107. Optimal Binary Search Tree

PROGRAM:-

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

    # cost[i][i] is equal to freq[i]
    for i in range(n):
        cost[i][i] = freq[i]

    # Now we need to consider chains of length 2, 3, ..., n
    for L in range(2, n + 1):
        for i in range(n - L + 1):
            j = i + L - 1
            cost[i][j] = float('inf')

            # Try making all keys in interval keys[i..j] as root
            for r in range(i, j + 1):
                c = 0
                if r > i:
                    c += cost[i][r - 1]
                if r < j:
                    c += cost[r + 1][j]
                c += sum(freq[i:j + 1])

                if c < cost[i][j]:
                    cost[i][j] = c

    return cost[0][n - 1]

# Example usage:
keys = [10, 12, 20]
freq = [34, 8, 50]
min_cost = optimal_bst(keys, freq)
print(f"The cost of the optimal BST is {min_cost}")
```

OUTPUT:-

```
The cost of the optimal BST is 142

=== Code Execution Successful ===
```

TIME COMPLEXITY:-$O(n^3)$