**Student Name:- CHAUDHARY HAMDAN**

**Student Roll No.:- 1905387**

**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**

// Author: Chaudhary Hamdan

#include <stdio.h>

#include <time.h>

#include <limits.h>

#include <stdlib.h>

#define sf(x) scanf("%d", &x)

#define pf printf

#define pfs(x) printf("%d ", x)

#define pfn(x) printf("%d\n", x)

#define pfc(x) printf("%d, ", x)

#define F(i,x,y) for(int i = x; i < y; i++)

#define FI(i,x,y,inc) for(int i = x; i < y; i += inc)

#define RF(i,x,y) for(int i = x; i >= y; i--)

#define pfarr(i,a,n) for(int i = 0; i < n-1; i++) pfs(a[i]); pfn(a[n-1]);

void i\_o\_from\_file() {

#ifndef ONLINE\_JUDGE

freopen("C:\\Users\\KIIT\\input", "r", stdin);

freopen("C:\\Users\\KIIT\\output", "w", stdout);

#endif

}

int maxSum(int a[], int n) {

int prev = INT\_MIN, curr = 0;

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

{

curr = curr + a[i];

if (prev < curr)

prev = curr;

if (curr < 0)

curr = 0;

}

return prev;

}

int main() {

i\_o\_from\_file();

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

int n;

sf(n);

int arr[n];

F(i, 0, n) {

sf(arr[i]);

}

time\_t start, end;

double time;

start = clock();

pf("Max Subarray sum : %d\n", maxSum(arr, n));

end = clock();

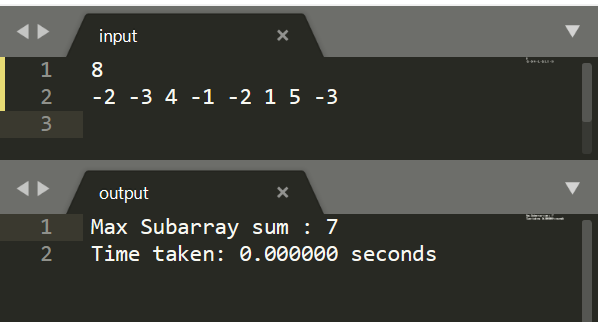
time = (end - start) \* 1.0 / CLOCKS\_PER\_SEC;

pf("Time taken: %f seconds", time);

return 0;

}

**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**

// Author: Chaudhary Hamdan

#include <stdio.h>

#include <time.h>

#include <limits.h>

#include <stdlib.h>

#define sf(x) scanf("%d", &x)

#define pf printf

#define pfs(x) printf("%d ", x)

#define pfn(x) printf("%d\n", x)

#define pfc(x) printf("%d, ", x)

#define F(i,x,y) for(int i = x; i < y; i++)

#define FI(i,x,y,inc) for(int i = x; i < y; i += inc)

#define RF(i,x,y) for(int i = x; i >= y; i--)

#define pfarr(i,a,n) for(int i = 0; i < n-1; i++) pfs(a[i]); pfn(a[n-1]);

void i\_o\_from\_file() {

#ifndef ONLINE\_JUDGE

freopen("C:\\Users\\KIIT\\input", "r", stdin);

freopen("C:\\Users\\KIIT\\output", "w", stdout);

#endif

}

int maxDiff(int a[], int n)

{

int max = INT\_MIN, c;

F(i, 0, n) {

F(j, i + 1, n) {

c = a[j] - a[i];

max = (c > max) ? c : max;

}

}

return max;

}

int main() {

i\_o\_from\_file();

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

int n;

sf(n);

int arr[n];

F(i, 0, n) {

sf(arr[i]);

}

time\_t start, end;

double time;

start = clock();

pf("Max Difference : %d\n", maxDiff(arr, n));

end = clock();

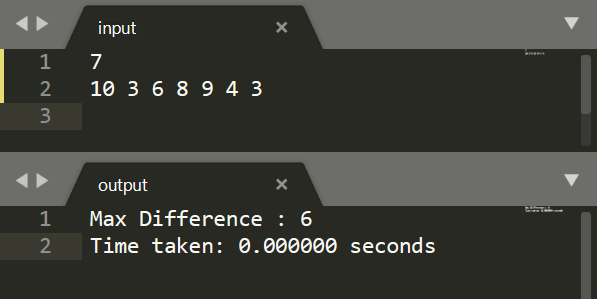
time = (end - start) \* 1.0 / CLOCKS\_PER\_SEC;

pf("Time taken: %f seconds", time);

return 0;

}

**Output**

****

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

**Program**

// Author: Chaudhary Hamdan

#include <stdio.h>

#include <time.h>

#include <limits.h>

#include <stdlib.h>

#define sf(x) scanf("%d", &x)

#define pf printf

#define pfs(x) printf("%d ", x)

#define pfn(x) printf("%d\n", x)

#define pfc(x) printf("%d, ", x)

#define F(i,x,y) for(int i = x; i < y; i++)

#define FI(i,x,y,inc) for(int i = x; i < y; i += inc)

#define RF(i,x,y) for(int i = x; i >= y; i--)

#define pfarr(i,a,n) for(int i = 0; i < n-1; i++) pfs(a[i]); pfn(a[n-1]);

void i\_o\_from\_file() {

#ifndef ONLINE\_JUDGE

freopen("C:\\Users\\KIIT\\input", "r", stdin);

freopen("C:\\Users\\KIIT\\output", "w", stdout);

#endif

}

int gcd(int a, int b)

{

if (b == 0)

return a;

return gcd(b, a % b);

}

int lcm(int a, int b) {

return (a \* b / gcd(a, b));

}

int GCDN(int a[], int n) {

int gcd\_ = a[0];

F(i, 1, n) {

gcd\_ = gcd(gcd\_, a[i]);

}

return gcd\_;

}

int LCMN(int a[], int n) {

int lcm\_ = a[0];

F(i, 1, n) {

lcm\_ = lcm(lcm\_, a[i]);

}

return lcm\_;

}

int main() {

i\_o\_from\_file();

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

int n;

sf(n);

int arr[n];

F(i, 0, n) {

sf(arr[i]);

}

time\_t start, end;

double time;

start = clock();

pf("GCD of numbers: %d\n", GCDN(arr, n));

end = clock();

time = (end - start) \* 1.0 / CLOCKS\_PER\_SEC;

pf("Time taken: %f seconds\n", time);

start = clock();

pf("LCM of numbers: %d\n", LCMN(arr, n));

end = clock();

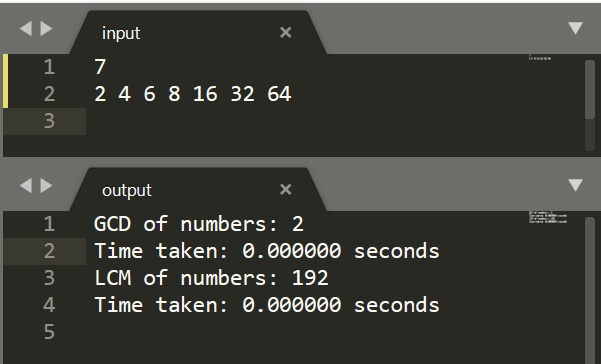
time = (end - start) \* 1.0 / CLOCKS\_PER\_SEC;

pf("Time taken: %f seconds\n", time);

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**

// Author: Chaudhary Hamdan

#include <stdio.h>

#include <time.h>

#include <limits.h>

#include <stdlib.h>

#define sf(x) scanf("%d", &x)

#define pf printf

#define pfs(x) printf("%d ", x)

#define pfn(x) printf("%d\n", x)

#define pfc(x) printf("%d, ", x)

#define F(i,x,y) for(int i = x; i < y; i++)

#define FI(i,x,y,inc) for(int i = x; i < y; i += inc)

#define RF(i,x,y) for(int i = x; i >= y; i--)

#define pfarr(i,a,n) for(int i = 0; i < n-1; i++) pfs(a[i]); pfn(a[n-1]);

void i\_o\_from\_file() {

#ifndef ONLINE\_JUDGE

freopen("C:\\Users\\KIIT\\input", "r", stdin);

freopen("C:\\Users\\KIIT\\output", "w", stdout);

#endif

}

int main() {

i\_o\_from\_file();

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

int n;

sf(n);

float arr[10][10];

float a[10][10];

F(i, 0, n) {

F(j, 0, n) {

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

a[i][j] = arr[i][j] - (int)(arr[i][j]);

}

}

float row[n], col[n];

F(i, 0, n) {

float s = 0.0;

F(j, 0, n) {

s += a[i][j];

}

row[i] = s;

}

F(j, 0, n) {

float s = 0.0;

F(i, 0, n) {

s += (arr[i][j] - (int)(arr[i][j]));

}

col[j] = s;

}

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

for (int j = 0; j < row[i]; j++) {

if (col[j] == 0) {

a[i][j] = 0;

continue;

}

a[i][j] = 1;

col[j]--;

}

}

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

for (int j = 0; j < col[i]; j++) {

if (row[j] == 0) {

a[j][i] = 1;

continue;

}

a[j][i] = 0;

row[j]--;

}

}

pf("Rounded up matrix :\n");

F(i, 0, n) {

F(j, 0, n) {

pf("%d ", (int)(arr[i][j]) + (int)(a[i][j]));

}

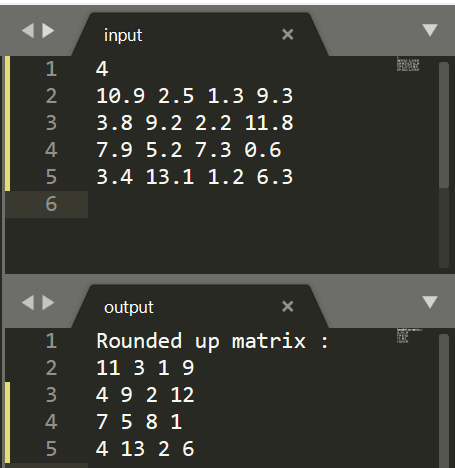
pf("\n");

}

return 0;

}

**Output**

****