.

First to Penalty

-12

Contents

1   Template

```cpp
#include "bits/stdc++.h"
//assert(x>0) si falla da RTE
using namespace std;
#define endl '\n'
#define DBG(x) cerr<<#x<< "=" << (x) << endl;
#define RAYA cerr<<"====================="<<endl;
#define RAYAS cerr<<"......................"<<endl;
//#define DBG(x) ;
//#define RAYA ;
//#define RAYAS ;

//-------------------SOLBEGIN----------------------------
int main() {
  ios_base::sync_with_stdio(false); cout.tie(NULL); cin.tie(NULL);
  int tC;

  cin >> tC;
  while (tC--) {

  }

}
//-------------------EOSOLUTION---------------------------
```

2   Data structures

2.1   Simplified DSU (Stolen from GGDem)

```cpp
int uf[MAXN];
void uf_init(){memset(uf,-1,sizeof(uf));}
int uf_find(int x){return uf[x]<0?x:uf[x]=uf_find(uf[x]);}
bool uf_join(int x, int y){
  x=uf_find(x);y=uf_find(y);
  if(x==y)return false;
  if(uf[x]>uf[y])swap(x,y);
  uf[x]+=uf[y];uf[y]=x;
  return true;
}
```

2.2   Disjoint Set Union

```cpp
class disjSet {
  int* sz;
  int* par;
public:
  int len;
  disjSet(int tam){
      sz = new int[tam + 4]();
      par = new int[tam + 4]();
      len = 0;
      for(int i = 0; i<=tam; i++){
          par[i] = i;
          sz[i] = 1;
          len++;
      }
  }
  int finds(int el){
      if (el == par[el]) return el;
      return par[el] = finds(par[el]);
  }
  void unions(int a, int b){
      a = finds(a);
    b = finds(b);
      if (a == b) return;
      len--;
      //se hace que el gde sea padre del pequeno
      if (sz[a] > sz[b]) swap(a,b);
      par[a] = b;
      sz[b] += sz[a];
  }
  ~disjSet(){
      delete[] size;
      size = nullptr;
      delete[] parent;
      parent = nullptr;
  }
};
```

2.3   Segment tree

```cpp
//MAXN = 2^k, n = tam arreglo inicial
int stsize; long long int neut;int n;
long long int* st = new long long int[2*MAXN-1]();
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
        2), define fst para determinar
37  //la opracion del segment tree
```

3   Graphs

3.1   Graph Transversal

3.1.1   BFS

```
1   #define GS 400040
2   vector<int> graph[GS];
3   bitset <GS> vis;
4   //anchura O(V+E)
5   void dfs(int curr) {
6     queue<int> fringe;
7     fringe.push(curr);
8     while (fringe.size()) {
9       curr = fringe.front(); fringe.pop();
10      if (!vis[curr]) {
11        vis[curr] = 1;
12        for (int h : graph[curr]) fringe.push(h);
13      }
14    }
15  }
```

3.1.2   DFS

```
1   #define GS 400040
2   vector<int> graph[GS];
3   bitset <GS> vis;
4   //profundidad O(V+E)
5   void dfs(int curr) {
6     stack<int> fringe;
7     fringe.push(curr);
8     while (fringe.size()){
9       curr = fringe.top(); fringe.pop();
10      if (!vis[curr]) {
11        vis[curr] = 1;
12        for (int h : graph[curr]) fringe.push(h);
13      }
14    }
15  }
```

3.2   Topological Sort

```
1   #define GS 400040
2   vector<int> graph[GS];
3   bitset <GS> vis;
4   vector<int> topsort;
5   int e,n;
6   //profundidad
7   //O(N+E)
8   //Solo funciona con DAG's, no existe un top sort de un grafo Non-DAG
9   void todfs(int pa) {
10    vis[pa]=1;
```

```
11      for(int h: graph[pa]){if(!vis[h]){todfs(h);}}
12      topsort.push_back(pa);
13  }
14  void topologicalSort(){
15      vis.reset();
16      topsort.clear();
17      for(int i = 0; i<n; i++){if(!vis[i]){dfs(i);}}
18      reverse(topsort.begin(),topsort.end());
19  }
```

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 $f(n) = \sum d \mid n g(d)\mu\left(\frac{n}{d}\right)$.

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,
        long long int& y) {
3       x = 1, y = 0;
4       long long int x1 = 0, y1 = 1, a1 = a, b1 = b;
5       while (b1) {
6           long long int q = a1 / b1;
7           tie(x, x1) = make_tuple(x1, x - q * x1);
8           tie(y, y1) = make_tuple(y1, y - q * y1);
9           tie(a1, b1) = make_tuple(b1, a1 - q * b1);
10      }
11      return a1;
12  }
13  long long int modinverse(long long int b, long long int m){
14      long long int x,y;
15      long long int d = extEuclid(b,inf,x,y);
16      if(d!=1) return -1;
17      return ((x%inf)+inf)%inf;
18  }
```

4.4   Modular Binomial Coeficient and Permutations

```
1   long long int inf = 10000000007;
2   //cat[n] = bincoef(2*n,n)/(n+1), cat[0] = 1
3   class binCoef{
4       long long int lim;
5       long long int* fact;
6   public:
7       binCoef(long long int l){
8           lim = l; fact = new long long int[l+1];fact[0]= 1;
9           for(long long int i = 1; i<=l; i++) fact[i] = (fact[i-1]*i)%inf;
10      }
11      //perm = (fact[n] * modinverse(fac[n-k],inf)%inf;
12      long long int query(long long int n, long long int k){
13          if(n<k) return 0;
14          return (fact[n] * modinverse((fac[n-k]*fact[k])%inf,inf))%inf;
15      }
16  };
```

4.5   Non-Mod Binomial Coeficient and Permutations

```
1   //Solo usar con n<=20
2   //cat[n] = bincoef(2*n,n)/(n+1), cat[0] = 1
3   unsigned long long int bincoef(unsigned long long int n, unsigned long
        long int k){
```

```
4      if(n<k) return 0;
5      unsigned long long int num  = 1, den= 1;
6      for(unsigned long long int i = (n-k)+1; i<=n; i++) num*=i;
7      for(unsigned long long int i = 2; i<=k; i++) den*=i;
8      //perm = return num;
9      return num/den;
10 }
```

4.6   Modular Catalan Numbers

```
1  long long int inf = 10000000007;
2  class catalan{
3      long long int* cat; long long int lim
4  public:
5      catalan(long long int l){
6          lim = l; cat = new long long int[l+10];cat[0] = 1;
7          for(long long int i = 0;i<=l; i++) cat[i+1] = (((((4LL*i+2)%inf)
                *cat[i])%inf) *modinverse(n+2))%inf;
8      }
9      long long int query(long long int n){ return cat[n];}
10 };
```

4.7   Ceil Fraccionario

```
1  long long int techo(long long int num, long long int den){ return (num+
      den-1)/den;}
```

4.8   Numeros de Fibonacci

```
1  //en caso de ser usados mod un m pequeno
2  //recordar que los numeros de fibonacci se repiten por lo menos cada m^2
3  //O(n)
4  unsigned long long int fib(int n){
5    unsigned long long int a = 1,b = 1,aux;
6    if(n<=2){
7      return 1;
8    }
9    for(int i = 3; i<=n; i++){
10     aux = a+b;
11     a = b;
12     b = aux;
13   }
14   return b;
15 }
```

```
1  const long long int inf = 1000000007;
2  unordered_map<long long int,long long int> Fib;
3  //O(log n) :DD
4  long long int fib(long long int n)
5  {
6      if(n<2) return 1;
7      if(Fib.find(n) != Fib.end()) return Fib[n];
8      Fib[n] = (fib((n+1) / 2)*fib(n/2) + fib((n-1) / 2)*fib((n-2) / 2)) %
            inf;
9      return Fib[n];
10 }
```

4.9   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 };
```

4.10   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 };
```

4.11 Berlekamp Massey

```cpp
typedef long long int ll;
//Obtiene recurrencia lineal dados los primeros elementos en O(n^2)
vector<ll> berlekampMassey(const vector<ll> &s) {
    vector<ll> c;
    vector<ll> oldC;
    int f = -1;
    for (int i=0; i<(int)s.size(); i++) {
        ll delta = s[i];
        for (int j=1; j<=(int)c.size(); j++) delta -= c[j-1] * s[i-j];
        if (delta == 0) continue;
        if (f == -1) {
            c.resize(i + 1);
            mt19937 rng(chrono::steady_clock::now().time_since_epoch().
                count());
            for (ll &x : c) x = rng();
            f = i;
        } else {
            vector<ll> d = oldC;
            for (ll &x : d) x = -x;
            d.insert(d.begin(), 1);
            ll df1 = 0;
            for (int j=1; j<=(int)d.size(); j++) df1 += d[j-1] * s[f+1-j
                ];
            assert(df1 != 0);
            ll coef = delta / df1;
            for (ll &x : d) x *= coef;
            vector<ll> zeros(i - f - 1);
            zeros.insert(zeros.end(), d.begin(), d.end());
            d = zeros;
            vector<ll> temp = c;
            c.resize(max(c.size(), d.size()));
            for (int j=0; j<(int)d.size(); j++) c[j] += d[j];
            if (i - (int) temp.size() > f - (int) oldC.size()) {oldC =
                temp;f = i;}
        }
    }
    return c;
}
```

4.12 Modular Berlekamp Massey

```cpp
typedef long long int ll;
long long int inf = 1000000007;
vector<ll> bermas(vector<ll> x){
    vector<ll> ls,cur;
    int lf,ld;
    for(int i = 0; i<x.size(); i++){
        long long int t = 0;
        for(int j = 0; j<cur.size(); j++) t=(t+x[i-j-1]*(long long int)
            cur[j])%inf;
        if((t-x[i])%inf==0)continue;
        if(cur.size()==0){cur.resize(i+1);lf=i;ld=(t-x[i])%inf;continue
            ;}
        long long int k = (x[i]-t)*powermod(ld,inf-2)%inf;
        vector<ll>c(i-lf-1);c.push_back(k);
        for(int j = 0; j<ls.size(); j++) c.push_back(-ls[j]*k%inf);
        if(c.size()<cur.size()) c.resize(cur.size());
        for(int j = 0; j<cur.size();j++) c[j]=(c[j]+cur[j])%inf;
        if(i-lf+ls.size()>=cur.size())ls=cur,lf=i,ld=(t-x[i])%inf;
            cur=c;
    }
    for(int i =0; i<cur.size(); i++) cur[i]=(cur[i]%inf+inf)%inf;
    return cur;
}
```

5 Geometry

6 Strings

6.1 Explode by token

```cpp
//#include <sstream>

vector<string> explode(string const& s, char delim) {
  vector<string> result;
  istringstream iss(s);
  for (string token; getline(iss, token, delim); )
  {
    result.push_back(move(token));
  }
  return result;
}
```

6.2 Multiple Hashings DS

```
1  #include "bits/stdc++.h"
2  using namespace std;
3  struct multhash{
4      unsigned long long int h1,h2;
5      unsigned long long int alf[257];
6      bool operator < (multhash b) const {
7      if (h1 != b.h1) return h1 < b.h1;
8      return h2 < b.h2;
9      }
10     bool operator == (multhash b) const { return (h1== b.h1 && h2== b.h2)
           ;}
11     bool operator != (multhash b) const { return !(h1== b.h1 && h2== b.h2)
           ;}
12  public:
13      string s;
14      multhash(){
15          h1 = 0; h2 = 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 = 999727999;
22          p = 325255434;op = 325255434;
23          for(char l: s){
24              h1+=(p*alf[l])%inf;
25              p*=op;
26              p%=inf;
27          }
28
29          inf = 1070777777;
30          p = 10018302;op = 10018302;
31          for(char l: s){
32              h2+=(p*alf[l])%inf;
33              p*=op;
34              p%=inf;
35          }
36      }
37  };
38  //VALORES ALTERNATIVOS DE INF, LOG 17
39  //666666555557777777
40  //986143414027351997
41  //974383618913296759
42  //973006384792642181
43  //953947941937929919
44  //909090909090909091
45  //VALORES PARA P, USAR PRIMOS MAYORES A |Alfabeto|
46  //31,47,53,61,79
47  int main(){
48      return 0;
49  }
```

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  //O(|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      for(int r = 1; r<te.size(); r++){
19          for(int c = 1; c<pa.size(); c++){
20              dp[r][c] = dp[r-1][c-1]+score(te[r],pa[c]);
21              dp[r][c] = max(dp[r][c-1]+score(te[r],'*'),dp[r][c]);
22              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  //O(|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;
```

```cpp
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------------------------------
```

```cpp
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
          power of 2)
13    a = 15; n = 8-1;
14    a &= n;
15    cout << "a%n,␣a␣=␣15,␣n␣=␣2^3" << endl;
16    cout << a << endl;
17    //Apaga el bit menos significativo de a
18    a = 14;
19    b = (a & ((-1) * a));
20    a &= ~b;
21    cout << a << endl;
22    //enciende el ultimo cero de a
23    a = 9;
24    b = ~a;
```

```cpp
25    b = (b & ((-1) * b));
26    a = a | b;
27    cout << a<<endl;
28    //contar bits encendidos en a
29    cout << __builtin_popcount(a)<<endl;
30    //checar la paridad de a
31    cout << (__builtin_parity(a) ? "IMPAR" : "PAR") << endl;
32    //contar leading zeroes en a
33    cout << __builtin_clz(a)<<endl;
34    //contar 9,trailling zeroes en a
35    cout << __builtin_ctz(a)<<endl;
36  }
37  //------------------EOSOLUTION------------------------------
```

9   Testing