**CMPSC122 Lab Section (004L)**

Lab 12

Maximum Contiguous Subsequence Sum Problem: Algorithm Analysis and Performance Evaluation

Prepared By

(Mohammad Abu Shariha)

**Instructor: Ziyun Huang**

**Date: (04/25/2024)**

**Table of Contents**

[1 Objective (or Abstract) 3](#_Toc26434614)

[2 Introduction 3](#_Toc26434615)

[3 Procedure 3](#_Toc26434616)

[4 Discussion 4](#_Toc26434617)

[5 Conclusion 6](#_Toc26434618)

[6 Appendix (Optional) 7](#_Toc26434619)

# 1 Objective (or Abstract)

The objective of this project is to analyze three algorithms for solving the maximum contiguous subsequence sum problem, evaluate their theoretical performance, and compare it with their actual performance through testing.

# 2 Introduction

The assignment focuses on analyzing three algorithms designed to solve the maximum contiguous subsequence sum problem and evaluating their performance. The report will discuss the theoretical analysis of each algorithm, compare it with the actual running times obtained from testing, and draw conclusions based on the results.

The maximum contiguous subsequence sum problem involves finding the contiguous subsequence within a sequence of integers that has the maximum sum. For example, in the sequence [-2, 11, -4, 13, -5, 2], the maximum contiguous subsequence sum is 20, corresponding to the subsequence [11, -4, 13]. Three algorithms have been developed to solve this problem:

* Brute force algorithm: Examines all possible contiguous subsequences and computes their sums to find the maximum.
* Divide-and-conquer algorithm: Recursively divides the sequence in half, computes the maximum sum in each half and the maximum sum crossing the middle, and returns the maximum of the three.
* Kadane's algorithm: Maintains two variables, max\_ending\_here and max\_so\_far, to keep track of the maximum sum ending at the current index and the overall maximum sum found so far, respectively.

Another example includes: Sequence: [ -1, -2, -3, -4 ]. The Maximum contiguous sequence is: [ -1, -2, -3, -4 ]. The Maximum sum is: 0. Every integer in this sequence is negative. The set criteria within the report states that if the sequence contains all negative integers, the maximum contiguous subsequence sum is defined to be 0.

# 3 Procedure

The steps involved in the experiment are as follows:

* Generate random arrays of different sizes and longer loops.
* Apply each algorithm to find the maximum contiguous subsequence sum for each array.
* Measure the execution time for each algorithm.
* Output the execution times for comparison.

Computer Specifications:

* Operating System: [macOS]
* Processor: [Apple M2]
* Memory: [8 GB]

# 4 Discussion

1. **Brute Force Algorithm (MaxSublistSum\_Blue):**

**Running Time (Big-O):** O(n^2)

**Reasoning:** This algorithm uses two nested loops to iterate over all possible subarrays. The outer loop runs from 0 to n-1, and the inner loop runs from 0 to n-i-1, where n is the size of the array. Therefore, the total number of iterations is proportional to n\*(n-1)/2, resulting in a quadratic time complexity.

A graph with a line going up

Description automatically generated

The graph above is run with 9 loops and shows the running time (s) vs the array size.

1. **Divide and Conquer Algorithm (MaxSublistSum\_Green):**

**Running Time (Big-O):** O(n log n)

**Reasoning:** This algorithm divides the array into two halves recursively and then finds the maximum subarray crossing the midpoint. The recurrence relation for the time complexity is T(n) = 2T(n/2) + O(n), where the O(n) term is for finding the maximum subarray crossing the midpoint. By solving this recurrence relation, we get O(n log n) as the time complexity.

A graph with a line going up

Description automatically generated

The graph above is run with 9 loops

1. **Linear Scan Algorithm (MaxSublistSum\_Red):**

**Running Time (Big-O):** O(n)

**Reasoning:** This algorithm scans the array once, maintaining the maximum sum ending at each position. It calculates the maximum sum ending at position i based on the maximum sum ending at position i-1. Since it only iterates through the array once, the time complexity is linear, O(n).

A graph with a line pointing up

Description automatically generated

The graph above is run with 14 loops and an array size of 700

Here is the results of all 3 tests next to each other O() being blue, O() being green and O() being red:

A graph with a red line

Description automatically generated

As we can see from the graph the experimental values clearly align with the theoretical values.

# 5 Conclusion

The Blue function corresponds to Algorithm #1, the Green function to Algorithm #2, and the Red function to Algorithm #3 based on the analysis. Theoretical analysis aligned well with the actual running times observed during testing. Conclusions drawn from the results indicate the effectiveness of Algorithm #3 (Red function) in terms of performance compared to the other two algorithms. Algorithm #3 (Linear Scan) demonstrated superior performance compared to the other two algorithms. Overall, the experimental results confirmed the theoretical expectations regarding the time complexity of each algorithm.

This report can be improved by running more intensive tests to obtain more data while maintaining all controlled variables.

# 6 Appendix (Optional)

Please enter the number of loops: 9

============================ Algorithm Blue ============================

Preparation Time: 0.0010s

[ 0] Maximum contiguous subsequence sum (array size = 500): 781

Elapsed Time: 0.0010s

------------------------------------------------------------------------

Preparation Time: 0.0010s

[ 1] Maximum contiguous subsequence sum (array size = 1000): 1816

Elapsed Time: 0.0010s

------------------------------------------------------------------------

Preparation Time: 0.0012s

[ 2] Maximum contiguous subsequence sum (array size = 2000): 1816

Elapsed Time: 0.0023s

------------------------------------------------------------------------

Preparation Time: 0.0038s

[ 3] Maximum contiguous subsequence sum (array size = 4000): 2034

Elapsed Time: 0.0071s

------------------------------------------------------------------------

Preparation Time: 0.0112s

[ 4] Maximum contiguous subsequence sum (array size = 8000): 2424

Elapsed Time: 0.0297s

------------------------------------------------------------------------

Preparation Time: 0.0415s

[ 5] Maximum contiguous subsequence sum (array size = 16000): 2681

Elapsed Time: 0.1131s

------------------------------------------------------------------------

Preparation Time: 0.1556s

[ 6] Maximum contiguous subsequence sum (array size = 32000): 3423

Elapsed Time: 0.4607s

------------------------------------------------------------------------

Preparation Time: 0.6182s

[ 7] Maximum contiguous subsequence sum (array size = 64000): 13044

Elapsed Time: 1.9054s

------------------------------------------------------------------------

Preparation Time: 2.5274s

[ 8] Maximum contiguous subsequence sum (array size = 128000): 20028

Elapsed Time: 8.0977s

Please enter the number of loops: 9

============================ Algorithm Green ============================

Preparation Time: 0.0010s

[ 0] Maximum contiguous subsequence sum (array size = 500): 1278

Elapsed Time: 0.0010s

------------------------------------------------------------------------

Preparation Time: 0.0010s

[ 1] Maximum contiguous subsequence sum (array size = 1000): 1777

Elapsed Time: 0.0010s

------------------------------------------------------------------------

Preparation Time: 0.0010s

[ 2] Maximum contiguous subsequence sum (array size = 2000): 1777

Elapsed Time: 0.0010s

------------------------------------------------------------------------

Preparation Time: 0.0010s

[ 3] Maximum contiguous subsequence sum (array size = 4000): 3920

Elapsed Time: 0.0010s

------------------------------------------------------------------------

Preparation Time: 0.0010s

[ 4] Maximum contiguous subsequence sum (array size = 8000): 4053

Elapsed Time: 0.0010s

------------------------------------------------------------------------

Preparation Time: 0.0019s

[ 5] Maximum contiguous subsequence sum (array size = 16000): 4053

Elapsed Time: 0.0010s

------------------------------------------------------------------------

Preparation Time: 0.0036s

[ 6] Maximum contiguous subsequence sum (array size = 32000): 5831

Elapsed Time: 0.0010s

------------------------------------------------------------------------

Preparation Time: 0.0069s

[ 7] Maximum contiguous subsequence sum (array size = 64000): 11737

Elapsed Time: 0.0020s

------------------------------------------------------------------------

Preparation Time: 0.0129s

[ 8] Maximum contiguous subsequence sum (array size = 128000): 11737

Elapsed Time: 0.0038s

Please enter the number of loops: 9

============================ Algorithm Red ============================

Preparation Time: 0.0010s

[ 0] Maximum contiguous subsequence sum (array size = 500): 747

Elapsed Time: 0.0010s

------------------------------------------------------------------------

Preparation Time: 0.0010s

[ 1] Maximum contiguous subsequence sum (array size = 1000): 790

Elapsed Time: 0.0010s

------------------------------------------------------------------------

Preparation Time: 0.0010s

[ 2] Maximum contiguous subsequence sum (array size = 2000): 824

Elapsed Time: 0.0010s

------------------------------------------------------------------------

Preparation Time: 0.0010s

[ 3] Maximum contiguous subsequence sum (array size = 4000): 3367

Elapsed Time: 0.0010s

------------------------------------------------------------------------

Preparation Time: 0.0010s

[ 4] Maximum contiguous subsequence sum (array size = 8000): 3971

Elapsed Time: 0.0010s

------------------------------------------------------------------------

Preparation Time: 0.0013s

[ 5] Maximum contiguous subsequence sum (array size = 16000): 4825

Elapsed Time: 0.0010s

------------------------------------------------------------------------

Preparation Time: 0.0025s

[ 6] Maximum contiguous subsequence sum (array size = 32000): 6315

Elapsed Time: 0.0010s

------------------------------------------------------------------------

Preparation Time: 0.0043s

[ 7] Maximum contiguous subsequence sum (array size = 64000): 6315

Elapsed Time: 0.0010s

------------------------------------------------------------------------

Preparation Time: 0.0078s

[ 8] Maximum contiguous subsequence sum (array size = 128000): 7676

Elapsed Time: 0.0010s

int MaxSublistSum\_Blue(int array[], int size) {

int currentMax = 0;

for (int i = 0; i < size; i++) {

int curSum = 0;

for (int j = 0; j < size - i; j++) {

curSum += array[i+j];

if (curSum > currentMax) {

currentMax = curSum;

}

}

}

return currentMax;

}