = 0; i <= n - L; 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) +

sum(freq, i, j);

if (c < cost[i][j]) {

cost[i][j] = c;

root[i \* n + j] = r;

}

}

}

}

printf("Optimal BST structure:\n");

printOptimalBST(root, 0, n - 1, 0, n);

return cost[0][n - 1];

}

int main()

{

int n;

printf("Enter the number of keys: ");

scanf("%d", &n);

int keys[n], freq[n];

printf("Enter the keys:\n");

for (int i = 0; i < n; i++) {

scanf("%d", &keys[i]);

}

printf("Enter the frequencies:\n");

for (int i = 0; i < n; i++) {

scanf("%d", &freq[i]);

}

printf("Cost of optimal BST is %d\n", optimalBST(keys, freq, n));

return 0;

}

**Output:**

Enter the number of keys: 5

Enter the keys:

4

8

6

7

3

Enter the frequencies:

1

2

3

4

5

Optimal BST structure:

Level 0: 3

Level 1: 1

Level 2: 0

Level 3: Null

Level 3: Null

Level 2: 2

Level 3: Null

Level 3: Null

Level 1: 4

Level 2: Null

Level 2: Null

Cost of optimal BST is 30

--------------------------------

Process exited after 13.65 seconds with return value 0

Press any key to continue . . .

