Questions and Exercises to work out and turn in:

Grading Guidelines:

* A right answer will get full credit when:

1. It is right (worth 25%)
2. It is right **AND** neatly presented making it easy and pleasant to read. (worth an **extra** 15%)
3. There is an **obvious and clear link** between 1) the information provided in the exercise and in class and 2) the final answer. A clear link is built by properly writing, justifying, and documenting an answer (worth an **extra** 60%).
4. Calculation mistakes will be minimally penalized (2 to 5% of full credit) while errors on units will be more heavily penalized.

You are welcome/encouraged to discuss exercises with other students or the instructor. But, ultimately, **personal** writing is expected.

* USE THIS FILE AS THE STARTING DOCUMENT YOU WILL TURN IN. **DO NOT DELETE ANYTHING FROM THIS FILE:** JUST **INSERT** EACH ANSWER **RIGHT AFTER ITS QUESTION/PROMPT**.
* IF USING HAND WRITING (STRONGLY DISCOURAGED), **USE THIS FILE** BY CREATING SUFFICIENT SPACE AND WRITE IN YOUR ANSWERS.
* FAILING TO FOLLOW TURN IN DIRECTIONS /GUIDELINES WILL COST **A 30% PENALTY.**

Objectives of this assignment:

* to use and manipulate the concepts presented in this module
* to propose and write algorithms in pseudocode
* to analyze the time complexity of algorithms
* to analyze the space complexity of algorithms
* to learn autonomously new concepts

What you need to do:

Answer the questions and/or solve the exercises described below.

Exercise 1 (30 points)

Consider a modification of the rod-cutting problem in which, in addition to a price pi for each rod, each cut incurs a fixed cost of c. The revenue associated with a solution is now the sum of the prices of the pieces minus the costs of making the cuts. Give a dynamic-programming algorithm to solve this modified problem. Precisely comment and explain the modifications/additions you make to the original algorithm to meet the requirements.

Include here the pseudocode

Make sure you comment, justify, and explain the modifications/additions.

Cite your references.

// This is all based on the values represented in figure 15.1.

// The overall this function that will calculate maximum revenue including cutting cost.

Algorithm ModifiedRodCutting(p, n, c)

// Creates an array initialized with nothing that will store "maximum revenue" for each length.

Initialize R[0...n] with zeros

// This will loop at each length defined as j from 1 to n.

for j from 1 to n do

// Initializes a maximum revenue for length j that could never be reached to start.

Set maxRevenue to -Infinity

// Then loop over every possible cut length defined "i" from 1 to j.

for i from 1 to j do

// Calculate revenue based on whether a cut is made.

if j == i then

// When no cut is made, the maxRevenue will be price of rod of length i.

Set revenue to p[i]

else

// After a cut is made, then make sure to include the cost of that cut.

Set revenue to p[i] + R[j - i] - c

// Make sure to then set the maximum revenue for length j.

if revenue > maxRevenue then

Set maxRevenue to revenue

// // Store the new maxRevenue that was found for rod of length j in the array.

Set R[j] to maxRevenue

// This simply returns the maximum revenue for a rod of given length n.

return R[n]

Exercise 2 (70 points)

1. (10 points if correct) **Implement** in your preferred language MEMOIZED-CUT-ROD. Insure your program can be compiled and executed on an Engineering Unix Tux machine.

Turn in this program separately on Canvas with your homework

I have attached the java file to this assignment. The name of the file was MemoizedCutRod.java.

1. (20 points) Test your implementation that it yields the same results as in the textbook. State here whether your implementation yields the same results (max revenue) as in the textbook.

Insert **here** screenshots to show that your program yields the same results.

The screenshots must include the Tux machine name, your Auburn username and the date. For the date, type the command "date" just before executing your program. Your screenshot(s) must be as readable as this template screenshot:



A screenshot of a computer

Description automatically generated

Above is the screenshot for the first program that was implemented. My results did match what the book detailed as apart of the first figure in 15.1. I included extra “toString” to show all the test data used.

1. (35 points) Modify your MEMOIZED-CUT-ROD implementation (from Question 1) to return not only the value but the actual solution (how to cut), too.

Include/Insert here the pseudocode of your program

Turn in the modified implementation separately on Canvas.

Insert **here** screenshots to show that your program yields correct results for the example on the textbook.

The screenshots must include the Tux machine name, your Auburn username and the date. For the date, type the command "date" just before executing your program.

// The overall UPDATED function that will calculate maximum revenue that also includes cutting cost and tge return the solution

Algorithm UpdatedMemoizedCutRod(priceArray, rodLength)

// Create arrays to store the maximum revenue for each length and the cuts

Initialize revenueArray[0...rodLength] with -Infinity

Initialize solutionArray[0...rodLength] with 0

// Calculate the maximum revenue and fill the solution array

maxRevenue <- MemoizedCutRodAux(priceArray, rodLength, revenueArray, solutionArray)

// Construct the solution by tracking the cuts

cuts <- empty list

length <- rodLength

while length > 0

cuts.append(solutionArray[length])

length <- length - solutionArray[length]

// Return both the maximum revenue and the cuts

return (maxRevenue, cuts)

// Aux function to calculate the maximum revenue and takes note of cuts.

Function MemoizedCutRodAux(priceArray, rodLength, revenueArray, solutionArray)

// If the revenue for length rodLength is already calculated, return it

if revenueArray[rodLength] >= 0

return revenueArray[rodLength]

// Initialize the maximum revenue

maxRevenue <- 0

if rodLength == 0

maxRevenue <- 0

else

maxRevenue <- -Infinity

// Loop over each possible cut length i

for i from 1 to rodLength

currentRevenue <- priceArray[i] + MemoizedCutRodAux(priceArray, rodLength - i, revenueArray, solutionArray)

if currentRevenue > maxRevenue

maxRevenue <- currentRevenue

solutionArray[rodLength] <- i

// Store the calculated maximum revenue

revenueArray[rodLength] <- maxRevenue

return maxRevenue

The screenshot below shows the updated version of the algorithm and solution.

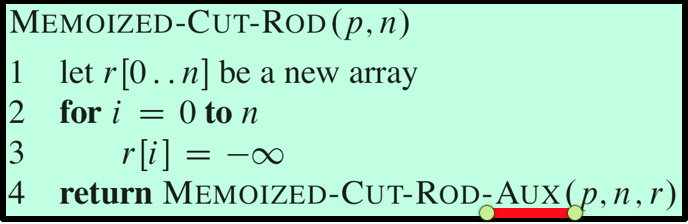
A screenshot of a computer

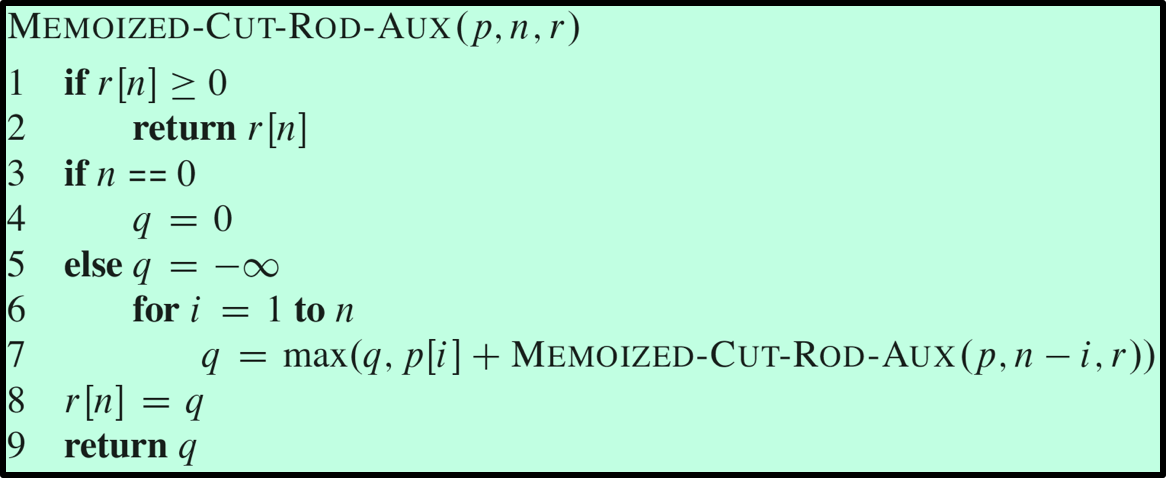
Description automatically generated

1. (5 points) In addition to the pseudocode in THIS file, turn in the **source code** of your implementation. Include a small report stating 1) whether your code works, 2) how to compile and execute your program on a Tux machine.

include here how to compile and execute your two programs on Tux machines

There are two files I included along with this word document that includes both my screenshots and pseudocode. The first file is MemoizedCutRod.java. The second file is named UpdatedMemoizedCutRod.java that includes the updated version that also returns the valid solution as described. The console return “toString” should simply return and display the results that match the values used in figure 15.1 from the book. My results did in fact match the book.





What you need to turn in:

* Electronic copy of this file (including your answers) (standalone). Submit the file as a Microsoft Word or PDF file.
* Source code of required program(s) with directions about how to compile and execute them.
* Recall that answers must be well written, documented, justified, and presented to get full credit.
* How this assignment will be graded:
* A right answer will get full credit when:
* It is right (worth 25%)
* It is right AND neatly presented making it easy and pleasant to read. (worth 15%)
* There is an obvious and clear link between 1) the information provided in the exercise and in class and 2) the final answer. A clear link is built by properly writing, justifying, and documenting an answer (worth 60%).
* Calculation mistakes will be minimally penalized (2 to 5% of full credit) while errors on units will be more heavily penalized.
* You are welcome/encouraged to discuss exercises with other students or the instructor. But, ultimately, personal writing is expected.

**Appendix**: Grading: What is an OBVIOUS and CLEAR LINK?

Here is an example to explain what an **obvious and clear link** is and how we grade your work.

Consider the following problem:

"(100 points) John travels from Auburn to Atlanta in his car at a speed of 60 mph. Leaving at 8am, at what time will John reach Atlanta".

Here are the answers of three students and their scores:

* **Student 1** answers: "9:48am". Student 1 will get 25 points.
* **Student 2**answers: "John will reach Atlanta at 9:48am". Student 2 will get 25+15 = 40 points
* **Student 3** answers: "The time t to travel a distance d at speed v is equal to d/v = d/60mph. The problem does not provide the distance d from Auburn to Atlanta. Based on GoogleMaps, the distance from Auburn to Atlanta is approximately 108 miles (**document is attached**).



Therefore, the time t = 108 miles/60mph \* 60 minutes/hour= 108 minutes. Since John left at 8am, he will then reach Atlanta at 8am + 108 minutes = 8 am + 60 minutes + 48 minutes = 9:48".

**Student 3** will get 25 + 15 + 60 = 100 points

Do you see the **direct** **link** going from the data provided in the question to the final answer, using general knowledge/formula and documents?.... Can you now solve the following problem and get 100 points?

"(100 points) Alice travels from Auburn to Atlanta in her car at a speed of 60 mph. Leaving at 8am, at what time will Alice reach Atlanta assuming that she had a flat tire that delayed her 30 minutes".