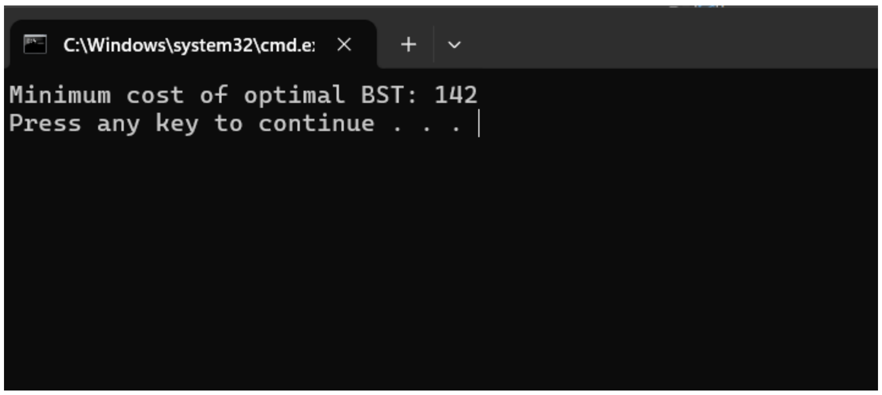
107)Optimal binary search tree

CODE:

```
def optimal_bst(keys, freq, n):
    cost = [[0] * (n) for _ in range(n)]
    dp = [[0] * (n) for _ in range(n)]
    root = [[-1] * (n) for _ in range(n)]

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

    for length in range(2, n+1):
        for i in range(n-length+1):
            j = i + length - 1
            dp[i][j] = float('inf')

            for k in range(i, j+1):
                left_cost = dp[i][k-1] if k > i else 0
                right_cost = dp[k+1][j] if k < j else 0
                total_cost = left_cost + right_cost + sum(freq[i:j+1])

                if total_cost < dp[i][j]:
                    dp[i][j] = total_cost
                    cost[i][j] = total_cost
                    root[i][j] = k

    return dp[0][n-1], root

keys = [10, 12, 20]
freq = [34, 8, 50]
n = len(keys)
min_cost, root = optimal_bst(keys, freq, n)
print(f"Minimum cost of optimal BST: {min_cost}")
```

OUTPUT:



```
Minimum cost of optimal BST: 142
Press any key to continue . . .
```

TIME COMPLEXITY : O(n^3)