

# First to Penalty



## Contents

|     |                                    |   |
|-----|------------------------------------|---|
| 1   | Template                           | 2 |
| 2   | Data structures                    | 2 |
| 2.1 | Simplified DSU (Stolen from GGDem) | 2 |
| 2.2 | Disjoint Set Union                 | 2 |
| 2.3 | Segment tree                       | 2 |
| 3   | Graphs                             | 3 |
| 4   | Math                               | 3 |
| 4.1 | Identities                         | 3 |
| 4.2 | Binary Exponentiation and modArith | 3 |
| 4.3 | Modular Inverse (dividir mod)      | 3 |
| 4.4 | Sieve Of Eratosthenes              | 4 |
| 4.5 | Sieve-based Factorization          | 4 |
| 4.6 | Berlekamp Massey                   | 4 |
| 4.7 | Modular Berlekamp Massey           | 4 |
| 5   | Geometry                           | 5 |
| 6   | Strings                            | 5 |
| 6.1 | Explode by token                   | 5 |
| 6.2 | Multiple Hashings DS               | 5 |
| 6.3 | Permute chars of string            | 6 |
| 6.4 | Longest common subsequence         | 6 |
| 6.5 | KMP                                | 6 |
| 7   | Flow                               | 6 |
| 8   | Miscellaneous                      | 6 |

|     |                  |   |
|-----|------------------|---|
| 8.1 | Bit Manipulation | 6 |
| 9   | Testing          | 8 |

## 1 Template

```

1 #include "bits/stdc++.h"
2 //assert(x>0) si falla da RTE
3 using namespace std;
4 #define endl '\n'
5 #define DBG(x) cerr<<#x<< "=" << (x) << endl;
6 #define RAYA cerr<<"===== "<<endl;
7 #define RAYAS cerr<<"..... "<<endl;
8 // #define DBG(x) ;
9 // #define RAYA ;
10 // #define RAYAS ;
11
12 //-----SOLBEGIN-----
13 int main() {
14     ios_base::sync_with_stdio(false); cout.tie(NULL); cin.tie(NULL);
15     int tC;
16
17     cin >> tC;
18     while (tC--) {
19
20     }
21 }
22
23 //-----EOSOLUTION-----

```

## 2 Data structures

### 2.1 Simplified DSU (Stolen from GGDem)

```

1 int uf[MAXN];
2 void uf_init(){memset(uf,-1,sizeof(uf));}
3 int uf_find(int x){return uf[x]<0?x:uf[x]=uf_find(uf[x]);}
4 bool uf_join(int x, int y){
5     x=uf_find(x);y=uf_find(y);
6     if(x==y)return false;
7     if(uf[x]>uf[y])swap(x,y);
8     uf[x]+=uf[y];uf[y]=x;
9     return true;
10 }

```

### 2.2 Disjoint Set Union

```

1 class disjSet {
2     int* sz;
3     int* par;
4 public:
5     int len;
6     disjSet(int tam){
7         sz = new int[tam + 4]();
8         par = new int[tam + 4]();
9         len = 0;
10        for(int i = 0; i<=tam; i++){
11            par[i] = i;
12            sz[i] = 1;
13            len++;
14        }
15    }
16    int finds(int el){
17        if (el == par[el]) return el;
18        return par[el] = finds(par[el]);
19    }
20    void unions(int a, int b){
21        a = finds(a);
22        b = finds(b);
23        if (a == b) return;
24        len--;
25        //se hace que el gde sea padre del pequeno
26        if (sz[a] > sz[b]) swap(a,b);
27        par[a] = b;
28        sz[b] += sz[a];
29    }
30    ~disjSet(){
31        delete[] size;
32        size = nullptr;
33        delete[] parent;
34        parent = nullptr;
35    }
36 };

```

### 2.3 Segment tree

```

1 //MAXN = 2^k, n = tam arreglo inicial
2 int stsize; long long int neut;int n;
3 long long int* st = new long long int[2*MAXN-1]();
4 long long int fst(long long int a, long long int b);

```

```

5 long long int build(int sti,int csize){
6     if(csize == 1) return st[sti];
7     return st[sti] = fst(build(sti*2+1,csize/2),build(sti*2+2,csize/2));
8 }
9 void innit(){
10    for(int i = 0; i<stsize; i++) st[i] = neut;
11    /*int d = 0;
12    for(int i = stsize-n; i<stsize && d<n; i++){
13        st[i] = arr[d];d++;
14    }*/
15    build(0,n);
16 }
17 void upd(int ind, long long int val){
18    ind = stsize-n+ind;
19    st[ind] = val;ind--;ind/=2;
20    while(true){
21        st[ind] = fst(st[ind*2+1],st[ind*2+2]);
22        ind--;
23        if(ind<0) break;
24        ind/=2;
25    }
26 }
27 long long int rqu(int l, int r,int sti, int ls, int rs){
28    if(l<=ls && rs<= r) return st[sti];
29    if(r<ls || l>rs) return neut;
30    int m = (rs+ls)/2;
31    return fst(rqu(l,r,sti*2+1,ls,m),rqu(l,r,sti*2+2,m+1,rs));
32 }
33 long long int query(int l, int r){
34    return rqu(l,r,0,0,n-1);
35 }
36 //uso, inicializa neut, determina n (asegurate que sea una potencia de
37 //2), define fst para determinar
38 //la opracion del segment tree

```

### 3 Graphs

## 4 Math

### 4.1 Identities

$$C_n = \frac{2(2n-1)}{n+1} C_{n-1}$$

$$C_n = \frac{1}{n+1} \binom{2n}{n}$$

$$C_n \sim \frac{4^n}{n^{3/2}\sqrt{\pi}}$$

$$\sigma(n) = O(\log(\log(n))) \text{ (number of divisors of } n)$$

$$F_{2n+1} = F_n^2 + F_{n+1}^2$$

$$F_{2n} = F_{n+1}^2 - F_{n-1}^2$$

$$\sum_{i=1}^n F_i = F_{n+2} - 1$$

$$F_{n+i}F_{n+j} - F_nF_{n+i+j} = (-1)^n F_i F_j$$

(Möbius Inv. Formula) Let  $g(n) = \sum_{d|n} f(d)$ , then  $f(n) = \sum_{d|n} d \cdot ng(d) \mu\left(\frac{n}{d}\right)$ .

### 4.2 Binary Exponentiation and modArith

```

1 long long int inf = 10000000007;
2 //suma (a+b)%m
3 //resta ((a-b)%m+m)%m
4 //mult (a*b)%m
5 long long binpow(long long b, long long e) {
6     long long res = 1; b%=inf;
7     while (e > 0) {
8         if (e & 1) res = (res * b)%inf;
9         b = (b * b)%inf;
10        e >>= 1;
11    }
12    return res;
13 }

```

### 4.3 Modular Inverse (dividir mod)

```

1 long long int inf = 10000000007;
2 long long int gcd(long long int a, long long int b, long long int& x,
3     long long int& y) {
4     x = 1, y = 0;
5     long long int x1 = 0, y1 = 1, a1 = a, b1 = b;
6     while (b1) {
7         long long int q = a1 / b1;
8         tie(x, x1) = make_tuple(x1, x - q * x1);
9         tie(y, y1) = make_tuple(y1, y - q * y1);
10        tie(a1, b1) = make_tuple(b1, a1 - q * b1);
11    }
12    return a1;
13 }
14 long long int modinverse(long long int b, long long int m){
15     long long int x,y;
16     long long int d = extEuclid(b,inf,x,y);
17     if(d!=1) return -1;

```

```

17     return ((x%inf)+inf)%inf;
18 }
    
```

#### 4.4 Sieve Of Eratosthenes

```

1 #define MAXN 10e6
2 class soe{
3 public:
4     bitset<MAXN> isPrime;
5     soe(){
6         for(int i = 3; i<MAXN; i++) isPrime[i] = (i%2);
7         isPrime[2] = 1;
8         for(int i = 3; i*i<MAXN; i+=2)
9             if(isPrime[i])
10                 for(int j = i*i; j<MAXN; j+=i)
11                     isPrime[j] = 0;
12     }
13 };
    
```

#### 4.5 Sieve-based Factorization

```

1 #define MAXN 10e6
2 class soe{
3 public:
4     int smolf[MAXN];
5     soe(){
6         for(int i = 2; i<MAXN; i++) smolf[i] = (i%2==0?2:i);
7
8         for(int i = 3; i*i<MAXN; i+=2)
9             if(smolf[i]==i)
10                 for(int j = i*i; j<MAXN; j+=i)
11                     smolf[j] = min(smolf[j],smolf[i]);
12     }
13 };
    
```

#### 4.6 Berlekamp Massey

```

1 typedef long long int ll;
2 //Obtiene recurrencia lineal dados los primeros elementos en O(n^2)
3 vector<ll> berlekampMassey(const vector<ll> &s) {
4     vector<ll> c;
5     vector<ll> oldC;
6     int f = -1;
7     for (int i=0; i<(int)s.size(); i++) {
    
```

```

8         ll delta = s[i];
9         for (int j=1; j<=(int)c.size(); j++) delta -= c[j-1] * s[i-j];
10        if (delta == 0) continue;
11        if (f == -1) {
12            c.resize(i + 1);
13            mt19937 rng(chrono::steady_clock::now().time_since_epoch().
14                count());
15            for (ll &x : c) x = rng();
16            f = i;
17        } else {
18            vector<ll> d = oldC;
19            for (ll &x : d) x = -x;
20            d.insert(d.begin(), 1);
21            ll df1 = 0;
22            for (int j=1; j<=(int)d.size(); j++) df1 += d[j-1] * s[f+1-j
23                ];
24            assert(df1 != 0);
25            ll coef = delta / df1;
26            for (ll &x : d) x *= coef;
27            vector<ll> zeros(i - f - 1);
28            zeros.insert(zeros.end(), d.begin(), d.end());
29            d = zeros;
30            vector<ll> temp = c;
31            c.resize(max(c.size(), d.size()));
32            for (int j=0; j<(int)d.size(); j++) c[j] += d[j];
33            if (i - (int) temp.size() > f - (int) oldC.size()) {oldC =
34                temp;f = i;}
35        }
36    }
37    return c;
38 }
    
```

#### 4.7 Modular Berlekamp Massey

```

1 typedef long long int ll;
2 long long int inf = 1000000007;
3 vector<ll> bermas(vector<ll> x){
4     vector<ll> ls,cur;
5     int lf,ld;
6     for(int i = 0; i<x.size(); i++){
7         long long int t = 0;
8         for(int j = 0; j<cur.size(); j++) t=(t+x[i-j-1]*(long long int)
9             cur[j])%inf;
    
```

```

9         if((t-x[i])%inf==0)continue;
10        if(cur.size()==0){cur.resize(i+1);lf=i;ld=(t-x[i])%inf;continue;
11        };}
12        long long int k = (x[i]-t)*powermod(ld,inf-2)%inf;
13        vector<ll>c(i-lf-1);c.push_back(k);
14        for(int j = 0; j<ls.size(); j++) c.push_back(-ls[j]*k%inf);
15        if(c.size()<cur.size()) c.resize(cur.size());
16        for(int j = 0; j<cur.size();j++) c[j]=(c[j]+cur[j])%inf;
17        if(i-lf+ls.size()>=cur.size())ls=cur,lf=i,ld=(t-x[i])%inf;
18        cur=c;
19    }
20    for(int i =0; i<cur.size(); i++) cur[i]=(cur[i]%inf+inf)%inf;
21    return cur;
22 }

```

## 5 Geometry

## 6 Strings

### 6.1 Explode by token

```

1 // #include <sstream>
2
3 vector<string> explode(string const& s, char delim) {
4     vector<string> result;
5     istringstream iss(s);
6     for (string token; getline(iss, token, delim); )
7     {
8         result.push_back(move(token));
9     }
10    return result;
11 }

```

### 6.2 Multiple Hashings DS

```

1 struct multhash{
2     unsigned long long int h1,h2,h3;
3     unsigned long long int alf[257];
4     bool operator < (multhash b) const { // override less than operator
5         if (h1 != b.h1) return h1 < other.h1;
6         if (h2 != b.h2) return h2 < other.h2;
7         return h3 < b.h3;
8     }

```

```

9     bool operator == (multhash b) const { // override equal operator
10        return (h1== b.h1 && h2== b.h2 && h3== b.h3)
11    }
12    public:
13        string s;
14        multhash(){
15            h1 = 0; h2 = 0;h3 = 0; s = "";
16            for(char l = 'a'; l<='z'; l++) alf[l] = l-'a'+1;
17        }
18        void innit(){
19            unsigned long long int inf,p,op;
20
21            inf = 666666555557777777;
22            p = 47;op = 47;
23            for(char l: s){
24                h1+=(p*alf[l])%inf;
25                p*=op;
26                p%=inf;
27            }
28
29            inf = 986143414027351997;
30            p = 53;op = 53;
31            for(char l: s){
32                h2+=(p*alf[l])%inf;
33                p*=op;
34                p%=inf;
35            }
36
37            inf = 909090909090909091;
38            p = 79;op = 79;
39            for(char l: s){
40                h3+=(p*alf[l])%inf;
41                p*=op;
42                p%=inf;
43            }
44        }
45    };
46    //VALORES POSIBLES DE INF, MIENTRAS MAS CERCANOS A 10^17 MEJOR
47    //666666555557777777
48    //986143414027351997
49    //974383618913296759
50    //973006384792642181
51    //953947941937929919

```

```
52 //9090909090909091
53 //VALORES PARA P, USAR PRIMOS MAYORES A |Alfabeto|
54 //31,47,53,61,79
```

### 6.3 Permute chars of string

```
1 void permute(string str){
2     // Sort the string in lexicographically
3     // ascennding order
4     sort(str.begin(), str.end());
5
6     // Keep printing next permutation while there
7     // is next permutation
8     do {
9         cout<<str<<endl;
10    } while (next_permutation(str.begin(), str.end()));
11 }
```

### 6.4 Longest common subsequence

```
1 //0(|te|*|pa|)
2 //cambiar score para otros problemas, str all match = +2, miss/ins/del =
   -1
3 //usar char que no este en el alfabeto para denotar del/ins
4 string te,pa;
5 long long int ninf = -10e13;
6 long long int score(char a, char b){
7     if(a=='*' || b=='*') return 0;
8     if(a==b) return 1;
9     return ninf;
10 }
11 long long int lcs(){
12     long long int** dp;te = "*" + te; pa = "*" + pa;
13     long long int res = 0;
14
15     dp = new long long int*[te.size()];
16     for(int i = 0; i<te.size(); i++) dp[i] = new long long int[pa.size()
17         ]();
18
19     for(int r = 1; r<te.size(); r++){
20         for(int c = 1; c<pa.size(); c++){
21             dp[r][c] = dp[r-1][c-1]+score(te[r],pa[c]);
22             dp[r][c] = max(dp[r][c-1]+score(te[r],'*'),dp[r][c]);
23             dp[r][c] = max(dp[r-1][c]+score('*',pa[c]),dp[r][c]);
```

```
23     }
24     }
25
26     return dp[te.size()-1][pa.size()-1];
27 }
```

### 6.5 KMP

```
1 string T,P;
2 int bt[MAXN];
3 //0(|Text|+|Pattern|)
4 void KMPpre(){
5     int i = 0,j = 0; bt[0] = -1;
6     while(i<P.size()){
7         while(j>=0 && P[i]!=P[(j>=0?j:0)]) j = bt[j];
8         i++;j++; bt[i] = j;
9     }
10 }
11 int kmp(){
12     int res =0, i = 0, j = 0;
13     while(i<T.size()){
14         while(j>=0 && T[i] != P[(j>=0?j:0)]) j = bt[j];
15         i++; j++;
16         if(j==P.size()){//match, do anything
17             res++;j = bt[j];
18         }
19     }
20     return res;
21 }
```

## 7 Flow

## 8 Miscellaneous

### 8.1 Bit Manipulation

```
1 #include "bits/stdc++.h"
2 using namespace std;
3 #define endl '\n'
4
5
6 int main() {
7     ios_base::sync_with_stdio(false); cout.tie(NULL); cin.tie(NULL);
```

```

8 //Se representan bitmasks de 30 a 62 bits
9 //usando signed int y signed long long int
10 //para evitar problemas con el complemento de dos
11 signed int a, b;
12 //para multiplicar un numero por dos solo es necesario aplicar un
13 // shifteo de sus bits a la izquierda
14 a = 1;
15 a= a << 3;
16 cout << a << endl;
17 //para dividir un numero entre dos es necesario aplicar un
18 //shifteo a la derecha
19 a = 32;
20 a = a >> 3;
21 cout << a << endl;
22 //para encender el bit n de a, solo hay que igualar a = a | pow(2,n-1)
23 //prende el tercer bit
24 a = 1;
25 b = 1 << 2;
26 a = a | b;
27 cout << a << endl;
28 //para apagar el bit n de a, solo hay que a &= ~pow(2,n-1)
29 //prende el tercer bit
30 a = 5;
31 b = 1 << 2;
32 a &= ~b;
33 cout << a << endl;
34 //para revisar si el bit n de a esta encendido
35 //revisa si el tercer bit esta encendido
36 a = 5;
37 b = 1 << 2;
38 a = a & b;
39 cout << (a?"SI":"NO") << endl;
40 //para volter el bit n de a, solo hay que igualar a = a ^ pow(2,n-1)
41 //apaga el tercer bit
42 a = 5;
43 b = 1 << 2;
44 a = a ^ b;
45 cout << a << endl;
46 //para obtener el bit menos significativo que esta encendido a& -a
47 a = 12;
48 cout << log2(a & ((-1) * a))+1 << endl;
49 //para prender todos los bits hasta n
50 a = (1<<4)-1;
    
```

```

51 cout << a << endl;
52 }
53 //-----EOSOLUTION-----

1 #include "bits/stdc++.h"
2 using namespace std;
3 #define endl '\n'
4 #pragma GCC optimize("O3")
5 #pragma GCC target("popcnt")
6
7 //no usar con visual c++
8 //solo con g++ like compilers
9 int main() {
10     ios_base::sync_with_stdio(false); cout.tie(NULL); cin.tie(NULL);
11     signed long long int a, b, n;
12     //Obtain the remainder (modulo) of a when it is divided by n (n is a
13         power of 2)
14     a = 15; n = 8-1;
15     a &= n;
16     cout << "a%n, a_u=15, n_u=2^3" << endl;
17     cout << a << endl;
18     //Apaga el bit menos significativo de a
19     a = 14;
20     b = (a & ((-1) * a));
21     a &= ~b;
22     cout << a << endl;
23     //enciende el ultimo cero de a
24     a = 9;
25     b = ~a;
26     b = (b & ((-1) * b));
27     a = a | b;
28     cout << a<<endl;
29     //contar bits encendidos en a
30     cout << __builtin_popcount(a)<<endl;
31     //chechar la paridad de a
32     cout << (__builtin_parity(a) ? "IMPAR" : "PAR") << endl;
33     //contar leading zeroes en a
34     cout << __builtin_clz(a)<<endl;
35     //contar 9,trailling zeroes en a
36     cout << __builtin_ctz(a)<<endl;
37 }
38 //-----EOSOLUTION-----
    
```

9 Testing

|  |
|--|
|  |
|--|