Análisis de Algoritmos

Ejercicio 09: "Diseño de soluciones DyV"

Nombre: Luis Fernando Ramírez Cotonieto

Fecha de entrega:24 de Mayo del 2021

Grupo:3CM13

INSTITUTO POLITÉCNICO NACIONAL

ESCOM

Luis Fernando Ramírez Cotonieto

Problema 01: Divide y vencerás

Divide and conquer 1

Puntos		Límite de memoria	32MB
Límite de tiempo (caso)	1 s	Límite de tiempo (total)	60s

Descripción

Edgardo se puso un poco intenso este semestre y puso a trabajar a sus alumnos con problemas de mayor dificultad.

La tarea es simple, dado un arreglo A de números enteros debes imprimir cual es la suma máxima en cualquier subarreglo contiguo.

Por ejemplo si el arreglo dado es {-2, -5, **6, -2, -3, 1, 5**, -6}, entonces la suma máxima en un subarreglo contiguo es **7**.

Entrada

La primera línea contendrá un numero N.

En la siguiente línea N enteros representando el arreglo A.

Salida

La suma máxima en cualquier subarreglo contiguo.

Ejemplo

Entrada	Salida
8 -2 -5 6 -2 -3 1 5 -6	7
5 1 2 3 4 5	15

Límites

- $1 \leq N \leq 10^5$
- $|A_i| \leq 10^9$

omegaUp

Aceptación de juez:

Explicación:

Los números enteros son leídos conforme van llegando, para esto se manejan dos variables, “Sumapar” que mantendrá una suma parcial de los números que llevemos hasta el momento, tiene dos posibilidades para ser modificado, la primera es que el siguiente numero sea mayor que la suma de la actual y de ese numero, o sumando ese numero. La segunda variable, “Sumamax” llevará la suma máxima, el numero es modificado solo si la suma parcial es mayor a la suma máxima, en este caso, se quedará ese número.

Luis Fernando Ramírez Cotonieto

Código:

```
#include <iostream>
#include <string>
#include <sstream>
#include <iomanip>
#include <math.h>
using namespace std;

typedef long long int lli;

int main(int argc, char const *argv[])
{
    int n;
    scanf ("%d",&n);
    lli Sumapar, aux;
    scanf ("%lld",&Sumapar);
    lli Sumamax = Sumapar;
    for (int i = 1; i < n; ++i)
    {
        scanf ("%lld",&aux);
        if (aux > Sumapar + aux)
        {
            Sumapar = aux;
        }else{
            Sumapar += aux;
        }
        if (Sumapar > Sumamax)
        {
            Sumamax = Sumapar;
        }
    }
    printf("%lld\n", Sumamax);
    return 0;
}
```

Análisis de complejidad:

Las operaciones son constantes, y el ciclo for depende de una variable “n”, al realizar la iteración “n-1” veces, obtenemos:

$$ft(n) = O(n)$$

Luis Fernando Ramírez Cotonieto

Problema 02: Amigos y Regalos

Amigos y Regalos

Puntos	25.6	Límite de memoria	32MB
Límite de tiempo (caso)	1s	Límite de tiempo (total)	60s

Descripción

Tienes dos amigos. A ambos quieres regalarles varios números enteros como obsequio. A tu primer amigo quieres regalarle C_1 enteros y a tu segundo amigo quieres regalarle C_2 enteros. No satisfecho con eso, también quieres que todos los regalos sean únicos, lo cual implica que no podrás regalar el mismo entero a ambos de tus amigos.

Además de eso, a tu primer amigo no le gustan los enteros que son divisibles por el número primo X. A tu segundo amigo no le gustan los enteros que son divisibles por el número primo Y. Por supuesto, tu no le regalaras a tus amigos números que no les gusten.

Tu objetivo es encontrar el mínimo número V, de tal modo que puedas dar los regalos a tus amigos utilizando únicamente enteros del conjunto 1, 2, 3, ..., V. Por supuesto, tú podrías decidir no regalar algunos enteros de ese conjunto.

Un número entero positivo mayor a 1 es llamado primo si no tiene divisores enteros positivos además del 1 y el mismo.

Entrada

Una línea que contiene cuatro enteros positivos C_1, C_2, X, Y. Se garantiza que X y Y son números primos.

Salida

Una línea. Un entero que representa la respuesta al problema.

Ejemplo

Entrada	Salida	Descripción
3 1 2 3	5	Teniendo el conjunto de números: {1, 2, 3, 4, 5}, podemos regalarle los enteros {1, 3, 5} al amigo 1, y los enteros {2, 4} al amigo 2. Éste es el conjunto más pequeño que cumple con la solución.
1 1 2 3	2	Teniendo el conjunto de números: {1, 2}, podemos regalarle el entero {1} al amigo 1, y el entero {2} al amigo 2. Éste es el conjunto más pequeño que cumple con la solución.

Aceptación de juez:

Luis Fernando Ramírez Cotonieto

Explicación:

El algoritmo presenta una solución recursiva donde primero encuentra la condición base para devolverla como el valor mínimo posible para cumplir todas las condiciones, al ser creciente podemos comprobar si el valor es valido para los sucesivos, pero si no es valido, por igual manera ninguno lo será. Por la propiedad de ser creciente, podemos utilizar la búsqueda binaria de tal manera que el arreglo se vaya acomodando de izquierda a derecha, los elementos se van comparando en parejas, una de las secciones será de los números validos para un sujeto, incluyendo parte de los números primos que no acepta el otro sujeto y viceversa, sin dejar de tener en consideración que algunos puedan ser para ambos. Al final se dará un número booleano con estas consideraciones, dependiendo si es invalido o valido para regalo1 y regalo2, se otorgará el resultado.

Código:

```
#include <iostream>
#include <string>
#include <sstream>
#include <iomanip>
#include <math.h>
using namespace std;


typedef long long int lc;

const lc INF = 1LL << 60;


lc c1, c2, x, y;

bool Posicion(lc numero){
   lc unvalid = numero / (x * y);

   lc regalo1 = (c1 - ( (numero / y) - unvalid) );
   if (regalo1 < 0)
   {
      regalo1 = 0;
   }

   lc regalo2 = (c2 - ( (numero / x) - unvalid) );
   if (regalo2 < 0)
   {
```

```
        regalo2 = 0;
    }

    lc intersection = (numero - (numero / y) - (numero / x) + unvalid );

    return ( (regalo1 + regalo2) <= intersection);
}

lc BusquedaBinaria(lc left, lc right){
    lc mid = left + ((right - left) / 2);

    if ( Posicion(mid) && !Posicion(mid-1) )
    {
        return mid;
    }else if ( Posicion(mid) )
    {
        return BusquedaBinaria(left, mid - 1);
    }else{
        return BusquedaBinaria(mid + 1, right);
    }
}

int main( )
{
    cin>>c1>>c2>>x>>y;
    cout<<BusquedaBinaria(c1+c2,INF)<<endl;
    return 0;
}
```

Análisis de complejidad:

Tenemos nuestra función del main una muy simple, ppor lo que tenemos:

$$ft(n) = O(1)$$

Este programa realiza a la vez una busqueda binaria, conocemos con anterioridad que la busqueda binaria tiene una forma de:

$$ft(n) = \theta(\log n)$$

Podemos entonces obtener que la función del rpograma será:

$$ft(n) = O(\log n)$$

Luis Fernando Ramírez Cotonieto

Problema 07: INVCT

INVCNT - Inversion Count

#graph-theory #number-theory #shortest-path #sorting #bitmasks

Let A[0...n - 1] be an array of n distinct positive integers. If i < j and A[i] > A[j] then the pair (i, j) is called an inversion of A. Given n and an array A your task is to find the number of inversions of A.

Input

The first line contains t, the number of testcases followed by a blank space. Each of the t tests start with a number n (n <= 200000). Then n + 1 lines follow. In the ith line a number A[i - 1] is given (A[i - 1] <= 10^7). The (n + 1)th line is a blank space.

Output

For every test output one line giving the number of inversions of A.

Example

Input:

```
2

3
3
1
2

5
2
3
8
6
1
```

Output:

```
2
5
```

SPOJ.com - Problem RMQSQ

...

spoj.com

Aceptación de juez:

ID	DATE	USER	PROBLEM	RESULT	TIME	MEM	LANG
27949814	2021-05-25 01:49:44	Luis Coto	Inversion Count	**accepted** edit ideone it	0.15	8.0M	CPP

Luis Fernando Ramírez Cotonieto

Explicación:

El algoritmo lee los datos de entrada y colocarlos en un arreglo, posteriormente, ese arreglo lo colocaremos en otro junto a su índice de inicio en el arreglo original, éste será ordenado descendentemente con base en el número más alto que encontremos en el arreglo. A continuación tiene el procedimiento de las iteraciones. A la hora de insertar un valor, se realiza un procedimiento similar a la query, con base en el índice original dividiremos el segmento en el árbol y aumentaremos una unidad en la posición final, que se sumarán recursivamente en la llamada original para fijar el valor resultante en lo que se considera como el nodo padre.
Todos los valores que nos regresan las queries se suman en una variable que será la que retornemos para ser impresa y otorgarnos el resultado final.

Código:

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

typedef long long typ;
typedef long long siz;

siz countinv = 0;
void count_invertion_merge_it(typ lis1[],typ lis2[],typ lis[], siz n1, siz n2, siz n){
    siz p=0,q=0,r=0,counter=0;
    while(p<n1 && q<n2){
        if(lis1[p] <= lis2[q]){

            countinv += counter;
            lis[r++] = lis1[p++];

        }
        else{

                counter++;
            lis[r++] = lis2[q++];
        }
    }
    while(p<n1){
```

```
        countinv += counter;
        lis[r++] = lis1[p++];

    }
    while(q<n2){
        lis[r++] = lis2[q++];
    }

}

void merge_sort(typ lis[], siz n){
    if(n>1){
        siz pos = n/2;
        siz k = 0;
        typ lis1[pos],lis2[n-pos];
        for(siz i=0;i<pos;++i){
            lis1[i] = lis[k++];
        }
        for(siz i=0;i<n-pos;++i){
            lis2[i] = lis[k++];
        }
        merge_sort(lis1,pos);
        merge_sort(lis2,n-pos);
        count_invertion_merge_it(lis1,lis2,lis,pos,n-pos,n);
}

int main(){

    siz n,t;
    cin>>t;
    while(t--){
        char ignore[10];
        cin.getline(ignore,10);
        cin>>n;
        countinv = 0;
```

```
        typ Ar[n];
        for(siz i=0; i<n; ++i)
            cin>>Ar[i];
        merge_sort(Ar,n);


        cout<<countinv<<endl;


    }
return 0;
}
```

Análisis de complejidad:

$$ft(n) = O(n \log n)$$

Luis Fernando Ramírez Cotonieto

Problema 08: TRIPINV

TRIPINV - Mega Inversions

no tags

The n^2 upper bound for any sorting algorithm
is easy to obtain: just take two elements
that are misplaced with respect to each other
and swap them. Conrad conceived an algorithm
that proceeds by taking not two, but three misplaced
elements. That is, take three elements
ai > aj > ak with i < j < k and place them in
order ak; aj ; ai. Now if for the original algorithm
the steps are bounded by the maximum number
of inversions n(n-1)/2, Conrad is at his wits' end as
to the upper bound for such triples in a given sequence. He asks you to write a program
that counts the number of such triples.

Input

The rst line of the input is the length of the sequence, 1 <= n <= 10^5.
The next line contains the integer sequence a1; a2..an.
You can assume that all ai belongs [1; n].

Output

Output the number of inverted triples.

Example

```
Input:
4
3 3 2 1
Output:
2
```

SPOJ.com - Problem RMQSQ

...

spoj.com

Aceptación de juez:

ID	DATE	USER	PROBLEM	RESULT	TIME	MEM	LANG
27949789	2021-05-25 01:24:10	Luis Coto	Mega Inversions	**accepted** edit ideone it	0.14	6.8M	CPP

Luis Fernando Ramírez Cotonieto

Explicación:

En este código realmente requerí poder visualizar un ejemplo similar, la idea es al principio, comprimir la matriz mapeando los números. Después de eso, lea la matriz en orden inverso y consulte el valor de ese índice mapeado en el árbol de segmentos y se almacena en una matriz. La matriz se va ordenando y arreglando.

Código:

```
#define mx 100006

ll tree[2][mx];
ll ara[mx],ara2[mx];

void update(int id, int idx, ll val)
{
    for(; idx<mx && idx; idx+=idx&-idx)
        tree[id][idx]+=val;
}

ll query(int id, int idx)
{
    ll ret=0;
    for(; idx; idx-=idx&-idx)
        ret+=tree[id][idx];
    return ret;
}




int main()
{




    int n;
    sf(n);
```

```
    loop1(i,n)
    {
        sfl(ara[i]);

    }



    ll ans=0;

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

        int id=ara[i];

        ara2[i]=query(0,id);
        update(0,id+1,1);

    }

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

        int id=ara[i];
        ans+=query(1,id);
        update(1,id+1,ara2[i]);

    }

    printf("%lld\n",ans);

    return 0;
}
```

Análisis de complejidad:

$$ft(n) = O(n \log n)$$