A Project Report on

\*\*\*\*\*\*\*ROD CUTTING PROBELM\*\*\*\*\*\*\*\*\*\*

*submitted*

by

**Team Leader (20AT1A0565)**

**Student 1 (20AT1A0578)**

**Student 2 (20AT1A05A4)**

**Student 3 (20AT1A05C2)**

**Under the Guidance of**

**Mrs.K.Lakshmi M.Tech**

**Assistant Professor**

****

**DEPARTMENT OF ELECTRONICS AND COMMUNICATIONENGINEERING**

**G. PULLAIAH COLLEGE OF ENGINEERING AND TECHNOLOGY**

**(Autonomous)**

(Approved by AICTE | NAAC Accreditation with ‘A’ Grade | Accredited by NBA (ECE,CSE & EEE) |

Permanently Affiliated to JNTUA)

**2020-2024**

**G.PULLAIAH COLLEGE OF ENGINEERING AND TECHNOLOGY**

**(Autonomous)**

(Approved by AICTE | NAAC Accreditation with ‘A’ Grade | Accredited by NBA (ECE, CSE & EEE) |

Permanently Affiliated to JNTUA)

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**

****

**CERTIFICATE**

This is to certify that the project report entitled **“ROD CUTTING PROBELM”** being submitted by **Team Leader (20AT1A0565) Student 1 (20AT1A0578) Student 2 (20AT1A05A4) Student 3 (20AT1A05C2)** in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in Computer Science and Engineering of G.Pullaiah College of Engineering and Technology, Kurnool is a record of bonafide work carried out by them under my guidance and supervision.

.

**Mrs K.LAKSKMI M.Tech** **Mrs.M.SrilakshmiM.Tech**

Project Supervisor Head of the Department

INTERNAL EXAMINER EXTERNAL EXAMINER

**ACKNOWLEDGEMENT**

We are extremely grateful to Chairman, **Sri G.V.M.Mohan Kumar,** of G.Pullaiah College of Engineering and Technology, Kurnool, Andhra Pradesh for their good blessings.

We owe indebtedness to our Principal **Dr.C.SrinivasaRao, M.E., Ph.D.**G.Pullaiah College of Engineering and Technology, Kurnool for providing us the required facilities.

We would like to express our deep sense of gratitude and our sincere thanks to HOD **Smt.M.Srilakshmi,M.Tech., Ph.D.** Department of Computer Science and Engineering, G.Pullaiah College of Engineering and Technology, Kurnool for providing the necessary facilities and encouragement towards the project work.

We thank our project supervisor, **Mrs.K.Lakshmi**. for her guidance, valuable suggestions and support in the completion of the project.

We gratefully acknowledge and express our thanks to teaching and non-teaching staff of CSE Department.

We would like to express our love and affection to our parents for their encouragement throughout this project work.

**Project Associates**

**Team Leader (20AT1A0565)**

**Student 1 (20AT1A0578)**

**Student 2 (20AT1A05A4)**

**Student 3 (20AT1A05C2)**

ABSTRACT

The objective of image fusion for medical images is to combine multiple images acquired from different sources into a single image suitable for improved diagnosis. Most of the state-of-the-art image fusing techniques are based on non-fuzzy sets, and the fused image so obtained lags with complementary information. Fuzzy sets are strong-minded to be more appropriate for medical image processing as more hesitations are considered compared with non-fuzzy sets. In this project, a procedure for efficiently fusing multimodal medical images is presented. In the proposed method, images are initially converted into intuitionistic fuzzy images (IFIs), and a new objective function called intuitionistic fuzzy entropy (IFE) is employed to obtain the optimum value of the parameter in the membership and non-membership functions. Next, the IFIs are compared using fitness function, contrast visibility (CV). Then, teaching learning based optimization (TLBO) is introduced to optimize fusion coefficients, which will be changed under teaching phase and learner phase of TLBO, so that the weighted coefficients can be automatically adjusted according to fitness function. Finally, the fused image is obtained using optimal coefficients. Simulations on several pairs of multimodal medical images are performed and compared with the existing fusion methods. The superiority of the proposed method is presented and is justified. Fused image quality is also verified with various quality metrics, such as feature mutual information (FMI), spatial frequency (SF), entropy, edge-based image fusion (QAB/F), modified spatial frequency (MSF) and computation(CT)

**INTRODUCTION**

Rod cutting problem is very much related to any real-world problem we face. You have a rod of some size and you want to cut it into parts and sell in such a way that you get the maximum revenue out of it. Now, here is the catch, prices of different size of pieces are different and it is a possibility that a cutting into smaller pieces can fetch more revenue than selling a bigger piece, so a different strategy is needed.

**DEFINATION**

A naive solution to this problem is to generate all configurations of different pieces and find the highest-priced configuration. This solution is exponential in terms of time complexity. Let us see how this problem possesses both important properties of a Dynamic Programming (DP) Problem and can efficiently be solved using Dynamic Programming

**Applications**

1. The approach explained here can be applicable to many dynamic programming questions directly like the fibonacci series, and indirectly be used to understand other questions like coin change problem, longest common subsequence(LCS) etc.
2. The dynamic programming approach is very useful when it comes to optimization problems like the graph algorithms(All pair shortest path algorithm) that are extensively applied in real-life systems.
3. The others include 0/1 knapsack problem, Mathematical optimization problem, Reliability design problem, Flight control and robotics control, Time sharing: It schedules the job to maximize CPU usage.

**Implementations**

Implementations are available for many [programming languages](https://en.wikipedia.org/wiki/Programming_language).

* For [C++](https://en.wikipedia.org/wiki/C%2B%2B), in the [boost::graph](http://www.boost.org/libs/graph/doc/) library
* For [C#](https://en.wikipedia.org/wiki/C_Sharp_(programming_language)), at [QuickGraph](http://www.codeplex.com/quickgraph)
* For [C#](https://en.wikipedia.org/wiki/C_Sharp_(programming_language)), at [QuickGraphPCL](https://www.nuget.org/packages/QuickGraphPCL/3.6.61114.2) (A fork of QuickGraph with better compatibility with projects using Portable Class Libraries.)
* For [Java](https://en.wikipedia.org/wiki/Java_(programming_language)), in the [Apache Commons Graph](http://commons.apache.org/sandbox/commons-graph/) library
* For [JavaScript](https://en.wikipedia.org/wiki/JavaScript), in the [Cytoscape](https://en.wikipedia.org/wiki/Cytoscape) library
* For [MATLAB](https://en.wikipedia.org/wiki/MATLAB), in the [Matlab\_bgl](http://www.mathworks.com/matlabcentral/fileexchange/10922) package
* For [Perl](https://en.wikipedia.org/wiki/Perl), in the [Graph](https://metacpan.org/module/Graph) module
* For [Python](https://en.wikipedia.org/wiki/Python_(programming_language)), in the [SciPy](https://en.wikipedia.org/wiki/SciPy) library (module [scipy.sparse.csgraph](http://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.csgraph.floyd_warshall.html#scipy.sparse.csgraph.floyd_warshall)) or [NetworkX](https://en.wikipedia.org/wiki/NetworkX) library
* For [R](https://en.wikipedia.org/wiki/R_programming_language), in packages [e1071](https://cran.r-project.org/web/packages/e1071/index.html) and [Rfast](https://cran.r-project.org/web/packages/Rfast/index.html)

## \*\*ROD CUTTING PROBLEM CAN BE SOLVED IN TWO METHODS

## 1:RECURSIVE APPROACH

## 2:DYNAMIC PROGRAMMING APPROACH

## Rod-Cutting Problem

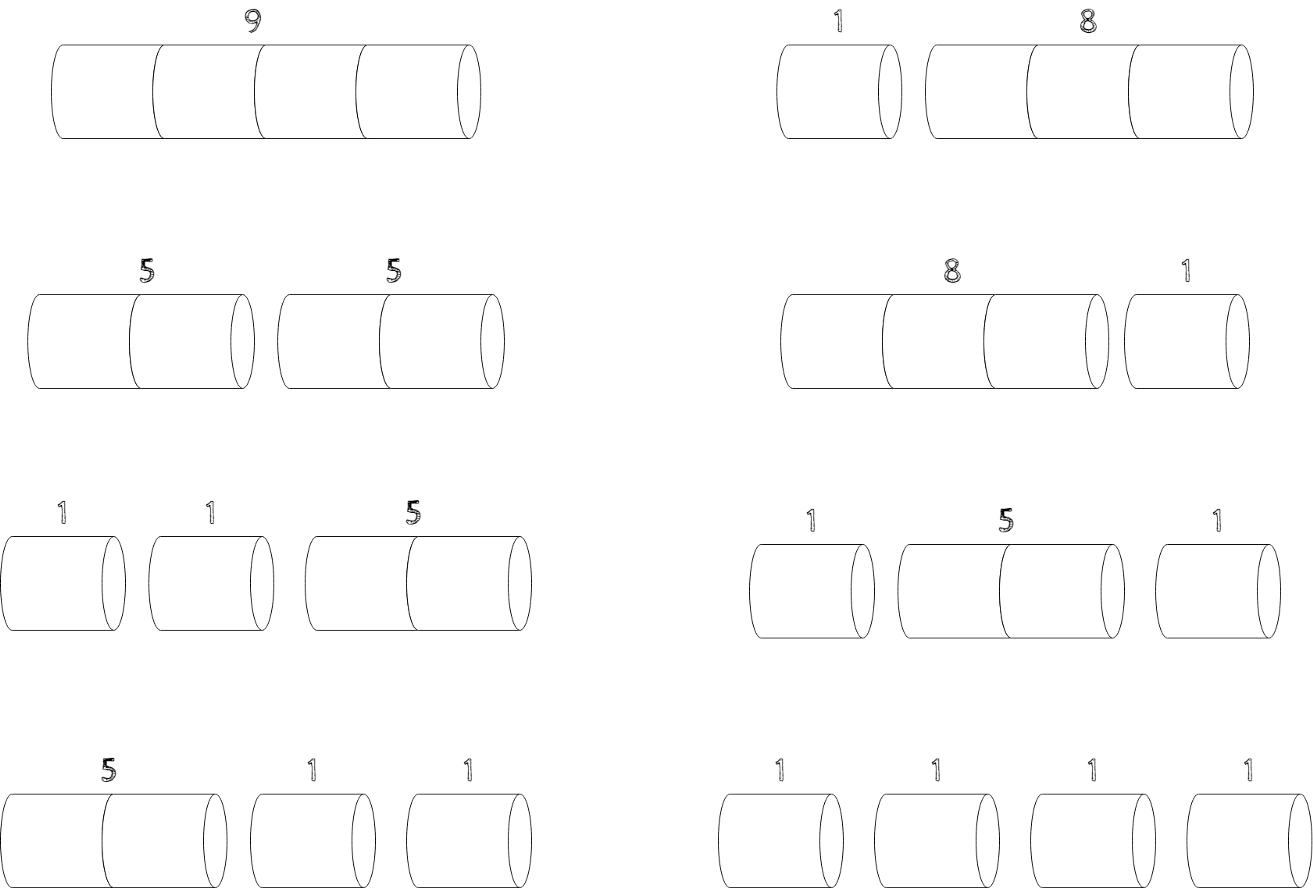
There is a rod of length **n** (whatever the unit; say meters) and there is a price list of rod length **i** meters where **1 ≤ i ≤ n**. We have to find the optimal way of cutting the rod we have (length of **n** meters) to maximize the profit.

Length (**i** meters) 1 2 3 4 5 6 7 8 9 10  
Price (**pᵢ**) 1 5 8 9 10 17 17 20 24 30

For example, let’s see the explanation for the solution first.

The rod length available is **4**m. First, let’s see how we can cut it. How many possibilities are there to cut this rod? 2ⁿ/2 different ways.

That is; if **n**= 4, ²⁴/2 = 16/2 = 8 different ways.

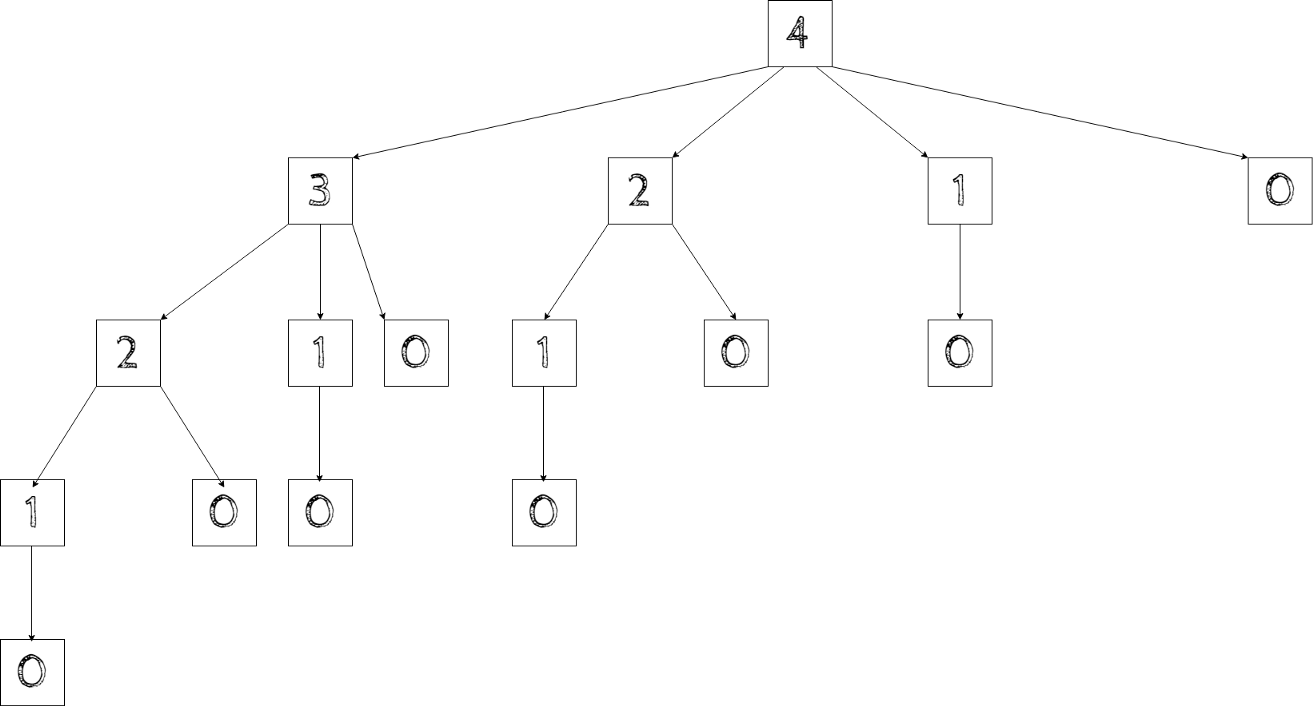


Cut Profit  
4 9  
2, 2 5 + 5 = 10  
1, 3 1 + 8 = 9  
3, 1 8 + 1 = 9  
1, 1, 2 1 + 1 + 5 = 7  
1, 2, 1 1 + 5 + 1 = 7  
2, 1, 1 5 + 1 + 1 = 7  
1, 1, 1, 1 1 + 1 + 1 + 1 = 4

Rod-Cutting Problem — Recursive Approach

We are given an array price[], where the rod of length i has a value of price[i-1]. The idea is simple; one by one, partition the given rod of length n into two parts; i and n-i. Now we have n-i as rod length and have to recur again till it reaches a point where we have 0 as the rod length. Then we have to take max among the computed price. So the recursive function would be

rodCut(n) = max { price[i – 1] + rodCut(n – i) } where 1 <= i <= n



Program for rod cutting problem in recursive method approach

import java.util.Scanner;

public class rodCut {

if (n == 0) {

return 0;

}

int maxValue = Integer.MIN\_VALUE;

for (int i = 1; i <= n; i++) {

int cost = price[i - 1] + rodCut(price, n - i);

if (cost > maxValue) {

maxValue = cost;

}

}

return maxValue;

}

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

int price [] = { 1, 5, 8, 9, 10, 17, 17, 20 };

System.out.print("Give the rod length : ");

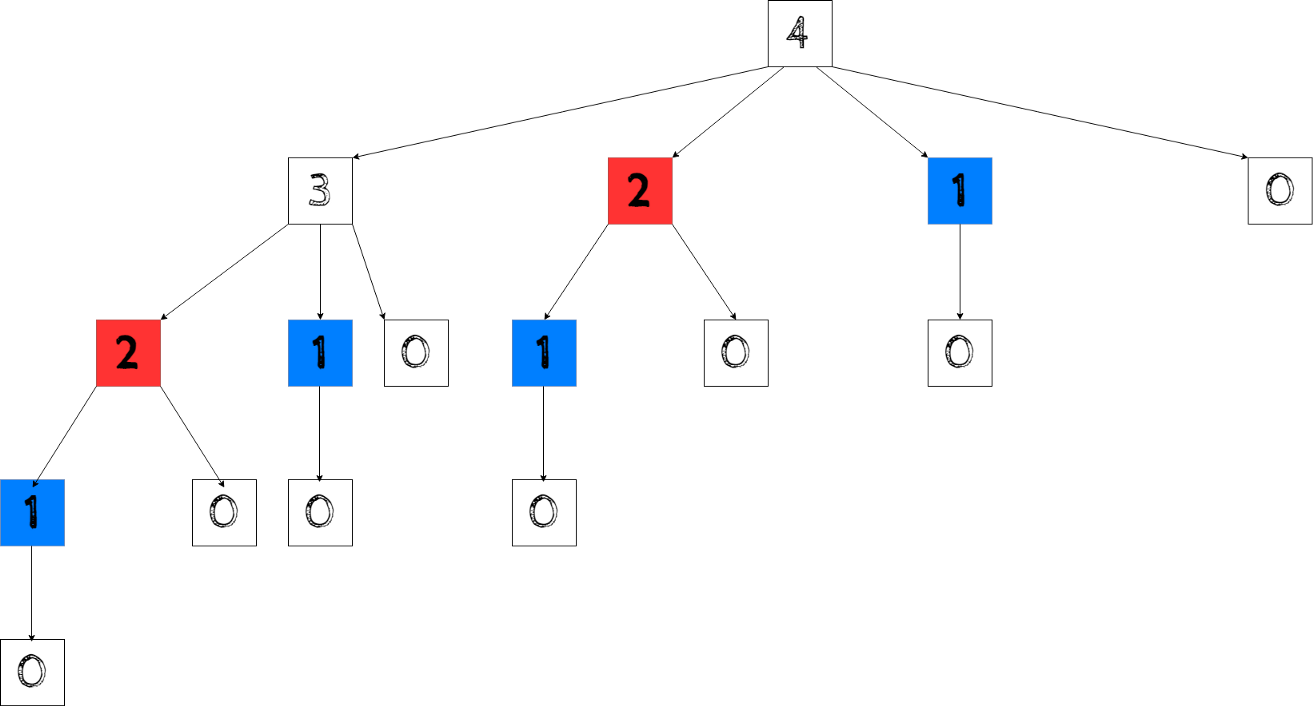
int n = input.nextInt();

System.out.println("Profit is " + rodCut(price, n));

}

}

\*Rod-Cutting Problem — Dynamic Programming Approach



As we can see, the same substructure (highlighted in the same color, red and blue) is getting computed repeatedly. So the problem exhibits overlapping subproblems. We know that problems with optimal substructure and overlapping subproblems can be solved by Dynamic Programming, where subproblem solutions are memoized rather than computing again.

Let’s solve this problem in a bottom-up approach. In the bottom-up approach, we solve similar subproblems first, and then we solve the larger sub-problems from them. The following approach computes T[i], which stores maximum profit obtained from the rod of length i for each 1≤i≤n, and then it uses the values of smaller values i which are already computed.

Cut Profit  
4 9  
2, 2 5 + 5= 10

1, 3 1 + 8= 9  
3, 1 8 + 1= 9

1, 1, 2 1 + 1 + 5 = 7  
1, 2, 1 1 + 5 + 1 = 7  
2, 1, 1 5 + 1 + 1 = 7

1, 1, 1, 1 1 + 1 + 1 + 1 = 4

Program for rod cutting problem in dynamic programming approach

import java.util.Scanner;

public class dynamicRodCutting {

public static int rodCut(int[] price, int n) {

int[] T = new int[n + 1];

for (int i = 1; i <= n; i++){

for (int j = 1; j <= i; j++) {

T[i] = Integer.max(T[i], price[j - 1] + T[i - j]);

}

}

return T[n];

}

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

System.out.print("Give the rod length : ");

int n = input.nextInt();

System.out.print("Profit is " + rodCut(price, n));

}

}

Time Complexity

This bottom-up approach costs O(n²) and required O(n) extra space; where n is the rod size.