Introduccion a la programacion competitiva

8/7/19

Cantidad de operaciones

<= 10⁷ entra seguro

10⁷ – 10⁹ ~

> 10⁹ probablemente no

Overflow

int (<= 2. 10⁹)

long long (<=10^18)

(ver iteraciones por operador/iterador usado a la hora de contar)

overflow con bitmask

long long MASK = 1 << i;

Los numeros sueltos son int.

si i >= 32, hay que acompaniar el numero con ll

es decir

long long MASK = 1ll <<i;

Aritmetica modular:

SI piden ultimo digito es modulo 10

SI piden ultimos 3 digitos es modulo 1000

El modulo es ‘distributivo’ en suma, resta y producto

La division es avanzada

Para que solo de positivos

((x%M) + M)%M

Queda en el rango [0, M-1)

Formulas fundamentales

Sumatoria hasta n, n natural

(n(n+1))/2

Sumatorias importantes

Progresiones aritmeticas

Progresiones geometicas

Potencias de 2

Funciones importantes

sort(algorithm) [begin, end]

NO USAR CON SET Y MAP

lower\_bound (sale en log n)

upper\_bound

equal\_range

lower\_bound(vec.begin, vec.end, n) – vec.begin => posicion del lowebound n del vector

VECTOR

DEQUE (eficiente, tiene acceso por indice) es como un arreglo

find(algorithm) [begin, end, val]

max\_element

min\_element(algorithm) [begin, end]

TDD (test driven developement)

Flags del compilador

flags indicados en wiki.oia.unsam.edu.ar/cpp-avanzado/opciones-gcc

Macro DBG

Estructuras fundamentales

Vectores:

vector<datatype>

tiene push\_back pop\_back

Sirven como pila

Son amortizados O(1)

Queue (fifo):

queue<datatype>

tiene push, front y pop

le puedo sacar cosas por el frente

Deque

deque<datatype>

push\_front

push\_back

pop\_front

pop\_back

podes sacar cosas por posicion,etc

Cola de dos puntas

O(1)

Tipico en problemas de cartas y reparten por abajo y por arriba

Medio lentos:

Unordered Sets: HashSet

unordered\_set<datatype>

insertar, borrar, consultar pertenencia

O(1)

Unordered Map: HashMap

pueden guardar un valor asociado a cada elemento

TreeSet:

set<int>

insertar, borrar, consultar pertenencia

s.lower\_bound

s.upper\_bound

O(log n)

Nota:

lower\_bound(s.begin, s.end, x) ES LENTO!

Map:

la key es la primera componente, se puede buscar solo a partir de la key pq no estan ordenados

Sumas parciales (algoritmo)

Suma de i a j en el arreglo (en una query por ejemplo)

Intervalo de substring c++[)

partial\_sum (arma la tabla de las sumas pero con intervalo cerrado [ ] ) (numeric)

adjacent\_differences (volver de la tabla de sumas al array original) (numeric)

Se aplican a rangos (v.begin(), v.end(), it), para array tenes que pasar el array donde se copia

Ej.:

int a[] = {1,2,3,4,5};

int b[5];

partial\_sum(a,a+5,b);

Entrada y salida rapida;

Usar printf o scanf

Alternativa:

cout cin (son lentos)

PROBLEMA: endl

El endl hace una llamada inmediata a imprir, esto es lento, por lo que hay que usar

“\n”

Ademas hay que utilizar:

ios::sync\_with\_stdio(false)

Esto apaga la sincronizacion entre printf scanf y cin y cout

OJO: No usar printf, scanf y cin y cout si usas esta linea

Hay que apagar el flush del buffer al hacer cin utilizar:

cin.tie(nullptr) 0 cin.tie(null)

Tarea:

Se tiene un arreglo de numeros enteros positivos y negativos

Cual es el subarreglo de mayor suma

Se puede hacer en O(n)