

2. Knapsack Problem

Code:

```
def knapsack(weights, values, capacity):  
    n = len(weights)  
    dp = [[0 for _ in range(capacity + 1)] for _ in range(n + 1)]  
    for i in range(1, n + 1):  
        for w in range(1, capacity + 1):  
            if weights[i - 1] <= w:  
                dp[i][w] = max(dp[i - 1][w], dp[i - 1][w - weights[i - 1]] + values[i - 1])  
            else:  
                dp[i][w] = dp[i - 1][w]  
    return dp[n][capacity]  
weights = [1, 3, 4, 5]  
values = [1, 4, 5, 7]  
capacity = 7  
print(knapsack(weights, values, capacity))
```

output:

```
PS C:\Users\karth>  
PS C:\Users\karth> & C:/Users/karth/AppData/Local/Programs/Python/Python312/python.exe c:/Users/karth/OneDrive/Documents/OriginLab/daa.py  
9  
PS C:\Users\karth> █
```

Time complexity:

$F(n) = O(nw)$