.

First to Penalty

Contents

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  #define MAXN 262160
3  int stsize; long long int neut;int n;
4  long long int* st = new long long int[2*MAXN-1]();
```

```
5   long long int fst(long long int a, long long int b);
6   long long int build(int sti,int csize){
7       if(csize == 1) return st[sti];
8       return st[sti] = fst(build(sti*2+1,csize/2),build(sti*2+2,csize/2));
9   }
10  void innit(){
11      for(int i = 0; i<stsize; i++) st[i] = neut;
12      /*int d = 0;
13      for(int i = stsize-n; i<stsize && d<n; i++){
14          st[i] = arr[d];d++;
15      }*/
16      build(0,n);
17  }
18  void upd(int ind, long long int val){
19      ind = stsize-n+ind;
20      st[ind] = val;ind--;ind/=2;
21      while(true){
22          st[ind] = fst(st[ind*2+1],st[ind*2+2]);
23          ind--;
24          if(ind<0) break;
25          ind/=2;
26      }
27  }
28  long long int rqu(int l, int r,int sti, int ls, int rs){
29      if(l<=ls && rs<= r) return st[sti];
30      if(r<ls || l>rs) return neut;
31      int m = (rs+ls)/2;
32      return fst(rqu(l,r,sti*2+1,ls,m),rqu(l,r,sti*2+2,m+1,rs));
33  }
34  long long int query(int l, int r){
35      return rqu(l,r,0,0,n-1);
36  }
37  //uso, inicializa neut, n = primera potencia de 2 >= n del problema,
        stsize = 2*n-1
38  //llena arr de neutros hasta que su tam sea el nuevo n
39  //DEFINE LA FUNCION fst
```

2.4   Segment tree Lazy

```
1   //MAXN = 2^k, n = tam arreglo inicial
2   #define MAXN 262160
3   vector<int> arr;
4   int stsize; long long int neut;int n;
```

```
5   long long int* st = new long long int[2*MAXN-1]();
6   long long int* pendientes = new long long int[2*MAXN-1]();
7   long long int fst(long long int a, long long int b){return a+b;}
8   long long int build(int sti,int csize){
9       if(csize == 1) return st[sti];
10      return st[sti] = fst(build(sti*2+1,csize/2),build(sti*2+2,csize/2));
11  }
12  bool hasChildren(int sti){sti*=2;sti++;sti++;return sti<stsize;}
13  void innit(){
14      for(int i = 0; i<stsize; i++) st[i] = neut;
15      int d = 0;
16      for(int i = stsize-n; i<stsize && d<n; i++) {st[i] = arr[d];d++;}
17      build(0,n);
18  }
19  void updrec(int l,int r, int sl, int sr,int sti, long long int val){
20      if(sr<l || r< sl) return;
21      if(l<= sl && sr <=r){
22          st[sti] += val*(sr-sl+1);
23          if(hasChildren(sti)){pendientes[sti*2+1]+=val;pendientes[sti
                *2+2]+=val;}
24          return;
25      }
26
27      int sm = (sl+sr)/2;
28      updrec(l,r,sl,sm,sti*2+1,val);
29      updrec(l,r,sm+1,sr,sti*2+2,val);
30      st[sti] = fst(st[sti*2+1],st[sti*2+2]);
31  }
32  void upd(int l, int r, long long int val){updrec(l,r,0,n-1,0,val);}
33
34  long long int rqu(int l, int r,int sti, int ls, int rs){
35      if(r<ls || l>rs) return neut;
36      if(l<=ls && rs<= r){
37          return st[sti]+pendientes[sti]*(rs-ls+1);
38      }
39
40      st[sti] += pendientes[sti]*(rs-ls+1);
41      if(hasChildren(sti)){pendientes[sti*2+1]+=pendientes[sti];pendientes
            [sti*2+2]+=pendientes[sti];}
42      pendientes[sti] = 0;
43
44      int m = (rs+ls)/2;
45      return fst(rqu(l,r,sti*2+1,ls,m),rqu(l,r,sti*2+2,m+1,rs));
```

```
46  }
47  long long int query(int l, int r){
48      return rqu(l,r,0,0,n-1);
49  }
50  //uso, inicializa neut, n = primera potencia de 2 >= n del problema,
        stsize = 2*n-1
51  //llena arr de neutros hasta que su tam sea el nuevo n
52  //DEFINE LA FUNCION fst
```

2.5   Trie

```
1   struct triver {
2       char alphabet;
3       bool ter;
4       vector<triver*> child;
5       triver(char a): alphabet(a) { child.assign(26, NULL); ter = false; }
6   };
7   class trie{
8   private:
9       triver* root;
10  public:
11      trie() { root = new triver('!');}
12      void insert(string s){
13          triver* curr = root;
14          for(char l: s){
15              if(curr->child[l-'A'] == NULL) curr->child[l-'A'] = new
                    triver(l);
16              curr = curr->child[l-'A'];
17          }
18          curr->ter = true;
19      }
20      bool search(string s){
21          triver* curr = root;
22          for(char l: s){
23              if(curr == NULL) break;
24              curr = curr->child[l-'A'];
25          }
26          if(curr == NULL) return false;
27          return curr->ter;
28      }
29  };
```

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  }
```

3.3   APSP: Floyd Warshall

```
1   #define GS 1000
2   #define INF 100000000
3   //destino, costo
4   int graph[GS][GS];
5   //All Pairs Dist
6   int dist[GS][GS];
7   //Toma en cuenta nodos [0-tam] inclusivo, modificar de acuerdo a las
        necesidades
8   //Ten cuidado con el valor que le pones a INF, puede provocar overflows
        o puede no ser lo suficientemente grande.
9   void Floyd_Warshall(int tam){
10    for(int i = 0; i<=tam; i++)
11      for(int f = 0; f<=tam; f++)
12        dist[i][f] = INF;
13
14    for(int i = 0; i<=tam; i++)
15      for(int f = 0; f<=tam; f++)
16        dist[i][f] = graph[i][f];
17
18    //para reconstruir el camino solo basta con guardar intermedio como
          el padre de ini si el cambio se hizo, -1 otherwise
19    for(int intermedio = 0; intermedio<=tam; intermedio++)
20      for(int ini = 0;  ini<=tam; ini++)
21        for(int fin = 0; fin<=tam; fin++)
22          dist[ini][fin] = min(dist[ini][fin],dist[ini][intermedio
                ]+dist[intermedio][fin]);
23  }
```

3.4   SSSP

3.4.1   Lazy Dijkstra

```
1   #define GS 1000
2   #define INF 100000000
3   //destino, costo
4   vector<pair<int,int>> graph[GS];
5   int dist[GS];
6   void dijkstra(int origen,int tam){
7     for(int i = 0; i<=tam; i++){
8       dist[i] = INF;
9     }
10    priority_queue<pair<int,int>,vector<pair<int,int>>, greater<pair<int
          ,int>>> pq;
11    pair<int,int> curr;
12
13    pq.push(make_pair(0,origen));
14
15    while(pq.size()){
16      curr = pq.top();pq.pop();
17      if(curr.first >= dist[curr.second]) continue;
18
19      dist[curr.second] = curr.first;
20      for(pair<int,int> h: graph[curr.second]){
21        if((h.second+curr.first)<dist[h.first]) pq.push({h.second+
              curr.first,h.first});
22      }
23    }
24  }
25  //Esta es la implementacion huevona
26  //Resuelve Single Source Shortest Paths con aristas positivas
27  //Como es la lazy implementation, si funciona con edges negativos
        siempre y cuando no hayan ciclos negativos
28  //Si hay ciclos negativos se va atascar en un ciclo infinito
29  //Si no los hay puede que funcione en O((V+E)log(V)) o puede que se
        exponencial, si no jala prueba BellmanFord
```

3.4.2   Bellman-Ford

```cpp
//esta es la implementacion huevona
#define GS 1000
//cuidado con overflows!!
#define INF 100000000
#define NINF -100000000
//destino, costo
vector<pair<int,int>> graph[GS];
int dist[GS];
struct edge{
    int from,to,cost;
};
//Corre en O(VE)
void bellmanFord(int origen,int tam){
    for(int i = 0; i<=tam; i++){
        dist[i] = INF;
    }
    dist[origen] = 0;
    edge aux;
    vector<edge> aristas;
    bool optimal;

    for(int i = 0; i<=tam; i++){
        for(pair<int,int> h: graph[i]){
            aux.from = i; aux.to  = h.first;aux.cost = h.second;
            aristas.push_back(aux);
        }
    }

    //Si se relajan todos las aristas V-1 veces en un orden arbitrario
    //Se asegura que la distancia optima para cada vertice sera
        alcanzada
    for(int i = 0; i<tam && !optimal; i++){
        optimal = true;
        for(edge elem: aristas){
            if(dist[elem.from] + elem.cost < dist[elem.to]){
                dist[elem.to] = dist[elem.from] + elem.cost;
                //si algun vertice fue actualizado significa que puede
                    que
                //las distancias aun no sean optimas
                optimal = false;
            }
        }
    }
```

```cpp

    //Se corre de nuevo para asegurar encontrar todos los ciclos
        negativos
    for(int i = 0; i<tam && !optimal; i++){
        optimal = true;
        for(edge elem: aristas){
            if(dist[elem.from] + elem.cost < dist[elem.to]){
                //Si aun despues de correr V-1 veces se puede actualizar
                //Significa que esta en un ciclo negativo
                dist[elem.to] = NINF;
                //si algun vertice fue actualizado significa que puede
                    que
                //las distancias aun no sean optimas
                optimal = false;
            }
        }
    }
}
```

3.5   Strongly Connected Components: Kosaraju

```cpp
#define GS 2010
vector<int> graph[GS];
vector<int> graphI[GS];
vector<int> orden;
bitset<GS> vis;

void invertirGrafo(int n){
    for(int p = 1;p<= n; p++)
        for(int h: graph[p])graphI[h].push_back(p);
}
void obtOrd(int p,int n){
    vis[p] = 1;
    for(int h: graph[p]){
        if(!vis[h] && h<=n) obtOrd(h,n);
    }
    orden.push_back(p);
}
int findSCC(int n){
    int res = 0;
    invertirGrafo(n);
    orden.clear();
```

```
22    for(int i = 1; i<=n; i++) vis[i] =0;
23    for(int i = 1; i<=n; i++) if(!vis[i]) obtOrd(i,n);
24    reverse(orden.begin(),orden.end());
25    //cuenta los connected components
26    //vector<int> lscc;
27    stack<int> fringe;
28    int curr;
29    for(int i = 1; i<=n; i++) vis[i] =0;
30    for(int i: orden){
31        //lscc.clear();
32        if(!vis[i]){
33            fringe.push(i);
34            while (fringe.size()){
35                curr = fringe.top();fringe.pop();
36                //lscc.push_back(curr);
37                if (!vis[curr]) {
38                    vis[curr] = 1;
39                    for (int h : graphI[curr]) fringe.push(h);
40                }
41            }
42            res++;
43        }
44        //hacer lo que sea con lcss
45    }
46    return res;
47 }
48
49 //OJO esto solo jala con directed graphs
50 //por definicion todas las undirected graphs tienen un solo SCC
51 //NOTAR QUE LOS GRAFOS QUE USA CUMPLEN CON: 0<=VERTICE<=n
```

3.6 Articulation Points and Bridges: ModTarjan

```
1  #define GS 50
2  vector<int> graph[GS];
3  bitset<GS> vis, isArtic;
4  vector<int> padre;
5  //id por tiempo, menor id accesible
6  //ya sea por descendientes o por back edges
7  vector<int> tId,lId;
8  //cantidad de hijos que tiene en el bfs spanning tree
9  int rootChildren;
10 int cnt;
11 int dfsRoot;
12 void findAP_B(int p){
13     cnt++;vis[p] = 1;tId[p] = cnt;lId[p] = tId[p];
14
15     for(int hijo: graph[p]){
16         if(!vis[hijo]){
17             padre[hijo] = p;
18             if(p == dfsRoot) rootChildren++;
19
20             findAP_B(hijo);
21
22             //esto significa que ni por un back edge el hijo accede al
                   padre
23             //por lo que si el padre fuese eliminado el hijo quedaria
                   aislado
24             if(lId[hijo] >= tId[p]) isArtic[p] = 1;
25             if(lId[hijo] > tId[p]){
26                 //esto significa que si se eliminase el camino de padre
                       ->hijo
27                 //se lograria desconectar el grafo, aka bridge
28             }
29             lId[p] = min(lId[p],lId[hijo]);
30         }else{
31             //si hay un ciclo indirecto, actualiza el valor para el
                   padre
32             if(hijo != padre[p]) lId[p] = min(lId[p],tId[hijo]);
33         }
34     }
35 }
36 //OJO esto solo jala con Undirected graphs
37 /*
38     MAIN
39     for(int i = 0; i<n; i++){
40         if(!vis[i]){
41             rootChildren = 0;
42             dfsRoot = i;
43             findAP_B(i);
44             //el algoritmo no puede detectar si el nodo que lo origino
45             //es un articulation point, por lo que queda checar si
46             //en el spanning tree que genero tiene mas de un solo hijo
47             isArtic[i] = (rootChildren>1?1:0);
48         }
49     }
```

```
50  */
```

4   Math

4.1   Identities

$$C_n = \frac{2(2n-1)}{n+1}C_{n-1}$$
$$C_n = \frac{1}{n+1}\binom{2n}{n}$$
$$C_n \sim \frac{4^n}{n^{3/2}\sqrt{\pi}}$$
$$\sigma(n) = O(\log(\log(n))) \text{ (number of divisors of } n)$$
$$F_{2n+1} = F_n^2 + F_{n+1}^2$$
$$F_{2n} = F_{n+1}^2 - F_{n-1}^2$$
$$\sum_{i=1}^{n} F_i = F_{n+2} - 1$$
$$F_{n+i}F_{n+j} - F_n F_{n+i+j} = (-1)^n F_i F_j$$

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

Permutaciones objetos repetidos $P(n,k) = \frac{P(n,k)}{n_1!n_2!...}$

Separadores, Ecuaciones lineares a variables $= b$ $\left(\binom{a}{b}\right) = \binom{a+b-1}{b} = \binom{a+b-1}{a-1}$

4.2   Binary Exponentiation and modArith

```cpp
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)

```cpp
1  long long int inf = 1000000007;
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) {
```

```cpp
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 = gcd(b,inf,x,y);
16     if(d!=1) return -1;
17     return ((x%inf)+inf)%inf;
18 }
```

4.4   Modular Binomial Coeficient and Permutations

```cpp
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

```cpp
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   Cycle Finding

```
1  void cyclef(long long int sem){
2      long long int hare = f(sem),tort=f(sem);hare = f(hare);
```

```
3        //liebre avanza dos pasos, tortuga solo uno
4        while(hare!=tort){
5            tort = f(tort); hare = f(f(hare));
6        }
7        //Se detiene en el inicio del ciclo
8        tort = sem;
9        while(hare!=tort){
10            tort = f(tort); hare = f(hare);
11        }
12
13        int len = 1;
14        tort = f(sem);
15        while(hare!=tort){
16            tort=f(tort);
17            len++;
18        }
19    }
```

4.12   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().
                   count());
14              for (ll &x : c) x = rng();
15              f = i;
16          } else {
17              vector<ll> d = oldC;
18              for (ll &x : d) x = -x;
19              d.insert(d.begin(), 1);
20              ll df1 = 0;
21              for (int j=1; j<=(int)d.size(); j++) df1 += d[j-1] * s[f+1-j
                   ];
```

```
22              assert(df1 != 0);
23              ll coef = delta / df1;
24              for (ll &x : d) x *= coef;
25              vector<ll> zeros(i - f - 1);
26              zeros.insert(zeros.end(), d.begin(), d.end());
27              d = zeros;
28              vector<ll> temp = c;
29              c.resize(max(c.size(), d.size()));
30              for (int j=0; j<(int)d.size(); j++) c[j] += d[j];
31              if (i - (int) temp.size() > f - (int) oldC.size()) {oldC =
                   temp;f = i;}
32          }
33      }
34      return c;
35 }
```

4.13   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)
                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          long long int k = (x[i]-t)*powermod(ld,inf-2)%inf;
12          vector<ll>c(i-lf-1);c.push_back(k);
13          for(int j = 0; j<ls.size(); j++) c.push_back(-ls[j]*k%inf);
14          if(c.size()<cur.size()) c.resize(cur.size());
15          for(int j = 0; j<cur.size();j++) c[j]=(c[j]+cur[j])%inf;
16          if(i-lf+ls.size()>=cur.size())ls=cur,lf=i,ld=(t-x[i])%inf;
17          cur=c;
18      }
19      for(int i =0; i<cur.size(); i++) cur[i]=(cur[i]%inf+inf)%inf;
20      return cur;
21 }
```

4.14   Matrix exponentiation

```
1   typedef vector<vector<long long int>> Matrix;
2   long long int inf = 1000000007;
3   Matrix ones(int n) {
4     Matrix r(n,vector<long long int>(n));
5     for(int i= 0; i<n; i++){
6           r[i][i]=1;
7       }
8     return r;
9   }
10  Matrix operator*(Matrix &a, Matrix &b) {
11    int n=a.size(),m=b[0].size(),z=a[0].size();
12    Matrix r(n,vector<long long int>(m));
13    for(int i=0; i<n; i++){
14          for(int j=0; j<m; j++){
15              for(int k=0;k<z; k++){
16                  r[i][j]+=((a[i][k]%inf)*(b[k][j]%inf))%inf;
17                  r[i][j]%=inf;}}}
18    return r;
19  }
20  Matrix be(Matrix b, long long int e) {
21    Matrix r=ones(b.size());
22    while(e){if(e&1LL)r=r*b;b=b*b;e/=2;}
23    return r;
24  }
25
26  //Matrix mat(n,vector<long long int>(n));
```

4.15   Ecuaciones Diofantinas

```
1   long long int gcd(long long int a, long long int b, long long int& x,
        long long int& y) {
2     x = 1, y = 0;
3     long long int x1 = 0, y1 = 1, a1 = a, b1 = b;
4     while (b1) {
5       int q = a1 / b1;
6       tie(x, x1) = make_tuple(x1, x - q * x1);
7       tie(y, y1) = make_tuple(y1, y - q * y1);
8       tie(a1, b1) = make_tuple(b1, a1 - q * b1);
9     }
10    return a1;
11  }
12  long long int d;
13  bool findAnySol(long long int a, long long int& x, long long int b, long
```

```
    long int& y, long long int c) {
14    long long int g = gcd(abs(a), abs(b), x, y);
15    if (c % g != 0) return false;
16    x *= c;
17    y *= c;
18    x /= g;
19    y /= g;
20    d = c / g;
21    if (a < 0) x = -x;
22
23    if (b < 0) y = -y;
24    return true;
25  }
26  //-------------------SOLBEGIN-----------------------------
27  int main() {
28    ios_base::sync_with_stdio(false); cout.tie(NULL); cin.tie(NULL);
29    long long int m, a, k, n;
30    long long int f, h,res;
31    //estira en n, y despues cada m
32    //estira en k+a, y despues cada a
33    cin >> n >> m >> a >> k;
34    while (n != 0 && m != 0 && a != 0 && k != 0) {
35      m = -m;
36      if (!findAnySol(m, f, a, h, k + a - n)) {
37        cout << "Impossible" << endl;
38      }else {
39        res = f * m+n;
40        while (res > 0) res -= m * d;
41        while (res < 0) res += m * d;
42
43        cout << res << endl;
44      }
45      cin >> n >> m >> a >> k;
46    }
47
48  }
49  //-------------------EOSOLUTION-----------------------------
```

4.16   FFT, Stolen from GGDem

```
1   // SPOJ VFMUL - AC
2   // http://www.spoj.com/problems/VFMUL/
3   #include <bits/stdc++.h>
```

```
4    #define fst first
5    #define snd second
6    #define fore(i,a,b) for(int i=a,ThxDem=b;i<ThxDem;++i)
7    #define pb push_back
8    #define ALL(s) s.begin(),s.end()
9    #define FIN ios_base::sync_with_stdio(0);cin.tie(0);cout.tie(0)
10   #define SZ(s) int(s.size())
11   using namespace std;
12   typedef long long ll;
13   typedef pair<int,int> ii;
14
15   // MAXN must be power of 2 !!
16   // MOD-1 needs to be a multiple of MAXN !!
17   // big mod and primitive root for NTT:
18   const int MOD=998244353,RT=3,MAXN=1<<20;
19   typedef vector<int> poly;
20   // FFT
21   struct CD {
22     double r,i;
23     CD(double r=0, double i=0):r(r),i(i){}
24     double real()const{return r;}
25     void operator/=(const int c){r/=c, i/=c;}
26   };
27   CD operator*(const CD& a, const CD& b){
28     return CD(a.r*b.r-a.i*b.i,a.r*b.i+a.i*b.r);}
29   CD operator+(const CD& a, const CD& b){return CD(a.r+b.r,a.i+b.i);}
30   CD operator-(const CD& a, const CD& b){return CD(a.r-b.r,a.i-b.i);}
31   const double pi=acos(-1.0);
32   // NTT
33   /*
34   struct CD {
35     int x;
36     CD(int x):x(x){}
37     CD(){}
38     int get()const{return x;}
39   };
40   CD operator*(const CD& a, const CD& b){return CD(mulmod(a.x,b.x));}
41   CD operator+(const CD& a, const CD& b){return CD(addmod(a.x,b.x));}
42   CD operator-(const CD& a, const CD& b){return CD(submod(a.x,b.x));}
43   vector<int> rts(MAXN+9,-1);
44   CD root(int n, bool inv){
45     int r=rts[n]<0?rts[n]=pm(RT,(MOD-1)/n):rts[n];
46     return CD(inv?pm(r,MOD-2):r);
47   }
48   */
49   CD cp1[MAXN+9],cp2[MAXN+9];
50   int R[MAXN+9];
51   void dft(CD* a, int n, bool inv){
52     fore(i,0,n)if(R[i]<i)swap(a[R[i]],a[i]);
53     for(int m=2;m<=n;m*=2){
54       double z=2*pi/m*(inv?-1:1); // FFT
55       CD wi=CD(cos(z),sin(z)); // FFT
56       // CD wi=root(m,inv); // NTT
57       for(int j=0;j<n;j+=m){
58         CD w(1);
59         for(int k=j,k2=j+m/2;k2<j+m;k++,k2++){
60           CD u=a[k];CD v=a[k2]*w;a[k]=u+v;a[k2]=u-v;w=w*wi;
61         }
62       }
63     }
64     if(inv)fore(i,0,n)a[i]/=n; // FFT
65     //if(inv){ // NTT
66     //  CD z(pm(n,MOD-2)); // pm: modular exponentiation
67     //  fore(i,0,n)a[i]=a[i]*z;
68     //}
69   }
70   poly multiply(poly& p1, poly& p2){
71     int n=p1.size()+p2.size()+1;
72     int m=1,cnt=0;
73     while(m<=n)m+=m,cnt++;
74     fore(i,0,m){R[i]=0;fore(j,0,cnt)R[i]=(R[i]<<1)|((i>>j)&1);}
75     fore(i,0,m)cp1[i]=0,cp2[i]=0;
76     fore(i,0,p1.size())cp1[i]=p1[i];
77     fore(i,0,p2.size())cp2[i]=p2[i];
78     dft(cp1,m,false);dft(cp2,m,false);
79     fore(i,0,m)cp1[i]=cp1[i]*cp2[i];
80     dft(cp1,m,true);
81     poly res;
82     n-=2;
83     fore(i,0,n)res.pb((ll)floor(cp1[i].real()+0.5)); // FFT
84     //fore(i,0,n)res.pb(cp1[i].x); // NTT
85     return res;
86   }
87
88   char s[MAXN],t[MAXN],r[MAXN];
89
```

```
90  int main(){
91    int tn;
92    scanf("%d",&tn);
93    while(tn--){
94      vector<int> a,b,c;
95      scanf("%s%s",s,t);
96      for(int i=0;s[i];++i)a.pb(s[i]-'0');reverse(a.begin(),a.end());
97      for(int i=0;t[i];++i)b.pb(t[i]-'0');reverse(b.begin(),b.end());
98      c=multiply(a,b);
99      while(!c.empty()&&!c.back())c.pop_back();
100     if(c.empty()){puts("0");continue;}
101     int n=0;
102     ll x=0;
103     fore(i,0,c.size()){
104       x+=c[i];
105       r[n++]=x%10;
106       x/=10;
107     }
108     while(x){
109       r[n++]=x%10;
110       x/=10;
111     }
112     reverse(r,r+n);
113     bool p=false;
114     fore(i,0,n){
115       putchar(r[i]+'0');
116     }
117     puts("");
118   }
119   return 0;
120 }
```

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;
3    unsigned long long int alf[257];
4    bool operator < (multhash b) const {
5    if (h1 != b.h1) return h1 < b.h1;
6    return h2 < b.h2;
7    }
8    bool operator == (multhash b) const { return (h1== b.h1 && h2== b.h2)
         ;}
9    bool operator != (multhash b) const { return !(h1== b.h1 && h2== b.h2)
         ;}
10 public:
11   string s;
12   multhash(){
13       h1 = 0; h2 = 0;s = "";
14       for(char l = 'a'; l<='z'; l++) alf[l] = l-'a'+1;
15   }
16   void innit(){
17       unsigned long long int inf,p,op;
18
19       inf = 999727999;
20       p = 325255434;op = 325255434;
21       for(char l: s){
22           h1+=(p*alf[l])%inf;
23           p*=op;
24           p%=inf;
25       }
26
27       inf = 1070777777;
28       p = 10018302;op = 10018302;
29       for(char l: s){
30           h2+=(p*alf[l])%inf;
31           p*=op;
32           p%=inf;
```

```
33          }
34        }
35 };
36 //VALORES ALTERNATIVOS DE INF, LOG 17
37 //666666555557777777
38 //986143414027351997
39 //974383618913296759
40 //973006384792642181
41 //953947941937929919
42 //909090909090909091
43 //VALORES PARA P, USAR PRIMOS MAYORES A |Alfabeto|
44 //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  //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 = -1; 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   Clasicos

7.1   Job scheduling

7.1.1   One machine, linear penalty

```
//cuando se tiene que encontrar un orden optimo
//para trabajos con una funcion lineal de penalty, basta con hacer un
    sort en O(n log n)
struct trabajo{
    long long int penalty,tiempo;
    int ind;
};
bool comp(const trabajo a, const trabajo b){
    if (a.tiempo * b.penalty == a.penalty * b.tiempo) return a.ind<b.ind
        ;
    return a.tiempo * b.penalty < a.penalty * b.tiempo;
}
```

7.1.2   One machine, deadlines

```
//calcula la maxima cantidad de jobs que se pueden hacer dados sus
    deadlines y duraciones en O(n log n)
struct Job {
    int deadline, duration, idx;

    bool operator<(Job o) const {
        return deadline < o.deadline;
    }
};
vector<int> compute_schedule(vector<Job> jobs) {
    sort(jobs.begin(), jobs.end());

    set<pair<int,int>> s;
    vector<int> schedule;
    for (int i = jobs.size()-1; i >= 0; i--) {
        int t = jobs[i].deadline - (i ? jobs[i-1].deadline : 0);
        s.insert(make_pair(jobs[i].duration, jobs[i].idx));
        while (t && !s.empty()) {
            auto it = s.begin();
            if (it->first <= t) {
                t -= it->first;
                schedule.push_back(it->second);
            } else {
                s.insert(make_pair(it->first - t, it->second));
                t = 0;
            }
            s.erase(it);
        }
    }
    return schedule;
}
```

7.1.3   One machine, profit

```
// Dado n Jobs y su profit, calcula cual es el mayor profit que se puede
    obtener en O(n^2)
struct Job{int start, finish, profit;};
bool jobComparataor(Job s1, Job s2){return (s1.finish < s2.finish);}
// Find the latest job (in sorted array) that doesn't
// conflict with the job[i]. If there is no compatible job,
// then it returns -1.
vector <Job> arr;
int* memo;
int latestNonConflict( int i){
  for (int j = i - 1; j >= 0; j--)
    if (arr[j].finish <= arr[i - 1].start)
      return j;
  return -1;
}
// A recursive function that returns the maximum possible
// profit from given array of jobs.  The array of jobs must
// be sorted according to finish time.
int findMaxProfitRec( int n){
  // Base case
  if (n == 1) return arr[n - 1].profit;
    if (memo[n]>=0) return memo[n];
  // Find profit when current job is included
  int inclProf = arr[n - 1].profit;
  int i = latestNonConflict(n);
  if (i != -1) inclProf += findMaxProfitRec( i + 1);

  // Find profit when current job is excluded
  int exclProf = findMaxProfitRec( n - 1);

  return memo[n]=max(inclProf, exclProf);
}
```

```
32
33  // The main function that returns the maximum possible
34  // profit from given array of jobs
35  int findMaxProfit( int n){
36    sort(arr.begin(),arr.end(), jobComparataor);
37    return findMaxProfitRec(n);
38  }
```

7.1.4   Two machines, min time

```
1  //Obtiene el ordenamiento optimo de Jobs en dos maquinas en O(n log n)
2  struct Job {
3      int a, b, idx;
4      bool operator<(Job o) const {return min(a, b) < min(o.a, o.b);}
5  };
6  vector<Job> johnsons_rule(vector<Job> jobs) {
7      sort(jobs.begin(), jobs.end());
8      vector<Job> a, b;
9      for (Job j : jobs) {
10         if (j.a < j.b)
11             a.push_back(j);
12         else
13             b.push_back(j);
14     }
15     a.insert(a.end(), b.rbegin(), b.rend());
16     return a;
17 }
18
19 pair<int, int> finish_times(vector<Job> const& jobs) {
20     int t1 = 0, t2 = 0;
21     for (Job j : jobs) {
22         t1 += j.a;
23         t2 = max(t2, t1) + j.b;
24     }
25     return make_pair(t1, t2);
26 }
```

8   Flow

8.1   Dinic, thx GGDem

```
1  #define pb push_back
2  #define mp make_pair
3  #define fst first
4  #define snd second
5  #define ALL(s) s.begin(),s.end()
6  #define SZ(x) int((x).size())
7  #define fore(i,a,b) for(int i=a,to=b;i<to;++i)
8  using namespace std;
9  typedef long long ll;
10
11 #define INF (1LL<<62)
12 // Min cut: nodes with dist>=0 vs nodes with dist<0
13 // Matching MVC: left nodes with dist<0 + right nodes with dist>0
14 struct Dinic{
15   int nodes,src,dst;
16   vector<int> dist,q,work;
17   struct edge {int to,rev;ll f,cap;};
18   vector<vector<edge>> g;
19   Dinic(int x):nodes(x),g(x),dist(x),q(x),work(x){}
20   void add_edge(int s, int t, ll cap){
21     g[s].pb((edge){t,SZ(g[t]),0,cap});
22     g[t].pb((edge){s,SZ(g[s])-1,0,0});
23   }
24   bool dinic_bfs(){
25     fill(ALL(dist),-1);dist[src]=0;
26     int qt=0;q[qt++]=src;
27     for(int qh=0;qh<qt;qh++){
28       int u=q[qh];
29       fore(i,0,SZ(g[u])){
30         edge &e=g[u][i];int v=g[u][i].to;
31         if(dist[v]<0&&e.f<e.cap)dist[v]=dist[u]+1,q[qt++]=v;
32       }
33     }
34     return dist[dst]>=0;
35   }
36   ll dinic_dfs(int u, ll f){
37     if(u==dst)return f;
38     for(int &i=work[u];i<SZ(g[u]);i++){
39       edge &e=g[u][i];
40       if(e.cap<=e.f)continue;
41       int v=e.to;
42       if(dist[v]==dist[u]+1){
43         ll df=dinic_dfs(v,min(f,e.cap-e.f));
44         if(df>0){e.f+=df;g[v][e.rev].f-=df;return df;}
45       }
46     }
```

```
47        return 0;
48      }
49      ll max_flow(int _src, int _dst){
50        src=_src;dst=_dst;
51        ll result=0;
52        while(dinic_bfs()){
53          fill(ALL(work),0);
54          while(ll delta=dinic_dfs(src,INF))result+=delta;
55        }
56        return result;
57      }
58    };
59
60    //--------------------SOLBEGIN--------------------------------
61    int main() {
62      ios_base::sync_with_stdio(false); cout.tie(NULL); cin.tie(NULL);
63        //l set,r set
64        int n,m;
65        cin>>n>>m;
66        m+=n;
67        Dinic d(n+m+2);
68        for(int i = 1; i<=n; i++) d.add_edge(0,i,1);
69        for(int i = n+1; i<=m; i++) d.add_edge(i,m+1,1);
70
71        int fin,q;
72        for(int i = 1; i<=n; i++){
73            cin>>q;
74            while(q--){
75                cin>>fin;
76                d.add_edge(i,n+fin,1);
77            }
78        }
79        int res =d.max_flow(0,m+1);
80        m-=n;
81        //how many were left unmatched
82        cout<<m-res<<endl;
83    }
84    //--------------------EOSOLUTION--------------------------------
```

9   Miscellaneous

9.1   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;
```

```cpp
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;
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-------------------------------
```

10 Testing