Analysis of Algorithms
Lab 06: Analyzing Catalan Numbers – Time Complexity of Recursive and DP Solutions

Andrés Oswaldo Calderón Romero, Ph.D.

October 2, 2025

1 Introduction

In this lab, we will explore the concept of **Catalan numbers**, a fundamental sequence in combinatorial mathematics, and analyze different approaches to computing them. Catalan numbers frequently appear in various counting problems, such as parenthesis matching, binary search tree formations, and non-intersecting chord arrangements.

To reinforce the concept of **dynamic programming (DP)**, we will begin by reviewing its core principles through an external video resource. Then, we will examine two approaches to computing Catalan numbers: a naive recursive solution and an optimized dynamic programming approach. Through this comparison, you will gain insights into the significance of optimizing recursive algorithms to improve efficiency.

By the end of this lab, you should be able to:

- Understand the recurrence relation that defines Catalan numbers.
- Implement a recursive solution to compute Catalan numbers.
- Analyze the inefficiencies of recursion and apply dynamic programming to optimize the computation.
- Determine the time complexity of both approaches.
- Compile your findings into a structured report.

Take your time to work through the exercises, collaborate with your group, and think critically about the problem-solving strategies. Time to dive in!

2 Reinforcing DP Notions

The first part of this lab involves watching a YouTube video[1] that covers the fundamental concepts of dynamic programming along with a few illustrative examples. One of these examples, the n-th Fibonacci number, has already been discussed in class. You can watch the video here.

Remember that you can enable captions and translate the subtitles into Spanish if needed. There is no deliverable for this part—just watch and enjoy the video.

3 Catalan Numbers

The Catalan numbers form a mathematical sequence of positive integers used to determine the number of ways various combinations can be arranged. The n-th term in the sequence, denoted as C_n, is given by the formula:

$$C_n = \frac{(2n)!}{(n+1)!n!}$$

The initial values of the Catalan sequence for $n = 0, 1, 2, 3, 4, 5, \ldots$ are:

$$1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, \ldots$$

Catalan numbers arise in numerous combinatorial counting problems, including:

- Determining the number of valid ways to arrange n pairs of parentheses so that they are correctly matched.
- Counting the number of unique Binary Search Trees (BSTs) that can be formed using n keys.
- Finding the number of full binary trees, where each node has either two children or none, with $n + 1$ leaves.
- Calculating the number of ways to draw n non-intersecting chords in a circle with $2n$ points.

These numbers have many more applications in combinatorics and discrete mathematics. You can find more information about Catalan number applications here.

[1] Tech With Nicola YouTube Channel, "Mastering Dynamic Programming - How to Solve Any Interview Problem (Part 1), 2024."

3.1 Naïve Approach Using Recursion

Catalan numbers satisfy the following recursive formula:

$$C = \begin{cases} C_0 & = 1 \text{ for } n = 0 \\ C_1 & = 1 \text{ for } n = 1 \\ C_n & = \sum_{i-0}^{n-1} C_i C_{n-i-1} \text{ for } n > 2 \end{cases}$$

Algorithm 1 presents the pseudocode for computing Catalan numbers using the recursive formula.

Algorithm 1 Recursive Catalan Number Computation

```
1: function findCatalan(n)
2:     if n ≤ 1 then
3:         return 1
4:     end if
5:     res ← 0
6:     for i ← 0 to n − 1 do
7:         res ← res + (findCatalan(i) × findCatalan(n - i - 1))
8:     end for
9:     return res
10: end function
```

Exercises

1. State the recurrence relation for Algorithm 1.
2. Determine the time complexity of Algorithm 1.

3.2 Optimized Approach Using Dynamic Programming

The recursive implementation repeatedly computes the same subproblems, resulting in redundant calculations. Since the problem exhibits overlapping subproblems, dynamic programming can be applied to enhance efficiency by storing previously computed values.

Step-by-Step Approach

1. Declare an array `catalan[]` to store the Catalan numbers.
2. Initialize the base cases: `catalan[0] = 1` and `catalan[1] = 1`.
3. Iterate from $i = 2$ to the given Catalan number n.
4. For each i, iterate j from 0 to $i - 1$, updating `catalan[i]` by summing the product of `catalan[j]` and `catalan[i - j - 1]`.

5. After completing the iterations, return `catalan[n]`.

Exercises

1. Write an algorithm to compute Catalan numbers using Dynamic Programming.
2. State the recurrence relation for this algorithm.
3. Determine the time complexity of the dynamic programming approach.

4 What Do We Expect?

You will apply what you have learned from the previous lab. The solution for finding Catalan numbers is readily available on the Internet or can be obtained by asking an AI. However, take your time to work through the problem independently and with your group before considering alternative sources. Only if you feel completely lost or want to review your responses should you consult those resources.

Always think critically, and most importantly, take enough time to fully understand the information you find or receive.

Compile all answers into a well-structured report and submit it in **PDF** format via the corresponding Brightspace assignment before the next lab on **October 16, 2025**.

Happy Hacking 😎!