106. Optimal Binary Search Tree

AIM: To Find the search cost in a binary search tree using dynamic programming

PROGRAM:

```
def optimal_bst(keys, freq):
    n = len(keys)
    cost = [[0] * n for _ in range(n)]

    for i in range(n):
        cost[i][i] = freq[i]

    for length in range(2, n + 1):
        for i in range(n - length + 1):
            j = i + length - 1
            cost[i][j] = float('inf')
            sum_freq = sum(freq[i:j+1])
            for r in range(i, j + 1):
                c = cost[i][r - 1] if r > i else 0
                c += cost[r + 1][j] if r < j else 0
                if c < cost[i][j]:
                    cost[i][j] = c

            cost[i][j] += sum_freq

    return cost[0][n - 1]

keys = ["key1", "key2", "key3", "key4", "key5"]
freq = [4, 2, 6, 3, 1]

min_cost = optimal_bst(keys, freq)
print(f"Minimum average search cost for optimal BST: {min_cost}")
```

OUTPUT:

```
Minimum average search cost for optimal BST: 29
```

TIME COMPLEXITY: O(n^3)