Student Name:-Srishti Shrivastava

Student Roll No.:- 1905647

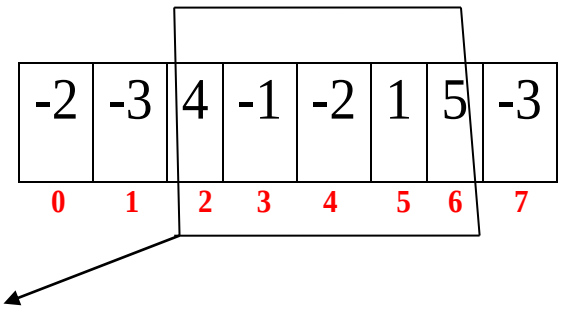
Algorithm Lab. Class Assignment-5

CSE Group 1

Date: - 6th August 2021

1. **Write a C program to find the sum of contiguous subarray within a one dimensional (1-D) array of numbers which has the largest sum. Find the time complexty of your program.**

Example



4 + (-1) + (-2) + 1 + 5 = 7

So the maximum contiguous subarray sum is 7

Program

```c
#include <stdio.h>
#include<limits.h>
#include<time.h>


#define ARRAY_SIZE(a)  sizeof(a)/sizeof(a[0])

#define MAX(a,b)   (a>b)?a:b
int maxSubArraySum(int a[], int n)
{
    int max_so_far = a[0];
    int curr_max = a[0];
    int i = 0;
    for (i = 1; i < n; i++)
    {
        curr_max = MAX(a[i], curr_max+a[i]);
        max_so_far = MAX(max_so_far, curr_max);
    }
    return max_so_far;
}
int main()
{
```

```
    int arr[] = { -2,  -3, 4, -1, -2, 1, 0, 5, -3};
    clock_t start, end;
    double total_cputime;
    start = clock();

    int arr_size = ARRAY_SIZE(arr);
    const int maxSum = maxSubArraySum(arr, arr_size);
    printf("\nMax sum = %d\n", maxSum);
    end = clock();
    printf("\nstarting time=%ld", start);
    printf("\nEnd time =%ld", end);

    total_cputime = ((double)(end - start));
    printf("\ntotal_cputime =%f", total_cputime);

    total_cputime = ((double)(end - start)) / CLOCKS_PER_SEC;
    printf("\ntotal_cputime in second =%f", total_cputime);

    return 0;
}
```

Output

```
C:\Users\hp\Documents\DAA LAB>cd "c:\Users\hp\Documents\DAA LAB\LAB-5\" && gcc que1

Max sum = 7

starting time=0
End time =1
total_cputime =1.000000
total_cputime in second =0.001000
c:\Users\hp\Documents\DAA LAB\LAB-5>
```

2. **Write a program to find out the largest difference between two elements A[i] and A[j] (A[j]-A[i]) of the array of integers A in O(n) time such that j > i. For example: Let A is an array of integers:**

   int[] a = { 10, 3, 6, 8, 9, 4, 3 };

   if i=1, j=3, then diff = a[j] - a[i] = 8 - 3 = 5

   if i=4, j=6, then diff = a[j] - a[i] = 3 - 9 = -6

   ……..

.....

if i=1, j=4, then diff = a[j] - a[i] = 9 - 3 = 6

.....

.....

6 is the largest number between all the differences, that is the answer.

Find the time complexty of your program.

Program

```c
#include <stdio.h>
#include <limits.h>
#include<time.h>


int max(int x, int y) {
    return (x > y) ? x : y;
}

int getMaxDiff(int arr[], int n)
{
    int diff = INT_MIN;
    if (n == 0) {
        return diff;
    }

    int max_so_far = arr[n-1];


    for (int i = n - 2; i >= 0; i--)
    {

        if (arr[i] >= max_so_far) {
            max_so_far = arr[i];
        }

        else {
            diff = max (diff, max_so_far - arr[i]);
        }
    }


    return diff;
}

int main()
{
```

```
    clock_t start, end;
    double total_cputime;
    start = clock();
    int arr[] = {10, 3, 6, 8, 9, 4, 3  };
    int n = sizeof(arr) / sizeof(arr[0]);

    int result = getMaxDiff(arr, n);
    if (result != INT_MIN) {
        printf("The maximum difference is %d", result);
    }
    end = clock();
    printf("\nstarting time=%ld", start);
    printf("\nEnd time =%ld", end);

    total_cputime = ((double)(end - start));
    printf("\ntotal_cputime =%f", total_cputime);

    total_cputime = ((double)(end - start)) / CLOCKS_PER_SEC;
    printf("\ntotal_cputime in second =%f", total_cputime);

    return 0;
}
```

Output

```
C:\Users\hp\Documents\DAA LAB>cd "c:\Users\hp\Documents\DAA LAB\LAB-5\" && gcc que2.c
The maximum difference is 6
starting time=0
End time =1
total_cputime =1.000000
total_cputime in second =0.001000
c:\Users\hp\Documents\DAA LAB\LAB-5>
```

3. Find the GCD and LCM of n numbers where (n>=2).

Program

```
#include <stdio.h>
#include <time.h>
int gcd(int a, int b)
{
    if (b == 0)
        return a;
    return gcd(b, a % b);
}
```

```c
int lcm(int a, int b) {
    return (a * b / gcd(a, b));
}
int GCDN(int a[], int n) {
    int gcd_ = a[0];
    for(int i=1;i<n;i++){
        gcd_ = gcd(gcd_, a[i]);
    }
    return gcd_;
}
int LCMN(int a[], int n) {
    int lcm_ = a[0];
    for(int i=1;i<n;i++){
        lcm_ = lcm(lcm_, a[i]);
    }
    return lcm_;
}
int main() {
    int n;
    printf("enter no of elements:\n");
    scanf("%d",&n);
    int arr[n];
    printf("elements are:-\n");
    for(int i=0;i<n;i++) {
        scanf("%d",&arr[i]);
    }
    time_t start, end;
    double time;
    start = clock();
    printf("GCD of numbers: %d\n", GCDN(arr, n));
    end = clock();
    time = (end - start) * 1.0 / CLOCKS_PER_SEC;
    printf("Time taken: %f seconds\n", time);
    start = clock();
    printf("LCM of numbers: %d\n", LCMN(arr, n));
    end = clock();
    time = (end - start) * 1.0 / CLOCKS_PER_SEC;
    printf("Time taken: %f seconds\n", time);
    return 0;
}
```

Output

```
C:\Users\hp\Documents\DAA LAB>cd "c:\Users\hp\Documents\DAA LAB\LAB-5\" && gcc que3.c
enter no of elements:
3
elements are:-
12
15
30
GCD of numbers: 3
Time taken: 0.001000 seconds
LCM of numbers: 60
Time taken: 0.000000 seconds
```

4. Consider an n × n matrix A = (a_{ij}), each of whose elements a_{ij} is a nonnegative real number, and suppose that each row and column of A sums to an integer value. We wish to replace each element a_{ij} with either ⌉ a_{ij} ⌉ or ⌋ a_{ij} ⌋ without disturbing the row and column sums. Here is an example:

$$\begin{pmatrix} 10.9 & 2.5 & 1.3 & 9.3 \\ 3.8 & 9.2 & 2.2 & 11.8 \\ 7.9 & 5.2 & 7.3 & 0.6 \\ 3.4 & 13.1 & 1.2 & 6.3 \end{pmatrix} \rightarrow \begin{pmatrix} 11 & 3 & 1 & 9 \\ 4 & 9 & 2 & 12 \\ 7 & 5 & 8 & 1 \\ 4 & 13 & 2 & 6 \end{pmatrix}$$

Write a program by defining an user defined function that is used to produce the rounded matrix as described in the above example. Find out the time complexity of your algorithm/function.

Program

```c
#include <stdio.h>
#include<stdlib.h>
#include<time.h>
#include<math.h>

int main(){
       clock_t start ,end;
    double taken;
    start=clock();
    int n=4;
    double arr[4][4]={{10.9,2.5,1.3,9.3},{3.8,9.2,2.2,11.8},{7.9,5.2,
7.3,0.6},{3.4,13.1,1.2,6.3}};
    for (int i = 0; i < n; i++)
{
    for (int j = 0; j < n; j++)
    {
        arr[i][j]=floor(arr[i][j]);
    }
```

```
}
for (int i = 0; i < 4; i++)
{
    for (int j = 0; j < 4; j++)
    {
        printf("%.6g ",arr[i][j]);
    }
    printf("\n");
}


    end=clock();
    taken=((double)(end-start))/CLOCKS_PER_SEC;
    printf("Total time taken %f",taken);
}
```

Output

```
C:\Users\hp\Documents\DAA LAB\LAB-5>cd "c:\Users\hp\Documents\DAA LAB\LAB-5\" && gcc que4.c -
10 2 1 9
3 9 2 11
7 5 7 0
3 13 1 6
Total time taken 0.005000
C:\Users\hp\Documents\DAA LAB\LAB-5>
```