VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

AAT Report

on

Analysis and Design of Algorithm

for

Optimal Cutting of Materials Using Dynamic Programming

Submitted by

Shreesha H Shetty (1BM21CS0209)
Sushanth (1BM21CS227)

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

B.M.S. COLLEGE OF ENGINEERING

(Autonomous Institution under VTU)

BENGALURU-560019

June-2023 to September-2023

Chapter 1:

Problem Statement: Optimal Cutting of Materials Using Dynamic Programming

You are given a long piece of material of length L units. Your goal is to cut this material into smaller pieces of various lengths to maximize the total value. Each piece of material can have a different value associated with it, and the value depends on its length.

However, there are specific lengths at which the material can be cut. These available cutting lengths are given in an array cuttingLengths[]. You can make as many cuts as needed, but they must be made only at the available cutting lengths.

Your task is to determine the arrangement of cuts that will maximize the total value of the cut pieces.

Chapter 2:

Design Technique and Algorithm

Step 1: Input

Prompt the user to enter the length of the rod and the number of available cutting sizes (numSizes).

Create a vector of SizeCost structures to store the cutting size and its cost for each size.

Read the cutting size and cost for each size from the user.

Step 2: Initialization

Initialize two 2D vectors:

dp, which stores the maximum cost achieved for each rod length and size.

cuts, which stores the index of the previous size that resulted in the maximum cost.

Initialize a vector selectedCounts to store the count of selected pieces for each cutting size.

Step 3: Dynamic Programming (DP) Calculation

Iterate over rod lengths from 1 to the given rod length.

For each rod length, iterate over each available cutting size (sizeIndex).

Check if the current cutting size can fit within the current rod length.

Calculate the maximum cost if the current cutting size is used and store it in current_val.

Compare current_val with the previously calculated maximum cost for the same rod length and size. If current_val is greater, update dp with this value and store the index of the previous size in cuts.

After the DP calculation, you will have the maximum cost for the given rod length in dp.

Step 4: Counting Selected Pieces

To determine the count of selected pieces for each cutting size, start from the end of the cuts array and backtrack to find the selected sizes.

Begin with the last rod length and size (sizeIndex).

If the index in cuts at this position is different from the previous rod length, it means this size was selected. Increment the count in selectedCounts for that size.

Move to the previous size (sizeIndex) and repeat the process until you reach the beginning of the rod.

Step 5: Output

Print the maximum cost obtained by cutting the rod.

Print the selected cutting lengths and their quantities based on the selectedCounts array.

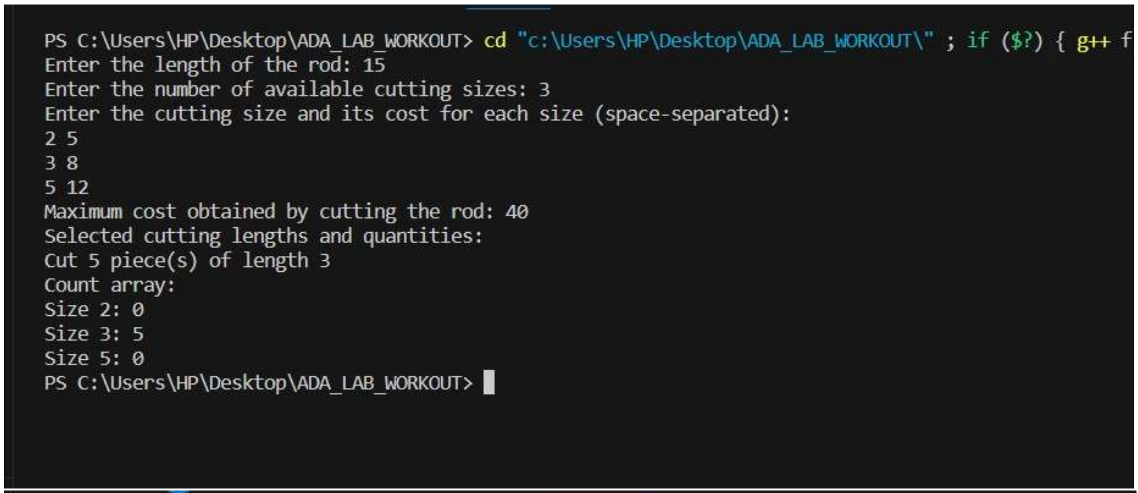
Print the count array, showing the count of selected pieces for each cutting size.

Chapter 3:

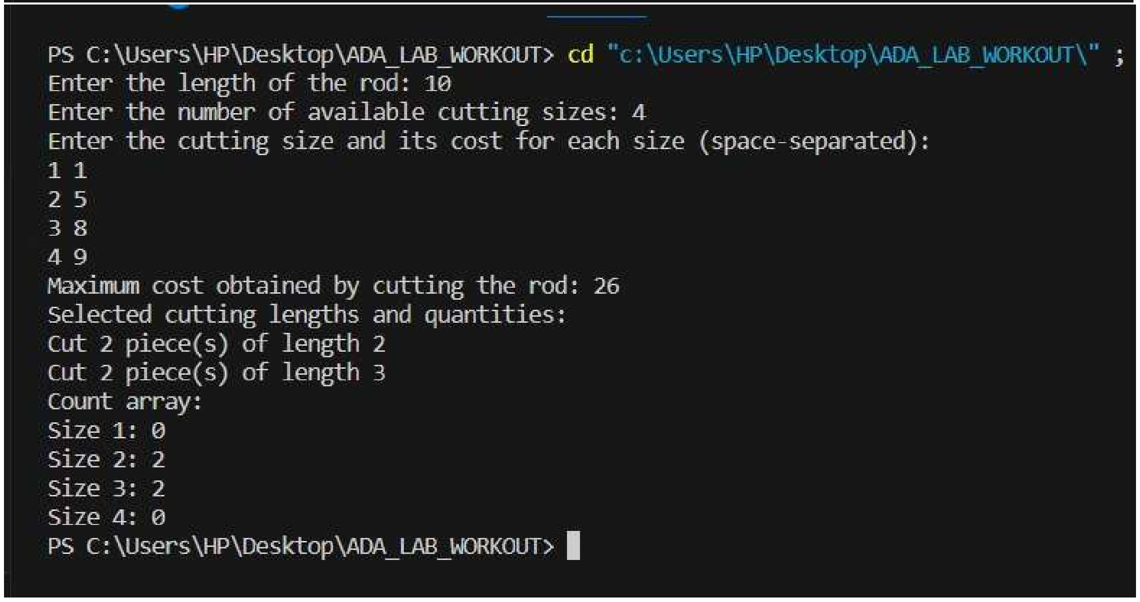
Implementation

```
#include <iostream>
#include <vector>

using namespace std;

struct SizeCost {
    int size;
    int cost;
};

int maxCost(int rodLength, const vector<SizeCost>& sizeCosts, vector<vector<int>>& dp,
vector<vector<int>>& cuts, vector<int>& selectedCounts) {
    for (int i = 1; i <= rodLength; ++i) {
        for (int j = 0; j < sizeCosts.size(); ++j) {
            if (sizeCosts[j].size <= i) {
                int current_val = sizeCosts[j].cost + dp[i - sizeCosts[j].size][j];
                if (current_val > dp[i][j]) {
                    dp[i][j] = current_val;
                    cuts[i][j] = i - sizeCosts[j].size; // Store the index of the previous size
                }
            }
            if (j > 0 && dp[i][j - 1] > dp[i][j]) {
                dp[i][j] = dp[i][j - 1];
                cuts[i][j] = cuts[i - 1][j]; // Copy the index from the previous column
            }
        }
    }

    // Calculate the count array
    int length = rodLength;
    int sizeIndex = sizeCosts.size() - 1;

    while (length > 0 && sizeIndex >= 0) {
        if (cuts[length][sizeIndex] == length - sizeCosts[sizeIndex].size) {
            selectedCounts[sizeIndex]++;
            length -= sizeCosts[sizeIndex].size;
        } else {
            sizeIndex--;
        }
    }

    return dp[rodLength][sizeCosts.size() - 1];
}

int main() {
    int rodLength;
    cout << "Enter the length of the rod: ";
    cin >> rodLength;

    int numSizes;
    cout << "Enter the number of available cutting sizes: ";
    cin >> numSizes;

    vector<SizeCost> sizeCosts(numSizes);
    cout << "Enter the cutting size and its cost for each size (space-separated):\n";
    for (int i = 0; i < numSizes; ++i) {
        cin >> sizeCosts[i].size >> sizeCosts[i].cost;
    }

    vector<vector<int>> dp(rodLength + 1, vector<int>(numSizes, 0));
    vector<vector<int>> cuts(rodLength + 1, vector<int>(numSizes, 0));
    vector<int> selectedCounts(numSizes, 0);

    int maxCostValue = maxCost(rodLength, sizeCosts, dp, cuts, selectedCounts);
    cout << "Maximum cost obtained by cutting the rod: " << maxCostValue << endl;

    cout << "Selected cutting lengths and quantities:\n";
    for (int i = 0; i < numSizes; ++i) {
        if (selectedCounts[i] > 0) {
            cout << "Cut " << selectedCounts[i] << " piece(s) of length " << sizeCosts[i].size <<
endl;
        }
    }

    cout << "Count array:\n";
    for (int i = 0; i < numSizes; ++i) {
        cout << "Size " << sizeCosts[i].size << ": " << selectedCounts[i] << endl;
    }

    return 0;
}
```

Outputs:



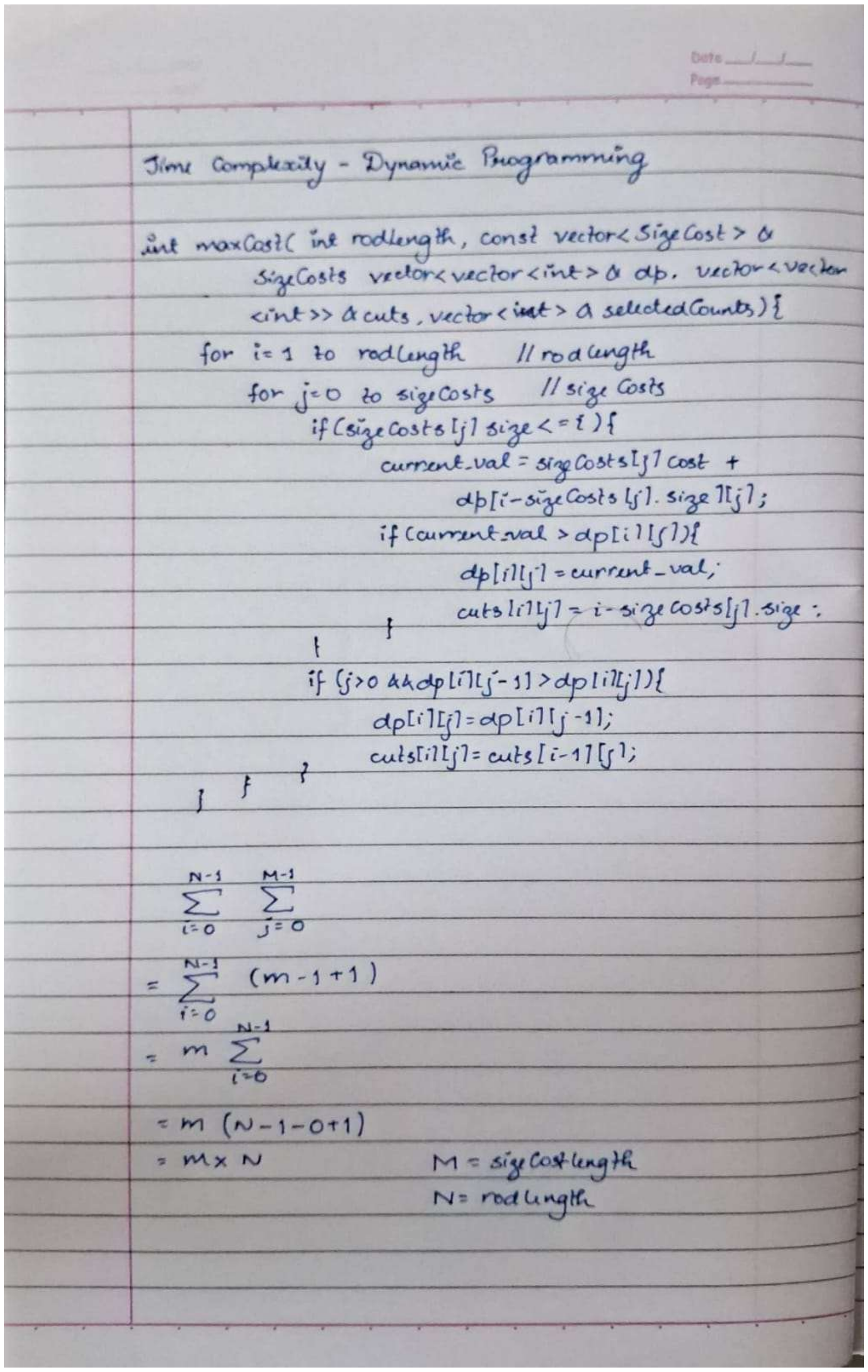
```
PS C:\Users\HP\Desktop\ADA_LAB_WORKOUT> cd "c:\Users\HP\Desktop\ADA_LAB_WORKOUT\" ; if ($?) { g++ f
Enter the length of the rod: 15
Enter the number of available cutting sizes: 3
Enter the cutting size and its cost for each size (space-separated):
2 5
3 8
5 12
Maximum cost obtained by cutting the rod: 40
Selected cutting lengths and quantities:
Cut 5 piece(s) of length 3
Count array:
Size 2: 0
Size 3: 5
Size 5: 0
PS C:\Users\HP\Desktop\ADA_LAB_WORKOUT>
```



```
PS C:\Users\HP\Desktop\ADA_LAB_WORKOUT> cd "c:\Users\HP\Desktop\ADA_LAB_WORKOUT\" ;
Enter the length of the rod: 10
Enter the number of available cutting sizes: 4
Enter the cutting size and its cost for each size (space-separated):
1 1
2 5
3 8
4 9
Maximum cost obtained by cutting the rod: 26
Selected cutting lengths and quantities:
Cut 2 piece(s) of length 2
Cut 2 piece(s) of length 3
Count array:
Size 1: 0
Size 2: 2
Size 3: 2
Size 4: 0
PS C:\Users\HP\Desktop\ADA_LAB_WORKOUT>
```

Chapter 4:

Time Complexity



Time Complexity - Dynamic Programming

```
int maxCost( int rodlength, const vector< SizeCost > &
        SizeCosts vector< vector <int> & dp, vector< vector
        <int>> & cuts, vector< int > & selectedCounts){
    for i= 1 to rodLength       // rodLength
        for j=0 to sizeCosts    // size Costs
            if (sizeCosts[j] size <= i ){
                current_val = sizeCosts[j] cost +
                        dp[i-sizeCosts[j]. size ][j];
                if (current_val > dp[i][j]){
                        dp[i][j] = current_val;
                        cuts[i][j] = i-sizeCosts[j].size ;
                }
            }
            if (j>0 && dp[i][j-1] > dp[i][j]){
                dp[i][j] = dp[i][j-1];
                cuts[i][j] = cuts[i-1][j];
            }
        }
    }
```

$$\sum_{i=0}^{N-1} \sum_{j=0}^{M-1}$$

$$= \sum_{i=0}^{N-1} (m-1+1)$$

$$= m \sum_{i=0}^{N-1}$$

$$= m(N-1-0+1)$$

$$= m \times N$$

M = sizeCostlength

N = rodLength

References:

https://takeuforward.org/data-structure/rod-cutting-problem-dp-24/

https://www.geeksforgeeks.org/cutting-a-rod-dp-13/