EAST WEST UNIVERSITY

Project

Project Name: Maximum Sum Interval

Submitted To

Redwan Ahmed Rizvee
Lecturer
Department of Computer Science and Engineering
East West University

Submitted By

Group: 1
Section: 05

Name	ID
Mahmuda Islam Rodela	2020-2-60-038
Ratul Saha	2020-1-60-036
Aisha Siddika Mim	2020-1-60-040

Date of submission: Friday, 2 September, 2022

Problem Statement

The name of the given problem is "**Maximum Sum Interval**". The purpose of this project is to find the contiguous subarray with the biggest total within a one-dimensional array of numbers. Generally, a subarray refers that slice of a contiguous array which maintains the order of the elements. It can take both positive and negative numbers. In our project, we have some properties like:

- if the array contains only non-positive numbers, then the problem is the number in the array with the smallest magnitude.
- If the array only contains non-negative numbers, the problem is solved easily and the sum of all the elements in the list is the maximum sum.
- An empty set is not valid.
- There can be many sub-arrays with the same maximum sum to the problem.

Algorithm Discussion

To solve this project, we have to used Kadane's Algorithm. It is an iterative dynamic programing algorithm. So, from the Divide and Conquer, Dynamic Programming and Ad Hoc, we have used the Dynamic Programming method. In this algorithm:

Pseudocode:

```
MaxSubArray(array)
 Max=0
 Curr_max=0
 For I in array
  Curr_max += array[i]
   If curr_max<0
    Curr_max =0
     If max<curr_max
      Max = curr_max
```

Return max;

Step by Step flow:

For example: Let us assume that there is an array with 8 number of index and the elements are

-2, -5, 16, 1, 3, -1, -4, 7

Now we initialize curr_sum and max_sum with the first element or the element present in index 0.

So, now curr_sum = -2 and max_sum = -2

Then we will iterate a loop from 1 to n - 1 (array with n number of index, in this case n = 7)

For i = 1,

curr_sum = -2

array[1] = -5

curr_sum + array[1] = -2 + (-5) = -7

and -2 < -5 which means curr_sum + array[1] < array[1]

so, curr_sum = array[1] = -5

curr_sum < max_sum as -2 > -5

so max_sum will not update. It will be -2.

For i = 2,

curr_sum = -5

array[2] = 16

curr_sum + array[2] = -5 + 16 = 11

and 16 > -5 which means curr_sum + array[2] < array[2]

so, curr_sum = array[2] = 16

currentSum > max_sum as 16 > -2

so max_sum will update. It will be 16.

For i = 3,

curr_sum = 16

array[3] = 1

curr_sum + array[3] = 16 + 1= 17

and 17 > 1 which means currentSum + array[3] > array[3]

so, curr_sum = currentSum + array[3] = 17

curr_sum > max_sum as 17 > 16

so max_sum will update. It will be 17.

For i = 4,

curr_sum = 17

array[4] = 3

curr_sum + array[4] = 17 + 3 = 20

and 17 > 1 which means curr_sum + array[4] > array[4]

so, curr_sum = curr_sum + array[4] = 20

curr_sum > max_sum as 20 > 17

so max_sum will update. It will be 20.

For i = 5,

curr_sum = 20

array[5] = -1

curr_sum + array[5] = 20 - 1 = 19

and 19 > -1 which means curr_sum + array[5] > array[5]

so, curr_sum = currentSum + array[5] = 19

curr_sum < max_sum as 20 > 19

so max_sum will not update. It will be 20.

For i = 6,

curr_sum = 19

array[6] = - 4

curr_sum + array[6] = 19 - 4 = 15

and 15 > -4 which means curr_sum + array[6] > array[6]

so, curr_sum = curr_sum + array[6] = 15

curr_sum < max_sum as 20 > 15

so max_sum will not update. It will be 20.

For i = 7,

curr_sum = 15

array[6] = 7

currentSum + array[6] = 15 + 7 = 22

and 22 > 7 which means curr_sum + array[7] > array[7]

so, curr_sum = curr_sum + array[7] = 22

curr_sum > max_sum as 22 > 20

so max_sum will update. It will be 22.

So, the result is 22 and the contiguous array is 16, 1, 3, -1, -4, 7.

Complexity Analysis

In this project, we can find runtime and memory complexity. The kadane's algorithm has a runtime complexity of O(n). The loop from the program runs i=0 to (n – 1) time as we are taking an array with n number of indices. We have used a single loop to solve the problem. The run time complexity of our program is O(n).

Memory complexity can be defined the additional amount of memory other than the input that a program takes. The kadane's algorithm has a runtime complexity of O(1). Here, we have taken two variables to store the maximum sum interval and current sum in a specific index. So, the memory complexity of our project is O(1).

Implementation

The simple approach to solve this problem is to run one loop and use if-else. We have followed the below steps to solve the problem.

```
void maxSumInterval(int sizeofArray, vector<int> array)
{

    int max_sum = array[0]; //stores maximum sum contiguous sum
    int curr_sum = array[0]; //stores current maximum sum ending here

    for (int i = 1; i < sizeofArray; i++)//iterates over the array and add the value of the current element to curr_sum and check
    {
        if((curr_sum + array[i]) > array[i])
        {

            curr_sum = curr_sum + array[i]; // if array element is smaller than sum the of array element and curr_sum, update curr_sum
        }
        else
        {
            curr_sum = array[i]; // if array element is greater than the sum of array element and currentSum, updates currentSum
        }
        if( curr_sum > max_sum)
        {
            max_sum = curr_sum; //if sum is greater than max_sum, updates max_sum equals to curr_sum
        }

    }
    cout<<max_sum; //prints the value of maximum sum
}
```

1. We used a function named maxSumInterval to find the maximum sum for a non-empty array.
2. First, we took two variables named maxSum and currSum . First one is for storing the maximum sum contiguous sum and the second one is for storing the current maximum sum ending here.
3. Secondly, we ran a loop for i from 1 to the length of the array to add the value of the current element to curr_sum and check.
4. Next, we checked,
    - If the array element is smaller than sum of array element and curr_sum, update curr_sum equals to the summation of current sum and that element.

- If the array element is greater than the sum of array element and curr_sum, curr_sum equals that element.
- Finally,if the sum is greater than max_sum, updates max_sum equals to curr_sum.

5. Lastly this maxSumInterval function prints the value of max_sum.
6. If the array is empty then this will print “invalid”.

Applications

We can use Kadane's algorithm in many ways in our daily application. It can be used to tackle problems such as "Ordering Station Travel" and "Hotels along the Coast." It also can be use for an image processing algorithm. It is also used for business analysis nowadays.

THE END