Aditi Jorapur
CS 146 Section 4
Potika
10/23/22

Project 2 Code Report

Introduction: The purpose of this project was to solve the maximum subarray problem using different techniques. The three different algorithms I implemented in this project were the brute force method, the divide and conquer method, and Kadane's Algorithm. Working on this project and having to implement solving the same problem in three different ways and three different runtimes, helped me better understand how to find the maximum subarray. It also helped me to see the different ways the code was implemented based on the runtimes of the three algorithms. After implementing these three algorithms, I also created three different methods to calculate the runtimes of the algorithms and in order to test the algorithms, I made JUnit tests. We were given a file that we had to use to test and in order to do that, I created a class that read the file.

Maximum Subarray Class: This class contains three different methods to compute the maximum subarray of a given array. It aslo contains three methods to calculate the runtime of the three algorithms that calculate the Maximum Subarray.

public static int[] bruteForceMaxSubarray(int[] array): *Big O Notation- O(n^2)*
The Brute Force Algorithm for finding the Maximum Subarray has a O(n^2) runtime. First, it creates a variable called maxSum that is equal to the smallest value. It also creates arrive and depart variables which indicate the start and end of the max subarray respectfully. It creates a new array called ans which will return the maxsum, arrive, and depart variables. Then it traverses through the array and initializes the current sum of the array to 0. It will traverse through the array again with another pointer that starts at the index of i. It checks to see if the currentSum is greater than the max sum and if it is, the currentSum is the new maxSum and it sets the values for arrive and departure. Checks to see if the sum is not negative and if it is, depart and maxSum are set to -1 and 0. This method then returns the array of the maxSum, arrive, and depart of the max sub-array of the array.

public static int[] maxCrossSubarray(int[] array, int low, int mid, int high): *Big O Notation- O(n)*
The maxCrossSubarray is a helper method for the divide and conquer algorithm when calculating the maximum subarray. This method calculates the max of the left and right ends of an array by using the temporary sum and updating the final sum or the two sides. It returns the answer as an array that contains the maxLeft value, maxRight value, and the sum of the left and right. It is used to find the max sum of the left and right arrays if the max sum happens to be when they cross.

public static int[] divdeAndConquerMaxSubarray(int[] array, int low, int high): *Big O Notation- O(nlgn)*

The divide and conquer algorithm is a recursive algorithm that calculates the maximum subarray and takes in three parameters. It first checks if the max is in the left side of the array, the right side, or the middle (cross). When only one element is being compared, the array is returned with the starting and ending values and the maxSum. It then calculates the leftArray, crossArray, and rightArray and compares all three. Finally, it returns the maxSum, the highest and lowest in the array.

public static int[] kadaneMaxSubarray(int array[]): *Big O Notation-* O(n)
Kadane's algorithm traverses through the array and calculates the maximum temporary sum. If the temp sum is less than 0 then the starting value is moved to the next index because it cannot be less than 0. If the max Sum is less than the temp sum at any point then the new max sum is the temp sum and the other values are updated.In the end, the arrive value is finalized and an array is created with the three values: maxSum, arrive and depart indices and is returned.

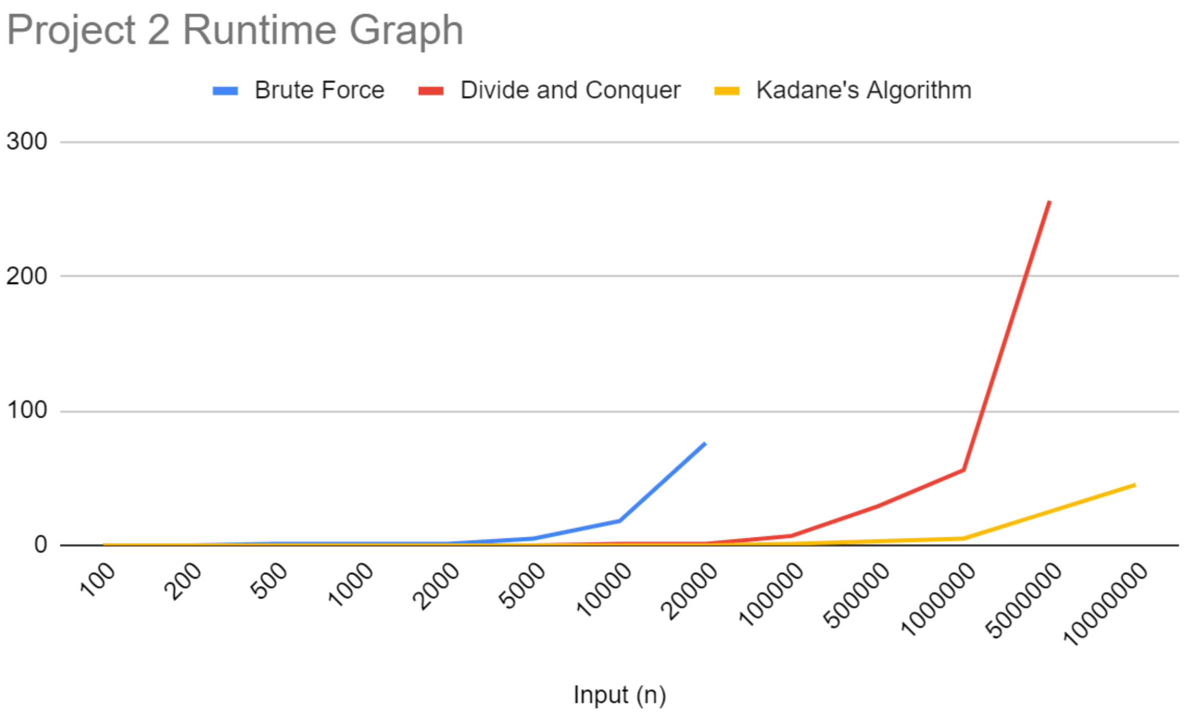
public void runtimeBruteForce(int num)
This method calculates the average runtime for the brute force maximum subarray algorithm. It creates an integer array with random values and uses those values to calculate the runtime of brute force in milliseconds.

public void runtimeDivideAndConquer(int num)
This method calculates the average runtime for the divide and conquer maximum subarray algorithm. It creates an integer array with random values and uses those values to calculate the runtime of divide and conquer in milliseconds.

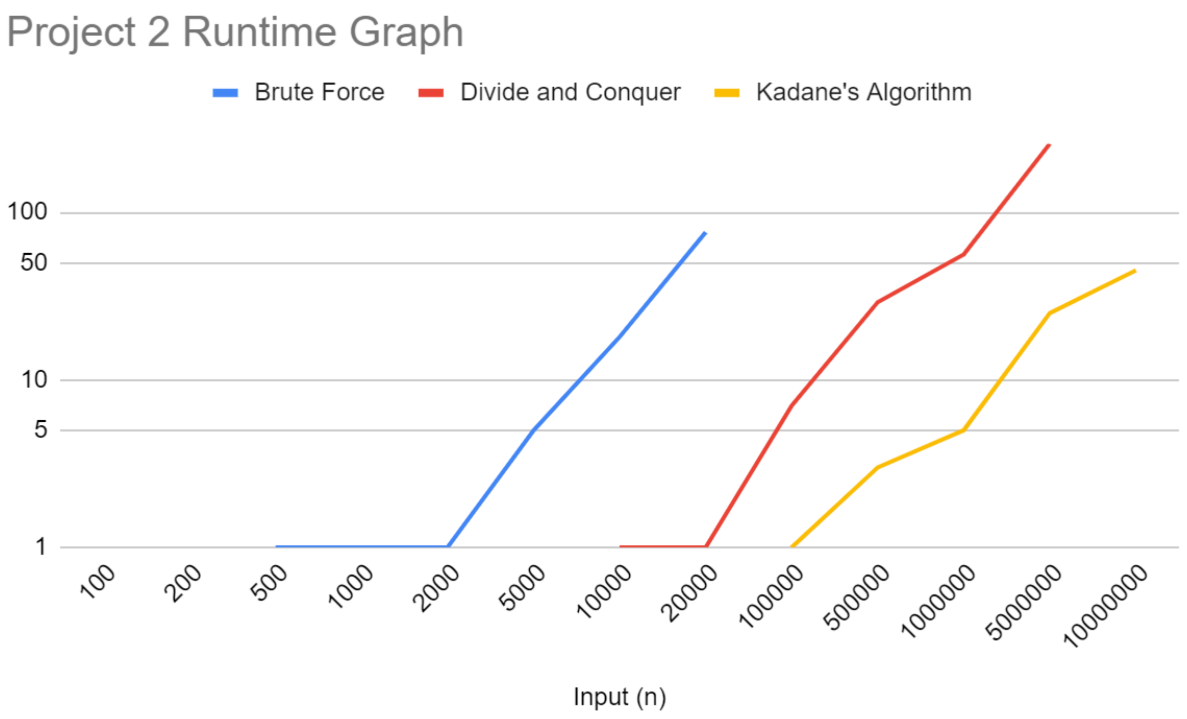
public void runtimeKadane(int num)
This method calculates the average runtime for Kadane's maximum subarray algorithm. It creates an integer array with random values and uses those values to calculate the runtime of Kadane's algorithm in milliseconds.

MaxSum FileReader Class:
This class aids in reading the maxSumtest.txt file that contains the test cases for the three Maximum Subarray algorithms.

public static ArrayList<String> processInput(String fileName) throws IOException, NumberFormatException:
This class processes a File and returns a String arraylist of the input. It throws an IOException and NumberFormatException. It first creates an ArrayList of lines, reads the lines and then while the line is 0, it adds to element to the arraylist.


Project 2 Runtime Graph
Brute Force
Divide and Conquer
Kadane's Algorithm
300
200
100
0
100
200
500
1000
2000
5000
10000
20000
100000
500000
1000000
5000000
10000000
Input (n)

Log Scale:


Project 2 Runtime Graph
Brute Force
Divide and Conquer
Kadane's Algorithm
100
50
10
5
1
100
200
500
1000
2000
5000
10000
20000
100000
500000
1000000
5000000
10000000
Input (n)

Conclusion:
In conclusion, this project was challenging because finding the three different ways to calculate the maximumSubarray was hard. The brute force method took me a while to figure out but I was able to figure it out. The divide and conquer method was hard to find because it needed an additional method to calculate the cross array. I originally didn’t know how to read the file to make the test cases but after trial and error I was able to figure it out.I had the most trouble converting the strings to an array of integers and running through the file to make sure I was finding the right values.