

---

# Data Structures & Algorithms II

## IS 2110

### Group Assignment 2

Group - 04

#### Group Members

Team member	Contribution	Email Addresses	Index number
AHILAN A.	Question 1 <b>Naive method</b>	archanaahil@gmail.com	17020042
JAYATHILAKA H.A.P.C.J.	Question 2	mailtochamodij@gmail.com	17020387
RATHNAYAKE A.R.M.M.K	Question 1 <b>Divide method</b>	kavindirathnayake0@gmail.com	17020727

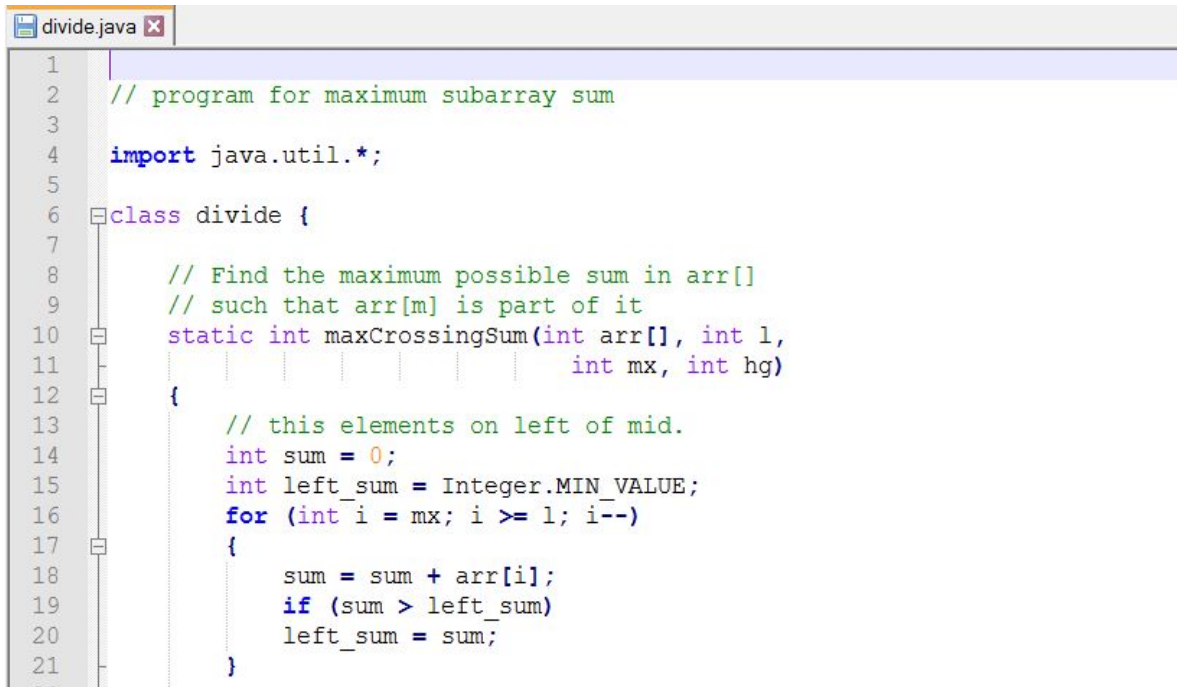
### Question 1:

Suppose that you are given a one-dimensional array which contains both positive and negative integers. You are required to find the sum of a contiguous subarray of elements that has the largest sum.

Here I have used java in this program. This gave for a one-dimensional array. Using the **Divide and Conquer** approach, we can find the maximum subarray sum in  $O(N \log N)$  time. First, use for finding the maximum possible in the array.

All the value which created array by finding the left element and the right element in the array. If check the array, you can find one element and the middle point of the array.

Find maximum subarray sum such that the subarray crosses the midpoint. We can easily find the crossing sum in linear time. The idea is simple, find the maximum sum starting from midpoint and ending at some point on left of mid, then find the maximum sum starting from  $\text{mid} + 1$  and ending with sum point on right of  $\text{mid} + 1$ . Finally, combine the two and return.



```
1
2 // program for maximum subarray sum
3
4 import java.util.*;
5
6 class divide {
7
8     // Find the maximum possible sum in arr[]
9     // such that arr[m] is part of it
10    static int maxCrossingSum(int arr[], int l,
11                             int mx, int hg)
12    {
13        // this elements on left of mid.
14        int sum = 0;
15        int left_sum = Integer.MIN_VALUE;
16        for (int i = mx; i >= l; i--)
17        {
18            sum = sum + arr[i];
19            if (sum > left_sum)
20                left_sum = sum;
21        }
22    }
23 }
```

```

22
23 // this elements on right of mid
24 sum = 0;
25 int right_sum = Integer.MIN_VALUE;
26 for (int i = mx + 1; i <= hg; i++)
27 {
28     sum = sum + arr[i];
29     if (sum > right_sum)
30         right_sum = sum;
31 }
32
33 // Return sum of elements on left
34 // Return and right of mid
35 return left_sum + right_sum;
36 }
37

```

```

37
38 // Returns sum of maxium sum subarray
39
40 static int maxSubArraySum(int arr[], int l,
41                             int hg)
42 {
43     //Only one element
44     if (l == hg)
45         return arr[l];
46
47     // Find middle point
48     int mx = (l + hg)/2;
49

```

```

50
51 // Maximum subarray sum in left half
52 // Maximum subarray sum in right half
53 // Maximum subarray sum such that the
54
55 return Math.max(Math.max(maxSubArraySum(arr, l, mx),
56                          maxSubArraySum(arr, mx+1, hg)),
57               maxCrossingSum(arr, l, mx, hg));
58 }
59
60 /* Driver program to test maxSubArraySum */
61 public static void main(String[] args)
62 {
63     int arr[] = {-10, -13, 23, -10, -11, 8, 13, -17};
64     int n = arr.length;
65     int max_sum = maxSubArraySum(arr, 0, n-1);
66
67     System.out.println("sum of contiguous subarray of elements : "+
68                       max_sum);
69 }
70 }
71

```

*Out put*

**C:\** Command Prompt

```

Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\ASUS>cd Desktop

C:\Users\ASUS\Desktop>javac divide.java

C:\Users\ASUS\Desktop>java divide
sum of contiguous subarray of elements : 23

C:\Users\ASUS\Desktop>_

```

## Naive method

Using for loop and finding the maximum value and calculating the sums of those elements.

```
#include<iostream>
#include<climits>

using namespace std;

int Find_maxSubarraySum(int a[], int size)
{
    int max = INT_MIN, max_end = 0;

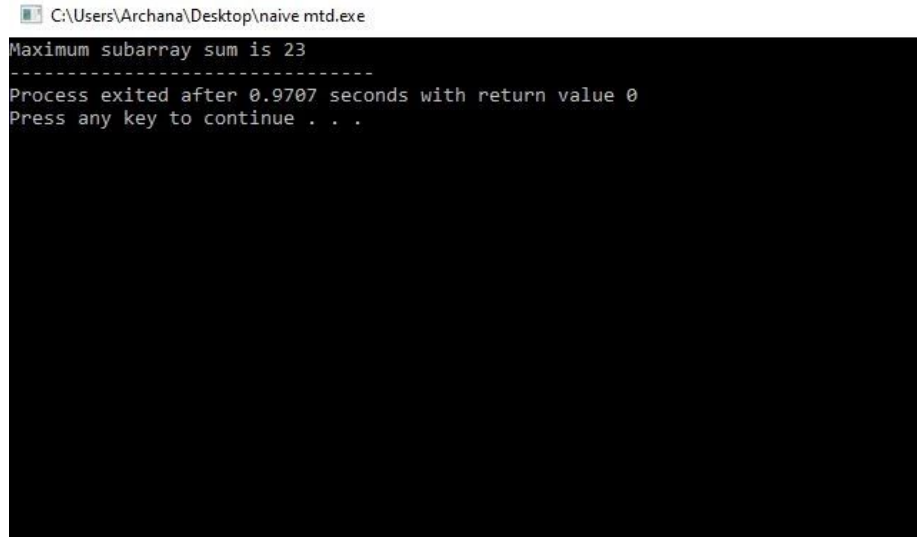
    //here for each element we are finding the maximum value and calculating the sums of those elements
    for (int i = 0; i < size; i++)
    {
        max_end = max_end + a[i];
        if (max < max_end)
            max = max_end;

        if (max_end < 0)
            max_end = 0;
    }
    return max;
}
```

In the main function initializing the array and calculating the size of the given array. Then find the maximum subarray sum.

```
int main()
{
    //initializing the array
    int a[] = {-10,-13,23,-10,-11,8,13,-17};
    //calculating the size of the given array
    int n = sizeof(a)/sizeof(a[0]);
    //here Find_maxSubarraySum() returns the maximum sum for the max_sum variable
    int max_sum = Find_maxSubarraySum(a, n);
    cout << "Maximum subarray sum is " << max_sum;
    return 0;
}
```

## Output:



```
C:\Users\Archana\Desktop\naive mtd.exe
Maximum subarray sum is 23
-----
Process exited after 0.9707 seconds with return value 0
Press any key to continue . . .
```

## Question 2:

You are given an array of independent jobs where every job is associated with a deadline and profit. The profit could be claimed only if the job is finished before the deadline. It is also given that every job takes only a single unit of time and hence the minimum possible deadline for any job is 1. You are required to find out how to maximize total profit if only one job can be scheduled at a time.

**The language used:** C++ language to implement this program.

**Method:** Firstly, I create a job structure array with the jobID, deadline, and profit.

```
#include <iostream>
#include <stdlib.h>
#include <string.h>
using namespace std;

//Creating the job structure array
] struct job{
    string jobid;
    int deadline;
    int profit;
- };
```

I initialized the values to an array named as jobs in the main method. Next calculated the number of jobs available in the array using divide the size of the job by size of the array in the zeroth position. Next, I sorted the JobsID according to the descending order of the profit and print the sorted descending order JobID with the Profit and Deadline.

```

DSA3.cpp
25 int main(){
26     int i,j;
27     job temp;
28     int max_deadlinelevel=0;
29     int slot[max_deadlinelevel];
30
31     //Initialzing the value to the array
32     job jobs[]={{"J1",2,90}, {"J2",2,20}, {"J3",1,40}, {"J4",1,30}, {"J5",4,65}, {"J6",1,35}, {"J7",3,50}};
33     int n=sizeof(jobs)/sizeof(jobs[0]);
34     printf("Number of jobs available:%d\n",n);
35
36     //Sorted the Jobs Id according to the decending order of the profit
37     for(i=1;i<n;i++){
38         for(j=0;j<n-i;j++){
39             if(jobs[j+1].profit>jobs[j].profit){
40                 temp=jobs[j+1];
41                 jobs[j+1]=jobs[j];
42                 jobs[j]=temp;
43             }
44         }
45     }
46
47

```

To find out the maximum level of the deadline used an if method and print the maximum deadline level. Create empty slots according to the number of maximum deadline..

```

    for(i=0;i<n;i++){
        //Print the sortde jobs in decending Order
        printf("JobID %s Profit %d Deadline %d \n",jobs[i].jobid.c_str(),jobs[i].profit,jobs[i].deadline);
        //Find the maximum deadline level
        if(jobs[i].deadline>max_deadlinelevel){
            max_deadlinelevel=jobs[i].deadline;
        }
    }
    //Print the maximum deadline level
    printf("Maximum Deadline : %d\n",max_deadlinelevel);

    //Creating empty slot according to the number of maximum deadline level
    for (int i=0; i<=max_deadlinelevel; i++) {
        slot[i] = false;
    }

```

To put the values to the array, get the first value of the sorted descending array of profit and compare its deadline with the maximum deadline. If it is below put the number to relevant array position according to the deadline. In here we take array index as the deadline. Before getting the next value always we have to find out is there are empty slots in the array. Using the for loop goes to the next value of descending order

array of profit. If the array is full the loop will be break. From that, we can get the jobs finished deadline according to the Greedy method.

**Output :**

```
Number of jobs available:7
JobID J1 Profit 90 Deadline 2
JobID J5 Profit 65 Deadline 4
JobID J7 Profit 50 Deadline 3
JobID J3 Profit 40 Deadline 1
JobID J6 Profit 35 Deadline 1
JobID J4 Profit 30 Deadline 1
JobID J2 Profit 20 Deadline 2
Maximum Deadline : 4
Following is maximum profit sequence of jobs:
J3 J2 J7 J5
-----
Process exited after 2.645 seconds with return value 3221225477
Press any key to continue . . .
```