106. Optimal binary search tree

PROGRAM:

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

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

    for L in range(2, n + 1):
        for i in range(n - L + 1):
            j = i + L - 1
            cost[i][j] = float('inf')
            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
                c += sum(freq[i:j + 1])
                if c < cost[i][j]:
                    cost[i][j] = c

    return cost[0][n - 1]

keys = [10, 12, 20]
freq = [34, 8, 50]

result = optimal_bst(keys, freq)
print("Cost of optimal BST is:", result)
```

OUTPUT:

```
Cost of optimal BST is: 142

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

TIMECOMPLEXITY:O(n^3)