# California State University, Fresno

# DEPARTMENT OF COMPUTER SCIENCE

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Class: | **Algorithms & Data Structures** | | | Semester: | **Fall 2021** |
|  | | | | | |
| Points |  | Document author: | **Marlon Branham** | | |
|  | Author’s email: | **zekebranham@mail.fresnostate.edu** | | |
| Laboratory number: | **Laboratory 2** | | |
|  | | | | | |

**1. Statement of Objectives**

In this lab, we were to create a program that uses a dynamic programming approach to implement functions that parenthesize matrices for matrix chain order multiplication. The purpose of this lab is to gain an understanding of not necessarily just the matrix chain order multiplication but to explore dynamic programming approaches and implement them. In this report, I will be going over my experimental procedures, an analysis of results, encountered problems, and a conclusion of my findings.

**2. Experimental Procedure**

In this lab, we were to implement a parenthesizing function and a matrix chain order function. We use the parenthesizing function to find the optimal way to parenthesize the matrices before multiplication. The matrix chain order function is to calculate the amount of scalar multiplication necessary for each parenthetic approach. My first function is named ‘print\_opt\_parens’ that takes in a double pointer of an array ‘s’ (the 2D array/matrix), an integer i for iterating, an integer j for again iterating, and a reference to a character for the name of each matrix. This is a recursive function that parentheses the matrices.

A computer screen with white and blue text

Description automatically generated

My second primary function is my ‘matrix\_chain-order’ function that takes in an array p[ ], the size of the array, a double pointer to a 2D array, and a double pointer to another 2D array. This function does the matrix chain order operations for scalar multiplication iteratively. In order to use arrays throughout the program rather than the easier implementation of vectors, I needed to add two other functions for allocating the arrays and for deallocating them since I needed to dynamically allocate the memory for the arrays before runtime.

A computer screen with text

Description automatically generated

**3. Analysis**

My results came out to be as expected. For my example input, I used an array p[ ] with the size of 5 to hold the values {10,20,30,40,20}. My results show to be: (((AB)C)D) with an optimal cost of 26000 scalar multiplications. As far as the time complexities of the functions, the ‘matric\_chain\_order’ function is a triple-nested loop, making it have a very slow complexity of O(n^3). The ‘print\_opt\_parens’ function uses recursive calls, and in its worst case is called once for each matrix making the true worst-case to be O(4n), but in terms of time complexity, it is a linear O(n) time.

****

**4. Encountered Problems**

The only issue encountered for the program was using arrays for the entirety of the program. Since arrays must be initialized prior to runtime, I had to find methods to dynamically allocate the memory and deallocate the memory to prevent memory leaks. This was a bit challenging given that the arrays are 2D using pointers and references, so I created two separate functions to allocate and deallocate the memory. Other than this, no issues in the programming aspect.

**5. Conclusions**

In this lab, I learned more about dynamic programming and how it can be used to improve/optimize your code. Using these two functions shows how useful dynamic programming is to a program. I was able to successfully allocate 2D arrays, parenthesize matrix chain order multiplication, and calculate the cost of the choice of parenthesizing.

**6. References**

<https://www.geeksforgeeks.org/matrix-chain-multiplication-dp-8/>

<https://www.geeksforgeeks.org/how-to-declare-a-2d-array-dynamically-in-c-using-new-operator/>

<https://favtutor.com/blogs/dynamic-2d-array-cpp>