Notebook

UTN FRRO - HOLA

2024

Contents

1 Basics

1.1 Template

```cpp
#include <bits/stdc++.h>
#define forr(i, a, b) for (int i = (a); i < (b); i++)
#define forn(i, n) forr(i, 0, n)
#define dforn(i, n) for (int i = (n) - 1; i >= 0; i--)
#define forall(it, v) for (auto it = v.begin(); it != v.end(); it++)

#ifdef EBUG
// local
#else
// judge
#endif

using namespace std;

int main() {
#ifdef EBUG
    freopen("input.txt", "r", stdin);
#endif

    ios::sync_with_stdio(false);
    cin.tie(NULL);
    cout.tie(NULL);
    return 0;
}
```

1.2 Compilation

```
g++ -DEBUG <ej>.cpp -o a && time ./a
```

2 Math

2.1 Identidades

$\sum_{i=0}^{n} \binom{n}{i} = 2^n$
$\sum_{i=0}^{n} i\binom{n}{i} = n * 2^{n-1}$
$\sum_{i=m}^{n} i = \frac{n(n+1)}{2} - \frac{m(m-1)}{2} = \frac{(n+1-m)(n+m)}{2}$
$\sum_{i=0}^{n} i = \sum_{i=1}^{n} i = \frac{n(n+1)}{2}$
$\sum_{i=0}^{n} i^2 = \frac{n(n+1)(2n+1)}{6} = \frac{n^3}{3} + \frac{n^2}{2} + \frac{n}{6}$
$\sum_{i=0}^{n} i(i-1) = \frac{8}{6}(\frac{n}{2})(\frac{n}{2}+1)(n+1)$ (doubles) $\rightarrow$ Sino ver caso impar y par
$\sum_{i=0}^{n} i^3 = \left(\frac{n(n+1)}{2}\right)^2 = \frac{n^4}{4} + \frac{n^3}{2} + \frac{n^2}{4} = \left[\sum_{i=1}^{n} i\right]^2$
$\sum_{i=0}^{n} i^4 = \frac{n(n+1)(2n+1)(3n^2+3n-1)}{30} = \frac{n^5}{5} + \frac{n^4}{2} + \frac{n^3}{3} - \frac{n}{30}$
$\sum_{i=0}^{n} i^p = \frac{(n+1)^{p+1}}{p+1} + \sum_{k=1}^{p} \frac{B_k}{p-k+1}\binom{p}{k}(n+1)^{p-k+1}$

2.2 Tablas y cotas (Primos, Divisores, Factoriales, etc)

Factoriales

0! = 1	11! = 39.916.800
1! = 1	12! = 479.001.600 ($\in$ `int`)
2! = 2	13! = 6.227.020.800
3! = 6	14! = 87.178.291.200
4! = 24	15! = 1.307.674.368.000
5! = 120	16! = 20.922.789.888.000
6! = 720	17! = 355.687.428.096.000
7! = 5.040	18! = 6.402.373.705.728.000
8! = 40.320	19! = 121.645.100.408.832.000
9! = 362.880	20! = 2.432.902.008.176.640.000 ($\in$ `tint`)
10! = 3.628.800	21! = 51.090.942.171.709.400.000

Primos

2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97 101 103 107 109 113 127 131 137 139 149 151 157 163 167 173 179 181 191 193 197 199 211 223 227 229 233 239 241 251 257 263 269 271 277 281 283 293 307 311 313 317 331 337 347 349 353 359 367 373 379 383 389 397 401 409 419 421 431 433 439 443 449 457 461 463 467 479 487 491 499 503 509 521 523 541 547 557 563 569 571 577 587 593 599 601 607 613 617 619 631 641 643 647 653 659 661 673 677 683 691 701 709 719 727 733 739 743 751 757 761 769 773 787 797 809 811 821 823 827 829 839 853 857 859 863 877 881 883 887 907 911 919 929 937 941 947 953 967 971 977 983 991 997 1009 1013 1019 1021 1031 1033 1039 1049 1051 1061 1063 1069 1087 1091 1093 1097 1103 1109 1117 1123 1129 1151 1153 1163 1171 1181 1187 1193 1201 1213 1217 1223 1229 1231 1237 1249 1259 1277 1279 1283 1289 1291 1297 1301 1303 1307 1319 1321 1327 1361 1367 1373 1381 1399 1409 1423 1427 1429 1433 1439 1447 1451 1453 1459 1471 1481 1483 1487 1489 1493 1499 1511 1523 1531 1543 1549 1553 1559 1567 1571 1579 1583 1597 1601 1607 1609 1613 1619 1621 1627 1637 1657 1663 1667 1669 1693 1697 1699 1709 1721 1723 1733 1741 1747 1753 1759 1777 1783 1787 1789 1801 1811 1823 1831 1847 1861 1867 1871 1873 1877 1879 1889 1901 1907 1913 1931 1933 1949 1951 1973 1979 1987 1993 1997 1999 2003 2011 2017 2027 2029 2039 2053 2063 2069 2081

Primos cercanos a 10^n

9941 9949 9967 9973 10007 10009 10037 10039 10061 10067 10069 10079
99961 99971 99989 99991 100003 100019 100043 100049 100057 100069
999959 999961 999979 999983 1000003 1000033 1000037 1000039
9999943 9999971 9999973 9999991 10000019 10000079 10000103 10000121
99999941 99999959 99999971 99999989 100000007 100000037 100000039 100000049
999999893 999999929 999999937 1000000007 1000000009 1000000021 1000000033

Cantidad de primos menores que 10^n

$\pi(10^1) = 4$; $\pi(10^2) = 25$; $\pi(10^3) = 168$; $\pi(10^4) = 1229$; $\pi(10^5) = 9592$
$\pi(10^6) = 78.498$; $\pi(10^7) = 664.579$; $\pi(10^8) = 5.761.455$; $\pi(10^9) = 50.847.534$
$\pi(10^{10}) = 455.052{,}511$; $\pi(10^{11}) = 4.118.054.813$; $\pi(10^{12}) = 37.607.912.018$

2.3 Reglas de divisibilidad

Nro	Regla	Ejemplo
1	Todos los números	5: porque si divides 5:1=5 y ese número es un múltiplo o divisor de cualquier número.
2	El número termina en una cifra par.	378: porque la última cifra (8) es par.
3	La suma de sus cifras es un múltiplo de 3.	480: porque 4+8+0=12 es múltiplo de 3.
4	Sus últimos dos dígitos son 0 o un múltiplo de 4.	300 y 516 son divisibles entre 4 porque terminan en 00 y en 16, respectivamente, siendo este último un múltiplo de 4 (16=4*4).
5	La última cifra es 0 o 5.	485: porque termina en 5.
7	Un número es divisible entre 7 cuando, al separar la última cifra de la derecha, multiplicarla por 2 y restarla de las cifras restantes la diferencia es igual a 0 o es un múltiplo de 7. Otro sistema: Si la suma de la multiplicación de los números por la serie 2,3,1,-2,-3,-1... da 0 o un múltiplo de 7.	34349: separamos el 9, y lo duplicamos (18), entonces 3434-18=3416. Repetimos el proceso separando el 6 (341'6) y duplicándolo (12), entonces 341-12=329, y de nuevo, 32'9, 9*2=18, entonces 32-18=14; por lo tanto, 34349 es divisible entre 7 porque 14 es múltiplo de 7. Ejemplo método 2: 34349: [(2*3)+(3*4)+(1*3)-(2*4)-(3*9)]= 6+12+3-8-27 = -14.8
8	Para saber si un número es divisible entre 8 hay que comprobar que sus tres últimas cifras sean divisibles entre 8. Si sus tres últimas cifras son divisibles entre 8 entonces el número también es divisible entre 8.	Ejemplo: El número 571.328 es divisible por 8 ya que sus últimas tres cifras (328) son divisibles por 8 (32 = 8*4 y 8 = 8*1). Realizando la división comprobamos que 571.328 : 8 = 71.416
9	Un número es divisible por 9 cuando al sumar todas sus cifras el resultado es múltiplo de 9.	504: sumamos 5+0+4=9 y como 9 es múltiplo de 9 504 es divisible por 9 5346: sumamos 5+3+4+6=18 y como 18 es múltiplo de 9, 5346 es divisible por 9.
10	La última cifra es 0.	4680: porque termina en 0
11	Sumando las cifras (del número) en posición impar por un lado y las de posición par por otro. Luego se resta el resultado de ambas sumas obtenidas. Si el resultado es cero o un múltiplo de 11, el número es divisible entre este. Si el número tiene solo dos cifras y estas son iguales será múltiplo de 11.	42702: 4+7+2=13 · 2+0=2 · 13-2=11 → 42702 es múltiplo de 11. 66: porque las dos cifras son iguales. Entonces 66 es múltiplo de 11.
13	Un número es divisible entre 13 cuando, al separar la última cifra de la derecha, multiplicarla por 9 y restarla de las cifras restantes la diferencia es igual a 0 o es un múltiplo de 13	3822: separamos el último dos (382'2) y lo multiplicamos por 9, 2×9=18, entonces 382-18=364. Repetimos el proceso separando el 4 (36'4) y multiplicándolo por 9, 4×9=36, entonces 36-36=0; por lo tanto, 3822 es divisible entre 13.
17	Un número es divisible entre 17 cuando, al separar la última cifra de la derecha, multiplicarla por 5 y restarla de las cifras restantes la diferencia es igual a 0 o es un múltiplo de 17	2142: porque 214'2, 2*5=10, entonces 214-10=204, de nuevo, 20'4, 4*5=20, entonces 20-20=0; por lo tanto, 2142 es divisible entre 17.
19	Un número es divisible entre 19 si al separar la cifra de las unidades, multiplicarla por 2 y sumar a las cifras restantes el resultado es múltiplo de 19.	3401: separamos el 1, lo doblamos (2) y sumamos 340+2= 342, ahora separamos el 2, lo doblamos (4) y sumamos 34+4=38 que es múltiplo de 19, luego 3401 también lo es.

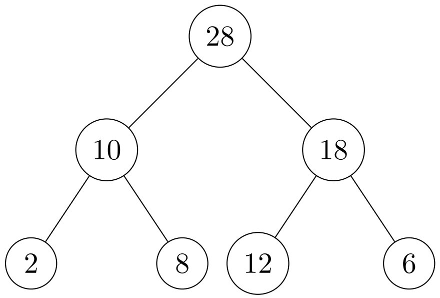
Continúa		
Nro	**Regla**	**Ejemplo**
20	Un número es divisible entre 20 si sus dos últimas cifras son ceros o múltiplos de 20. Cualquier número par que tenga uno o más ceros a la derecha, es múltiplo de 20.	57860: Sus 2 últimas cifras son 60 (Que es divisible entre 20), por lo tanto 57860 es divisible entre 20.
23	Un número es divisible entre 23 si al separar la cifra de las unidades, multiplicar por 7 y sumar las cifras restantes el resultado es múltiplo de 23.	253: separamos el 3, lo multiplicamos por 7 y sumamos 25+21= 46, 46 es múltiplo de 23 así que es divisible entre 23.
25	Un número es divisible entre 25 si sus dos últimas cifras son 00, o en múltiplo de 25 (25,50,75,...)	650: Es múltiplo de 25 por lo cual es divisible. 400 también será divisible entre 25.
29	Un número es divisible entre 29 cuando, al separar la última cifra de la derecha, multiplicarla por 3 y restarla de las cifras restantes la diferencia es igual a 0 o es un múltiplo de 29	2436: separamos el 6 (243'6) y lo multiplicamos por 3, 6×3=18, entonces 243-18=225. Repetimos el proceso separando el 5 (22'5) y multiplicándolo por 3, 5×3=15, entonces 22-15=7, que no es divisible entre 29.

2.4 Coprimos

Son aquellos números enteros a y b cuyo único factor en común que tienen es 1. Equivalentemente son coprimos, si, y solo si, su máximo común divisor (MCD) es igual a 1. Dos números coprimos no tienen por qué ser primos absolutos de forma individual. 14 y 15 son compuestos, sin embargo son coprimos, pues: $gcd(14, 15) = 1$

3 Estructuras

3.1 Segment Tree




```cpp
#include <bits/stdc++.h>
#define INF 100000000
using namespace std;

int n, t[4*10000];

void buildST(int a[], int v, int tl, int tr) { // este te hace las sumas
    if (tl == tr) {
        t[v] = a[tl];
    } else {
        int tm = (tl + tr) / 2;
        buildST(a, v * 2, tl, tm);
        buildST(a, v * 2 + 1, tm + 1, tr);
        t[v] = t[v * 2] + t[v * 2 + 1]; //aca esta la parte en que las suma
    }
}

int queryST(int v, int tl, int tr, int l, int r){
    if( l > r){
        return 0;
    }
    if( l == tl && r == tr){
        return t[v];
    }
    int tm = (tl + tr)/2;
    return queryST(v*2, tl, tm, l, min(r, tm)) +
    + queryST(v*2+1, tm+1, tr, max(l, tm+1), r); // esto lo hace para sumar

}

void updateST(int v, int tl, int tr, int pos, int new_val){
    if(tl == tr){
        t[v] = new_val;
    } else {
        int tm = (tl + tr) / 2;
        if(pos <= tm){
            updateST(v*2, tl, tm, pos, new_val);
        }else{
            updateST(v*2 + 1, tm+1, tr, pos, new_val);
```

```
        }
        t[v] = t[v*2] + t[v*2+1];
    }
}
```

4 Grafos

4.1 Recorrer Grafos

Dado un Grafo como lista de adjacencias

```
#include <bits/stdc++.h>
#define MAXN 100000
using namespace std;

vector<int> G[MAXN];
bool visited[MAXN];
```

Podemos recorrerlo con DFS o con BFS.

4.1.1 DFS

```
void dfs(int nodo){
    if(visited[nodo]) return;
    visited[nodo] = true;

    for(auto it : G[nodo]){
        if(!visited[it]){
            // logic
        }
    }
}
```

4.1.2 BFS

```
void bfs(int nodo){
    cola.push(nodo);

    while(!cola.empty()){
        Nodo actual = cola.front();
        cola.pop();

        for (int vecino : G[actual]) {
            if (!visited[vecino]) {
                visited[vecino] = true;
                cola.push(vecino);
            }
        }
    }
}
```

4.2 Camino Mínimo

4.2.1 Bellman-Ford

El algoritmo de Bellman-Ford encuentra el camino desde un nodo de origen a todos los nodos del grafo.

Complejidad = O(nm)

```cpp
#include <bits/stdc++.h>
#define INF 100000000
using namespace std;

vector<tuple<int, int, int>> edges;
int n;

void bellman_ford(int x){
    vector<int> distance(n, INF);
    distance[x] = 0;

    for (int i = 1; i <= n-1; i++) {
        for (auto e : edges) {
            int a, b, w;
            tie(a, b, w) = e;
            distance[b] = min(distance[b], distance[a]+w);
        }
    }
}
```

4.2.2 Ciclos negativos

El algoritmo es capaz de detectar ciclos negativos. Para eso Se debe correr una vez m'as

4.2.3 Dijkstra

El algorimo necesita que todos los pesos sean ¿ 0

Complejidad = O(n + m log m)

```cpp
#include <bits/stdc++.h>
#define MAXN 100000
using namespace std;
typedef pair<int, int> ii;

vector<pair<int, int>> G[MAXN]; //Lista de pares, dest, peso
bool visited[MAXN];
int n;

void dijkstra(int x){
    // La PQ esta ordenada de menor a mayor
    priority_queue<ii, vector<ii>, greater<ii>> q;
    vector<int> distance(n, MAXN);
    distance[x] = 0;
    q.push({0,x});

    while (!q.empty()) {
        int a = q.top().second; q.pop();
        if (visited[a]) continue;
        visited[a] = true;

        for (auto u : G[a]) {
            int b = u.first, w = u.second;
            if (distance[a]+w < distance[b]) {
                distance[b] = distance[a]+w;
                q.push({distance[b], b});
            }
        }
    }
}
```

5 Geometria

5.1 Punto

```
#include <bits/stdc++.h>
#define ll long long
#define ld double

struct pto {
    ll x, y;
    pto() : x(0), y(0) {} //Constructor pto a = pto(); ==> ) a.x = 0, a.y = 0
    pto(ll _x, ll _y) : x(_x), y(_y) {}
    pto operator+(pto b) { return pto(x+b.x, y+b.y); }
    pto operator-(pto b) { return pto(x-b.x, y-b.y); }
    pto operator+(ll k) { return pto(x+k, y+k); }
    pto operator*(ll k) { return pto(x*k, y*k); }
    pto operator/(ll k) { return pto(x/k, y/k); }
    ll operator*(pto b) { return x*b.x + y*b.y; }
    pto proj(pto b) { return b*((*this)*b) / (b*b); }
    ll operator^(pto b) { return x*b.y - y*b.x; }
    ld norm() { return sqrt(x*x + y*y); }
    ld dist(pto b) { return (b - (*this)).norm(); }
};
```

5.2 Line

```
#include "pto.cpp"

int sgn(T x) { return x < 0 ? -1 : !!x; }

struct line {
    T a, b, c;  // Ax+By=C
    line() {}
    line(T a_, T b_, T c_) : a(a_), b(b_), c(c_) {}

    // TO DO: check negative C (multiply everything by -1)
    line(pto u, pto v) : a(v.y - u.y), b(u.x - v.x), c(a * u.x + b * u.y) {}
    int side(pto v) { return sgn(a * v.x + b * v.y - c); }
    bool inside(pto v) { return abs(a * v.x + b * v.y - c) <= EPS; }
    bool parallel(line v) { return abs(a * v.b - v.a * b) <= EPS; }
    pto inter(line v) {
        T det = a * v.b - v.a * b;
        if (abs(det) <= EPS) return pto(INF, INF);
        return pto(v.b * c - b * v.c, a * v.c - v.a * c) / det;
    }
};
```

5.3 Segment

```
#include "pto.cpp"
#include "line.cpp"

struct segment {
    pto s, e;
    segment(pto s_, pto e_) : s(s_), e(e_) {}

    pto closest(pto b) {
        pto bs = b - s, es = e - s;
        ld l = es * es;
        if (abs(l) <= EPS) return s;
        ld t = (bs * es) / l;
        if (t < 0.) return s;       // comment for lines
        else if (t > 1.) return e;  // comment for lines
        return s + (es * t);
    }
    bool inside(pto b) { //Return true if pto b is inside the segment
        return abs(s.dist(b) + e.dist(b) - s.dist(e)) < EPS;
    }

    pto inter(segment b) {  // if a and b are collinear, returns one point
        if ((*this).inside(b.s)) return b.s;
        if ((*this).inside(b.e)) return b.e;
        pto in = line(s, e).inter(line(b.s, b.e));
        if ((*this).inside(in) && b.inside(in)) return in;
        return pto(INF, INF);
    }
};
```

5.4 Circle

```
#define sqr(a) ((a)*(a))
pto perp(pto a){return pto(-a.y, a.x);}
line bisector(pto a, pto b){
    line l = line(a, b); pto m = (a+b)/2;
    return line(-l.b, l.a, -l.b*m.x+l.a*m.y);
}

struct circle{
    pto o; T r;
    circle(){}
    circle(pto a, pto b, pto c) {
        o = bisector(a, b).inter(bisector(b, c));
        r = o.dist(a);
    }
    bool inside(pto p) { return (o-p).norm_sq() <= r*r+EPS; }
    bool inside(circle c) { // this inside of c
        T d = (o - c.o).norm_sq();
        return d <= (c.r-r) * (c.r-r) + EPS;
    }
    // circle containing p1 and p2 with radius r
    // swap p1, p2 to get snd solution
    circle* circle2PtoR(pto a, pto b, T r_) {
```

```
        ld d2 = (a-b).norm_sq(), det = r_*r_/d2 - ld(0.25);
        if(det < 0) return nullptr;
        circle *ret = new circle();
        ret->o = (a+b)/ld(2) + perp(b-a)*sqrt(det);
        ret->r = r_;
        return ret;
    }
    pair<pto, pto> tang(pto p) {
        pto m = (p+o)/2;
        ld d = o.dist(m);
        ld a = r*r/(2*d);
        ld h = sqrtl(r*r - a*a);
        pto m2 = o + (m-o)*a/d;
        pto per = perp(m-o)/d;
        return make_pair(m2 - per*h, m2 + per*h);
    }
    vector<pto> inter(line l) {
        ld a = l.a, b = l.b, c = l.c - l.a*o.x - l.b*o.y;
        pto xy0 = pto(a*c/(a*a + b*b), b*c/(a*a + b*b));
        if(c*c > r*r*(a*a + b*b) + EPS) {
            return {};
        }else if(abs(c*c - r*r*(a*a + b*b)) < EPS) {
            return { xy0 + o };
        }else{
            ld m = sqrtl((r*r - c*c/(a*a + b*b))/(a*a + b*b));
            pto p1 = xy0 + (pto(-b,a)*m);
            pto p2 = xy0 + (pto(b,-a)*m);
            return { p1 + o, p2 + o };
        }
    }
    vector<pto> inter(circle c) {
        line l;
        l.a = o.x - c.o.x;
        l.b = o.y - c.o.y;
        l.c = (sqr(c.r)-sqr(r)+sqr(o.x)-sqr(c.o.x)+sqr(o.y)-sqr(c.o.y))/2.0;
        return (*this).inter(l);
    }
    ld inter_triangle(pto a, pto b) { // area of intersection with oab
        if(abs((o-a)^(o-b)) <= EPS) return 0.;
        vector<pto> q = {a}, w = inter(line(a,b));
        if(sz(w) == 2) forn(i,sz(w)) if((a-w[i])*(b-w[i]) < -EPS) q.pb(w[i]);
        q.pb(b);
        if(sz(q) == 4 && (q[0]-q[1])*(q[2]-q[1]) > EPS) swap(q[1], q[2]);
        ld s = 0;
        forn(i, sz(q)-1){
            if(!inside(q[i]) || !inside(q[i+1])) {
                s += r*r*angle((q[i]-o),q[i+1]-o)/T(2);
            }
            else s += abs((q[i]-o)^(q[i+1]-o)/2);
        }
        return s;
    }
};
vector<ld> inter_circles(vector<circle> c){
    vector<ld> r(sz(c)+1); // r[k]: area covered by at least k circles
    forn(i, sz(c)) {      // O(n^2 log n) (high constant)
        int k = 1;
        cmp s(c[i].o, pto(1,0));
        vector<pair<pto,int>> p = {
            {c[i].o + pto(1,0)*c[i].r, 0},
            {c[i].o - pto(1,0)*c[i].r, 0}};
        forn(j, sz(c)) if(j != i) {
            bool b0 = c[i].inside(c[j]), b1 = c[j].inside(c[i]);
            if(b0 && (!b1 || i<j)) k++;
            else if(!b0 && !b1) {
                vector<pto> v = c[i].inter(c[j]);
                if(sz(v) == 2) {
                    p.pb({v[0], 1}); p.pb({v[1], -1});
                    if(s(v[1], v[0])) k++;
                }
            }
        }
        sort(p.begin(), p.end(), [&](pair<pto,int> a, pair<pto,int> b) {
                return s(a.fst,b.fst); });
        forn(j,sz(p)) {
            pto p0 = p[j? j-1: sz(p)-1].fst, p1 = p[j].fst;
            ld a = angle(p0 - c[i].o, p1 - c[i].o);
            r[k]+=(p0.x-p1.x)*(p0.y+p1.y)/ld(2)+c[i].r*c[i].r*(a-sinl(a))/ld(2);
            k += p[j].snd;
        }
    }
    return r;
}
```

6 Utils

6.1 Binary Search

```
#include <bits/stdc++.h>
using namespace std;

int bs(vector<int> &v, int val){
    int l = 0, r = v.size() - 1, mid = (l+r)/2;
    while(l <= r){
        if(val < v[mid]){
            r = mid - 1;
        }else{
            l = mid + 1;
        }
        mid = (l+r)/2;
    }
    if(val < v[mid]){
        mid --;
    }

    return mid;
}
```

6.2 Sort

Ordenar un vector de pair por su segunda componente

```
vector<pair<int, int>> v;

bool sortbysec(const pair<int,int> &a, const pair<int,int> &b){
  return (a.second < b.second);
}

sort(v.begin(), v.end(), sortbysec);
```