Lab Report-09

(0/1 Knapsack Problem)

CSE-2212 (Design and Analysis of Algorithms Lab)

Submitted By:

Name: Eyasir Ahamed
Exam Roll: 413
Class Roll: 15
Registration No: 202004017

Submitted To:

Sharad Hasan
Ex. Lecturer
Dept. of CSE
Sheikh Hasina University, Netrokona

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SHEIKH HASINA UNIVERSITY
NETROKONA, BANGLADESH

#9_0/1 Knapsack Problem

Problem Definition:

Given a set of items, each with a weight and a value, and a knapsack with a maximum weight capacity, the problem is to select the maximum total value of items that can be accommodated in the knapsack without exceeding its capacity.

Formal Statement of Algorithm (0/1 Knapsack using Dynamic Programming):

- Define a function knapsack that takes vectors of weights (wt) and values (val), the number of items n, and the capacity of the knapsack W as parameters.
- Initialize a 2D DP table dp with dimensions n x W+1, where dp[i][j] represents the maximum value that can be obtained with the first i items and a knapsack capacity of j.
- Base Condition: For the first item (i = 0), initialize dp[0][j] with the value of the first item if its weight is less than or equal to j.
- Iterate over the remaining items (i = 1 to n-1) and knapsack capacities (cap = 0 to W).
- For each item and capacity, calculate the maximum value that can be obtained by either:
    - Not taking the current item (notTaken = dp[ind - 1][cap]).

    - Taking the current item if its weight is less than or equal to the current capacity (taken = val[ind] + dp[ind - 1][cap - wt[ind]]).
- Update dp[ind][cap] with the maximum of notTaken and taken.
- The final result is in dp[n-1][W], representing the maximum value of items the thief can steal without exceeding the knapsack capacity.

Complexity Analysis:

- Time Complexity: The algorithm fills in a 2D DP table of size n x W, so the time complexity is O(nW), where n is the number of items and W is the capacity of the knapsack.
- Space Complexity: The space complexity is also O(nW) because of the DP table.

Actual Code and Output

```cpp
#include <bits/stdc++.h>
using namespace std;

int knapsack(vector<int>& wt, vector<int>& val, int n, int W) {
    vector<vector<int>> dp(n, vector<int>(W + 1, 0));

    for (int i = wt[0]; i <= W; i++) {
        dp[0][i] = val[0];
    }

    for (int ind = 1; ind < n; ind++) {
        for (int cap = 0; cap <= W; cap++) {
            int notTaken = dp[ind - 1][cap];
            int taken = INT_MIN;

            if (wt[ind] <= cap) {
                taken = val[ind] + dp[ind - 1][cap - wt[ind]];
            }

            dp[ind][cap] = max(notTaken, taken);
        }
    }

    return dp[n - 1][W];
}

int main() {
    vector<int> wt = {1, 2, 4, 5};
    vector<int> val = {5, 4, 8, 6};
    int W = 5;
    int n = wt.size();

    cout << "The Maximum value of items the thief can steal is " << knapsack(wt, val, n, W);

    return 0;
}
```

```
The Maximum value of items the thief can steal is 13[Finished in 1.2s]
```