**Student Name:- Prameet Upadhyay**

**Student Roll No.:- 1905692**

**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**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| -2 | -3 | 4 | -1 | -2 | 1 | 5 | -3 |

**0 1 2 3 4 5 6 7**

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

**So the maximum contiguous subarray sum is 7**

**Program**

**#include** <stdio.h>

**#include**<time.h>

**#include**<limits.h>

**int** algo(**int** *arr***[]**, **int** *n*)

{

**int** max\_sum **=** INT\_MIN;

**int** curr\_sum **=** 0;

**for** (**int** i **=** 0; i **<** *n*; i**++**)

    {

        curr\_sum **+=** *arr*[i];

**if** (max\_sum **<** curr\_sum)

            max\_sum **=** curr\_sum;

**if** (curr\_sum **<** 0)

            curr\_sum **=** 0;

    }

**return** max\_sum;

}

**int** main()

{

**time\_t** strt, end;

**int** n;

    scanf("%d", **&**n);

**int** arr[n];

**for** (**int** i **=** 0; i **<** n; i**++**)

    {

        scanf("%d", **&**arr[i]);

    }

    strt **=** clock();

**int** ans **=** algo(arr, n);

    end **=** clock();

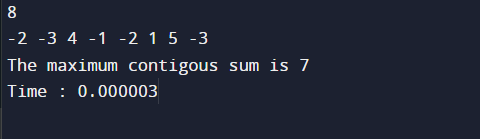
**double** t **=** end **-** strt;

    printf("The maximum contigous sum is %d", ans);

    printf("\nTime : %f",(t**/**CLOCKS\_PER\_SEC));

}

**Output**



1. **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**

**………**

**………**

1. **is the largest number between all the differences, that is the answer.**

**Find the time complexty of your program.**

**Program**

**#include** <stdio.h>

**#include**<time.h>

**int** algo(**int** *arr***[]**, **int** *n*)

{

**int** max\_diff **=** *arr*[1] **-** *arr*[0];

**int** min **=** *arr*[0];

**for** (**int** i **=** 1; i **<** *n*; i**++**)

    {

**int** curr\_diff **=** *arr*[i] **-** min;

**if** (max\_diff **<** curr\_diff)

            max\_diff **=** curr\_diff;

**if** (*arr*[i] **<** min)

            min **=** *arr*[i];

    }

**return** max\_diff;

}

**int** main()

{

**time\_t** strt, end;

**int** n;

    scanf("%d", **&**n);

**int** arr[n];

**for** (**int** i **=** 0; i **<** n; i**++**)

    {

        scanf("%d", **&**arr[i]);

    }

    strt **=** clock();

**int** ans **=** algo(arr, n);

    end **=** clock();

**double** t **=** end **-** strt;

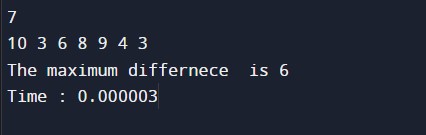
    printf("The maximum differnece  is %d", ans);

    printf("\nTime : %f",(t**/**CLOCKS\_PER\_SEC));

**return** 0;

}

**Output**

****

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

**Program**

**#include** <stdio.h>

**int** gcd(**int** *a*, **int** *b*)

{

**if** (*a* **==** 0)

**return** *b*;

**return** gcd(*b* **%** *a*, *a*);

}

**int** findGCD(**int** *A***[]**, **int** *n*)

{

**int** result **=** *A*[0];

**for** (**int** i **=** 1; i **<** *n*; i**++**)

    {

        result **=** gcd(*A*[i], result);

**if** (result **==** 1)

        {

**return** 1;

        }

    }

**return** result;

}

**int** findlcm(**int** *A***[]**, **int** *n*)

{

**int** ans **=** *A*[0];

**for** (**int** i **=** 1; i **<** *n*; i**++**)

        ans **=** (((*A*[i] **\*** ans)) **/**(gcd(*A*[i], ans)));

**return** ans;

}

**int** main()

{

**int** n;

    scanf("%d",**&**n);

**int** A[n];

**for**(**int** i **=**0;i**<**n;i**++**){

        scanf("%d",**&**A[i]);

    }

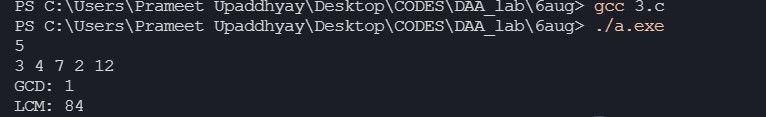
    printf("GCD: %d\n",findGCD(A,n));

    printf("LCM: %d",findlcm(A,n));

**return** 0;

}

**Output**

****

1. **Consider an n × n matrix A = (aij), each of whose elements aij 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 aij with either ˥ aij ˥ or ˩ aij ˩ without disturbing the row and column sums. Here is an example:**

****

**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**

**#include** <stdio.h>

**int** roundNo(**float** *num*)

{

**return** *num* **<** 0 **?** *num* **-** 0.5 **:** *num* **+** 0.5;

}

**void** algo(**float** *A*[3][4])

{

**for** (**int** i **=** 0; i **<** 3; i**++**)

    {

**for** (**int** j **=** 0; j **<** 4; j**++**)

        {

*A*[i][j] **=** roundNo(*A*[i][j]);

        }

    }

**for** (**int** i **=** 0; i **<** 3; i**++**)

    {

**for** (**int** j **=** 0; j **<** 4; j**++**)

        {

            printf("%0.2f ",*A*[i][j]);

        }

        printf("\n");

    }

}

**int** main()

{

**float** A[3][4];

**for** (**int** i **=** 0; i **<** 3; i**++**)

    {

**for** (**int** j **=** 0; j **<** 4; j**++**)

        {

            scanf("%f", **&**A[i][j]);

        }

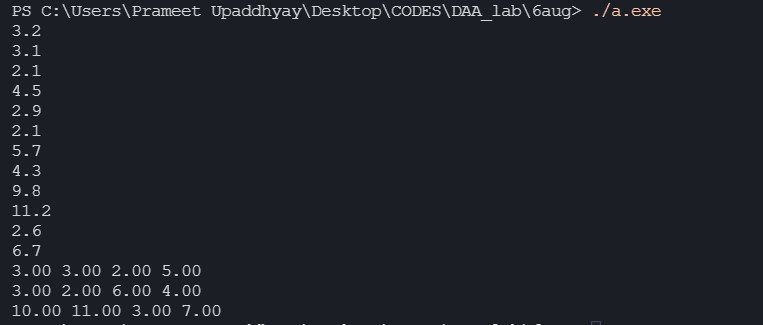
    }

    algo(A);

**return** 0;

}

**Output**

****