

EXPERIMENT NO.03

Roll No.:24141001

Batch:I1

Title: Knapsack problem

Objective:

- Understand the Knapsack optimization problem.
- Study the difference between 0/1 Knapsack and Fractional Knapsack.
- Apply the Greedy strategy to solve the Fractional Knapsack problem.
- Implement the algorithm in C.
- Analyze time and space complexity.
- Learn real-life applications such as resource allocation, budgeting, and cargo loading.

Theory:

Two Types of Knapsack Problems

1. 0/1 Knapsack (Not Greedy)

- Item can be either taken completely or not taken at all.
- Cannot take fractions.
- Solved using Dynamic Programming.

2. Fractional Knapsack (Greedy)

- Items can be divided, meaning fractions are allowed.
- Greedy method applies here:
Always pick the item with highest profit/weight ratio first.

Applications:

Resource allocation

Investment and budgeting

Loading cargo boxes

Project selection in limited time

Memory management

Cloud computing task allocation

Program:

```
#include <stdio.h>
int max(int a, int b) {
    return (a > b) ? a : b;
}

// Recursive function to solve knapsack
int knapsack(int W, int wt[], int val[], int n) {
    // Base case: no items or no capacity
    if (n == 0 || W == 0)
        return 0;

    // If weight of the nth item is more than capacity, skip it
    if (wt[n - 1] > W)
        return knapsack(W, wt, val, n - 1);

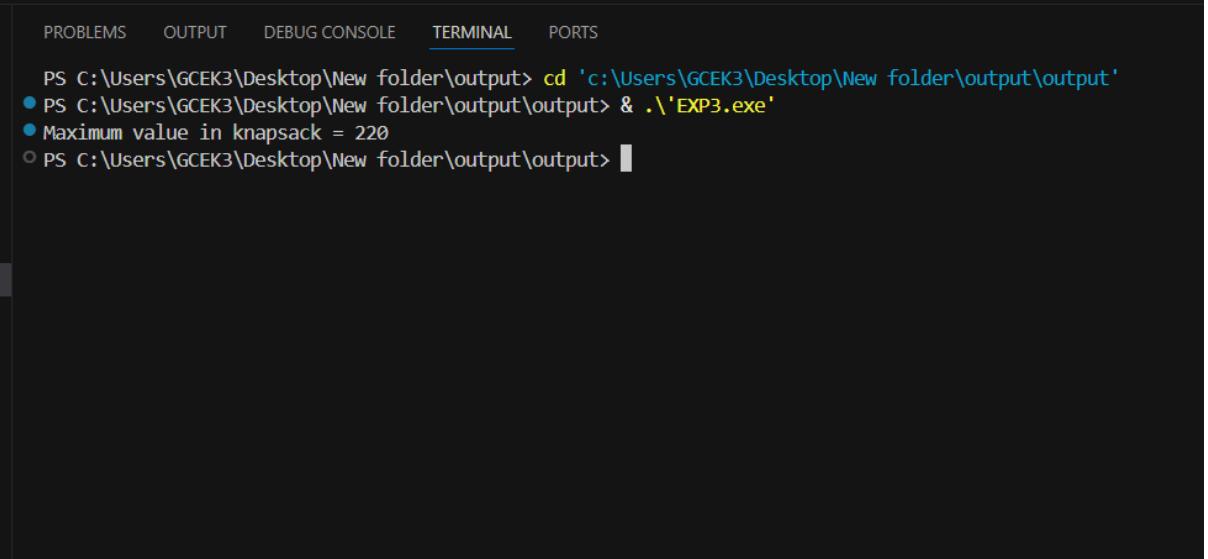
    // Return max of two cases:
    // (1) nth item included
    // (2) not included
    else
        return max(
            val[n - 1] + knapsack(W - wt[n - 1], wt, val, n - 1),
            knapsack(W, wt, val, n - 1)
        );
}

int main() {
    int val[] = {60, 100, 120};
    int wt[] = {10, 20, 30};
    int W = 50;
    int n = sizeof(val) / sizeof(val[0]);

    int result = knapsack(W, wt, val, n);
    printf("Maximum value in knapsack = %d\n", result);

    return 0;
}
```

}



The screenshot shows a terminal window with the following content:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\GCEK3\Desktop\New folder\output> cd 'c:\Users\GCEK3\Desktop\New folder\output\output'
● PS C:\Users\GCEK3\Desktop\New folder\output\output> & .\EXP3.exe
● Maximum value in knapsack = 220
○ PS C:\Users\GCEK3\Desktop\New folder\output\output>
```

The terminal window has tabs at the top: PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is underlined), and PORTS. The command entered was `cd 'c:\Users\GCEK3\Desktop\New folder\output\output'`, followed by `& .\EXP3.exe`. The output shows the maximum value found by the algorithm is 220.

Conclusion:

The Greedy Method efficiently solves the Fractional Knapsack Problem by selecting items based on the highest profit-to-weight ratio. It guarantees an optimal solution when fractional items are allowed. This problem has many real-life applications in resource optimization, data science, and logistics.