UFAL Notebook

Contents

1 Binary Search and Ternary Search

1.1 LowerBound

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<pi, int>
#define fir first
#define sec second
#define DEBUG 0
#define MAXN 1000001
#define mod 1000000007
// first element >= x
vector<int> k(MAXN);
int lower(int l, int r, int x) // first element >= x
{
  while (l < r)
  {
    int mid = (l + r) >> 1;
    (x <= k[mid]) ? r = mid : l = mid + 1;
  }
  return k[l];
}
```

1.2 Aplications

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

#define lli long long int
#define pb push_back
#define in insert
#define pi pair<int, int>
#define pii pair<int, pi>
#define mp make_pair
#define fir first
#define sec second
#define MAXN 1001

// 1 - ts para double
long double ts()
{
  long double l = 0, r = DBL_MAX;
  for (int i = 0; i < 2000; i++)
  {
    long double l1 = (l * 2 + r) / 3.0;
    long double l2 = (l + 2 * r) / 3.0;
    if (possible(l1))
      r = l2;
    else
      l = l1;
  }
  return l;
}
// 2- bb para double
long double bb()
{
  long double i = 0, f = DBL_MAX, m;
  while (f - i > 0.000000001)
  {
    m = (i + f) / 2.0;
    if (possible(m))
      f = m;
    else
      i = m;
  }
  return i;
}
// 3 - bb pra int
lli bb()
{
  lli i = 0, f = INT_MAX, m;
  while (i < f)
  {
    m = (i + f) / 2;
    if (possible(m))
      f = m;
    else
      i = m + 1;
  }
  return i;
}
// 4 - ts pra int (valor minimo da funcao f(x)), sendo x
```

```
    um inteiro
int l = 1, r = INT_MAX;
while (r - l > 15)
{
  int l1 = (l * 2 + r) / 3;
  int l2 = (l + 2 * r) / 3;
  (calc(l1) < calc(l2)) ? r = l2 : l = l1;
}
for (int i = l; i <= r; i++)
// vejo qual a melhor opcao de l ate r em o(n)


// busca ternaria para int, usando busca binaria:
int l = 0, r = 1e9;
while (l < r)
{
  int mid = (l + r) >> 1;
  (calc(mid) < calc(mid + 1)) ? r = mid : l = mid + 1;
}
return calc(l);
```

1.3 TS

```
// busca ternaria
// divide em 3 partes, 2 mids
// mid1 = l + (r-l)/3
// mid2 = r - (r-l)/3
#include <bits/stdc++.h>
using namespace std;

#define lli long long int
#define pb push_back
#define in insert
#define pi pair<int, int>
#define pii pair<int, pi>
#define mp make_pair
#define fir first
#define sec second
#define MAXL 100001

int n, key;
vector<int> ar;

int ts()
{
  int l = 0, r = n - 1;
  while (r >= l)
  {
    int mid1 = l + (r - l) / 3;
    int mid2 = r - (r - l) / 3;
    if (ar[mid1] == key)
      return mid1;
    if (ar[mid2] == key)
      return mid2;
    if (key < ar[mid1])
      r = mid1 - 1;
    else if (key > ar[mid2])
      l = mid2 + 1;
    else
    {
      l = mid1 + 1;
      r = mid2 - 1;
    }
  }
  return -1; // nao encontrado
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  cin >> n;
  ar.resize(n);
  for (int i = 0; i < n; i++)
    cin >> ar[i];
  sort(ar.begin(), ar.end());
  cin >> key;
  cout << ts() << endl;
  return 0;
}
```

1.4 UpperBound

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<pi, int>
#define fir first
#define sec second
#define DEBUG 0
#define MAXN 1000001
```

```
#define mod 1000000007
// last element <= x
vector<int> k(MAXN);
int upper(int l, int r, int x)
{
  while (l < r)
  {
    int mid = (l + r + 1) >> 1;
    (k[mid] <= x) ? l = mid : r = mid - 1;
  }
  return k[l];
}
```

1.5 STL

```
// lower - primeiro maior ou igual a x
// upper - ultimo menor ou igual a x

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

#define lli long long int
#define pb push_back

vector <int> v ;
int main()
{
    int n , aux ;
    cin >> n ;

    for (int i = 0 ; i < n ; i++)
    {
        cin >> aux ;
        v.pb(aux);
    }

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

    int q ;
    cin >> q ;

    while (q--)
    {
        cin >> aux ;
        vector <int> :: iterator low = lower_bound (v.
           begin() , v.end() , aux) ;
        vector <int> :: iterator up = upper_bound (v.
           begin() , v.end() , aux) ;
        cout << (low - v.begin()) << " " << (up - v.
           begin()) - 1 << endl ;
    }

    return 0 ;
}
```

1.6 BS

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

#define lli long long int
#define pb push_back

vector <int> v;
int binarysearch (int n , int x)
{
    int i = 0 ;
    int f = n - 1 ;
    int m ;

    while(i <= f)
    {
        m = (i + f) / 2 ;

        if(v[m] == x) return m + 1 ;
        if(v[m] < x)  i = m + 1 ;
        if(v[m] > x)  f = m - 1 ;
    }

    return 0 ;
}
int main ()
{
    int n , aux , m ;

    cin >> n ;

    for (int i = 0 ; i < n ; i++)
    {
        cin >> aux ;
        v.pb(aux);
    }

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

    cin >> m ;
    cout << binarysearch(n , m) << endl ;
```

```
    return 0 ;
}
```

2 Miscellaneous

2.1 meetinthemiddle

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<pi, int>
#define fir first
#define sec second
#define DEBUG 0
#define MAXN 1000001

int n, t;
vector<int> v;
vector<int> a;
vector<int> b;

void solve2(int i, int j, int k)
{
  if (i == j)
  {
    b.pb(k);
    return;
  }
  solve2(i + 1, j, k);
  solve2(i + 1, j, k + v[i]);
}
void solve(int i, int j, int k)
{
  if (i == j)
  {
    a.pb(k);
    return;
  }
  solve(i + 1, j, k);
  solve(i + 1, j, k + v[i]);
}
int upper(int l, int r, int x)
{
  while (l < r)
  {
    int mid = (l + r + 1) >> 1;
    (b[mid] <= x) ? l = mid : r = mid - 1;
  }
  return b[l];
}
int meetinthemiddle()
{
  solve(0, (n >> 1) + 1, 0);
  solve2((n >> 1) + 1, n, 0);
  sort(b.begin(), b.end());
  int ans = 0;
  for (auto const &i : a)
  {
    if (i > t)
      continue;
    ans = max(ans, i);
    int kappa = i + upper(0, b.size() - 1, t - i);
    if (kappa <= t)
      ans = max(ans, kappa);
  }
  return ans;
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  cin >> n >> t;
  v.resize(n);
  for (int i = 0; i < n; i++)
    cin >> v[i];
  cout << meetinthemiddle() << endl;
  return 0;
}
```

2.2 bitmasks

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
```

```
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 200005
#define mod 998244353

signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n, mask;
  vector<int> masks;

  // quantidade de bits setados na mask
  cout << __builtin_popcount(mask) << endl;

  // para printar o valor do bit i
  for (int i = 0; i < n; i++)
    cout << ((mask >> i) & 1) << " ";
  cout << endl;

  // quando eh necessario percorrer todas as submasks
     ate (1 << n)
  // e fazer algo com todas as submasks dessa mask
  // util em problemas de dp com mask por exemplo
  for (int i = 0; i < n; i++)
  {
    for (int j = 0; j < (1 << n); j++)
    {
      if ((j >> i & 1) == 0)
      {
        //alguma coisa aqui sabendo que a mask(j) eh uma
            submask de(j ^ 1 << i)
      }
    }
  }

  // para percorrer por todas as submasks de uma mask
  for (int s = mask; s; s = (s - 1) & mask)
  {
    // alguma coisa aqui sabendo que s eh uma submask de
        mask
  }

  // quando eh necessario percorrer todas as submasks
     ate (1 << n)
  // e fazer algo com todas as submasks dessa mask O(3^n
     )
  // util em problemas de dp com mask por exemplo
  for (int m = 0; m < (1 << n); m++)
  {
    for (int s = m; s; s = (s - 1) & m)
    {
      // alguma coisa aqui sabendo que mask s eh uma
         submask de m
    }
  }

  // comprimindo as masks de um vector baseada em uma
     mask qualquer
  for (int i = 0; i < masks.size(); i++)
  {
    int compressed = 0, curr_bit = 0;
    for (int j = 0; j < n; j++)
    {
      if (!(mask & (1LL << j)))
        continue;
      if (masks[i] & (1LL << j))
        compressed |= (1LL << curr_bit);
      curr_bit++;
    }
    // alguma coisa sabendo que a mask compressed eh a
       mask comprimida da mask atual
  }
  return 0;
}
```

2.3 two pointers

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define PI acos(-1)
#define pb push_back
#define int long long int
#define pi pair<int, int>
#define pii pair<int, pi>
```

```
#define fir first
#define sec second
#define MAXN 100001
#define mod 1000000007

signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n, m;
  cin >> n >> m;
  vector<int> v(n);
  vector<int> vv(m);
  for (int i = 0; i < n; i++)
    cin >> v[i];
  for (int i = 0; i < m; i++)
    cin >> vv[i];
  int ans = 0, prev = LLONG_MAX, curr = 0;
  for (int l = 0, r = 0; l < m; l++)
  {
    if (vv[l] != prev)
      curr = 0;
    while (r < n && v[r] <= vv[l])
    {
      if (v[r] == vv[l])
        curr++;
      r++;
    }
    ans += curr;
    prev = vv[l];
  }
  cout << ans << endl;
}
//You are given two arrays a and b, sorted in non-
    decreasing order. Find the number of pairs (i,j) for
     which ai=bj.
```

2.4 inversion count

```
// seja S = a1, a2 , ... , an
// uma inversao S e um par (i,j) com i < j e ai > aj

// Solucao O(n2) nao ideal:
//for(int i=0;i<n;i++)
//      for(int j=i+1;j<n;j++)
//              if(v[i]>v[j]) ans++;

// Em vez de trabalharmos com o vetor inteiro(n2), vamos
     dividir o vetor ao meio e trabalhar com suas
    metades,
// que chamaremos de u1 e u2.

// Queremos saber o valor de inv, o numero de inversoes
    em v. Ha tres tipos de inversoes (i,j)(i,j) em v:
// aquelas em que i e j estao ambos em u1, aquelas em
    que i e j estao ambos em u2 e aquelas
// em que i esta em u1 e j esta em u2.
#include <bits/stdc++.h>
using namespace std;

#define lli long long int
#define pb push_back
#define in insert
#define pi pair<int, int>
#define pii pair<int, pi>
#define mp make_pair
#define fir first
#define sec second
#define MAXN 100001
#define INF 1000000000

int merge_sort(vector<int> &v)
{
  int ans = 0;

  if (v.size() == 1)
  {
    return 0;
  }

  vector<int> u1, u2;

  for (int i = 0; i < v.size() / 2; i++)
  {
    u1.pb(v[i]);
  }
  for (int i = v.size() / 2; i < v.size(); i++)
  {
    u2.pb(v[i]);
  }

  ans += merge_sort(u1);
  ans += merge_sort(u2);

  u1.pb(INF);
  u2.pb(INF);

  int ini1 = 0, ini2 = 0;

  for (int i = 0; i < v.size(); i++)
```

```
    {
      if (u1[ini1] <= u2[ini2])
      {
        v[i] = u1[ini1];
        ini1++;
      }
      else
      {
        v[i] = u2[ini2];
        ini2++;
        ans += u1.size() - ini1 - 1;
      }
    }

    return ans;
}
signed main()
{
    int n;
    cin >> n;
    vector<int> v(n);
    for (int i = 0; i < n; i++)
      cin >> v[i];
    cout << merge_sort(v) << endl;
    return 0;
}
```

2.5 sprague grundy

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 500009
#define mod 1000000001

vector<int> v = {2, 3, 4, 5, 6};
unordered_map<int, bool> vis;
unordered_map<int, int> dp;

int g(int x) // achar o grundy number na marra
{
    if (x == 0)
        return 0;
    vector<bool> ok(4, 0);
    int mex = 0;
    for (auto const &i : v)
    {
        int curr = g(x / i);
        if (curr < 4)
            ok[curr] = 1;
        while (ok[mex])
            mex++;
    }
    vis[x] = 1;
    return dp[x] = mex;
}
int solve(int x) // padraozin
{
    vector<int> ini = {0, 1, 2, 2, 3, 3, 0, 0, 0, 0, 0,
       0};
    while (x >= 12)
        x /= 12;
    return ini[x];
}
signed main()
{
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);
    int q;
    cin >> q;
    while (q--)
    {
        int n;
        cin >> n;
        int x = 0;
        for (int i = 0; i < n; i++)
        {
            int k;
            cin >> k;
            x ^= solve(k);
        }
        (x > 0) ? cout << "Henry\n" : cout << "Derek\n";
    }
    return 0;
}
/*
```

```
game theory (um exemplo simples de problema pra ficar no
    repo)

- pro nim classico
- existem n pilhas cada uma possui x[i] blocos
- em uma play posso escolher uma pilha e tirar uma
   quantidade qualquer de blocos dela
- quem ganha?
- o jogador que comeca ganha se o xor dos tamanhos das
   pilhas for != 0

- teorema sprague-grundy (transformar um jogo qualquer
   em nim)
- seja v um estado que eu tou do jogo, podemos calcular
   o grundy number desse estado
- seja o conjuntos de estados adjacentes a v {u1, u2,
   ..., un}
- g(v) = mex(g(u1), g(u2), ..., g(un))
- se v nao tem nenhum extado adjacente, entao g(v) = 0
- g(v) -> grundy number do estado v
- com isso se tivemos varios estados iniciais (varias
   pilhas)
- podemos simplesmente achar o grundy number de cada um
   deles e depois saber quem ganha
- pelo valor do xor dos grundy numbers

- exemplo: floor division game
- existem n numeros e em uma play posso escolher um
   deles e dividir por 2, 3, 4, 5 ou 6
- quem ganha?
- achar o grundy number de cada um dos n numeros
- se o xor for != 0, ganha quem comeca jogando
- caso contrario, o outro jogador ganha

- as vzs e util tbm ver se existe um padrao (em caso de
   altas constantes)
- notando o padrao, da pra achar o grundy number de
   forma mais eficiente e resolver o problema
*/
```

2.6 stack trick

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

#define PI acos(-1)
#define int long long int
#define pb push_back
#define pi pair<int, int>
#define fir first
#define sec second
#define MAXN 300001
#define mod 1000000007

int n;
vector<int> v;
vector<int> ans;

void solve()
{
  stack<pi> s;
  for (int i = n - 1; i >= 0; i--)
  {
    while (!s.empty() && s.top().fir <= v[i])
      s.pop();
    (!s.empty()) ? ans[i] = s.top().sec : ans[i] = -1;
    s.push({v[i], i});
  }
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  cin >> n;
  v.resize(n);
  ans.resize(n);
  for (int i = 0; i < n; i++)
    cin >> v[i];
  solve();
  for (auto const &i : ans)
    cout << i << " ";
  cout << endl;
}
// WITHOUT SEGMENT TREE
// for each index (0 <= i < n), find another index (0 <=
    j < n)
// which v[j] > v[i] and j > i and j is as close as
   possible to i.
// if this index does not exist, print -1

/*
5
1 3 3 4 5
*/
/*
1 3 3 4 -1
*/
```

3 STL

3.1 ordered set

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<pi, int>
#define fir first
#define sec second
#define DEBUG false
#define MAXN 200002

template <class T> // template do ordered set
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  ordered_set<int> s; // ordered_set
  s.insert(1);
  s.insert(1);
  s.insert(2);
  s.insert(4);
  s.insert(3);
  for (auto const &i : s) // nao adiciona elementos
      repetidos, que nem o set normal
    cout << i << " ";
  cout << endl;
  cout << *(s.find_by_order(0)) << endl; // iterator do
      elemento 0
  cout << *(s.find_by_order(1)) << endl; // iterator do
      elemento 1
  cout << s.order_of_key(4) << endl;     // quantidade
      de elementos que sao menores do que 4
  cout << s.order_of_key(6) << endl;     // quantidade
      de elementos que sao menores do que 4
}
// find_by_order : O(log n), retorna (um iterator) qual
    o k-esimo elemento do set
// order_of_key: O(log n), retorna qual a quantidade de
    elementos menores do que x no set
```

3.2 STL

```
1)Vector

vector <int> v; //Criacao o vector

v.push_back(10); //Adiciono o elemento 10 no final do
    vector v
v.size() // retorna o tamanho do vector
v.resize(10); //Muda o tamanho do vector v para 10.
v.pop_back(); //Apaga o ultimo elemento do vector V.
v.clear(); // apaga todos os elementos do vector v .
sort(v.begin(), v.end()); //Ordena todo o vector v


2)Pair

pair <string, int> p; // criando a pair relacionando um
    first com um second

p.first = "Joao"; // adicionando elementos
p.second = 8 ; //adicionando elementos

// utilidade: vector de pair
vector< pair <int, string> > v; // criando o vector v de
     pair
v.push_back(make_pair(a,b)); // dando push back em uma
    pair no vector usando make_pair
sort(v.begin(), v.end()); // tambem e possivel ordenar o
     vector de pair

3)Queue / FIla

queue <int> f; // criando a queque

f.push(10); // adiciona alguem na fila
f.pop(); // remove o elemento que esta na frente da fila
f.front(); // olha qual o elemento esta na frete da fila
f.empty() // retorna true se a fila estiver vazia e
    false se nao estiver vazia


4)Stack / Pilha

stack <int> p ; // criando a stack

pilha.push(x); //Adiciona o elemento x no topo da pilha
```

```
pilha.pop(); //Remove elemento do topo da pilha
pilha.top(); // retorna o elemento do topo da pilha
pilha.empty(); // verifica se a pilha esta vazia ou nao

5) Set

set <int> s ; // criando a set
// obs: a set nao adiciona elementos repetidos

s.insert(10); //Adiciona o elemento 10 no set
s.find(10) // Para realizar uma busca no set utilizamos
    o comando find,
o find retorna um ponteiro que aponta para o elemento
    procurado caso o elemento esteja no set ou para o
    final do set, caso o elemento procurado nao esteja
    no set , em complexidade O(log n)

if(s.find(10) != s.end()) // procurando pelo 10, se ele
    estiver no set
s.erase(10); //Apaga o elemento 10 do set em O(log n)
s.clear(); // Apaga todos os elementos
s.size(); //  Retorna a quantidade de elementos
s.begin(); // Retorna um ponteiro para o inicio do set
s.end(); // Retorna um ponteiro para o final do set


6)Map
map <string, int> m; //Cria uma variavel do tipo map que
     mapeia strings em int
// Em um map cada elemento esta diretamente ligado a um
    valor, ou seja, cada elemento armazenado no map
    possui um valor correspondente
// Se tivermos um map de strings em inteiros e inserimos
     os pair ("Joao", 1), ("Alana", 10), ("Rodrigo", 9)
// Caso facamos uma busca pela chave "Alana" receberemos
     o numero 10 como retorno.

m.insert(make_pair("Alana", 10)); //Inserimos uma
    variavel do tipo pair diretamente no map, O(log n)
M["Alana"] = 10; //Relacionando o valor 10 a chave "
    Alana"
if(m.find("Alana") != m.end()){ //Se a chave "Alana" foi
     inserida no map
cout << m["Alana"] << endl; //Imprime o valor
    correspondente a chave "Alana", no caso, o valor 10.
m.erase("Alana"); //Apaga o elemento que possui a chave
    "Alana" do map
m.clear(); // Apaga todos os elementos
m.size(); //  Retorna a quantidade de elementos
m.begin(); // Retorna um ponteiro para o inicio do map
m.end(); // Retorna um ponteiro para o final do map


7)Priority Queue
priority_queue <int> q; // declarando a priority queue
// Para utilizar a priority_queue do C++ e importante
    apenas saber que o maior elemento sempre estara na
    primeiro posicao.
// Com execao disso, todos os outros metodos sao
    semelhantes ao uso de uma queue comum, porem para
    manter a estrutura organizada, a complexidade da
    operacao de insercao e O(logn).
p.push(i) // adiciono o elemento i na priority_queue
p.pop(); // apago o primeiro da fila
p.top(); // vejo quem esta no topo
```

4 Utils

4.1 runner2

```python
# This script does the following:
# 1 - Run a code with all inputs files from a folder
# 2 - Compare the output for each test case with the
    answer
import os

code = "a.cpp" # Path to your code
input_folder = "input" # Path to folder which the input
    files are
output_folder = "output" # Path to folder which the
    output files are
input_prefix = "L_" # prefix of all input files names
output_prefix = "L_" # prefix of all input files names
tests = 56 # Number of test cases

def compile_code():
    os.system('g++ ' + code + ' -o code -O2')

def get_ans(output):
    out = open(output, "r")
    ret = out.read()
    out.close()
    return ret

def get_code_output(input):
    output = os.popen('./code <' + input).read()
    return output

def main():
```

```
    compile_code()
    # tests indexed from 1
    for i in range (1, tests + 1):
        ans = get_ans(output_folder + '/' +
            output_prefix + str(i))
        code_output = get_code_output(input_folder + '/'
            + input_prefix + str(i))
        print('Case' + str(i) + ': ')
        if ans == code_output:
            print('ACCEPTED')
        else :
            print('FAILED\n')
            print('ANSWER:')
            print(ans)
            print('\nCODE OUTPUT:')
            print(code_output)
        print()

if __name__ == '__main__':
    main()
```

4.2 int128

```
// https://codeforces.com/blog/entry/75044
// functions to print and read a __int128 in c++
__int128 read()
{
  __int128 x = 0, f = 1;
  char ch = getchar();
  while (ch < '0' || ch > '9')
  {
    if (ch == '-')
      f = -1;
    ch = getchar();
  }
  while (ch >= '0' && ch <= '9')
  {
    x = x * 10 + ch - '0';
    ch = getchar();
  }
  return x * f;
}
void print(__int128 x)
{
  if (x < 0)
  {
    cout << "-";
    x = -x;
  }
  if (x > 9)
    print(x / 10);
  char at = (x % 10) + '0';
  cout << at;
}
```

4.3 execution time

```
// https://codeforces.com/blog/entry/57647
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 300005
#define mod 1000000007

signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  // just call clock in the beginning
  clock_t time = clock();

  // ...
  // ...
  // some code here ...
  // ...
  // ...

  // execution time:
  cout << setprecision(3) << fixed << (double)(clock() -
      time) / CLOCKS_PER_SEC << endl;
  return 0;
}
```

4.4 runner

```
# This script does the following:
```

```python
# 1 - Generate a random testcase
# 2 - Run some "naive" code with this input
# 3 - Run your code with this input
# 4 - Compare the outputs
import os

naive = "brute.cpp" # path to naive code
code = "d.cpp" # path to your code
generator = "g.cpp" # path to test generator

def compile_codes():
    os.system('g++ ' + generator + ' -o generator -O2')
    os.system('g++ ' + naive + ' -o naive -O2')
    os.system('g++ ' + code + ' -o code -O2')

def generate_case():
    os.system('./generator > in');

def get_naive_output():
    output = os.popen('./naive <in').read()
    return output

def get_code_output():
    output = os.popen('./code <in').read()
    return output

def main():
    compile_codes()

    while True:
        generate_case()
        naive_output = get_naive_output()
        code_output = get_code_output()

        if naive_output == code_output:
            print('ACCEPTED')
        else :
            print('FAILED\n')
            print('ANSWER:')
            print(naive_output)
            print('\nCODE OUTPUT:')
            print(code_output)
            break

if __name__ == '__main__':
    main()
```

5 Math

5.1 totient

```cpp
#define MAXN 100000

int phi[MAXN];

void calc()
{
  for (int i = 0; i < MAXN; i++)
    phi[i] = i;
  for (int i = 2; i < MAXN; i++)
  {
    if (phi[i] == i)
    {
      for (int j = i; j < MAXN; j += i)
        phi[j] -= phi[j] / i;
    }
  }
}
int calc_phi(int n)
{
  int ans = n;
  for (int i = 2; i * i <= n; i++)
  {
    if (n % i == 0)
    {
      while (n % i == 0)
        n /= i;
      ans -= ans / i;
    }
  }
  if (n > 1)
    ans -= ans / n;
  return ans;
}
```

5.2 iterative fft

```cpp
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
```

```
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define PI acos(-1)
#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<pi, int>
#define fir first
#define sec second
#define MAXN 125001
#define mod 1000000007
#define cd complex<double>

namespace fft
{
  int n;

  void fft(vector<cd> &a, bool invert)
  {
    int n = a.size();
    for (int i = 1, j = 0; i < n; i++)
    {
      int bit = n >> 1;
      for (; j & bit; bit >>= 1)
        j ^= bit;
      j ^= bit;
      if (i < j)
        swap(a[i], a[j]);
    }
    for (int len = 2; len <= n; len <<= 1)
    {
      double ang = 2 * PI / len * (invert ? -1 : 1);
      cd wlen(cos(ang), sin(ang));
      for (int i = 0; i < n; i += len)
      {
        cd w(1);
        for (int j = 0; j < len / 2; j++)
        {
          cd u = a[i + j], v = a[i + j + len / 2] * w;
          a[i + j] = u + v;
          a[i + j + len / 2] = u - v;
          w *= wlen;
        }
      }
    }
    if (invert)
      for (cd &x : a)
        x /= n;
  }

  vector<double> mul(vector<double> a, vector<double> b)
  {
    vector<cd> fa(a.begin(), a.end()), fb(b.begin(), b.
       end());
    n = 1;
    while (n < a.size() + b.size())
      n <<= 1;
    fa.resize(n);
    fb.resize(n);
    fft(fa, false);
    fft(fb, false);
    for (int i = 0; i < n; i++)
      fa[i] *= fb[i];
    fft(fa, true);
    vector<double> ans(n);
    for (int i = 0; i < n; i++)
      ans[i] = fa[i].real();
    return ans;
  }
} // namespace fft
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
}
```

5.3 fraction

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 200006
#define mod 1000000007

struct fraction
{
```

```
int x, y; // x / y

fraction() {}
fraction(int x, int y) : x(x), y(y) {}
bool operator==(fraction o) { return (x * o.y == o.x *
    y); }
bool operator!=(fraction o) { return (x * o.y != o.x *
    y); }
bool operator>(fraction o) { return (x * o.y > o.x * y
   ); }
bool operator>=(fraction o) { return (x * o.y >= o.x *
    y); }
bool operator<(fraction o) { return (x * o.y < o.x * y
   ); }
bool operator<=(fraction o) { return (x * o.y <= o.x *
    y); }
fraction operator+(fraction o)
{
  fraction ans;
  ans.y = (y == o.y) ? y : y * o.y;
  ans.x = (x) * (ans.y / y) + (o.x) * (ans.y / o.y);
  // ans.simplify();
  return ans;
}
fraction operator*(fraction o)
{
  fraction ans;
  ans.x = x * o.x;
  ans.y = y * o.y;
  // ans.simplify();
  return ans;
}
fraction inv()
{
  fraction ans = fraction(x, y);
  swap(ans.x, ans.y);
  return ans;
}
fraction neg()
{
  fraction ans = fraction(x, y);
  ans.x *= -1;
  return ans;
}
void simplify()
{
  if (abs(x) > 1e9 || abs(y) > 1e9) // slow
      simplification
  {
    int g = __gcd(y, x);
    x /= g;
    y /= g;
  }
 }
 // substraction and division can be easily done
};
signed main()
{
 ios_base::sync_with_stdio(false);
 cin.tie(NULL);
 return 0;
}
```

5.4 gaussian elimination2

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
   rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 230
#define mod 1000000001
#define EPS 1e-9

bitset<MAXN> ans;

int gauss(vector<bitset<MAXN>> &a)
{
  ans.reset();
  int n = a.size(), m = a[0].size() - 1, ret = 1;
  vector<int> where(m, -1);
  for (int col = 0, row = 0; col < m && row < n; col++)
  {
    for (int i = row; i < n; i++)
    {
      if (a[i][col])
      {
        swap(a[i], a[row]);
```

```
        break;
      }
    }
    if (!a[row][col])
      continue;
    where[col] = row;
    for (int i = 0; i < n; i++)
      if (i != row && a[i][col])
        a[i] ^= a[row];
    ++row;
  }
  for (int i = 0; i < m; i++)
  {
    if (where[i] != -1)
      ans[i] = (a[where[i]][m] / a[where[i]][i]);
    else
      ret = 2;
  }
  for (int i = 0; i < n; i++)
  {
    double sum = 0;
    for (int j = 0; j < m; j++)
      sum += (ans[j] * a[i][j]);
    if (abs(sum - a[i][m]) > EPS)
      ret = 0;
  }
  return ret;
}
signed main()
{
}
```

5.5 modular arithmetic

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second

#define MAXN 500001
#define mod 1000000007

struct modint
{
  int val;
  modint(int v = 0) { val = v % mod; }
  int pow(int y)
  {
    modint x = val;
    modint z = 1;
    while (y)
    {
      if (y & 1)
        z *= x;
      x *= x;
      y >>= 1;
    }
    return z.val;
  }
  int inv() { return pow(mod - 2); }
  void operator=(int o) { val = o % mod; }
  void operator=(modint o) { val = o.val % mod; }
  void operator+=(modint o) { *this = *this + o; }
  void operator-=(modint o) { *this = *this - o; }
  void operator*=(modint o) { *this = *this * o; }
  void operator/=(modint o) { *this = *this / o; }
  bool operator==(modint o) { return val == o.val; }
  bool operator!=(modint o) { return val != o.val; }
  int operator*(modint o) { return ((val * o.val) % mod)
      ; }
  int operator/(modint o) { return (val * o.inv()) % mod
      ; }
  int operator+(modint o) { return (val + o.val) % mod;
      }
  int operator-(modint o) { return (val - o.val + mod) %
       mod; }
};

modint f[MAXN];

void fat()
{
  f[0] = 1;
  for (int i = 1; i < MAXN; i++)
    f[i] = f[i - 1] * i;
}
modint ncr(int n, int k)
{
  modint d = f[k] * f[n - k];
```

```
  modint ans = f[n] / d;
  return ans;
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  return 0;
}
```

5.6 crivo

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

#define lli long long int
#define pb push_back
#define in insert
#define pi pair<int, int>
#define pd pair<double, int>
#define pii pair<int, pi>
#define mp make_pair
#define fir first
#define sec second
#define MAXN 100001
#define mod 1000000007

bitset <MAXN> prime;

void crivo ()
{
  prime.set();
  prime[0] = false;
  prime[1] = false;
  for (int i = 2 ; i < MAXN ; i++)
    if(prime[i])
      for(int j = 2 ; j * i < MAXN ; j++)
        prime[j * i] = false;
}
signed main()
{
  crivo();
  int q;
  cin >> q;
  while(q--)
  {
    int n;
    cin >> n;
    (prime[n]) ? cout << "YES\n" : cout << "NO\n" ;
  }
  return 0;
}
```

5.7 divisors

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

#define PI acos(-1)
#define int long long int
#define pb push_back
#define pi pair<int, int>
#define fir first
#define sec second
#define MAXN 5001
#define mod 1000000007

signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n;
  cin >> n;
  int ans = 0;
  for (int i = 1; i <= sqrt(n); i++)
  {
    if (!(n % i))
    {
      ans++;
      if (n / i != i)
        ans++;
    }
  }
  cout << ans << endl;
}
```

5.8 gaussian elimination

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
   rb_tree_tag, tree_order_statistics_node_update>;
```

```cpp
#define int long long int
#define double long double
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define DEBUG 1
#define MAXN 2001
#define mod 1000000007
#define EPS 1e-9

vector<double> ans;

int gauss(vector<vector<double>> a)
{
  int n = a.size(), m = a[0].size() - 1, ret = 1;
  ans.assign(m, 0);
  vector<int> where(m, -1);
  for (int col = 0, row = 0; col < m && row < n; col++,
      row++)
  {
    int sel = row;
    for (int i = row; i < n; i++)
      if (abs(a[i][col]) > abs(a[sel][col]))
        sel = i;
    if (abs(a[sel][col]) < EPS)
      continue;
    for (int i = col; i <= m; i++)
      swap(a[sel][i], a[row][i]);
    where[col] = row;
    for (int i = 0; i < n; i++)
    {
      if (i != row)
      {
        double c = a[i][col] / a[row][col];
        for (int j = col; j <= m; j++)
          a[i][j] -= a[row][j] * c;
      }
    }
  }
  for (int i = 0; i < m; i++)
  {
    if (where[i] != -1)
      ans[i] = (a[where[i]][m] / a[where[i]][i]);
    else
      ret = 2;
  }
  for (int i = 0; i < n; i++)
  {
    double sum = 0;
    for (int j = 0; j < m; j++)
      sum += (ans[j] * a[i][j]);
    if (abs(sum - a[i][m]) > EPS)
      ret = 0;
  }
  return ret; // 0 = nao existe solucao, 1 = existe uma
     solucao, 2 = existem multiplas solucoes
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  vector<vector<double>> a = {{1.0, 1.0, 20.0},  // 1x +
      1y = 20
                              {3.0, 4.0, 72.0}}; // 3x +
                                  4y = 72
  cout << gauss(a) << endl;
  for (auto const &i : ans) // x = 8 e y = 12
    cout << i << " ";
  cout << endl;
}
// eliminacao gaussiana
// para resolver sistemas com n equacoes e m incognitas

// para isso iremos utilizar uma representacao usando
// matrizes, no qual uma coluna extra e adicionada,
// representando os resultados de cada equacao.

// algoritimo:
// ideia: qualquer equacao pode ser reescrita como uma
   combinacao linear dela mesma
// 1- dividir a primeira linha(primeira equacao) por a
   [0][0]
// 2- adicionar a primeira linha as linhas restantes, de
    modo que, os
//    coeficientes da primeira coluna se tornem todos
   zeros, para que
//    isso aconteca, na i-esima linha devemos adicionar
   a primeira linha
//    multiplicada por (a[i][0] * -1)
// 3- com isso, o elemento a[0][0] = 1 e os demais
   elementos da primeira coluna
//    serao iguais a zero
// 4- continuamos o algoritimo a partir da etapa 1
   novamente, dessa vez
//    com a segunda coluna e a segunda linha, dividindo
   a linha por a[1][1]
//    e assim sucessivamente
// 5- ao final, teremos a resposta
```

```
// complexidade O(min(n, m) * n * m);
// se n == m, logo a complexidade sera O(n^3)
```

5.9 primefactors2

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define PI acos(-1)
#define pb push_back
#define int long long int
#define pi pair<int, int>
#define pii pair<pi, int>
#define fir first
#define sec second
#define DEBUG 0
#define MAXN 1000001
#define mod 1000000007

namespace primefactors
{
  bitset<MAXN> prime;
  vector<int> nxt(MAXN);
  vector<int> factors;

  void crivo()
  {
    prime.set();
    prime[0] = false, prime[1] = false;
    for (int i = 2; i < MAXN; i++)
    {
      if (prime[i])
      {
        nxt[i] = i;
        for (int j = 2; j * i < MAXN; j++)
        {
          prime[j * i] = false;
          nxt[j * i] = i;
        }
      }
    }
  }

  void fact(int n)
  {
    factors.clear();
    while (n > 1)
    {
      factors.pb(nxt[n]);
      n = n / nxt[n];
    }
  }
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  return 0;
}
```

5.10 matrix exponentiation2

```
// https://www.spoj.com/problems/ITRIX12E/
// count some {f(0) + f(1) + ... + f(n)} with just one
    matrix exponentiation
// creates an extra dimension in the matrix and
    initializes that column with 1s

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

#define PI acos(-1)
#define pb push_back
#define mp make_pair
#define int long long int
#define pi pair<int, int>
#define pii pair<pi, int>
#define fir first
#define sec second
#define MAXN 100001
#define MAXL 20
#define INF 200001
#define mod 1000000007

const int n = 11;
vector<vector<int>> ans(n, vector<int>(n));

vector<vector<int>> multiply(vector<vector<int>> a,
    vector<vector<int>> b)
{
  vector<vector<int>> res(n, vector<int>(n));
  for (int i = 0; i < n; i++)
  {
```

```
    for (int j = 0; j < n; j++)
    {
      res[i][j] = 0;
      for (int k = 0; k < n; k++)
        res[i][j] = (res[i][j] + (((a[i][k] % mod) * (b[
            k][j] % mod)) % mod)) % mod;
    }
  }
  return res;
}
vector<vector<int>> expo(vector<vector<int>> mat, int m)
{
  for (int i = 0; i < n; i++)
    for (int j = 0; j < n; j++)
      ans[i][j] = (i == j);
  while (m > 0)
  {
    if (m & 1)
      ans = multiply(ans, mat);
    m = m / 2;
    mat = multiply(mat, mat);
  }
  return ans;
}
bool is_prime(int n)
{
  for (int i = 2; i < n; i++)
    if (!(n % i))
      return false;
  return true;
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int q;
  cin >> q;
  while (q--)
  {
    int k;
    cin >> k;
    int resp = 0;
    vector<vector<int>> mat(n, vector<int>(n, 0));
    for (int i = 1; i <= 9; i++)
      for (int j = 1; j <= 9; j++)
        if (is_prime(i + j))
          mat[i][j] = 1;
    for (int i = 0; i <= 10; i++)
      mat[i][10] = 1;
    vector<vector<int>> ans = expo(mat, k - 1);
    for (int i = 0; i < n; i++)
      for (int j = 0; j < n; j++)
        resp = (resp + ans[i][j]) % mod;
    cout << resp - 7 << endl;
  }
  return 0;
}
```

5.11 segmentedsieve

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define PI acos(-1)
#define pb push_back
#define int long long int
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define DEBUG 0
#define MAXN 1000003
#define mod 1000000007

vector<int> prime;

void segmentedsieve(int l, int r)
{
  int lim = sqrt(r);
  vector<bool> mark(lim + 1, false);
  vector<int> primes;
  for (int i = 2; i <= lim; ++i)
  {
    if (!mark[i])
    {
      primes.pb(i);
      for (int j = i * i; j <= lim; j += i)
        mark[j] = true;
    }
  }
  vector<bool> isprime(r - l + 1, true);
  for (int i : primes)
```

```
    for (int j = max(i * i, (l + i - 1) / i * i); j <= r
        ; j += i)
      isprime[j - l] = false;
  if (l == 1)
    isprime[0] = false;
  for (int i = 0; i < isprime.size(); i++)
    if (isprime[i])
      prime.pb(i + l);
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int l, r;
  cin >> l >> r;
  segmentedsieve(l, r);
  for (auto const &i : prime)
    cout << i << " ";
  return 0;
}
```

5.12 pollard rho

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define int __int128
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 1000001
#define mod 998244353

int read() // __int128 functions
{
  int x = 0, f = 1;
  char ch = getchar();
  while (ch < '0' || ch > '9')
  {
    if (ch == '-')
      f = -1;
    ch = getchar();
  }
  while (ch >= '0' && ch <= '9')
  {
    x = x * 10 + ch - '0';
    ch = getchar();
  }
  return x * f;
}
void print(__int128 x) // __int128 functions
{
  if (x < 0)
  {
    cout << "-";
    x = -x;
  }
  stack<char> s;
  while (x)
  {
    s.push((x % 10) + '0');
    x = x / 10;
  }
  while (!s.empty())
  {
    cout << s.top();
    s.pop();
  }
}
namespace pollard_rho
{
  int multiplicate(int x, int y, int m)
  {
    return (x * y) % m;
  }
  int modpow(int x, int y, int m)
  {
    int z = 1;
    while (y)
    {
      if (y & 1)
        z = (z * x) % m;
      x = (x * x) % m;
      y >>= 1;
    }
    return z;
  }
  bool is_composite(int n, int a, int d, int s)
  {
    int x = modpow(a, d, n);
    if (x == 1 or x == n - 1)
```

```
    return false;
  for (int r = 1; r < s; r++)
  {
    x = multiplicate(x, x, n);
    if (x == n - 1LL)
      return false;
  }
  return true;
};
int miller_rabin(int n)
{
  if (n < 2)
    return false;
  int r = 0, d = n - 1LL;
  while ((d & 1LL) == 0)
  {
    d >>= 1;
    r++;
  }
  for (int a : {2, 3, 5, 7, 11, 13, 17, 19, 23, 29,
     31, 37})
  {
    if (n == a)
      return true;
    if (is_composite(n, a, d, r))
      return false;
  }
  return true;
}
int f(int x, int m)
{
  return multiplicate(x, x, m) + 1;
}
int rho(int n)
{
  int x0 = 1, t = 0, prd = 2;
  int x = 0, y = 0, q;
  while (t % 40 || __gcd(prd, n) == 1)
  {
    if (x == y)
    {
      x0++;
      x = x0;
      y = f(x, n);
    }
    q = multiplicate(prd, max(x, y) - min(x, y), n);
    if (q != 0)
      prd = q;
    x = f(x, n);
    y = f(y, n);
    t++;
  }
  return __gcd(prd, n);
}
vector<int> fact(int n)
{
  if (n == 1)
    return {};
  if (miller_rabin(n))
    return {n};
  int x = rho(n);
  auto l = fact(x), r = fact(n / x);
  l.insert(l.end(), r.begin(), r.end());
  return l;
}
}
signed main()
{
  //ios_base::sync_with_stdio(false);
  //cin.tie(NULL);
  while (1)
  {
    int n = read();
    if (n == 0)
      break;
    vector<int> factors = pollard_rho::fact(n);
    sort(factors.begin(), factors.end());
    int prev = -1, cnt = 0;
    for (auto const &i : factors)
    {
      if (prev != i)
      {
        if (prev != -1)
        {
          print(prev);
          printf("^");
          print(cnt);
          printf(" ");
        }
        prev = i;
        cnt = 0;
      }
      cnt++;
    }
    if (prev != -1)
    {
      print(prev);
      printf("^");
      print(cnt);
```

```
      printf(" ");
    }
    printf("\n");
  }
  return 0;
}
// sources:
// https://github.com/PauloMiranda98/Competitive-
   Programming-Notebook/blob/master/code/math/prime.h
// https://github.com/brunomaletta/Biblioteca/blob/
   master/Codigo/Matematica/pollardrho.cpp
// fast integer factorization with pollard-rho
// https://www.spoj.com/problems/FACT0/ - ok
// https://www.spoj.com/problems/FACT1/ - ok
// https://www.spoj.com/problems/FACT2/ - sigkill
// since the limit is at most 29 digits(in FACT2), we
   need to use __int128
```

5.13 xor trie

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
   rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 500001
#define mod 1000000007

struct node
{
  int me, cnt, id;
  int down[2];
  node(int c = 0) : me(c)
  {
    cnt = 0;
    id = -1;
    fill(begin(down), end(down), -1);
  }
};
struct trie_xor
{
  vector<node> t;

  trie_xor()
  {
    t.resize(1);
  }
  void add(int n, int id)
  {
    int v = 0;
    for (int i = 30; i >= 0; i--)
    {
      int bit = (n & (1 << i)) ? 1 : 0;
      if (t[v].down[bit] == -1)
      {
        t[v].down[bit] = t.size();
        t.emplace_back(bit);
      }
      v = t[v].down[bit];
      t[v].cnt++;
    }
    t[v].id = id;
  }
  void rem(int n, int id)
  {
    int v = 0;
    for (int i = 30; i >= 0; i--)
    {
      int bit = (n & (1 << i)) ? 1 : 0;
      v = t[v].down[bit];
      t[v].cnt--;
    }
  }
  int qry(int n) // maximum xor with n
  {
    int v = 0;
    for (int i = 30; i >= 0; i--)
    {
      int bit = (n & (1 << i)) ? 0 : 1;
      int nxt = t[v].down[bit];
      if (nxt != -1 && t[nxt].cnt > 0)
        v = nxt;
      else
        v = t[v].down[bit ^ 1];
    }
    return t[v].id;
  }
};
```

```
signed main()
{
}
// alguns problemas:
// https://codeforces.com/problemset/problem/706/D
// https://codeforces.com/contest/1625/problem/D
// https://codeforces.com/contest/888/problem/G
```

5.14 mobius

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 5000005
#define mod 1000000001

int lpf[MAXN];
int mobius[MAXN];
int g[MAXN];

void calc_lpf()
{
  for (int i = 2; i < MAXN; i++)
  {
    if (!lpf[i])
    {
      for (int j = i; j < MAXN; j += i)
      {
        if (!lpf[j])
          lpf[j] = i;
      }
    }
  }
}
void calc_mobius()
{
  calc_lpf();
  mobius[1] = 1;
  for (int i = 2; i < MAXN; i++)
  {
    if (lpf[i / lpf[i]] == lpf[i])
      mobius[i] = 0;
    else
      mobius[i] = -1 * mobius[i / lpf[i]];
  }
}
int count_pairs(int n)
{
  // f(n) -> contar pares (i, j) com __gcd(i, j) == 1 e
     1 <= i, j <= n
  int ans = 0;
  for (int d = 1; d <= n; d++)
  {
    // quadrado pq sao pares (2 caras)
    // mas se fossem x caras seria (n / d)^x
    int sq = (n / d) * (n / d);
    int x = mobius[d] * sq;
    ans += x;
  }
  return ans;
}
int gcd_sum(int n)
{
  // soma de todos os gcd(i, j) com 1 <= i, j <= n
  int ans = 0;
  for (int k = 1; k <= n; k++) // fixa o valor do gcd(i,
      j) e conta quantos pares com gcd(i, j) == k
  {
    int lim = n / k;
    int curr = k * count_pairs(lim);
    ans += curr;
  }
  return ans;
}
int lcm_sum(int n)
{
  // soma de todos os lcm(i, j) com 1 <= i, j <= n
  for (int i = 1; i <= n; i++)
    g[i] = 0;
  for (int i = 1; i <= n; i++)
  {
    for (int j = i; j <= n; j += i)
      g[j] += (mobius[i] * j * i);
  }
  int ans = 0;
  for (int l = 1; l <= n; l++)
```

```
    {
      int cima = (1 + n / l) * (n / l);
      int f = (cima / 2) * (cima / 2);
      f *= g[l];
      ans += f;
    }
    return ans;
}
signed main()
{
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);
    int q;
    cin >> q;
    calc_mobius();
    for (int i = 1; i <= q; i++)
    {
      int n;
      cin >> n;
      int ans = lcm_sum(n);
      for (int i = 1; i <= n; i++)
        ans -= i;
      ans /= 2;
      cout << "Case " << i << ": " << ans << endl;
    }
    return 0;
}
// https://codeforces.com/blog/entry/53925
// mobius inversion
// sejam f(x) e g(x) funcoes
// e g(x) e definida da seguinte maneira
// g(x) = soma dos f(d), no qual d eh um divisor de x

// temos que:
// f(n) = soma dos (g(d) * u(n / d)), no qual d eh um
    divisor de x
// u(x) -> mobius function

// propiedade legal:
// seja l(x) -> soma de u(d), para cada divisor d de x
// l(1) = 1
// l(x) = 0, x > 1

// problemas iniciais:
// https://vjudge.net/problem/AtCoder-abc162_e
// https://vjudge.net/problem/CodeChef-SMPLSUM
```

5.15 fft

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define PI acos(-1)
#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<pi, int>
#define fir first
#define sec second
#define DEBUG 0
#define MAXN 100001
#define mod 1000000007
#define cd complex<double> // numeros complexos na STL

void dft(vector<cd> &a)
{
  int n = a.size();
  if (n == 1)
    return;
  vector<cd> a0(n / 2), a1(n / 2);
  for (int i = 0; 2 * i < n; i++)
  {
    a0[i] = a[2 * i];
    a1[i] = a[2 * i + 1];
  }
  dft(a0);
  dft(a1);
  double ang = 2 * PI / n;
  cd w(1), wn(cos(ang), sin(ang));
  for (int i = 0; 2 * i < n; i++)
  {
    a[i] = a0[i] + w * a1[i];
    a[i + n / 2] = a0[i] - w * a1[i];
    w *= wn;
  }
}
void inverse_dft(vector<cd> &a)
{
  int n = a.size();
  if (n == 1)
    return;
  vector<cd> a0(n / 2), a1(n / 2);
  for (int i = 0; 2 * i < n; i++)
```

```
  {
    a0[i] = a[2 * i];
    a1[i] = a[2 * i + 1];
  }
  inverse_dft(a0);
  inverse_dft(a1);
  double ang = 2 * PI / n * -1;
  cd w(1), wn(cos(ang), sin(ang));
  for (int i = 0; 2 * i < n; i++)
  {
    a[i] = a0[i] + w * a1[i];
    a[i + n / 2] = a0[i] - w * a1[i];
    a[i] /= 2;
    a[i + n / 2] /= 2;
    w *= wn;
  }
}
vector<int> fft(vector<int> a, vector<int> b)
{
  vector<cd> fa(a.begin(), a.end()), fb(b.begin(), b.end
      ());
  int n = 1;
  while (n < a.size() + b.size())
    n <<= 1;
  fa.resize(n);
  fb.resize(n);
  dft(fa);                      // DFT(A)
  dft(fb);                      // DFT(B)
  for (int i = 0; i < n; i++) // DFT(A * B) = DFT(A) *
      DFT(B)
    fa[i] *= fb[i];
  inverse_dft(fa); // inverseDFT(DFT(A * B))
  vector<int> ans(n);
  for (int i = 0; i < n; i++)
    ans[i] = round(fa[i].real()); // arredondar para ter
        os coeficientes como inteiros
  return ans;
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n, m, caso = 1;
  while (cin >> n >> m)
  {
    cout << "Caso #" << caso << ": ";
    vector<int> a(n + 1);
    vector<int> b(m + 1);
    for (int i = 0; i <= n; i++)
      cin >> a[i];
    for (int i = 0; i <= m; i++)
      cin >> b[i];
    vector<int> ans = fft(a, b);
    for (int i = 0; i <= n + m; i++)
    {
      cout << ans[i];
      (i == n + m) ? cout << endl : cout << " ";
    }
    caso++;
  }
  return 0;
}
// fft
// multiplicar dois polinomios A e B
// basic approach:
// aplicar a propiedade distributiva e fazer essa
    multiplicacao em O(N^2)
// porem podemos melhorar
// vamos la
// 1  - todo polinomio de grau d que e representado na
    forma de coeficientes
//      de coeficientes possui uma representacao em
    forma de d - 1 pontos
// 2  - para esse conjunto de pontos, so existe um unico
     polinomio equivalente
// 3  - DFT -> transformacao da representacao de
    coeficientes para represntacao
//             de pontos
// 4  - com isso, para multiplicar os dois polinomios
    agora basta multiplicar
//      os conjuntos de pontos e com isso obtemos a
    representacao usando pontos
//      do polinomio resultante
// 5  - DFT(A * B) = DFT(A) * DFT(B);
// 6  - porem agora precisamos transformar a resposta
    obtida na multiplicacao dos pontos
//      para a representacao em que usa os coeficientes
// 7  - inverseDFT -> transformacao da representacao de
    pontos para represntacao
//                  de coeficientes
// 8  - A * B = inverseDFT(DFT(A) * DFT(B))
// 9  - FFT -> metodo para computar a DFT em O(N * low(N
    ))
// 10 - iremos usar divide and conquer para isso, vamos
    splitar o polinomio
//      atual em 2 polinomos de grau ((n / 2) - 1) , tal
     que, a soma deles
//      resulte no polinomio que tinhamos antes
// 11 - agora para achar a inverseDFT de uma DFT, iremos
     escrever a DFT
```

```
//      em forma de matriz, essa matriz e chamada de
    matriz de vandermonde
//      e em geral, podemos escrever a resposta como uma
     multiplicacao de
//      matrizes
// 12 - essa multplicacao de matrizes pode ser descrita
    como:
//      a^-1 * b = c
//      no qual:
//      a^-1 -> inversa da matriz a(DFT)
//      b -> valores dos coeficientes do polinomio A
//      c -> valores dos coeficientes da resposta
```

5.16 lagrange

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 600005
#define mod 1000000007

struct modint
{
  int val;
  modint(int v = 0) { val = v % mod; }
  int pow(int y)
  {
    modint x = val;
    modint z = 1;
    while (y)
    {
      if (y & 1)
        z *= x;
      x *= x;
      y >>= 1;
    }
    return z.val;
  }
  int inv() { return pow(mod - 2); }
  void operator=(int o) { val = o % mod; }
  void operator=(modint o) { val = o.val % mod; }
  void operator+=(modint o) { *this = *this + o; }
  void operator-=(modint o) { *this = *this - o; }
  void operator*=(modint o) { *this = *this * o; }
  void operator/=(modint o) { *this = *this / o; }
  bool operator==(modint o) { return val == o.val; }
  bool operator!=(modint o) { return val != o.val; }
  int operator*(modint o) { return ((val * o.val) % mod)
      ; }
  int operator/(modint o) { return (val * o.inv()) % mod
      ; }
  int operator+(modint o) { return (val + o.val) % mod;
      }
  int operator-(modint o) { return (val - o.val + mod) %
       mod; }
};
struct lagrange
{
  vector<modint> den;
  vector<modint> y;
  vector<modint> fat;
  vector<modint> inv_fat;

  lagrange(vector<modint> &v) // f(i) = v[i], gera um
      polinomio de grau n - 1
  {
    int n = v.size();
    calc(n);
    den.resize(n);
    y.resize(n);
    for (int i = 0; i < n; i++)
    {
      y[i] = v[i];
      den[i] = inv_fat[n - i - 1] * inv_fat[i];
      if ((n - i - 1) % 2 == 1)
      {
        int x = (mod - den[i].val) % mod;
        den[i] = x;
      }
    }
  }
  void calc(int n)
  {
    fat.resize(n + 1);
    inv_fat.resize(n + 1);
    fat[0] = 1;
```

```
    inv_fat[0] = 1;
    for (int i = 1; i <= n; i++)
    {
      fat[i] = fat[i - 1] * i;
      inv_fat[i] = fat[i].inv();
    }
  }
  modint get_val(int x) // complexity: O(n)
  {
    x %= mod;
    int n = y.size();
    vector<modint> l(n);
    vector<modint> r(n);
    l[0] = 1, r[n - 1] = 1;
    for (int i = 1; i < n; i++)
    {
      modint cof = (x - (i - 1) + mod);
      l[i] = l[i - 1] * cof;
    }
    for (int i = n - 2; i >= 0; i--)
    {
      modint cof = (x - (i + 1) + mod);
      r[i] = r[i + 1] * cof;
    }
    modint ans = 0;
    for (int i = 0; i < n; i++)
    {
      modint cof = l[i] * r[i];
      ans += modint(cof * y[i]) * den[i];
    }
    return ans;
  }
};
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n, k;
  cin >> n >> k;
  vector<modint> v;
  v.pb(0);
  int lim = k + 1;
  for (int i = 1; i <= lim; i++)
    v.pb(v.back() + modint(i).pow(k));
  lagrange l(v);
  cout << l.get_val(n).val << endl;
  return 0;
}
// https://codeforces.com/contest/622/problem/F
```

5.17 primefactors

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

#define PI acos(-1)
#define int long long int
#define pb push_back
#define mp make_pair
#define pi pair<int, int>
#define fir first
#define sec second
#define MAXN 501
#define MAXL 20
#define mod 1000000007

vector<int> facts;
void primefactors(int n)
{
  while (n % 2 == 0)
  {
    facts.pb(2);
    n = n / 2;
  }
  for (int i = 3; i <= sqrt(n); i += 2)
  {
    while (n % i == 0)
    {
      facts.pb(i);
      n = n / i;
    }
  }
  if (n > 2)
    facts.pb(n);
}
signed main()
{
  int n;
  cin >> n;
  primefactors(n);
  sort(facts.begin(), facts.end());
  for (auto const &i : facts)
    cout << i << endl;
  return 0;
}
```

5.18 operadores binarios

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

#define lli long long int
#define pb push_back
#define in insert
#define pi pair<int, int>
#define pd pair<double, int>
#define pii pair<int, pi>
#define mp make_pair
#define fir first
#define sec second
#define MAXN 200001
#define mod 1000000007

void shifts ()
{
  bitset <4> bs;
  bs.reset();
  bs[2] = true;
  bs[3] = true;
  cout << bs << endl ; // 1100
  bs >>= 1; // 0110
  bs <<= 1; // 1100
  bs >>= 2; // 0011
  bs <<= 2; // 1100
  bs >>= 3; // 0001
  bs <<= 3; // 1000
  cout << bs << endl ;
}
void op_xor ()
{
  // 0 ^ 0 = 0
  // 0 ^ 1 = 1
  // 1 ^ 0 = 1
  // 1 ^ 1 = 0
  bitset <4> bs , bs2;
  bs.reset();
  bs2.reset();
  bs[2] = true;
  bs[3] = true;
  bs2[1] = true;
  bs2[3] = true;
  bs ^= bs2; // bs = bs ^ bs2
  cout << bs.count() << endl ;
}
void op_and ()
{
  // 0 & 0 = 0
  // 0 & 1 = 0
  // 1 & 0 = 0
  // 1 & 1 = 1
  bitset <4> bs , bs2;
  bs.reset();
  bs2.reset();
  bs[2] = true;
  bs[3] = true;
  bs2[1] = true;
  bs2[3] = true;
  bs &= bs2; // bs = bs & bs2
  cout << bs.count() << endl ;
}
void op_or ()
{
  // 0 | 0 = 0
  // 0 | 1 = 1
  // 1 | 0 = 1
  // 1 | 1 = 1
  bitset <4> bs , bs2;
  bs.reset(); // poe tudo 0
  bs2.reset();
  bs[2] = true;
  bs[3] = true;
  bs2[1] = true;
  bs2[3] = true;
  bs |= bs2; // bs = bs | bs2
  cout << bs.count() << endl ; // quantidade de 1
}
signed main()
{
  op_or();
  op_and();
  op_xor();
  shifts();
  return 0;
}
```

5.19 matrix exponentiation

```cpp
// https://codeforces.com/gym/102644/problem/C
// achar o n-esimo termo da sequencia de fibonacci mod
   (10^9 + 7) em O(log(n))
// n <= 10^18
// podemos escrever a recorrencia de fibonnaci como uma
   exponenciacao de matriz
/*
  (  fib(n)  )       (1 1) ^ (n - 1)       (fib(1) = 1)
  (fib(n - 1))   =   (1 0)              *  (fib(0) = 1)
*/
```

```
// e possivel fazer essa exponenciacao em O(log(n)) com
    um algoritimo muito similar ao de exponenciacao
    rapida
// dai calculamos o n-esimo termo da sequencia de
    fibonacci mod (10^9 + 7) em O(log(n))

#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define PI acos(-1)
#define pb push_back
#define int long long int
#define pi pair<int, int>
#define pii pair<pi, int>
#define fir first
#define sec second
#define DEBUG 0
#define MAXN 201
#define mod 1000000007

namespace matrix
{
  vector<vector<int>> ans;

  int multi(int x, int y)
  {
    return (x * y) % mod;
  }
  int sum(int a, int b)
  {
    return (a + b >= mod) ? a + b - mod : a + b;
  }
  vector<vector<int>> multiply(vector<vector<int>> a,
      vector<vector<int>> b)
  {
    vector<vector<int>> res(a[0].size(), vector<int>(b
        [0].size()));
    for (int i = 0; i < a.size(); i++)
    {
      for (int j = 0; j < b[0].size(); j++)
      {
        res[i][j] = 0;
        for (int k = 0; k < a[0].size(); k++)
          res[i][j] = sum(res[i][j], multi(a[i][k], b[k
              ][j]));
      }
    }
    return res;
  }
  vector<vector<int>> expo(vector<vector<int>> mat, int
      m)
  {
    ans = vector<vector<int>>(mat.size(), vector<int>(
        mat[0].size()));
    for (int i = 0; i < mat.size(); i++)
      for (int j = 0; j < mat[0].size(); j++)
        ans[i][j] = (i == j);
    while (m > 0)
    {
      if (m & 1)
        ans = multiply(ans, mat);
      m = m / 2;
      mat = multiply(mat, mat);
    }
    return ans;
  }
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n;
  cin >> n;
  vector<vector<int>> mat = {{1, 1}, {1, 0}};
  vector<vector<int>> ans = matrix::expo(mat, n);
  cout << ans[0][1] << endl;
  return 0;
}
```

5.20 diophantine

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define pb push_back
```

```
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 200001
#define mod 998244353

namespace dio
{
  vector<pi> sols;

  int gcd(int a, int b, int &x, int &y)
  {
    if (b == 0)
    {
      x = 1, y = 0;
      return a;
    }
    int x1, y1, d = gcd(b, a % b, x1, y1);
    x = y1, y = x1 - y1 * (a / b);
    return d;
  }
  void one_sol(int a, int b, int c)
  {
    int x0, y0, g;
    g = gcd(abs(a), abs(b), x0, y0);
    if (c % g)
      return;
    x0 *= (c / g);
    y0 *= (c / g);
    if (a < 0)
      x0 *= -1;
    if (b < 0)
      y0 *= -1;
    sols.pb({x0, y0});
  }
  void more_sols(int a, int b, int c)
  {
    int g = __gcd(a, b);
    int x0 = sols[0].fir, y0 = sols[0].sec;
    for (int k = -200000; k <= 200000; k++)
    {
      int x = x0 + k * (b / g);
      int y = y0 - k * (a / g);
      sols.pb({x, y});
    }
  }
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int a, b, c;
  cin >> a >> b >> c;
  dio::one_sol(a, b, c);
  if (!dio::sols.size())
  {
    cout << "No\n";
    return 0;
  }
  dio::more_sols(a, b, c);
  bool can = false;
  for (auto const &i : dio::sols)
    can |= (i.fir >= 0 && i.sec >= 0);
  (can) ? cout << "Yes\n" : cout << "No\n";
  return 0;
}
// equacoes do tipo:
// ax + by = c
// o caso a = 0 e b = 0, nao eh tratado nesse codigo
// nesse caso quero checar se equacao diofantina tem uma
    solucao
// com x >= 0 e y >= 0
```

5.21 crt

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
   rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<pi, int>
#define fir first
#define sec second
#define MAXN 2000006
//#define mod 1000000007

namespace crt
{
  vector<pi> eq;

  int gcd(int a, int b, int &x, int &y)
```

```
  {
    if (b == 0)
    {
      x = 1, y = 0;
      return a;
    }
    int x1, y1, d = gcd(b, a % b, x1, y1);
    x = y1, y = x1 - y1 * (a / b);
    return d;
  }
  pi crt()
  {
    int a1 = eq[0].fir, m1 = eq[0].sec;
    a1 %= m1;
    for (int i = 1; i < eq.size(); i++)
    {
      int a2 = eq[i].fir, m2 = eq[i].sec;
      int g = __gcd(m1, m2);
      if (a1 % g != a2 % g)
        return {-1, -1};
      int p, q;
      gcd(m1 / g, m2 / g, p, q);
      int mod = m1 / g * m2;
      int x = (a1 * (m2 / g) % mod * q % mod + a2 * (m1
          / g) % mod * p % mod) % mod;
      a1 = x;
      if (a1 < 0)
        a1 += mod;
      m1 = mod;
    }
    return {a1, m1};
  }
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n;
  cin >> n;
  for (int i = 0; i < n; i++)
  {
    int a, b;
    cin >> a >> b;
    crt::eq.pb({a, b});
  }
  pi ans = crt::crt();
  if (ans.fir == -1)
    cout << "No solution\n";
  else
    cout << ans.fir << " " << ans.sec << endl;
  return 0;
}
// references:
// https://forthright48.com/chinese-remainder-theorem-
    part-2-non-coprime-moduli/
// https://cp-algorithms.com/algebra/chinese-remainder-
    theorem.html
// https://www.geeksforgeeks.org/chinese-remainder-
    theorem-set-1-introduction/

// teorema chines do resto(crt)
// para resolver sistemas de congruencias modulares
// o menor inteiro a que satisfaz:
// a mod p1 = x1
// a mod p2 = x2
// ...
// a mod pn = xn
// a funcao crt retorna um pair {a, mod}
// dai a solucao pode ser descrita como
// x = a % mod
// entao os valores possiveis sao:
// a, (a + mod), a + (2 * mod), a + (3 * mod), ...
```

5.22 ntt

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 250005
#define mod 998244353

struct modint
{
  int val;
  modint(int v = 0) { val = ((v % mod) + mod) % mod; }
  int pow(int y)
```

```
  {
    modint x = val;
    modint z = 1;
    while (y)
    {
      if (y & 1)
        z *= x;
      x *= x;
      y >>= 1;
    }
    return z.val;
  }
  int inv() { return pow(mod - 2); }
  void operator=(int o) { val = o % mod; }
  void operator=(modint o) { val = o.val % mod; }
  void operator+=(modint o) { *this = *this + o; }
  void operator-=(modint o) { *this = *this - o; }
  void operator*=(modint o) { *this = *this * o; }
  void operator/=(modint o) { *this = *this / o; }
  bool operator==(modint o) { return val == o.val; }
  bool operator!=(modint o) { return val != o.val; }
  int operator*(modint o) { return ((val * o.val) % mod)
      ; }
  int operator/(modint o) { return (val * o.inv()) % mod
      ; }
  int operator+(modint o) { return (val + o.val) % mod;
      }
  int operator-(modint o) { return (val - o.val + mod) %
       mod; }
};
namespace fft
{
  // para o modulo ser valido
  // precisa ser primo
  // precisa possuir a forma c * 2^k + 1
  // 998244353 - possui a forma - c * 2^k + 1 e eh primo
  int n;
  int root = -1;
  int root_1 = -1;
  int pw = __builtin_ctz(mod - 1);
  int root_pw = (1 << pw);

  void find_root()
  {
    if (root != -1)
      return;
    int r = 2;
    while (!(modint(r).pow((1 << pw)) == 1 && modint(r).
        pow((1 << (pw - 1))) != 1))
      r++;
    root = r;
    root_1 = modint(root).inv();
  }
  void ntt(vector<modint> &a, bool invert)
  {
    find_root();
    int n = a.size();
    for (int i = 1, j = 0; i < n; i++)
    {
      int bit = n >> 1;
      for (; j & bit; bit >>= 1)
        j ^= bit;
      j ^= bit;
      if (i < j)
        swap(a[i], a[j]);
    }
    for (int len = 2; len <= n; len <<= 1)
    {
      modint wlen = (invert) ? root_1 : root;
      for (int i = len; i < root_pw; i <<= 1)
        wlen *= wlen;
      for (int i = 0; i < n; i += len)
      {
        modint w = 1;
        for (int j = 0; j < len / 2; j++)
        {
          modint u = a[i + j];
          modint v = a[i + j + len / 2] * w;
          a[i + j] = u + v;
          a[i + j + len / 2] = u - v;
          w *= wlen;
        }
      }
    }
    if (invert)
    {
      modint n_1 = modint(n).inv();
      for (int i = 0; i < a.size(); i++)
        a[i] *= n_1;
    }
  }
  vector<modint> mul(vector<modint> a, vector<modint> b)
  {
    n = 1;
    while (n < 2 * max(a.size(), b.size()))
      n <<= 1;
    a.resize(n);
    b.resize(n);
    ntt(a, false);
    ntt(b, false);
```

```
    for (int i = 0; i < n; i++)
      a[i] *= b[i];
    ntt(a, true);
    return a;
  }
} // namespace fft
  // https://codeforces.com/contest/1613/problem/F
```

6 Dynamic programming and common problems

6.1 cht

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 1000005
#define mod 1000000007

struct line
{
  int m, b, p;
  line(int m, int b) : m(m), b(b) {}
  bool operator<(const line &o) const
  {
    if (m != o.m)
      return m > o.m;
    return b > o.b;
  }
  bool operator<(const int x) const { return p < x; }
  int eval(int x) const { return m * x + b; }
  int inter(const line &o) const
  {
    int x = b - o.b, y = o.m - m;
    return (x / y) - ((x ^ y) < 0 && x % y);
  }
};
struct cht
{
  deque<line> a;
  cht() {}
  int eval(int i, int x)
  {
    return a[i].m * x + a[i].b;
  }
  void add(line l)
  {
    while (1)
    {
      if (a.size() >= 1 && a[0].m == l.m && l.b > a[0].b
          )
      {
        a.pop_front();
      }
      else if (a.size() >= 1 && a[0].m == l.m && l.b <=
          a[0].b)
      {
        break;
      }
      else if (a.size() >= 2 && a[0].inter(l) >= a[1].
          inter(a[0]))
      {
        a.pop_front();
      }
      else
      {
        a.push_front(l);
        break;
      }
    }
  }
  int get(int x)
  {
    while (a.size() >= 2 && eval(a.size() - 1, x) <=
        eval(a.size() - 2, x))
      a.pop_back();
    return eval(a.size() - 1, x);
  }
};
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  return 0;
}
// cht
```

```
// queries ordenadas em ordem decrescente
// linhas ordenadas em ordem decrescente
```

6.2 subsequences string

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

#define PI acos(-1)
#define int long long int
#define pb push_back
#define mp make_pair
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 100
#define MAXL 20
#define mod 998244353

void count(string a, string b)
{
  int m = a.size();
  int n = b.size();
  int dp[m + 1][n + 1] = {{0}};
  for (int i = 0; i <= n; ++i)
    dp[0][i] = 0;
  for (int i = 0; i <= m; ++i)
    dp[i][0] = 1;
  for (int i = 1; i <= m; i++)
  {
    for (int j = 1; j <= n; j++)
    {
      if (a[i - 1] == b[j - 1])
        dp[i][j] = dp[i - 1][j - 1] + dp[i - 1][j];
      else
        dp[i][j] = dp[i - 1][j];
    }
  }
  cout << dp[m][n] << endl;
}
signed main()
{
  string a, b;
  cin >> a >> b;
  count(a, b);
  return 0;
}
```

6.3 sos dp

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 100001
#define mod 1000000007

signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  // exemplos de sos dp para calcular f[x] para cada
     mask x
  // a[x] eh o valor de uma funcao a para uma mask x
  // complexidade: O(M * 2^M), M = numero de bits

  // Exemplo 1:
  // nesse caso, f[x] eh a funcao que soma:
  // todos os a[i], tal que, (x & i) == i)
  // isso eh, i eh uma "mask filha" de x
  // pois todos os bits de i estao setados em x
  for (int mask = 0; mask < (1 << m); mask++)
  {
    f[mask] = a[mask];
  }
  for (int i = 0; i < m; ++i)
  {
    for (int mask = 0; mask < (1 << m); mask++)
    {
      if (mask & (1 << i))
        f[mask] += f[mask ^ (1 << i)];
    }
  }
```

```
  // Exemplo 2:
  // nesse caso, f[x] eh a funcao que soma:
  // todos os a[i], tal que, (x & i) == x)
  // isso eh, i eh uma "mask pai" de x
  // pois todos os bits de x estao setados em i
  for (int mask = 0; mask < (1 << m); mask++)
  {
    f[mask] = a[mask];
  }
  for (int i = 0; i < m; ++i)
  {
    for (int mask = 0; mask < (1 << m); mask++)
    {
      if (!(mask & (1 << i)))
        f[mask] += f[mask ^ (1 << i)];
    }
  }

  return 0;
}
// https://codeforces.com/blog/entry/45223
```

6.4 aliens trick

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 500001
#define mod 1000000007

int n, k, l;
string s;

pi solve(vector<int> &v, int lambda)
{
  // associar um custo lambda para ser subtraido quando
     realizamos uma operacao
  // dp[i] - melhor profit que tivemos considerando as i
     primeiras posicoes
  // cnt[i] - quantas operacoes utilizamos para chegarno
     valor de dp[i]
  vector<int> dp(n + 1);
  vector<int> cnt(n + 1);
  dp[0] = 0;
  cnt[0] = 0;
  for (int i = 1; i <= n; i++)
  {
    dp[i] = dp[i - 1];
    cnt[i] = cnt[i - 1];
    int id = i - 1;
    dp[i] += v[id];
    int lo = max(0ll, id - l + 1);
    int s = dp[lo] + (id - lo + 1) - lambda;
    if (s > dp[i])
    {
      dp[i] = s;
      cnt[i] = cnt[lo] + 1;
    }
  }
  return {dp[n], cnt[n]};
}
int aliens_trick(vector<int> &v)
{
  int l = 0, r = n;
  while (l < r)
  {
    int mid = (l + r) >> 1;
    pi ans = solve(v, mid);
    (ans.sec > k) ? l = mid + 1 : r = mid;
  }
  pi ans = solve(v, l);
  return ans.fir + (l * k);
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  cin >> n >> k >> l >> s;
  vector<int> a(n);
  vector<int> b(n);
  for (int i = 0; i < n; i++)
  {
    a[i] = 1, b[i] = 0;
    if (s[i] >= 'A' && s[i] <= 'Z')
    {
```

```
      a[i] ^= 1;
      b[i] ^= 1;
    }
  }
  cout << n - max(aliens_trick(a), aliens_trick(b)) <<
      endl;
  return 0;
}
// https://codeforces.com/contest/1279/problem/F
```

6.5 largest square

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

#define PI acos(-1)
#define pb push_back
#define int long long int
#define double long double
#define pi pair<int, int>
#define pii pair<pi, int>
#define fir first
#define sec second
#define MAXN 1001
#define mod 1000000007

signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n;
  cin >> n;
  int v[n][n];
  int dp[n][n];
  for (int i = 0; i < n; i++)
    for (int j = 0; j < n; j++)
      cin >> v[i][j];
  int ans = 0;
  for (int i = 0; i < n; i++)
  {
    for (int j = 0; j < n; j++)
    {
      dp[i][j] = v[i][j];
      if (i && j && dp[i][j])
        dp[i][j] = min({dp[i][j - 1], dp[i - 1][j], dp[i
            - 1][j - 1]}) + 1;
      ans = max(ans, dp[i][j]);
    }
  }
  cout << ans * ans << endl;
  return 0;
}
```

6.6 broken profile

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define PI acos(-1)
#define pb push_back
#define int long long int
#define pi pair<int, int>
#define pii pair<int, pair<int, pi>>
#define fir first
#define sec second
#define DEBUG 0
#define MAXN 1001
#define mod 1000000007

int n;
vector<int> validmasks;
int dp[MAXN][1 << 4];

void init() // preprocess valid masks
{
  for (int mask = 0; mask < (1 << 7); mask++)
  {
    int nxt_mask = 0, prev_mask = 0, valid = true;
    for (int k = 0; k < 7; k++)
    {
      if (mask & (1 << k))
      {
        if (k <= 3)
        {
          int idx = k, idx2 = k;
          if (nxt_mask & (1 << idx) || prev_mask & (1 <<
              idx2))
            valid = false;
          prev_mask = prev_mask | (1 << idx);
          nxt_mask = nxt_mask | (1 << idx2);
        }
        else
        {
```

```
          int idx = k - 4, idx2 = idx + 1;
          if (nxt_mask & (1 << idx) || nxt_mask & (1 <<
              idx2))
            valid = false;
          nxt_mask = nxt_mask | (1 << idx);
          nxt_mask = nxt_mask | (1 << idx2);
        }
      }
    }
    if (valid)
      validmasks.pb(mask);
  }
}
int solve(int i, int j)
{
  if (i == n)
    return (j == ((1 << 4) - 1)) ? 1 : 0;
  if (dp[i][j] != -1)
    return dp[i][j];
  int ret = 0;
  for (auto const &mask : validmasks)
  {
    int nxt_mask = 0, prev_mask = j, valid = true;
    for (int k = 0; k < 7; k++)
    {
      if (mask & (1 << k))
      {
        if (k <= 3)
        {
          int idx = k, idx2 = idx;
          if (prev_mask & (1 << idx) || nxt_mask & (1 <<
              idx2))
            valid = false;
          prev_mask = prev_mask | (1 << idx);
          nxt_mask = nxt_mask | (1 << idx2);
        }
        else
        {
          int idx = k - 4, idx2 = idx + 1;
          if (nxt_mask & (1 << idx) || nxt_mask & (1 <<
              idx2))
            valid = false;
          nxt_mask = nxt_mask | (1 << idx);
          nxt_mask = nxt_mask | (1 << idx2);
        }
      }
    }
    if (valid && prev_mask == ((1 << 4) - 1))
      ret += solve(i + 1, nxt_mask);
  }
  return dp[i][j] = ret;
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int q;
  cin >> q;
  init();
  for (int i = 1; i <= q; i++)
  {
    cin >> n;
    memset(dp, -1, sizeof(dp));
    cout << i << " " << solve(0, (1 << 4) - 1) << endl;
  }
  return 0;
}
// broken profile dp
// if you can fully fill an area with some figures
// finding number of ways to fully fill an area with
    some figures
// finding a way to fill an area with minimum number of
    figures
// ...
// https://www.spoj.com/problems/GNY07H/
// We wish to tile a 4xN grid with rectangles 2x1 (in
    either orientation)
// dp[i][mask]
// i denotes the current column
// mask denotes the situation of the previous column
// our mission is to fill all of the units of
// the previous column in a state [i][mask]
```

6.7 max matrix path

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

#define PI acos(-1)
#define pb push_back
#define int long long int
#define mp make_pair
#define pi pair<int, int>
#define pii pair<pi, int>
#define fir first
#define sec second
#define MAXN 301
#define MAXL 20
#define mod 1000000007
#define INF 1000000001
```

```
int n;
int grid[MAXN][MAXN];
int dp[MAXN][MAXN];

int solve(int i, int j)
{
  if (i == n - 1 && j == n - 1)
    return grid[i][j];
  if (dp[i][j] != -1)
    return dp[i][j];
  if (i + 1 < n && j + 1 < n)
    return dp[i][j] = grid[i][j] + max(solve(i + 1, j),
        solve(i, j + 1));
  if (i + 1 < n)
    return dp[i][j] = grid[i][j] + solve(i + 1, j);
  if (j + 1 < n)
    return dp[i][j] = grid[i][j] + solve(i, j + 1);
}
signed main()
{
  cin >> n;
  for (int i = 0; i < n; i++)
    for (int j = 0; j < n; j++)
      cin >> grid[i][j];
  memset(dp, -1, sizeof(dp));
  cout << solve(0, 0) << endl;
  return 0;
}
```

6.8 dynamic cht

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define endl '\n'
#define pb push_back
#define pf push_front
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second

#define MAXN 1000005
#define mod 1000000007

struct line
{
  mutable int m, b, p;
  bool operator<(const line &o) const
  {
    if (m != o.m)
      return m < o.m;
    return b < o.b;
  }
  bool operator<(const int x) const { return p < x; }
  int eval(int x) const { return m * x + b; }
  int inter(const line &o) const
  {
    int x = b - o.b, y = o.m - m;
    return (x / y) - ((x ^ y) < 0 && x % y);
  }
};
struct cht
{
  int INF = 1e18;
  multiset<line, less<>> l;
  void add(int m, int b)
  {
    auto y = l.insert({m, b, INF});
    auto z = next(y);
    if (z != l.end() && y->m == z->m)
    {
      l.erase(y);
      return;
    }
    if (y != l.begin())
    {
      auto x = prev(y);
      if (x->m == y->m)
        x = l.erase(x);
    }
    while (1)
    {
      if (z == l.end())
      {
        y->p = INF;
        break;
      }
      y->p = y->inter(*z);
      if (y->p < z->p)
        break;
      else
```

```
      z = l.erase(z);
    }
    if (y == l.begin())
      return;
    z = y;
    auto x = --y;
    while (1)
    {
      int ninter = x->inter(*z);
      if (ninter <= x->p)
        x->p = ninter;
      else
      {
        l.erase(z);
        break;
      }
      if (x == l.begin())
        break;
      y = x;
      x--;
      if (x->p < y->p)
        break;
      else
        l.erase(y);
    }
  }
  int get(int x)
  {
    if (l.empty())
      return 0;
    return l.lower_bound(x)->eval(x);
  }
};
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  return 0;
}
// sources:
// https://github.com/pauloamed/Training/blob/master/PD/
   cht.cpp
// https://github.com/brunomaletta/Biblioteca/blob/
   master/Codigo/DP/CHT-Dinamico.cpp

// cht dinamico
// dado uma coordenada x
// e um conjunto com varias equacoes lineares da forma:
   y = mx + c
// retorna o maior valor de y entre as equacoes do
   conjunto

// para o menor valor, multiplicar m e c de cada equacao
    por -1
// e multiplicar o resultado da query por -1

// problemas iniciais:
// https://atcoder.jp/contests/dp/tasks/dp_z
// https://codeforces.com/contest/1083/problem/E
```

6.9 lis

```
// dada uma sequencia s qualquer, descobrir o tamanho da
    maior subsequencia crescente de s
// uma subsequencia de s e qualquer subconjunto de
   elementos de s.
// Para cada novo numero, voce tem duas operacoes
   possiveis:
// 1 - Colocar o novo numero no topo de uma pilha se ele
    nao superar o que ja esta em seu topo;
// ou
// 2 - Criar uma nova pilha a direita de todas as outras
    e colocar o novo numero la.
// ao final do processo a nossa pilha tera os elementos
   da lis.
#include <bits/stdc++.h>
using namespace std;

#define lli long long int
#define pb push_back
#define in insert
#define pi pair<int, int>
#define pd pair<double, int>
#define pib pair<pi, bool>
#define mp make_pair
#define fir first
#define sec second
#define MAXN 200001
#define MAXL 1000001
#define mod 1000000007

vector<int> v;

int lis()
{
  vector<int> q;
  for (int i = 0; i < v.size(); i++)
  {
    vector<int>::iterator it = lower_bound(q.begin(), q.
       end(), v[i]);
```

```
    if (it == q.end())
      q.pb(v[i]);
    else
      *it = v[i];
  }
  for (int i = 0; i < q.size(); i++)
    cout << q[i] << " ";
  cout << endl;
  return q.size();
}
signed main()
{
  int n;
  cin >> n;
  v.resize(n);
  for (int i = 0; i < n; i++)
    cin >> v[i];
  cout << lis() << endl;
  return 0;
}
```

6.10 Knapsack

```
//O problema mais classico de Programacao Dinamica
    talvez seja o Knapsack.
//De maneira geral, um ladrao ira roubar uma casa com
    uma mochila
//que suporta um peso s. Ele ve n objetos na casa e sabe
     estimar o peso pi    e o valor vi
//de cada objeto i. Com essas informacoes, qual o maior
    valor que o ladrao pode roubar sem rasgar sua
    mochila?
#include <bits/stdc++.h>
using namespace std;

#define lli long long int
#define pb push_back
#define in insert
#define pi pair<int, int>
#define pii pair<int, pi>
#define mp make_pair
#define fir first
#define sec second
#define MAXN 1001
#define INF 1000000000

int n, l;
int value[MAXN];
int peso[MAXN];
int dp[MAXN][MAXN];

int knapsack(int i, int limit)
{
  if (dp[i][limit] >= 0) // se ja foi calculado
  {
    return dp[i][limit];
  }

  if (i == n or !limit) // se chegou no fim do array ou
      chegou no limite
  {
    return dp[i][limit] = 0;
  }

  int nao_coloca = knapsack(i + 1, limit); //
      recursivamente pra caso eu nao coloque o objeto i

  if (peso[i] <= limit) // se eu consigo botar o objeto
      i
  {
    int coloca = value[i] + knapsack(i + 1, limit - peso
        [i]);
    return dp[i][limit] = max(coloca, nao_coloca);
  }

  return dp[i][limit] = nao_coloca;
}
signed main()
{
  cin >> l >> n;
  for (int i = 0; i < n; i++)
  {
    cin >> peso[i] >> value[i];
  }
  memset(dp, -1, sizeof(dp));
  cout << knapsack(0, l) << endl;
  return 0;
}
```

6.11 tip

```
// dados os valores de moedas v1, v2, ... vn e possivel
    formar um valor m como combinacao de moedas
// para isso basta montar uma dp inicializada com -1
// nesse caso a dp so precisa de um parametro q e =
    valor restante ate o limite
// mas podem existir variacoes do problema q precise de
    mais coisas
// se em achar alguma combinacao valida retorna 1, se
    nao retorna 0
```

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

#define lli long long int
#define pb push_back
#define in insert
#define pi pair<int, int>
#define pd pair<double, int>
#define pib pair<pi, bool>
#define mp make_pair
#define fir first
#define sec second
#define MAXN 200001
#define MAXL 10001
#define mod 1000000007

int dp[MAXN];
vector<int> v;

int solve(int rem)
{
  if (rem == 0)
    return 1;
  if (rem < 0)
    return 0;
  if (dp[rem] >= 0)
    return dp[rem];
  for (int i = 0; i < v.size(); i++)
    if (solve(rem - v[i]))
      return dp[rem - v[i]] = 1;
  return dp[rem] = 0;
}
signed main()
{
  int n, m;
  cin >> n >> m;
  v.resize(n);
  for (int i = 0; i < n; i++)
    cin >> v[i];
  memset(dp, -1, sizeof(dp));
  (solve(m)) ? cout << "Yes\n" : cout << "No\n";
  return 0;
}
```

6.12 largest-sum-contiguous-subarray

```cpp
// dada uma sequencia s qual a maior soma que podemos
    obter escolhendo um subconjunto de termos adjacentes
     de s
// nesse caso o temos apenas duas opcoes
// nao usar o elemento v[i]
// ou
// usamos, adicionando a maior soma possivel que antes
    dele
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 200001
#define mod 1000000007

int kadane(vector<int> v)
{
  int n = v.size(), ans = 0, max_here = 0;
  for (int i = 0; i < n; i++)
  {
    max_here += v[i];
    if (ans < max_here)
      ans = max_here;
    if (max_here < 0)
      max_here = 0;
  }
  return ans;
}
int kadane_circular(vector<int> v)
{
  int n = v.size(), max_kadane = kadane(v);
  int max_wrap = 0, i;
  for (i = 0; i < n; i++)
  {
    max_wrap += v[i];
    v[i] = -v[i];
  }
  max_wrap += kadane(v);
  return max(max_wrap, max_kadane);
}
signed main()
{
```

```
  int n;
  cin >> n;
  vector<int> v(n);
  for (int i = 0; i < n; i++)
    cin >> v[i];
  cout << kadane_circular(v) << endl;
  return 0;
}
```

6.13 lcs

```
//Dadas duas sequencias s1 e s2, uma de tamanho n e
    outra de tamanho m, qual a maior subsequencia comum
    as duas?

// uma subsequencia de s e um subconjunto dos elementos
    de s na mesma ordem em que apareciam antes.
// isto significa que {1, 3, 5} e uma subsequencia de
    {1, 2, 3, 4, 5}, mesmo 1 nao estando do lado do 3.
#include <bits/stdc++.h>
using namespace std;

#define lli long long int
#define pb push_back
#define in insert
#define pi pair<int, int>
#define pii pair<int, pi>
#define mp make_pair
#define fir first
#define sec second
#define MAXN 1001
#define INF 1000000000

int v1[MAXN];
int v2[MAXN];
int dp[MAXN][MAXN];

void lcs(int m, int n)
{
  for (int i = 0; i <= m; i++)
  {
    for (int j = 0; j <= n; j++)
    {
      if (i == 0 || j == 0) //se uma das sequencias for
          vazia
        dp[i][j] = 0;
      else if (v1[i - 1] == v2[j - 1]) // se eh igual,
          adiciono a lcs e subtraio dos dois
        dp[i][j] = dp[i - 1][j - 1] + 1;
      else
        dp[i][j] = max(dp[i - 1][j], dp[i][j - 1]); //
            se nao retorno o maximo entre tirar um dos
            dois caras
    }
  }
  cout << dp[m][n] << endl;
}
signed main()
{
  int n, m;
  cin >> n >> m;
  for (int i = 0; i < n; i++)
    cin >> v1[i];
  for (int i = 0; i < m; i++)
    cin >> v2[i];
  lcs(n, m);
  return 0;
}
```

6.14 divideandconquer

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 200005
#define mod 1000000007

int s[8005];
int dp[3005][8005];

int cost(int l, int r)
{
  return (s[r + 1] - s[l]) * (r - l + 1);
}
void compute(int l, int r, int optl, int optr, int i)
{
```

```
  if (l > r)
    return;
  int mid = (l + r) >> 1;
  pair<int, int> ans = {1e18, -1}; // dp, k
  for (int q = optl; q <= min(mid, optr); q++)
  {
    if (q > 0)
      ans = min(ans, {dp[i - 1][q - 1] + cost(q, mid), q
          });
    else
      ans = min(ans, {cost(q, mid), q});
  }
  dp[i][mid] = ans.fir;
  compute(l, mid - 1, optl, ans.sec, i);
  compute(mid + 1, r, ans.sec, optr, i);
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n, g;
  cin >> n >> g;
  for (int i = 1; i <= n; i++)
  {
    cin >> s[i];
    s[i] += s[i - 1];
  }
  for (int i = 0; i <= g; i++)
  {
    for (int j = 0; j <= n; j++)
      dp[i][j] = 1e18;
  }
  for (int i = 1; i <= g; i++)
    compute(0, n - 1, 0, n - 1, i);
  cout << dp[g][n - 1] << endl;
  return 0;
}
// https://codeforces.com/gym/103536/problem/A
// https://codeforces.com/contest/321/problem/E

// otimizacao de dp usando divide and conquer
// para dps do tipo:
// dp[i][j] = min(dp[i - 1][k] + c(k, j)), para algum k
    <= j
// considerando opt(i, j) o menor valor de k que
    minimiza dp[i][j]
// podemos calcular opt(i, j) usando divide and conquer
// isso diminuiria a complexidade para O(k * n * log(n))
```

6.15 expected value

```
//https://atcoder.jp/contests/dp/tasks/dp_j
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define pb push_back
#define mp make_pair
#define pi pair<int, int>
#define pii pair<pi, int>
#define pci pair<char, int>
#define fir first
#define sec second
#define DEBUG 0
#define MAXN 301
#define mod 1000000007

int n;
vector<int> v;
vector<int> cnt(3);
double dp[MAXN][MAXN][MAXN];

double solve(int i, int j, int k)
{
  if (!i && !j && !k)
    return dp[i][j][k] = 0;
  if (dp[i][j][k] != -1)
    return dp[i][j][k];
  /*
  It is well-known from statistics that for the
     geometric distribution
  (counting number of trials before a success, where
     each independent trial is probability p)
  the expected value is i / p
  */
  double p = ((double)(i + j + k) / n);
  double ret = 1 / p; // expected number of trials
     before a success
  if (i)
  {
    double prob = (double)i / (i + j + k); //
```

```
        probabilidade de ser um prato com um sushi
    ret += (solve(i - 1, j, k) * prob);
  }
  if (j)
  {
    double prob = (double)j / (i + j + k); //
        probabilidade de ser um prato com dois sushis
    ret += (solve(i + 1, j - 1, k) * prob);
  }
  if (k)
  {
    double prob = (double)k / (i + j + k); //
        probabilidade de ser um prato com tres sushis
    ret += (solve(i, j + 1, k - 1) * prob);
  }
  return dp[i][j][k] = ret;
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  cin >> n;
  v.resize(n);
  for (int i = 0; i < n; i++)
    cin >> v[i], cnt[v[i] - 1]++;
  for (int i = 0; i < MAXN; i++)
    for (int j = 0; j < MAXN; j++)
      for (int k = 0; k < MAXN; k++)
        dp[i][j][k] = -1;
  cout << setprecision(15) << solve(cnt[0], cnt[1], cnt
      [2]) << endl;
  return 0;
}
```

6.16 Digitdp

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

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define fir first
#define sec second
#define MAXN 2001
#define mod 1000000007

int dp[20][20 * 9][2]; // a,b <= 10^18
vector<int> dig;
int solve(int i, int j, int k)
{
  if (i == dig.size())
    return (k) ? dp[i][j][k] = j : dp[i][j][k] = 0;
  if (dp[i][j][k] != -1)
    return dp[i][j][k];
  int sum = 0;
  if (k)
    for (int f = 0; f <= 9; f++)
      sum += solve(i + 1, j + f, k);
  if (!k)
    for (int f = 0; f <= dig[i]; f++)
      sum += solve(i + 1, j + f, (dig[i] != f) ? 1 : 0);
  return dp[i][j][k] = sum;
}
void get_digits(int n)
{
  dig.clear();
  while (n)
  {
    dig.pb(n % 10);
    n = n / 10;
  }
  reverse(dig.begin(), dig.end());
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int a, b;
  cin >> a >> b;
  get_digits(a);
  memset(dp, -1, sizeof(dp));
  int aa = solve(0, 0, 0);
  get_digits(b + 1);
  memset(dp, -1, sizeof(dp));
  int bb = solve(0, 0, 0);
  cout << bb - aa << endl;
  return 0;
}
```

7 Graph

7.1 centroid decomposition2

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
```

```
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 50005
#define mod 1000000007

int n, k, resp;
vector<int> adj[MAXN];
gp_hash_table<int, int> cnt;

namespace cd
{
  int sz;
  vector<int> adjl[MAXN];
  vector<int> father, subtree_size;
  vector<bool> visited;

  void dfs(int s, int f)
  {
    sz++;
    subtree_size[s] = 1;
    for (auto const &v : adj[s])
    {
      if (v != f && !visited[v])
      {
        dfs(v, s);
        subtree_size[s] += subtree_size[v];
      }
    }
  }
  int getCentroid(int s, int f)
  {
    bool is_centroid = true;
    int heaviest_child = -1;
    for (auto const &v : adj[s])
    {
      if (v != f && !visited[v])
      {
        if (subtree_size[v] > sz / 2)
          is_centroid = false;
        if (heaviest_child == -1 || subtree_size[v] >
            subtree_size[heaviest_child])
          heaviest_child = v;
      }
    }
    return (is_centroid && sz - subtree_size[s] <= sz /
        2) ? s : getCentroid(heaviest_child, s);
  }
  void dfs2(int s, int f, int d)
  {
    cnt[d]++;
    for (auto const &v : adj[s])
      if (v != f && !visited[v])
        dfs2(v, s, d + 1);
  }
  int solve(int s)
  {
    gp_hash_table<int, int> tot;
    int ans = 0;
    for (auto const &v : adj[s])
    {
      if (visited[v])
        continue;
      cnt.clear();
      dfs2(v, s, 1);
      for (int i = 1, j = k - 1; i < k; i++, j--)
        ans += (cnt[i] * tot[j]);
      for (auto const &i : cnt)
        tot[i.fir] += i.sec;
    }
    return ans + tot[k];
  }
  int decompose_tree(int s)
  {
    sz = 0;
    dfs(s, s);
    int cend_tree = getCentroid(s, s);
    visited[cend_tree] = true;
    resp += solve(cend_tree);
    for (auto const &v : adj[cend_tree])
    {
      if (!visited[v])
      {
        int cend_subtree = decompose_tree(v);
        adjl[cend_tree].pb(cend_subtree);
        adjl[cend_subtree].pb(cend_tree);
        father[cend_subtree] = cend_tree;
      }
    }
    return cend_tree;
  }
```

```
  void init()
  {
    subtree_size.resize(n);
    visited.resize(n);
    father.assign(n, -1);
    decompose_tree(0);
  }
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  cin >> n >> k;
  for (int i = 0; i < n - 1; i++)
  {
    int a, b;
    cin >> a >> b;
    a--, b--;
    adj[a].pb(b);
    adj[b].pb(a);
  }
  cd::init();
  cout << resp << endl;
  return 0;
}
// https://codeforces.com/contest/161/problem/D
// durante a decomposicao
// pega o centroid atual e resolve o problema pra ele
// isso eh:
// para cada centroid que eu achei, devo contar quantos
    caminhos
// de tamanho k passam por esse centroid
// somando todas essas respostas, a gente tem a resposta
    final
```

7.2 Floyd Warshall

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

#define pb push_back
#define lli long long int
#define MAXN 10000
#define INF 999999

int n , m , a , b , c ;
int dist [MAXN][MAXN] ;

void floyd_warshall ()
{
    for (int k = 0 ; k < n ; k++)
    {
        for (int i = 0 ; i < n ; i++)
        {
            for (int j = 0 ; j < n ; j++)
            {
                dist[i][j] = min(dist[i][j] , dist[i][k]
                    + dist[k][j]) ;
            }
        }
    }
}
void initialize ()
{
    for (int i = 0 ; i < n ; i++)
    {
        for (int j = 0 ; j < n ; j++)
        {
            if (i == j)
            {
                dist[i][j] = 0 ;
            }
            else
            {
                dist[i][j] = INF ;
            }

        }
    }
}
int main()
{
    cin >> n >> m ;

    initialize () ;

    for (int i = 0 ; i < m ; i++)
    {
        cin >> a >> b >> c ;
        dist [a][b] = min (dist[a][b] , c) ;
        dist [b][a] = min (dist[b][a] , c) ;
    }

    floyd_warshall () ;

    return 0;
}
```

7.3 strong orientation

```cpp
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 1000005
#define mod 1000000007

int n, m, timer, comps, bridges;
vector<pi> edges;
vector<pi> adj[MAXN];
int tin[MAXN];
int low[MAXN];
bool vis[MAXN];
char orient[MAXN];

void find_bridges(int v)
{
  low[v] = timer, tin[v] = timer++;
  for (auto const &p : adj[v])
  {
    if (vis[p.sec])
      continue;
    vis[p.sec] = true;
    orient[p.sec] = (v == edges[p.sec].first) ? '>' : '<
        ';
    if (tin[p.fir] == -1)
    {
      find_bridges(p.fir);
      low[v] = min(low[v], low[p.fir]);
      if (low[p.fir] > tin[v])
        bridges++;
    }
    else
    {
      low[v] = min(low[v], low[p.fir]);
    }
  }
}

signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  cin >> n >> m;
  for (int i = 0; i < m; i++)
  {
    int a, b;
    cin >> a >> b;
    a--, b--;
    edges.pb({a, b});
    adj[a].pb({b, i});
    adj[b].pb({a, i});
  }
  memset(tin, -1, sizeof(tin));
  memset(low, -1, sizeof(low));
  for (int v = 0; v < n; v++)
  {
    if (tin[v] == -1)
    {
      comps++;
      find_bridges(v);
    }
  }
  // numero minimo de scc = numero de componentes +
      numero de pontes
  cout << comps + bridges << endl;
  // > - a aresta foi orientada da esquerda pra direita
  // < - a aresta foi orientada da direita pra esquerda
  for (int i = 0; i < m; i++)
    cout << orient[i];
  cout << endl;
  return 0;
}
// to_test: https://szkopul.edu.pl/problemset/problem/
    nldsb4EW1YuZykBlf4lcZL1Y/site/?key=statement
// strong orientation:
// encontrar uma orientacao para as arestas tal que o
    numero
// minimo de scc e o menor possivel
```

7.4 scc

```cpp
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
```

```cpp
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 500005
#define mod 1000000007

int n, m;

bool vis[MAXN];
int root[MAXN];
vector<int> order;
vector<int> roots;
vector<int> comp;
vector<vector<int>> comps;
vector<int> adj[MAXN];
vector<int> adj_rev[MAXN];
vector<int> adj_scc[MAXN];

void dfs(int v)
{
  vis[v] = true;
  for (auto const &u : adj[v])
    if (!vis[u])
      dfs(u);
  order.pb(v);
}
void dfs2(int v)
{
  comp.pb(v);
  vis[v] = true;
  for (auto const &u : adj_rev[v])
    if (!vis[u])
      dfs2(u);
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  cin >> n >> m;
  for (int i = 0; i < m; i++)
  {
    int a, b;
    cin >> a >> b;
    adj[a].pb(b);
    adj_rev[b].pb(a);
  }
  for (int i = 0; i < n; i++)
  {
    if (!vis[i])
      dfs(i);
  }
  reverse(order.begin(), order.end());
  memset(vis, false, sizeof(vis));
  for (auto const &v : order)
  {
    if (!vis[v])
    {
      comp.clear();
      dfs2(v);
      comps.pb(comp);
      // making condensation graph
      /*
      int r = comp.back();
      for (auto const &u : comp)
        root[u] = r;
      roots.push_back(r);
      */
    }
  }
  // making condensation graph
  /*
  for (int v = 0; v < n; v++)
  {
    for (auto const &u : adj[v])
    {
      int root_v = roots[v];
      int root_u = roots[u];
      if (root_u != root_v)
        adj_scc[root_v].pb(root_u);
    }
  }
  */
  // printing scc
  cout << comps.size() << endl;
  for (auto const &comp : comps)
  {
    cout << comp.size() << " ";
    for (auto const &u : comp)
      cout << u << " ";
    cout << endl;
  }
  return 0;
}
// to test: https://judge.yosupo.jp/problem/scc
```

7.5 hld

```
//https://codeforces.com/contest/343/problem/D
#include <bits/stdc++.h>
using namespace std;

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<pi, int>
#define fir first
#define sec second
#define DEBUG 0
#define MAXN 500001
#define mod 1000000007

int n, q;
vector<int> adj[MAXN];

namespace seg
{
  int seg[4 * MAXN];
  int lazy[4 * MAXN];

  int single(int x)
  {
    return x;
  }
  int neutral()
  {
    return 0;
  }
  int merge(int a, int b)
  {
    return a + b;
  }
  void add(int i, int l, int r, int diff)
  {
    seg[i] = (r - l + 1) * diff;
    if (l != r)
    {
      lazy[i << 1] = diff;
      lazy[(i << 1) | 1] = diff;
    }
    lazy[i] = -1;
  }
  void update(int i, int l, int r, int ql, int qr, int
      diff)
  {
    if (lazy[i] != -1)
      add(i, l, r, lazy[i]);
    if (l > r || l > qr || r < ql)
      return;
    if (l >= ql && r <= qr)
    {
      add(i, l, r, diff);
      return;
    }
    int mid = (l + r) >> 1;
    update(i << 1, l, mid, ql, qr, diff);
    update((i << 1) | 1, mid + 1, r, ql, qr, diff);
    seg[i] = merge(seg[i << 1], seg[(i << 1) | 1]);
  }
  int query(int l, int r, int ql, int qr, int i)
  {
    if (lazy[i] != -1)
      add(i, l, r, lazy[i]);
    if (l > r || l > qr || r < ql)
      return neutral();
    if (l >= ql && r <= qr)
      return seg[i];
    int mid = (l + r) >> 1;
    return merge(query(l, mid, ql, qr, i << 1), query(
       mid + 1, r, ql, qr, (i << 1) | 1));
  }
} // namespace seg
namespace hld
{
  int cur_pos;
  vector<int> parent, depth, heavy, head, pos, sz;

  int dfs(int s)
  {
    int size = 1, max_c_size = 0;
    for (auto const &c : adj[s])
    {
      if (c != parent[s])
      {
        parent[c] = s;
        depth[c] = depth[s] + 1;
        int c_size = dfs(c);
        size += c_size;
        if (c_size > max_c_size)
          max_c_size = c_size, heavy[s] = c;
      }
    }
    return sz[s] = size;
  }
  void decompose(int s, int h)
```

```
{
  head[s] = h;
  pos[s] = cur_pos++;
  if (heavy[s] != -1)
    decompose(heavy[s], h);
  for (int c : adj[s])
  {
    if (c != parent[s] && c != heavy[s])
      decompose(c, c);
  }
}
void init()
{
  memset(seg::lazy, -1, sizeof(seg::lazy));
  parent.assign(MAXN, -1);
  depth.assign(MAXN, -1);
  heavy.assign(MAXN, -1);
  head.assign(MAXN, -1);
  pos.assign(MAXN, -1);
  sz.assign(MAXN, 1);
  cur_pos = 0;
  dfs(0);
  decompose(0, 0);
  for (int i = 0; i < 4 * n; i++)
    seg::lazy[i] = -1;
}
int query_path(int a, int b)
{
  int res = 0;
  for (; head[a] != head[b]; b = parent[head[b]])
  {
    if (depth[head[a]] > depth[head[b]])
      swap(a, b);
    int cur_heavy_path_max = seg::query(0, n - 1, pos[
        head[b]], pos[b], 1);
    res += cur_heavy_path_max;
  }
  if (depth[a] > depth[b])
    swap(a, b);
  int last_heavy_path_max = seg::query(0, n - 1, pos[a
      ], pos[b], 1);
  res += last_heavy_path_max;
  return res;
}
void update_path(int a, int b, int x)
{
  for (; head[a] != head[b]; b = parent[head[b]])
  {
    if (depth[head[a]] > depth[head[b]])
      swap(a, b);
    seg::update(1, 0, n - 1, pos[head[b]], pos[b], x);
  }
  if (depth[a] > depth[b])
    swap(a, b);
  seg::update(1, 0, n - 1, pos[a], pos[b], x);
}
void update_subtree(int a, int x)
{
  seg::update(1, 0, n - 1, pos[a], pos[a] + sz[a] - 1,
      x);
}
void query_subtree(int a, int x)
{
  seg::query(0, n - 1, pos[a], pos[a] + sz[a] - 1, 1);
}
} // namespace hld
signed main()
{
  cin >> n;
  for (int i = 0; i < n - 1; i++)
  {
    int a, b;
    cin >> a >> b;
    a--, b--;
    adj[a].pb(b);
    adj[b].pb(a);
  }
  hld::init();
  cin >> q;
  while (q--)
  {
    int a, b;
    cin >> a >> b;
    b--;
    if (a == 1)
    {
      hld::update_subtree(b, 1);
    }
    if (a == 2)
    {
      hld::update_path(0, b, 0);
    }
    if (a == 3)
    {
      cout << hld::query_path(b, b) << endl;
    }
  }
  return 0;
}
```

7.6 Prim

```
// algoritimo de prim

// 1 - definir a distancia de cada vertice como infinito
     (similar ao dijkstra).
// 2 - definir a distancia de 0 para o source(0).
// 3 - Em cada passo, encontrar o vertice u, que ainda
    nao foi processado, que possua a menor das
    distancias.
// 4 - ao termino fazer a soma de todas as distancias e
    encontrar qual a soma das distancias na MST.

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

#define lli long long int
#define pb push_back
#define pii pair<int, int>
#define mp make_pair
#define MAXN 100001
#define INF 999999
#define sec second
#define fir first

int n, m, a, b, c;
vector<pii> adj[MAXN];
int dist[MAXN];
bool processed[MAXN];

void prim()
{
  for (int i = 0; i < n; i++)
  {
    dist[i] = INF;
  }

  dist[0] = 0;

  priority_queue<pii, vector<pii>, greater<pii>> q;
  q.push(pii(dist[0], 0));

  while (1)
  {
    int davez = -1;

    while (!q.empty())
    {
      int atual = q.top().sec;
      q.pop();

      if (!processed[atual])
      {
        davez = atual;
        break;
      }
    }

    if (davez == -1)
    {
      break;
    }

    processed[davez] = true;

    for (int i = 0; i < adj[davez].size(); i++)
    {
      int distt = adj[davez][i].fir;
      int atual = adj[davez][i].sec;

      if (dist[atual] > distt && !processed[atual])
      {
        dist[atual] = distt;
        q.push(pii(dist[atual], atual));
      }
    }
  }

  int ans = 0;

  for (int i = 0; i < n; i++)
  {
    ans += dist[i];
  }

  cout << ans << endl;
}
int main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);

  cin >> n >> m;

  for (int i = 0; i < m; i++)
  {
    cin >> a >> b >> c;
    a--;
    b--;
```

```
    adj[a].pb(mp(c, b));
    adj[b].pb(mp(c, a));
  }

  prim();

  return 0;
}
```

7.7 articulation points

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 400005
#define mod 1000000007

int n, m, timer;
vector<int> adj[MAXN];
bool is_cutpoint[MAXN];
int tin[MAXN];
int low[MAXN];
bool vis[MAXN];

void dfs(int v, int p)
{
  vis[v] = true;
  tin[v] = timer, low[v] = timer++;
  int childs = 0;
  for (auto const &u : adj[v])
  {
    if (u == p)
      continue;
    if (vis[u])
    {
      low[v] = min(low[v], tin[u]);
    }
    else
    {
      dfs(u, v);
      low[v] = min(low[v], low[u]);
      if (low[u] >= tin[v] && p != -1)
        is_cutpoint[v] = true;
      childs++;
    }
  }
  if (p == -1 && childs > 1)
    is_cutpoint[v] = true;
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  cin >> n >> m;
  for (int i = 0; i < m; i++)
  {
    int a, b;
    cin >> a >> b;
    a--, b--;
    adj[a].pb(b);
    adj[b].pb(a);
  }
  memset(tin, -1, sizeof(tin));
  memset(low, -1, sizeof(low));
  for (int i = 0; i < n; i++)
  {
    if (!vis[i])
      dfs(i, -1);
  }
  return 0;
}
```

7.8 mincostflow

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define pb push_back
#define pi pair<int, int>
```

```cpp
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 301
#define mod 1000000007
#define INF 1e9

namespace mcf
{
  struct edge
  {
    int to, capacity, cost, res;
  };

  int source, destiny;
  vector<edge> adj[MAXN];
  vector<int> dist;
  vector<int> parent;
  vector<int> edge_index;
  vector<bool> in_queue;

  void add_edge(int a, int b, int c, int d)
  {
    adj[a].pb({b, c, d, (int)adj[b].size()});      // aresta normal
    adj[b].pb({a, 0, -d, (int)adj[a].size() - 1}); // aresta do grafo residual
  }
  bool dijkstra(int s) // rodando o dijkstra, terei o caminho de custo minimo
  {                    // que eu consigo passando pelas arestas que possuem capacidade > 0
    dist.assign(MAXN, INF);
    parent.assign(MAXN, -1);
    edge_index.assign(MAXN, -1);
    in_queue.assign(MAXN, false);
    dist[s] = 0;
    queue<int> q;
    q.push(s);
    while (!q.empty())
    {
      int u = q.front(), idx = 0;
      q.pop();
      in_queue[u] = false;
      for (auto const &v : adj[u])
      {
        if (v.capacity && dist[v.to] > dist[u] + v.cost)
        {
          dist[v.to] = dist[u] + v.cost;
          parent[v.to] = u;
          edge_index[v.to] = idx;
          if (!in_queue[v.to])
          {
            in_queue[v.to] = true;
            q.push(v.to);
          }
        }
        idx++;
      }
    }
    return dist[destiny] != INF; // se eu cheguei em destiny por esse caminho, ainda posso passar fluxo
  }
  int get_cost()
  {
    int flow = 0, cost = 0;
    while (dijkstra(source)) // rodo um dijkstra para saber qual o caminho que irei agora
    {
      int curr_flow = INF, curr = destiny;
      while (curr != source) // com isso, vou percorrendo o caminho encontrado para achar a aresta "gargalo"
      {
        int p = parent[curr];
        curr_flow = min(curr_flow, adj[p][edge_index[curr]].capacity);
        curr = p;
      }
      flow += curr_flow;                  // fluxo que eu posso passar por esse caminho = custo da aresta "gargalo"
      cost += curr_flow * dist[destiny]; // quanto eu gasto para passar esse fluxo no caminho encontrado
      curr = destiny;
      while (curr != source) // apos achar a aresta gargalo, passamos o fluxo pelo caminho encontrado
      {
        int p = parent[curr];
        int res_idx = adj[p][edge_index[curr]].res;
        adj[p][edge_index[curr]].capacity -= curr_flow;
        adj[curr][res_idx].capacity += curr_flow;
        curr = p;
      }
    }
    return cost; // ao final temos a resposta :)
  }
```

```
} // namespace mcf
signed main()
{
  int n;
  cin >> n;
  int v[n][n];
  mcf::source = 0, mcf::destiny = (2 * n) + 1;
  for (int i = 0; i < n; i++)
  {
    for (int j = 0; j < n; j++)
    {
      cin >> v[i][j];
      mcf::add_edge(i + 1, j + n + 1, 1, v[i][j]);
    }
  }
  for (int i = 1; i <= n; i++)
    mcf::add_edge(mcf::source, i, 1, 0);
  for (int i = n + 1; i <= n + n; i++)
    mcf::add_edge(i, mcf::destiny, 1, 0);
  cout << mcf::get_cost << endl;
}
```

7.9 eulertour

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<pi, int>
#define fir first
#define sec second
#define DEBUG 1
#define MAXN 100001
#define mod 1000000009
#define d 31

int n, idx;
vector<int> adj[MAXN];
int euler[2 * MAXN];
int entrei[MAXN];
int sai[MAXN];

void euler_tour(int s, int f)
{
  euler[idx] = s;
  entrei[s] = idx;
  idx++;
  for (auto const &v : adj[s])
  {
    if (v == f)
      continue;
    euler_tour(v, s);
  }
  euler[idx] = s;
  sai[s] = idx;
  idx++;
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n;
  cin >> n;
  for (int i = 0; i < n - 1; i++)
  {
    int a, b;
    cin >> a >> b;
    a--, b--;
    adj[a].pb(b);
    adj[b].pb(a);
  }
  euler_tour(0, -1);
  for (int i = 0; i < 2 * n; i++)
    cout << euler[i] << " ";
  cout << endl;
  return 0;
}
// euler tour of a tree
// muito util para algumas coisas
// exemplos:
// 1- soma da subarvore de v(com update)
// usando segment trees, podemos fazer uma query(entrei[
    v], sai[v])
// 2- LCA
// lca(u, v) = query(entrei[u], entrei[v])
// usando uma query de minimo e considerando as
    profundidade dos vertices
// a resposta sera o vertice de profundidade minima que
    encontrarmos no intervalo
// 3- agilidade para remover arestas/vertices/subtrees
    da arvore
```

```
// basta apenas tratar o segmento equivalente do jeito
    que for necessario
// 4- reroot a tree
// basta apenas rotacionar o euler path
```

7.10 TreeDiameter

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

#define PI acos(-1)
#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<pi, int>
#define fir first
#define sec second
#define MAXN 100001
#define mod 1000000007

int diameter, best;
vector<int> adj[MAXN];
bool visited[MAXN];

void dfs(int s, int c)
{
  if (c > diameter)
  {
    diameter = c;
    best = s;
  }
  visited[s] = true;
  for (auto const &i : adj[s])
    if (!visited[i])
      dfs2(i, c + 1);
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int q;
  cin >> q;
  while (q--)
  {
    int n;
    cin >> n;
    for (int i = 0; i < n; i++)
      adj[i].clear();
    for (int i = 0; i < n - 1; i++)
    {
      int a, b;
      cin >> a >> b;
      a--, b--;
      adj[b].pb(a);
      adj[a].pb(b);
    }
    diameter = 0, best = 0;
    memset(visited, false, sizeof(visited));
    dfs(1, 0);                          // achar o vertice
       mais distante a partir do vertice 0
    memset(visited, false, sizeof(visited));
    dfs(best, 0);                       // achar o mais
       distante a partir do primeiro vertice que
       achamos
    cout << diameter << endl;
  }
  return 0;
}
```

7.11 Grafo Bipartido

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 200006
#define mod 1000000007

struct dsu
{
  vector<pi> parent;
  vector<int> rank;
  vector<int> bipartite;

  dsu(int n)
  {
    parent.resize(n);
```

```
    rank.resize(n);
    bipartite.resize(n);
    for (int v = 0; v < n; v++)
    {
      parent[v] = {v, 0};
      rank[v] = 0;
      bipartite[v] = 1;
    }
  }
  dsu() {}
  pi find_set(int v)
  {
    if (v != parent[v].fir)
    {
      int parity = parent[v].sec;
      parent[v] = find_set(parent[v].fir);
      parent[v].sec ^= parity;
    }
    return parent[v];
  }
  void add_edge(int a, int b)
  {
    pi pa = find_set(a);
    a = pa.fir;
    int x = pa.sec;
    pi pb = find_set(b);
    b = pb.fir;
    int y = pb.sec;
    if (a == b)
    {
      if (x == y)
        bipartite[a] = 0;
    }
    else
    {
      if (rank[a] < rank[b])
        swap(a, b);
      parent[b] = {a, x ^ y ^ 1};
      bipartite[a] &= bipartite[b];
      if (rank[a] == rank[b])
        rank[a]++;
    }
  }
  bool is_bipartite(int v)
  {
    return bipartite[find_set(v).fir];
  }
};
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  return 0;
}
```

7.12 dsu

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 2001
#define mod 1000000007

struct dsu
{
  int tot;
  vector<int> parent;
  vector<int> sz;

  dsu(int n)
  {
    parent.resize(n);
    sz.resize(n);
    tot = n;
    for (int i = 0; i < n; i++)
    {
      parent[i] = i;
      sz[i] = 1;
    }
  }
  int find_set(int i)
  {
    return parent[i] = (parent[i] == i) ? i : find_set(
        parent[i]);
  }
  void make_set(int x, int y)
```

```
    {
      x = find_set(x), y = find_set(y);
      if (x != y)
      {
        if (sz[x] > sz[y])
          swap(x, y);
        parent[x] = y;
        sz[y] += sz[x];
        tot--;
      }
    }
};
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n;
  cin >> n;
  dsu d(n);
  int a, b;
  cin >> a >> b;
  d.make_set(a, b);
  d.find_set(a);
}
```

7.13 reroot

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define PI acos(-1)
#define pb push_back
#define int long long int
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define DEBUG 0
#define MAXN 200001
#define mod 1000000007

int n;
vector<int> adj[MAXN];
int sz[MAXN];
int dp[MAXN];

int dfs(int u, int v)
{
  sz[u] = 1;
  for (auto const &i : adj[u])
    if (i != v)
      sz[u] += dfs(i, u);
  return sz[u];
}
void reroot(int u, int v)
{
  for (auto const &i : adj[u])
  {
    if (i != v)
    {
      int a = sz[u], b = sz[i];
      dp[i] = dp[u];
      dp[i] -= sz[u], dp[i] -= sz[i];
      sz[u] -= sz[i], sz[i] = n;
      dp[i] += sz[u], dp[i] += sz[i];
      reroot(i, u);
      sz[u] = a, sz[i] = b;
    }
  }
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  cin >> n;
  for (int i = 0; i < n - 1; i++)
  {
    int a, b;
    cin >> a >> b;
    a--, b--;
    adj[a].pb(b);
    adj[b].pb(a);
  }
  dfs(0, -1);
  for (int i = 0; i < n; i++)
    dp[0] += sz[i]; // answer when tree is rooted on
        vertex 0
  reroot(0, -1);
  cout << *max_element(dp, dp + n) << endl;
  return 0;
}
// https://codeforces.com/contest/1187/problem/E
// f(v) = when tree is rooted at vertex v, the current
```

```
// answer is the sum of all subtrees sizes
// final answer = max(f(0), f(1), f(2), ..., f(n))
// easy approach: O(N^2)
// with reroot: O(N)
// 1 - run a dfs and calculate f(0)
// 2 - let be dp[i] = f(i)
// 3 - now, lets run a another dfs, and re-calculate the
// answer when tree is rooted at vertex i (dp[i])
// 4 - the final answer is the maximum value of dp[i]
```

7.14 hld edge

```
//https://www.spoj.com/problems/QTREE/
//Don't use cin/cout in this problem (gives TLE)
#include <bits/stdc++.h>
using namespace std;

#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define DEBUG 0
#define MAXN 10001
#define mod 1000000007

int n;
vector<pi> adj[MAXN];
vector<pi> edges;

namespace seg
{
  int seg[4 * MAXN];
  int lazy[4 * MAXN];
  int v[MAXN];

  int single(int x)
  {
    return x;
  }
  int neutral()
  {
    return -1;
  }
  int merge(int a, int b)
  {
    return max(a, b);
  }
  void add(int i, int l, int r, int diff)
  {
    seg[i] = (r - l + 1) * diff;
    if (l != r)
    {
      lazy[i << 1] = diff;
      lazy[(i << 1) | 1] = diff;
    }
    lazy[i] = -1;
  }
  void update(int i, int l, int r, int ql, int qr, int
      diff)
  {
    if (lazy[i] != -1)
      add(i, l, r, lazy[i]);
    if (l > r || l > qr || r < ql)
      return;
    if (l >= ql && r <= qr)
    {
      add(i, l, r, diff);
      return;
    }
    int mid = (l + r) >> 1;
    update(i << 1, l, mid, ql, qr, diff);
    update((i << 1) | 1, mid + 1, r, ql, qr, diff);
    seg[i] = merge(seg[i << 1], seg[(i << 1) | 1]);
  }
  int query(int l, int r, int ql, int qr, int i)
  {
    if (lazy[i] != -1)
      add(i, l, r, lazy[i]);
    if (l > r || l > qr || r < ql)
      return neutral();
    if (l >= ql && r <= qr)
      return seg[i];
    int mid = (l + r) >> 1;
    return merge(query(l, mid, ql, qr, i << 1), query(
        mid + 1, r, ql, qr, (i << 1) | 1));
  }
  void build(int l, int r, int i)
  {
    if (l == r)
    {
      seg[i] = single(v[l]);
      lazy[i] = -1;
      return;
    }
    int mid = (l + r) >> 1;
    build(l, mid, i << 1);
    build(mid + 1, r, (i << 1) | 1);
    seg[i] = merge(seg[i << 1], seg[(i << 1) | 1]);
    lazy[i] = -1;
```

```
  }
} // namespace seg
namespace hld
{
  int cur_pos;
  vector<int> parent, depth, heavy, head, pos, sz, up;

  int dfs(int s)
  {
    int size = 1, max_c_size = 0;
    for (auto const &c : adj[s])
    {
      if (c.fir != parent[s])
      {
        parent[c.fir] = s;
        depth[c.fir] = depth[s] + 1;
        int c_size = dfs(c.fir);
        size += c_size;
        if (c_size > max_c_size)
          max_c_size = c_size, heavy[s] = c.fir;
      }
    }
    return sz[s] = size;
  }
  void decompose(int s, int h)
  {
    head[s] = h;
    pos[s] = cur_pos++;
    seg::v[pos[s]] = up[s];
    for (auto const &c : adj[s])
    {
      if (c.fir != parent[s] && c.fir == heavy[s])
      {
        up[c.fir] = c.sec;
        decompose(heavy[s], h);
      }
    }
    for (auto const &c : adj[s])
    {
      if (c.fir != parent[s] && c.fir != heavy[s])
      {
        up[c.fir] = c.sec;
        decompose(c.fir, c.fir);
      }
    }
  }
  void init()
  {
    parent.assign(MAXN, -1);
    depth.assign(MAXN, -1);
    heavy.assign(MAXN, -1);
    head.assign(MAXN, -1);
    pos.assign(MAXN, -1);
    sz.assign(MAXN, 1);
    up.assign(MAXN, 0);
    cur_pos = 0;
    dfs(0);
    decompose(0, 0);
    seg::build(0, n - 1, 1);
  }
  int query_path(int a, int b)
  {
    int res = -1;
    for (; head[a] != head[b]; b = parent[head[b]])
    {
      if (depth[head[a]] > depth[head[b]])
        swap(a, b);
      res = max(res, seg::query(0, n - 1, pos[head[b]],
          pos[b], 1));
    }
    if (depth[a] > depth[b])
      swap(a, b);
    res = max(res, seg::query(0, n - 1, pos[a] + 1, pos[
        b], 1));
    return res;
  }
  void update_path(int a, int b, int x)
  {
    for (; head[a] != head[b]; b = parent[head[b]])
    {
      if (depth[head[a]] > depth[head[b]])
        swap(a, b);
      seg::update(1, 0, n - 1, pos[head[b]], pos[b], x);
    }
    if (depth[a] > depth[b])
      swap(a, b);
    seg::update(1, 0, n - 1, pos[a] + 1, pos[b], x);
  }
  void update_subtree(int a, int x)
  {
    seg::update(1, 0, n - 1, pos[a] + 1, pos[a] + sz[a]
        - 1, x);
  }
  int query_subtree(int a, int x)
  {
    return seg::query(0, n - 1, pos[a] + 1, pos[a] + sz[
        a] - 1, 1);
  }
} // namespace hld
signed main()
```

```
{
  int q;
  scanf("%d", &q);
  while (q--)
  {
    scanf("%d", &n);
    for (int i = 0; i < n; i++)
      adj[i].clear();
    edges.clear();
    for (int i = 0; i < n - 1; i++)
    {
      int a, b, c;
      scanf("%d %d %d", &a, &b, &c);
      a--, b--;
      adj[a].pb({b, c});
      adj[b].pb({a, c});
      edges.pb({a, b});
    }
    hld::init();
    while (true)
    {
      char k[10];
      scanf("%s", k);
      if (k[0] == 'Q')
      {
        int a, b;
        scanf("%d %d", &a, &b);
        a--, b--;
        printf("%d\n", hld::query_path(a, b));
      }
      else if (k[0] == 'C')
      {
        int a, b;
        scanf("%d %d", &a, &b);
        a--;
        hld::update_path(edges[a].fir, edges[a].sec, b);
      }
      else
      {
        break;
      }
    }
  }
  return 0;
}
```

7.15 dinic

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 705
#define mod 1000000007
#define INF 1e9

struct edge
{
  int to, from, flow, capacity, id;
};
struct dinic
{
  int n, src, sink;
  vector<vector<edge>> adj;
  vector<int> level;
  vector<int> ptr;

  dinic(int sz)
  {
    n = sz;
    adj.resize(n);
    level.resize(n);
    ptr.resize(n);
  }
  void add_edge(int a, int b, int c, int id)
  {
    adj[a].pb({b, (int)adj[b].size(), c, c, id});
    adj[b].pb({a, (int)adj[a].size() - 1, 0, 0, id});
  }
  bool bfs()
  {
    level.assign(n, -1);
    level[src] = 0;
    queue<int> q;
    q.push(src);
    while (!q.empty())
    {
```

```
    int u = q.front();
    q.pop();
    for (auto at : adj[u])
    {
      if (at.flow && level[at.to] == -1)
      {
        q.push(at.to);
        level[at.to] = level[u] + 1;
      }
    }
  }
  return level[sink] != -1;
}
int dfs(int u, int flow)
{
  if (u == sink || flow == 0)
    return flow;
  for (int &p = ptr[u]; p < adj[u].size(); p++)
  {
    edge &at = adj[u][p];
    if (at.flow && level[u] == level[at.to] - 1)
    {
      int kappa = dfs(at.to, min(flow, at.flow));
      at.flow -= kappa;
      adj[at.to][at.from].flow += kappa;
      if (kappa != 0)
        return kappa;
    }
  }
  return 0;
}
int run()
{
  int max_flow = 0;
  while (bfs())
  {
    ptr.assign(n, 0);
    while (1)
    {
      int flow = dfs(src, INF);
      if (flow == 0)
        break;
      max_flow += flow;
    }
  }
  return max_flow;
}
vector<pii> cut_edges() // arestas do corte minimo
{
  bfs();
  vector<pii> ans;
  for (int i = 0; i < n; i++)
  {
    for (auto const &j : adj[i])
    {
      if (level[i] != -1 && level[j.to] == -1 && j.
          capacity > 0)
        ans.pb({j.capacity, {i, j.to}});
    }
  }
  return ans;
}
vector<int> flow_edges(int n, int m) // fluxo em cada
    aresta, na ordem da entrada
{
  vector<int> ans(m);
  for (int i = 0; i < n; i++)
  {
    for (auto const &j : adj[i])
      if (!j.capacity)
        ans[j.id] = j.flow;
  }
  return ans;
}
};
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n, m;
  cin >> n >> m;
  dinic d(n);
  for (int i = 0; i < m; i++)
  {
    int a, b, c;
    cin >> a >> b >> c;
    a--, b--;
    d.add_edge(a, b, c, i);
  }
  d.src = 0, d.sink = n - 1;
  cout << d.run() << endl;
  vector<int> ans = d.flow_edges(n, m);
  for (auto const &i : ans)
    cout << i << endl;
  return 0;
}
```

7.16 hopcroft karp

```
#include <bits/stdc++.h>
```

```cpp
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 200001
#define mod 1000000007
#define INF 1e9

struct hopcroft_karp
{
  vector<int> match;
  vector<int> dist;
  vector<vector<int>> adj;
  int n, m, t;

  hopcroft_karp(int a, int b)
  {
    n = a, m = b;
    t = n + m + 1;
    match.assign(t, n + m);
    dist.assign(t, 0);
    adj.assign(t, vector<int>{});
  }
  void add_edge(int u, int v)
  {
    adj[u].pb(v);
    adj[v].pb(u);
  }
  bool bfs()
  {
    queue<int> q;
    for (int u = 0; u < n; u++)
    {
      if (match[u] == n + m)
        dist[u] = 0, q.push(u);
      else
        dist[u] = INF;
    }
    dist[n + m] = INF;
    while (!q.empty())
    {
      int u = q.front();
      q.pop();
      if (dist[u] < dist[n + m])
      {
        for (auto const &v : adj[u])
        {
          if (dist[match[v]] == INF)
          {
            dist[match[v]] = dist[u] + 1;
            q.push(match[v]);
          }
        }
      }
    }
    return dist[n + m] < INF;
  }
  bool dfs(int u)
  {
    if (u < n + m)
    {
      for (auto const &v : adj[u])
      {
        if (dist[match[v]] == dist[u] + 1 && dfs(match[v
            ]))
        {
          match[v] = u;
          match[u] = v;
          return true;
        }
      }
      dist[u] = INF;
      return false;
    }
    return true;
  }
  vector<pi> run()
  {
    int cnt = 0;
    while (bfs())
      for (int u = 0; u < n; u++)
        if (match[u] == n + m && dfs(u))
          cnt++;
    vector<pi> ans;
    for (int v = n; v < n + m; v++)
      if (match[v] < n + m)
        ans.pb({match[v], v});
    return ans;
  }
```

```
};
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  return 0;
}
// hopcroft-karp
// maximum bipartite matching
// O(sqrt(V) + E)
// 0-indexed
```

7.17 dsu rollback

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 600005
#define mod 1000000007

namespace dsu
{
  struct rollback
  {
    int u, v, rankv, ranku;
  };

  int num_sets;
  int parent[MAXN];
  int rank[MAXN];
  stack<rollback> op;

  int Find(int i)
  {
    return (parent[i] == i) ? i : Find(parent[i]);
  }
  bool Union(int x, int y)
  {
    int xx = Find(x);
    int yy = Find(y);
    if (xx != yy)
    {
      num_sets--;
      if (rank[xx] > rank[yy])
        swap(xx, yy);
      op.push({xx, yy, rank[xx], rank[yy]});
      parent[xx] = yy;
      if (rank[xx] == rank[yy])
        rank[yy]++;
      return true;
    }
    return false;
  }
  void do_rollback()
  {
    if (op.empty())
      return;
    rollback x = op.top();
    op.pop();
    num_sets++;
    parent[x.v] = x.v;
    rank[x.v] = x.rankv;
    parent[x.u] = x.u;
    rank[x.u] = x.ranku;
  }
  void init(int n)
  {
    for (int i = 0; i < n; i++)
    {
      parent[i] = i;
      rank[i] = 0;
    }
    num_sets = n;
  }
}
namespace seg
{
  struct query
  {
    int v, u, is_bridge;
  };

  vector<vector<query>> t(4 * MAXN);
  int ans[MAXN];

  void add(int i, int l, int r, int ql, int qr, query q)
  {
```

```
    if (l > r || l > qr || r < ql)
      return;
    if (l >= ql && r <= qr)
    {
      t[i].push_back(q);
      return;
    }
    int mid = (l + r) >> 1;
    add((i << 1), l, mid, ql, qr, q);
    add((i << 1) | 1, mid + 1, r, ql, qr, q);
  }
  void dfs(int i, int l, int r)
  {
    for (query &q : t[i])
      if (dsu::Union(q.v, q.u))
        q.is_bridge = 1;
    if (l == r)
      ans[l] = dsu::num_sets;
    else
    {
      int mid = (l + r) >> 1;
      dfs((i << 1), l, mid);
      dfs((i << 1) | 1, mid + 1, r);
    }
    for (query q : t[i])
      if (q.is_bridge)
        dsu::do_rollback();
  }
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n, q;
  cin >> n >> q;
  int time = 0;
  map<pi, int> tin;
  vector<int> queries;
  while (q--)
  {
    char t;
    cin >> t;
    if (t == '?')
    {
      queries.pb(++time);
    }
    else if (t == '+')
    {
      int a, b;
      cin >> a >> b;
      a--, b--;
      if (a > b)
        swap(a, b);
      tin[{a, b}] = ++time;
    }
    else
    {
      int a, b;
      cin >> a >> b;
      a--, b--;
      if (a > b)
        swap(a, b);
      seg::query kappa = {a, b, 0};
      seg::add(1, 0, MAXN - 1, tin[{a, b}], ++time,
          kappa);
      tin[{a, b}] = -1;
    }
  }
  for (auto const &i : tin)
  {
    if (i.sec != -1)
    {
      seg::query kappa = {i.fir.fir, i.fir.sec, 0};
      seg::add(1, 0, MAXN - 1, i.sec, ++time, kappa);
    }
  }
  dsu::init(n);
  seg::dfs(1, 0, MAXN - 1);
  for (auto const &i : queries)
    cout << seg::ans[i] << endl;
  return 0;
}
// https://codeforces.com/edu/course/2/lesson/7/3/
   practice/contest/289392/problem/C
// conectividade dinamica
// para uma query (u, v)
// podemos descrever em um intervalo [l, r]
// l = quando a aresta (u, v) foi adicionada
// r = quando a aresta (u, v) foi removida
// dai agora que temos um intervalo, podemos adicionar
// a query (u, v) em uma segtree "adaptada"
// no final rodamos um dfs nessa segtree e vamos
   atualizando as repostas das queries
// quando estamos em uma posicao na seg, dou union em
   todos os caras daquela posicao
// e em seguida chamo pros meus filhos, quando chego em
   uma folha, ela eh equivalente
// a uma unidade de "tempo", logo a resposta para aquele
    tempo eh a resposta atual no dsu
// e ao sair recursivamente, vou dando rollbacks no dsu
```

7.18 MatrixDijkstra

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

#define lli long long int
#define pb push_back
#define MAXN 10000000
typedef pair <int , int> pii ;

int t ;
int dist [MAXN] ;
bool visited [MAXN] ;
vector <pii> adj_list [MAXN] ;

void dijkstra (int s)
{
    dist[s] = 0 ;

    priority_queue <pii , vector<pii> , greater<pii>> q
        ;

    q.push(pii(dist[s], s)) ;

    while(1)
    {
        int davez = -1 ;
        int menor = INT_MAX ;

        while(!q.empty())
        {
            int atual = q.top().second ;
            q.pop() ;

            if(!visited[atual])
            {
                davez = atual;
                break;
            }

        }

        if(davez == -1)
        {
            break ;
        }

        visited[davez] = true ;

        for(int i = 0 ; i < adj_list[davez].size() ; i
            ++)
        {
            int distt  = adj_list[davez][i].first ;
            int atual = adj_list[davez][i].second ;

            if(dist[atual] > dist[davez] + distt)
            {
                dist[atual] = dist[davez] + distt ;
                q.push(pii(dist[atual] , atual)) ;
            }
        }
    }

}
void initialize ()
{
    for (int i = 0 ; i < t ; i++)
    {
        visited[i] = false ;
        dist[i] = INT_MAX ;
    }
}
int main()
{
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);

    int n , m ;
    cin >> n >> m ;
    t = n * m ;
    char array [t] ;

    for (int i = 0 ; i < t ; i++)
    {
        cin >> array[i] ;
    }

    for (int i = 0 ; i < t ; i++)
    {
        if (i >= m && array[i] != '#')
        {
            adj_list[i].pb(pii(1 , (i - m))) ;
        }
        if (i < (n * m) - m  && array[i] != '#')
        {
            adj_list[i].pb(pii(1 , (i + m))) ;
        }
        if (i % m != 0 && array[i] != '#')
        {
```

```
            adj_list[i].pb(pii(1 , (i - 1))) ;
        }
        if ((i + 1) % m != 0  && array[i] != '#')
        {
            adj_list[i].pb(pii(1 , (i + 1))) ;
        }

    }

    int q ;
    cin >> q ;

    while (q--)
    {
        int a , b , c , d , e ;
        cin >> a >> b >> c >> d >> e ;
        a-- , b-- , c-- , d-- ;

        int index1 = (m * a) + b ;
        int index2 = (m * c) + d ;

        adj_list[index1].pb(pii(e , index2)) ;
        adj_list[index2].pb(pii(e , index1)) ;
    }

    initialize () ;

    dijkstra(0) ;

    cout << dist[t - 1] << endl ;

    return 0 ;
}
```

7.19 bridges

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 400005
#define mod 1000000007

int n, m, timer;
vector<pi> edges;
vector<bool> is_bridge;
vector<pi> adj[MAXN];
int tin[MAXN];
int low[MAXN];
bool vis[MAXN];

void dfs(int v, int p)
{
  vis[v] = true;
  tin[v] = timer, low[v] = timer++;
  for (auto const &u : adj[v])
  {
    if (u.fir == p)
      continue;
    if (vis[u.fir])
    {
      low[v] = min(low[v], tin[u.fir]);
      continue;
    }
    dfs(u.fir, v);
    low[v] = min(low[v], low[u.fir]);
    if (low[u.fir] > tin[v])
      is_bridge[u.sec] = 1;
  }
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  cin >> n >> m;
  is_bridge.resize(m);
  for (int i = 0; i < m; i++)
  {
    int a, b;
    cin >> a >> b;
    a--, b--;
    edges.pb({a, b});
    adj[a].pb({b, i});
    adj[b].pb({a, i});
  }
  memset(tin, -1, sizeof(tin));
  memset(low, -1, sizeof(low));
  for (int i = 0; i < n; i++)
```

```
  {
    if (!vis[i])
      dfs(i, -1);
  }
  return 0;
}
```

7.20 Ford Fulkerson

```
// ford-fulkerson: obter qual o fluxo maximo de um
   vertice s ate um vertice d
// 1 - rodar um bfs para descobrir um novo caminho de s
   ate d
// 2 - apos isso pego a aresta de menor custo desse
   caminho e subtraio o valor dela nas outras arestas
   do caminho
// 3 - fluxo_maximo += custo da aresta de menor custo
   desse caminho
// 4 - rodar isso ate nao existirem mais caminhos
   disponiveis (com fluxo diferente de 0) entre s e d
// 5 - o fluxo maximo de s ate d sera a soma das arestas
    de menor custo de cada caminho feito

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

#define lli long long int
#define pb push_back
#define MAXN 10000
#define INF 999999

int n , m , a , b , c , s , d , max_flow , flow ;
vector <int> parent ;
vector <int> adj [MAXN] ;
int cost [MAXN][MAXN] ;
bool visited [MAXN] ;

void get_menor_custo (int v , int mincost)
{
    if (v == s)
    {
        flow = mincost ;
        return ;
    }
    else if (parent[v] != -1)
    {
        get_menor_custo(parent[v] , min(mincost , cost[
           parent[v]][v])) ;
        cost[parent[v]][v] -= flow ;
        cost[v][parent[v]] += flow ;
    }
}
void bfs ()
{
    visited[s] = true ;

    queue <int> q ;
    q.push(s) ;
    parent.assign(MAXN , -1) ;

    while (!q.empty())
    {
        int u = q.front() ;
        q.pop() ;

        if (u == d)
        {
            break ;
        }

        for (int j = 0 ; j < adj[u].size() ; j++)
        {
            int v = adj[u][j] ;

            if (cost[u][v] > 0 && !visited[v])
            {
                visited[v] = true ;
                q.push(v) ;
                parent[v] = u ;
            }
        }
    }
}
int ford_fulkerson ()
{
    max_flow = 0 ;

    while (1)
    {
        flow = 0 ;
        memset(visited , false , sizeof(visited));

        bfs() ;
        get_menor_custo(d , INF) ;

        if (flow == 0)
        {
            break ;
        }
```

```
        max_flow += flow ;
    }

    return max_flow ;
}
int main ()
{
    ios_base::sync_with_stdio(false) ;
    cin.tie(NULL) ;

    cin >> n >> m ;

    for (int i = 0 ; i < m ; i++)
    {
        cin >> a >> b >> c ;
        adj[a].pb(b);
        adj[b].pb(a);
        cost[a][b] = c ;
    }

    cin >> s >> d ;

    cout << ford_fulkerson() << endl ;

    return 0 ;
}
```

7.21 caminhoeuleriano

```
// caminho euleriano em um grafo
// passa por todas as arestas apenas uma unica vez e
   percorre todas elas
// condicao de existencia:
// todos os vertices possuem grau par (ciclo euleriano)
   comeca e acaba no mesmo vertice
// ou
// apenas 2 vertices possuem grau impar, todos os outros
    possuem grau par ou == 0.
// comeca num vertice de grau impar e termina num
   vertice de grau impar nesse caso.
// solucao:
// rodar um dfs com map de visited para as arestas
// no final por o source no vector path
// ao final teremos o caminho inverso no vector path
// note que o caminho inverso tambem e um caminho valido

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

#define lli long long int
#define pb push_back
#define in insert
#define pi pair<int, int>
#define pd pair<double, int>
#define pib pair<pi, bool>
#define mp make_pair
#define fir first
#define sec second
#define MAXN 10001
#define MAXL 1000001
#define mod 1000000007

int n, m, start;
vector<int> path;
vector<int> adj[MAXN];
map<pi, bool> visited;

void dfs(int s)
{
  for (int i = 0; i < adj[s].size(); i++)
  {
    int v = adj[s][i];
    if (!visited[mp(s, v)])
    {
      visited[mp(s, v)] = true;
      visited[mp(v, s)] = true;
      dfs(v);
    }
  }
  path.pb(s);
}
bool check()
{
  int odd = 0;
  for (int i = 0; i < n; i++)
    if (adj[i].size() & 1)
      odd++, start = i;
  return (odd == 0 || odd == 2);
}
signed main()
{
  cin >> n >> m;
  for (int i = 0; i < m; i++)
  {
    int a, b;
    cin >> a >> b;
    adj[a].pb(b);
    adj[b].pb(a);
  }
  start = 0;
```

```
  bool ok = check();
  (ok) ? cout << "Yes\n" : cout << "No\n";
  if (ok)
  {
    dfs(start);
    for (int i = 0; i < path.size(); i++)
      cout << path[i] << " ";
    cout << "\n";
  }
  return 0;
}
```

7.22 LCA

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define PI acos(-1)
#define pb push_back
#define int long long int
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define DEBUG 0
#define MAXN 100001
#define mod 1000000007

int n;
vector<int> adj[MAXN];

namespace lca
{
  int l, timer;
  vector<int> tin, tout, depth;
  vector<vector<int>> up;

  void dfs(int v, int p)
  {
    tin[v] = ++timer;
    up[v][0] = p;
    for (int i = 1; i <= l; i++)
      up[v][i] = up[up[v][i - 1]][i - 1];
    for (auto const &u : adj[v])
    {
      if (p == u)
        continue;
      depth[u] = depth[v] + 1;
      dfs(u, v);
    }
    tout[v] = ++timer;
  }
  bool is_ancestor(int u, int v)
  {
    return tin[u] <= tin[v] && tout[u] >= tout[v];
  }
  int binary_lifting(int u, int v)
  {
    if (is_ancestor(u, v))
      return u;
    if (is_ancestor(v, u))
      return v;
    for (int i = l; i >= 0; --i)
      if (!is_ancestor(up[u][i], v))
        u = up[u][i];
    return up[u][0];
  }
  void init()
  {
    tin.resize(n);
    tout.resize(n);
    depth.resize(n);
    timer = 0;
    l = ceil(log2(n));
    up.assign(n, vector<int>(l + 1));
    dfs(0, 0);
  }
  int dist(int s, int v)
  {
    int at = binary_lifting(s, v);
    return (depth[s] + depth[v] - 2 * depth[at]);
  }
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  cin >> n;
  for (int i = 0; i < n - 1; i++)
  {
    int a, b;
    cin >> a >> b;
    a--, b--;
```

```
    adj[a].pb(b);
    adj[b].pb(a);
  }
  lca::init();
  return 0;
}
```

7.23 Topological Sort

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

#define lli long long int
#define pb push_back
#define MAXN 10000

int n , m , a , b ;
vector <int> adj [MAXN] ;
int grau [MAXN];
vector <int> order ;

bool topological_sort ()
{
    int ini = 0 ;

    while (ini < order.size())
    {
        int atual = order[ini] ;
        ini++ ;

        for (int i = 0 ; i < adj[atual].size() ; i++)
        {
            int v = adj[atual][i] ;
            grau[v]-- ;

            if (grau[v] == 0)
            {
                order.pb(v) ;
            }
        }
    }

    return (order.size() == n) ? true : false ;
}
int main ()
{
    ios_base::sync_with_stdio(false) ;
    cin.tie(NULL) ;

    cin >> n >> m ;

    for (int i = 1 ; i <= m ; i++)
    {
        cin >> a >> b ;
        grau[a]++ ;
        adj[b].pb(a) ;
    }

    for (int i = 1 ; i <= n ; i++)
    {
        if (grau[i] == 0)
        {
            order.pb(i) ;
        }
    }

    if (topological_sort())
    {
        for (int i = 0 ; i < order.size() ; i++)
        {
            cout << order[i] << " " ;
        }

        cout << endl ;
    }
    else
    {
        cout << "Impossible\n" ;
    }

    return 0 ;
}
```

7.24 Dijkstra

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
   rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
```

```
#define sec second
#define DEBUG 1
#define MAXN 1001
#define mod 1000000007

int n, m;
vector<pi> adj[MAXN];
bool visited[MAXN];
int dist[MAXN];

void dijkstra(int s)
{
  for (int i = 0; i < n; i++)
  {
    dist[i] = INT_MAX;
    visited[i] = false;
  }
  priority_queue<pi, vector<pi>, greater<pi>> q;
  dist[s] = 0;
  q.push({dist[s], s});
  while (!q.empty())
  {
    int v = q.top().second;
    q.pop();
    if (visited[v])
      continue;
    visited[v] = true;
    for (auto const &u : adj[v])
    {
      if (dist[u.sec] > dist[v] + u.fir)
      {
        dist[u.sec] = dist[v] + u.fir;
        q.push({dist[u.sec], u.sec});
      }
    }
  }
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  cin >> n >> m;
  for (int i = 0; i < m; i++)
  {
    int a, b, c;
    cin >> a >> b >> c;
    a--, b--;
    adj[a].pb({c, b});
    adj[b].pb({c, a});
  }
  dijkstra(0);
}
```

7.25 centroid decomposition

```
// centroid de uma arvore -> e um no que ao ser removido da arvore, separaria as
// arvores resultantes de modo com que a maior arvore desse conjunto teria no maximo
// (n / 2) nos, sendo n o numero de nos da arvore. Para qualquer arvore com n nos,
// o centroid sempre existe.

//
////////////////////////////////////////////////////////////////////////////////


// centroid decomposition -> muito util para tentar diminuir a complexidade em certos
// tipos de consultas a serem feitas, uma maneira melhor de organizar a arvore.

// algoritimo:
// 1) o centroid e a raiz dessa nova arvore
// 2) achar o centroid das arvores menores que surgiram com a remocao do centroid "pai"
// 3) por uma aresta entre o centroid "filho" e o centroid "pai"
// 4) repetir isso ate todos os nos serem removidos
// 5) ao final teremos a centroid tree

#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>, rb_tree_tag, tree_order_statistics_node_update>;

#define PI acos(-1)
#define pb push_back
#define int long long int
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define DEBUG 0
#define MAXN 100001
```

```
#define mod 1000000007

int n;
vector<int> adj[MAXN];

namespace cd
{
  int sz;
  vector<int> adjl[MAXN];
  vector<int> father, subtree_size;
  vector<bool> visited;

  void dfs(int s, int f)
  {
    sz++;
    subtree_size[s] = 1;
    for (auto const &v : adj[s])
    {
      if (v != f && !visited[v])
      {
        dfs(v, s);
        subtree_size[s] += subtree_size[v];
      }
    }
  }
  int getCentroid(int s, int f)
  {
    bool is_centroid = true;
    int heaviest_child = -1;
    for (auto const &v : adj[s])
    {
      if (v != f && !visited[v])
      {
        if (subtree_size[v] > sz / 2)
          is_centroid = false;
        if (heaviest_child == -1 || subtree_size[v] >
            subtree_size[heaviest_child])
          heaviest_child = v;
      }
    }
    return (is_centroid && sz - subtree_size[s] <= sz /
        2) ? s : getCentroid(heaviest_child, s);
  }
  int decompose_tree(int s)
  {
    sz = 0;
    dfs(s, s);
    int cend_tree = getCentroid(s, s);
    visited[cend_tree] = true;
    for (auto const &v : adj[cend_tree])
    {
      if (!visited[v])
      {
        int cend_subtree = decompose_tree(v);
        adjl[cend_tree].pb(cend_subtree);
        adjl[cend_subtree].pb(cend_tree);
        father[cend_subtree] = cend_tree;
      }
    }
    return cend_tree;
  }
  void init()
  {
    subtree_size.resize(n);
    visited.resize(n);
    father.assign(n, -1);
    decompose_tree(0);
  }
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  cin >> n;
  for (int i = 0; i < n - 1; i++)
  {
    int a, b;
    cin >> a >> b;
    a--, b--;
    adj[a].pb(b);
    adj[b].pb(a);
  }
  cd::init();
  return 0;
}
```

7.26 DFS

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

#define MAXN 500000

int n , m ;
int visited [MAXN] ;
vector <int> adj_list [MAXN] ;

void dfs (int x)
{
    for (int i = 0 ; i < adj_list[x].size() ; i++)
```

```
    {
        int v = adj_list[x][i] ;

        if(visited[v] == -1)
        {
            visited[v] = visited[x] ;
            dfs(v) ;
        }
    }
}
void initialize ()
{
    for (int i = 1 ; i <= n ; i++)
    {
        visited[i] = -1 ;
    }
}
int main ()
{
    int a , b ;

    cin >> n >> m ;

    initialize();

    for (int i = 1 ; i <= m ; i++)
    {
        cin >> a >> b ;

        adj_list[a].push_back(b) ;
        adj_list[b].push_back(a) ;
    }

    dfs(1) ;

    return 0;
}
```

7.27 cycle detection

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;
#define PI acos(-1)
#define pb push_back
#define int long long int
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 205
#define MAXP 100001
#define mod 1000000007

int n, m, idx;
vector<int> cycles[MAXN];
vector<int> adj[MAXN];
int color[MAXN];
int parent[MAXN];
int ans[MAXN];

void dfs(int u, int p)
{
  if (color[u] == 2)
    return;
  if (color[u] == 1)
  {
    idx++;
    int curr = p;
    ans[curr] = idx;
    cycles[idx].pb(curr);
    while (curr != u)
    {
      curr = parent[curr];
      cycles[idx].pb(curr);
      ans[curr] = idx;
    }
    return;
  }
  parent[u] = p;
  color[u] = 1;
  for (auto const &v : adj[u])
    if (v != parent[u])
      dfs(v, u);
  color[u] = 2;
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  cin >> n >> m;
  for (int i = 0; i < m; i++)
  {
```

```
    int a, b;
    cin >> a >> b;
    a--, b--;
    adj[a].pb(b);
    adj[b].pb(a);
  }
  for (int i = 0; i < n; i++)
    if (!color[i])
      dfs(i, -1);
  cout << idx << endl;
  for (int i = 1; i <= idx; i++)
  {
    cout << cycles[i].size() << endl;
    for (auto const &j : cycles[i])
      cout << j + 1 << " ";
    cout << endl;
  }
  return 0;
}
```

7.28 Kruskal

```
// Algoritimo de kruskal - Achar a mst

// 1 - listar todas as arestas em ordem crescente.

// 2 - Cada aresta liga dois vertices x e y, checar se
    eles ja estao na mesma componente conexa
// (aqui, consideramos apenas as arestas ja colocadas na
     arvore).

// 3 - Se x e y estao na mesma componente, ignoramos a
    aresta e continuamos o procedimento
// (se a usassemos, formariamos um ciclo). Se estiverem
    em componentes distintas, colocamos a aresta
//na arvore e continuamos o procedimento.

// OBS: como a prioridade eh ordenar pelas menores
    distancias, basta botar o custo da aresta como
// first no vector das arestas para poder ordenar

// em suma: ordeno as arestas em ordem crescente com
    prioridade no custo, depois para cada aresta,
// se o find(x) != find(y) sendo x e y os vertices das
    arestas, eu adiciono eles a mst e dou um join
// nos dois, como as arestas tao ordenadas em ordem
    crescente, o primeiro que eu pego
// eh necessariamente a melhor opcao e assim a mst eh
    formada.

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

#define lli long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define mp make_pair
#define fir first
#define sec second
#define MAXN 100001

int n, m, a, b, c;
vector<pii> ar;
vector<pii> mst;
int pai[MAXN];
int peso[MAXN];

int find(int x)
{
  if (pai[x] == x)
  {
    return x;
  }
  return pai[x] = find(pai[x]);
}
void join(int a, int b)
{
  a = find(a);
  b = find(b);

  if (peso[a] < peso[b])
  {
    pai[a] = b;
  }
  else if (peso[b] < peso[a])
  {
    pai[b] = a;
  }
  else
  {
    pai[a] = b;
    peso[b]++;
  }
}
void initialize()
{
  for (int i = 1; i <= n; i++)
  {
    pai[i] = i;
```

```
  }
}
int main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);

  cin >> n >> m;

  for (int i = 0; i < m; i++)
  {
    cin >> a >> b >> c;
    ar.pb(mp(c, mp(a, b)));
  }

  sort(ar.begin(), ar.end());

  initialize();

  int size = 0;

  for (int i = 0; i < m; i++)
  {
    if (find(ar[i].sec.fir) != find(ar[i].sec.sec))
    {
      join(ar[i].sec.fir, ar[i].sec.sec);
      mst.pb(mp(ar[i].fir, mp(ar[i].sec.fir, ar[i].sec.
          sec)));
    }
  }

  for (int i = 0; i < mst.size(); i++)
  {
    cout << mst[i].sec.fir << " " << mst[i].sec.sec << "
        " << mst[i].fir << endl;
  }

  return 0;
}
```

7.29 BFS

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define DEBUG 1
#define MAXN 1001
#define mod 1000000007

int n, m;
vector<int> adj[MAXN];
bool visited[MAXN];

void bfs(int s)
{
  queue<int> q;
  q.push(s);
  while (!q.empty())
  {
    int v = q.front();
    q.pop();
    if (visited[v])
      continue;
    visited[v] = true;
    for (auto const &u : adj[v])
      if (!visited[u])
        q.push(u);
  }
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  cin >> n >> m;
  for (int i = 0; i < m; i++)
  {
    int a, b, c;
    cin >> a >> b >> c;
    a--, b--;
    adj[a].pb(b);
    adj[b].pb(a);
  }
  bfs(0);
}
```

8 Strings

8.1 suffix automaton

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 100001
#define mod 998244353

namespace sa
{
  struct state
  {
    int len, suf_link;
    map<char, int> nxt;
  };

  state st[2 * MAXN];
  int dp[2 * MAXN];
  int sz, last;

  void init()
  {
    memset(dp, -1, sizeof(dp));
    st[0].len = 0;
    st[0].suf_link = -1;
    sz++;
    last = 0;
  }
  void get_link(int curr, int p, char c)
  {
    while (p != -1 && !st[p].nxt.count(c))
    {
      st[p].nxt[c] = curr;
      p = st[p].suf_link;
    }
    if (p == -1)
    {
      st[curr].suf_link = 0;
      return;
    }
    int q = st[p].nxt[c];
    if (st[p].len + 1 == st[q].len)
    {
      st[curr].suf_link = q;
      return;
    }
    int clone = sz;
    sz++;
    st[clone].len = st[p].len + 1;
    st[clone].nxt = st[q].nxt;
    st[clone].suf_link = st[q].suf_link;
    while (p != -1 && st[p].nxt[c] == q)
    {
      st[p].nxt[c] = clone;
      p = st[p].suf_link;
    }
    st[q].suf_link = clone;
    st[curr].suf_link = clone;
  }
  void build(string &s)
  {
    for (auto const &c : s)
    {
      int curr = sz;
      sz++;
      st[curr].len = st[last].len + 1;
      get_link(curr, last, c);
      last = curr;
    }
  }
  void dfs2(int v)
  {
    if (dp[v] != -1)
      return;
    dp[v] = 1;
    for (auto const &u : st[v].nxt)
    {
      if (!u.sec)
        continue;
      dfs2(u.sec);
      dp[v] += dp[u.sec];
    }
  }
```

```
  void dfs(int v, int k, int &at, string &curr)
  {
    if (at == k)
      return;
    for (auto const &u : st[v].nxt)
    {
      if (!u.sec)
        continue;
      if (at + dp[u.sec] < k)
      {
        at += dp[u.sec];
        continue;
      }
      curr.pb(u.fir);
      at++;
      dfs(u.sec, k, at, curr);
      if (at == k)
        return;
      curr.pop_back();
    }
  }
  void find_kth(int k)
  {
    int at = 0;
    string curr = "";
    dfs(0, k, at, curr);
    cout << curr << endl;
  }
} // namespace sa
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  string s;
  cin >> s;
  sa::init();
  sa::build(s);
  sa::dfs2(0);
  int q;
  cin >> q;
  while (q--)
  {
    int k;
    cin >> k;
    sa::find_kth(k);
  }
  return 0;
}
// https://cp-algorithms.com/string/suffix-automaton.
   html
// suffix automaton
// definicao: um suffix automaton de uma string s e um
   automato finito deterministico
// que aceita todos os suffixos da string s.
// ou seja:
// um suffix automaton eh um grafo aciclico orientado
// tal que, um vertice representa um estado
// e uma aresta representa uma transicao (um caractere a
    mais em relacao ao estado(suffixo) atual)
// t0 -> estado inicial(string vazia), e todos os demais
    estados podem ser alcancados a partir de t0
// o suffix automaton minimiza o numero de vertices

// a propiedade mais importante de um suffix automaton
   eh a de que
// ele contem informacoes sobre todas as substrings de s
// pois, qualquer caminho comecando do estado t0
   corresponde a uma substring de s

// conceitos:

// 1 - endpos
// seja t uma substring de s, endpos(t) eh o conjunto de
    todas os indices(posicoes)
// na string s no qual todas as ocorrencias de t acabam
// por exemplo, se s = "abcbc" e t = "bc"
// logo endpos(t) = {2, 4}
// com isso se duas duas substrings t1 e t2 possuem os
   seus endpos iguais,
// chamamos de endpos-equivalent e dai podemos extrair
   algumas informacoes
// info 1: se duas substrings u e w u.size() <= w.size()
   , se u eh um sufixo de w, logo endpos(u) esta
   contido em endpos(w)
// info 2: se duas substrings u e w u.size() <= w.size()
   , se u nao eh um sufixo de w, logo nao existe
   interseccao entre endpos(u) e endpos(w)

// 2 - suffix link
// seja v algum estado != t0,  sabemos que v corresponde
    a classe de strings que possui os mesmos endpos
// seja w a maior dessas strings, com isso, todas as
   demais sao suffixos de w
// com isso um suffix_link(v) corresponde ao maior
   suffix de w que esta em outra classe de equivalencia
    pelos endpos
// com isso podemos abstrair algumas informacoes:
// info 1: os suffix links foram uma arvore enraizada em
    t0
// info 2: se construirmos uma arvore usando os sets
```

```
    endpos, a estrutura sera a arvore com os suffix
    links

// com isso, vamos ao algoritimo
// 1 - vai ser online, e iremos adicionar os caracteres
    de 1 por 1, da esquerda para a direita
// 2 - com isso para adicionar um novo char, seja v o
    ultimo estado que adicionamos antes do atual,
    adicionamos uma aresta
// do proximo em relacao a ele e iremos procurar pelo
    suffix link para adicionar
// 3 - complexidade O(n) ou O(n log k), se usarmos uma
    map para guardar as transicoes partindo de um estado

// exemplos de aplicacoes:

// 1 - checar se t aparece em s como substring:
// construa o suffix automaton de s, e vamos tentar
    fazer um caminho partindo de t0
// se em algum momento, nao existir transicao, logo nao
    existe
// se conseguir chegar no final, existe

// 2 - numero de substrings diferentes de s
// constura o suffix automaton de s, sabemos que, cada
    substring de s corresponde a um caminho no automato
// com isso, o numero de substrings distintas eh o
    numero de caminhos diferentes que comecam de t0
// e terminam em algum canto
// isso pode ser calculado facilmente com uma dpzinha

// 3 - tamanho total de todas as substrings distintas de
     s
// similar a solucao passada, podemos fazer isso com uma
     dpzinha :)

// 4 - a k-esima menor substring lexicografica
// a k-esima menor substring lexicograficamente
    corresponde ao k-esimo path no suffix automaton
// se considerarmos as transicoes sempre indo do menor
    char para o maior durante o percurso

// 5 - o menor cyclic shift
// construa o suffix automaton da string s + s (
    duplicada)
// com isso o suffix automaton vai conter todos os
    cyclic shifts da string s
// e agora o problema eh reduzido para: encontre o menor
     caminho lexicograficamente de tamanho s.size()

// 6 - numero de ocorrencias de uma substring t em s
// construa o suffix automaton da string s
// com isso, quando criamos um no que nao seja o t0 nem
    um clone
// inicializamos cnt[v] = 1
// depois vamos percorrer todo os estados em ordem
    decrescente de len
// e aplicando cnt[link(v)] += cnt[v]
// no final, para responder a query basta fazer o
    caminho ate o estado que quisermos e printar o cnt
    dele

// e mais uma porrada de aplicacoes alem dessas :)

// example of a problem: https://www.spoj.com/problems/
    SUBLEX/
// ver qual a k-th string lexicografica sem repeticao
// note que o k pode ser gigante
// ideia: calcular dp[v] -> quantidade de caminhos que
    comecam em v
// dai para cada query roda um dfs, sendo que, so vou
    pro proximo estado se at + dp[u] >= k
// caso contrario, posso ignorar
```

8.2 suffix array

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<pi, int>
#define pci pair<char, int>
#define fir first
#define sec second
#define MAXN 50005
#define mod 1000000007

void get_suf(string s)
{
  s += '$';
  int n = s.size();
```

```
  vector<int> p(n), c(n);
  vector<pci> a(n);
  for (int i = 0; i < n; i++)
    a[i] = {s[i], i};
  sort(a.begin(), a.end());
  for (int i = 0; i < n; i++)
    p[i] = a[i].sec;
  c[p[0]] = 0;
  for (int i = 1; i < n; i++)
    (a[i].fir == a[i - 1].fir) ? c[p[i]] = c[p[i - 1]] :
        c[p[i]] = c[p[i - 1]] + 1;
  int k = 0;
  while ((1 << k) < n)
  {
    vector<pii> v(n);
    for (int i = 0; i < n; i++)
      v[i] = {{c[i], c[(i + (1 << k)) % n]}, i};
    sort(v.begin(), v.end());
    for (int i = 0; i < n; i++)
      p[i] = v[i].sec;
    c[p[0]] = 0;
    for (int i = 1; i < n; i++)
      (v[i].fir == v[i - 1].fir) ? c[p[i]] = c[p[i - 1]]
          : c[p[i]] = c[p[i - 1]] + 1;
    k++;
  }
  for (int i = 0; i < n; i++)
    cout << p[i] << " ";
  cout << endl;
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  string s;
  cin >> s;
  get_suf(s);
  return 0;
}
```

8.3 kmp

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<pi, int>
#define fir first
#define sec second
#define MAXN 100005
#define mod 998244353

string s;
int n, m;
string a, b;
int c[MAXN][26];

vector<int> kmp(string &s)
{
  int n = s.size();
  vector<int> p(n);
  for (int i = 1; i < n; i++)
  {
    int j = p[i - 1];
    while (j > 0 && s[i] != s[j])
      j = p[j - 1];
    if (s[i] == s[j])
      j++;
    p[i] = j;
  }
  return p;
}
void compute(string s)
{
  s.pb('*');
  vector<int> p = kmp(s);
  for (int i = 0; i < s.size(); i++)
  {
    for (int cc = 0; cc < 26; cc++)
    {
      int j = i;
      while (j > 0 && 'a' + cc != s[j])
        j = p[j - 1];
      if ('a' + cc == s[j])
        j++;
      c[i][cc] = j;
    }
  }
}
```

```
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  string s;
  cin >> s;
  compute(s);
  return 0;
}
// kmp
// algoritmo eh online, vai coonstruindo da esquerda pra
    direita
// calcula pi[i], a seguinte funcao:
// seja a substring s.substr(0, i + 1)
// pi[i] = tamanho do maior prefixo que tbm eh um sufixo
    dessa substring

// dai por exemplo
// da pra contar a quantidade de matchings de s em t
// so concatenar as strings fazendo: t = s + "*" + t
// dai contar as posicoes com pi[i] = s.size()

// tambem eh possivel construir um automato do kmp
// do tipo
// se meu pi[i] == x, e leio a letra c
// dai devo ir pro estado p[i] == y
// as transicoes podem ser computadas e isso pode ser
   muito util
```

8.4 aho corasick

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
   rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<pi, int>
#define fir first
#define sec second
#define DEBUG 0
#define MAXN 5001
#define mod 1000000007

namespace aho
{
  int go(int v, char ch);
  const int K = 26; // tamanho do alfabeto
  struct trie
  {
    char me;            // char correspondente ao no
       atual
    int go[K];          // proximo vertice que eu devo
       ir estando em um estado (v, c)
    int down[K];        // proximo vertice da trie
    int is_leaf = 0;    // se o vertice atual da trie eh
       uma folha (fim de uma ou mais strings)
    int parent = -1;    // no ancestral do no atual
    int link = -1;      // link de sufixo do no atual (
       outro no com o maior matching de sufixo)
    int exit_link = -1; // folha mais proxima que pode
       ser alcancada a partir de v usando links de
       sufixo
    trie(int p = -1, char ch = '$') : parent(p), me(ch)
    {
      fill(begin(go), end(go), -1);
      fill(begin(down), end(down), -1);
    }
  };
  vector<trie> ac;
  void init() // criar a raiz da trie
  {
    ac.resize(1);
  }
  void add_string(string s) // adicionar string na trie
  {
    int v = 0;
    for (auto const &ch : s)
    {
      int c = ch - 'a';
      if (ac[v].down[c] == -1)
      {
        ac[v].down[c] = ac.size();
        ac.emplace_back(v, ch);
      }
      v = ac[v].down[c];
    }
    ac[v].is_leaf++;
  }
  int get_link(int v) // pegar o suffix link saindo de v
  {
    if (ac[v].link == -1)
      ac[v].link = (!v || !ac[v].parent) ? 0 : go(
```

```
        get_link(ac[v].parent), ac[v].me);
    return ac[v].link;
  }
  int go(int v, char ch) // proximo estado saindo do
      estado(v, ch)
  {
    int c = ch - 'a';
    if (ac[v].go[c] == -1)
    {
      if (ac[v].down[c] != -1)
        ac[v].go[c] = ac[v].down[c];
      else
        ac[v].go[c] = (!v) ? 0 : go(get_link(v), ch);
    }
    return ac[v].go[c];
  }
  int get_exit_link(int v) // suffix link mais proximo
      de v que seja uma folha
  {
    if (ac[v].exit_link == -1)
    {
      int curr = get_link(v);
      if (!v || !curr)
        ac[v].exit_link = 0;
      else if (ac[curr].is_leaf)
        ac[v].exit_link = curr;
      else
        ac[v].exit_link = get_exit_link(curr);
    }
    return ac[v].exit_link;
  }
  int query(string s) // query O(n + ans)
  {
    int ans = 0, curr = 0, at;
    for (auto const &i : s)
    {
      curr = go(curr, i);
      ans += ac[curr].is_leaf;
      at = get_exit_link(curr);
      while (at)
      {
        ans += ac[at].is_leaf;
        at = get_exit_link(at);
      }
    }
    return ans;
  }
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n, q;
  cin >> n >> q;
  aho::init();
  for (int i = 0; i < n; i++)
  {
    string s;
    cin >> s;
    aho::add_string(s);
  }
  while (q--)
  {
    string t;
    cin >> t;
    cout << aho::query(t) << endl;
  }
  return 0;
}
// automato de aho-corasick
// imagine o seguinte problema:
// temos um conjunto de n strings
// e q queries para processar
// em cada uma das q queries, voce recebe uma string s
// e quer saber, o numero de ocorrencias de
// alguma string do conjunto como
// substring de s e em tempo linear
```

8.5 stringhashing2

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 2001
#define mod 1000000009
```

```
struct modint
{
  int val;
  modint(int v = 0) { val = v % mod; }
  int pow(int y)
  {
    modint x = val;
    modint z = 1;
    while (y)
    {
      if (y & 1)
        z *= x;
      x *= x;
      y >>= 1;
    }
    return z.val;
  }
  int inv() { return pow(mod - 2); }
  void operator=(int o) { val = o % mod; }
  void operator=(modint o) { val = o.val % mod; }
  void operator+=(modint o) { *this = *this + o; }
  void operator-=(modint o) { *this = *this - o; }
  void operator*=(modint o) { *this = *this * o; }
  void operator/=(modint o) { *this = *this / o; }
  bool operator==(modint o) { return val == o.val; }
  bool operator!=(modint o) { return val != o.val; }
  int operator*(modint o) { return ((val * o.val) % mod)
      ; }
  int operator/(modint o) { return (val * o.inv()) % mod
      ; }
  int operator+(modint o) { return (val + o.val) % mod;
      }
  int operator-(modint o) { return (val - o.val + mod) %
       mod; }
};
struct string_hashing
{
  modint d;
  modint h;
  vector<modint> pref;
  vector<modint> pot;

  string_hashing() {}
  string_hashing(int base, string &s)
  {
    d = base;
    pref.resize(s.size() + 1);
    pref[0] = 0;
    for (int i = 0; i < s.size(); i++)
    {
      modint val = pref[i] * d;
      pref[i + 1] = val + s[i];
    }
    h = pref[s.size()];
    pot.resize(s.size() + 1);
    pot[0] = 1;
    for (int i = 1; i <= s.size(); i++)
      pot[i] = pot[i - 1] * d;
  }
  modint get(int l, int r)
  {
    return pref[r + 1] - (pref[l] * pot[r - l + 1]);
  }
  modint append(modint hb, int blen)
  {
    h = hb + (h * pot[blen]);
    return h;
  }
};
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  string s;
  cin >> s;
  string_hashing h(256, s); // (base, string)
  // string_hashing h(227, s); // (base, string)
  return 0;
}
```

8.6 lcp in suffix array

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
   rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<pi, int>
#define pci pair<char, int>
#define fir first
#define sec second
#define MAXN 50005
```

```cpp
#define mod 1000000007

void radix(vector<pii> &v)
{
  {
    int n = v.size();
    vector<int> cnt(n);
    for (auto const &i : v)
      cnt[i.fir.sec]++;
    vector<pii> ans(n);
    vector<int> pos(n);
    pos[0] = 0;
    for (int i = 1; i < n; i++)
      pos[i] = pos[i - 1] + cnt[i - 1];
    for (auto const &i : v)
    {
      int k = i.fir.sec;
      ans[pos[k]] = i;
      pos[k]++;
    }
    v = ans;
  }
  {
    int n = v.size();
    vector<int> cnt(n);
    for (auto const &i : v)
      cnt[i.fir.fir]++;
    vector<pii> ans(n);
    vector<int> pos(n);
    pos[0] = 0;
    for (int i = 1; i < n; i++)
      pos[i] = pos[i - 1] + cnt[i - 1];
    for (auto const &i : v)
    {
      int k = i.fir.fir;
      ans[pos[k]] = i;
      pos[k]++;
    }
    v = ans;
  }
}
vector<int> get_lcp(string s)
{
  s += '$';
  int n = s.size();
  vector<int> p(n), c(n);
  vector<pci> a(n);
  for (int i = 0; i < n; i++)
    a[i] = {s[i], i};
  sort(a.begin(), a.end());
  for (int i = 0; i < n; i++)
    p[i] = a[i].sec;
  c[p[0]] = 0;
  for (int i = 1; i < n; i++)
    (a[i].fir == a[i - 1].fir) ? c[p[i]] = c[p[i - 1]] :
        c[p[i]] = c[p[i - 1]] + 1;
  int k = 0;
  while ((1 << k) < n)
  {
    vector<pii> v(n);
    for (int i = 0; i < n; i++)
      v[i] = {{c[i], c[(i + (1 << k)) % n]}, i};
    radix(v);
    for (int i = 0; i < n; i++)
      p[i] = v[i].sec;
    c[p[0]] = 0;
    for (int i = 1; i < n; i++)
      (v[i].fir == v[i - 1].fir) ? c[p[i]] = c[p[i - 1]]
          : c[p[i]] = c[p[i - 1]] + 1;
    k++;
  }
  for (auto const &i : p) // suffix array
    cout << i << " ";
  cout << endl;
  vector<int> lcp(n);
  k = 0;
  for (int i = 0; i < n - 1; i++)
  {
    int idx = c[i], j = p[idx - 1];
    while (s[i + k] == s[j + k])
      k++;
    lcp[idx] = k;
    k = max(k - 1, 0ll);
  }
  for (int i = 1; i < n; i++) // lcp between 2 adjacent
     suffixes of suffix array
    cout << lcp[i] << " ";
  cout << endl;
  return lcp;
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  string s;
  cin >> s;
  int n = s.size();
  vector<int> v = get_lcp(s);
  return 0;
}
```

8.7 substring fft

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define PI acos(-1)
#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<pi, int>
#define fir first
#define sec second
#define MAXN 100005
#define mod 1000000007
#define cd complex<double>

const double eps = 1e-12;
const int alphabet_size = 26;

namespace fft
{
  void dft(vector<cd> &a)
  {
    int n = a.size();
    if (n == 1)
      return;
    vector<cd> a0(n / 2), a1(n / 2);
    for (int i = 0; 2 * i < n; i++)
    {
      a0[i] = a[2 * i];
      a1[i] = a[2 * i + 1];
    }
    dft(a0);
    dft(a1);
    double ang = 2 * PI / n;
    cd w(1), wn(cos(ang), sin(ang));
    for (int i = 0; 2 * i < n; i++)
    {
      a[i] = a0[i] + w * a1[i];
      a[i + n / 2] = a0[i] - w * a1[i];
      w *= wn;
    }
  }
  void inverse_dft(vector<cd> &a)
  {
    int n = a.size();
    if (n == 1)
      return;
    vector<cd> a0(n / 2), a1(n / 2);
    for (int i = 0; 2 * i < n; i++)
    {
      a0[i] = a[2 * i];
      a1[i] = a[2 * i + 1];
    }
    inverse_dft(a0);
    inverse_dft(a1);
    double ang = 2 * PI / n * -1;
    cd w(1), wn(cos(ang), sin(ang));
    for (int i = 0; 2 * i < n; i++)
    {
      a[i] = a0[i] + w * a1[i];
      a[i + n / 2] = a0[i] - w * a1[i];
      a[i] /= 2;
      a[i + n / 2] /= 2;
      w *= wn;
    }
  }
  vector<double> mul(vector<cd> a, vector<cd> b)
  {
    int n = 1;
    vector<cd> fa(a.begin(), a.end()), fb(b.begin(), b.
      end());
    while (n < a.size() + b.size())
      n <<= 1;
    fa.resize(n);
    fb.resize(n);
    dft(fa);
    dft(fb);
    for (int i = 0; i < n; i++)
      fa[i] *= fb[i];
    inverse_dft(fa);
    vector<double> ans(n);
    for (int i = 0; i < n; i++)
      ans[i] = fa[i].real();
    return ans;
  }
} // namespace fft
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  string s, t;
```

```
  cin >> s >> t;
  int n = s.size(), m = t.size();
  reverse(t.begin(), t.end());
  vector<cd> a(n);
  vector<cd> b(m);
  for (int i = 0; i < n; i++)
  {
    int ch = s[i] - 'a';
    double ang = (2 * PI * ch) / alphabet_size;
    a[i] = cd(cos(ang), sin(ang));
  }
  for (int i = 0; i < m; i++)
  {
    int ch = t[i] - 'a';
    double ang = (2 * PI * ch) / alphabet_size;
    b[i] = cd(cos(ang), -sin(ang));
  }
  vector<double> ans = fft::mul(a, b);
  int matches = 0;
  for (int i = m - 1; i < n; i++)
    matches += (abs(ans[i] - m) <= eps);
  cout << matches << endl;
  return 0;
}
// number of matches of a pattern in string
// using fft
```

8.8 suffix array2

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<pi, int>
#define pci pair<char, int>
#define fir first
#define sec second
#define MAXN 50005
#define mod 1000000007

void radix(vector<pii> &v)
{
  {
    int n = v.size();
    vector<int> cnt(n);
    for (auto const &i : v)
      cnt[i.fir.sec]++;
    vector<pii> ans(n);
    vector<int> pos(n);
    pos[0] = 0;
    for (int i = 1; i < n; i++)
      pos[i] = pos[i - 1] + cnt[i - 1];
    for (auto const &i : v)
    {
      int k = i.fir.sec;
      ans[pos[k]] = i;
      pos[k]++;
    }
    v = ans;
  }
  {
    int n = v.size();
    vector<int> cnt(n);
    for (auto const &i : v)
      cnt[i.fir.fir]++;
    vector<pii> ans(n);
    vector<int> pos(n);
    pos[0] = 0;
    for (int i = 1; i < n; i++)
      pos[i] = pos[i - 1] + cnt[i - 1];
    for (auto const &i : v)
    {
      int k = i.fir.fir;
      ans[pos[k]] = i;
      pos[k]++;
    }
    v = ans;
  }
}
void get_suf(string s)
{
  s += '$';
  int n = s.size();
  vector<int> p(n), c(n);
  vector<pci> a(n);
  for (int i = 0; i < n; i++)
    a[i] = {s[i], i};
  sort(a.begin(), a.end());
  for (int i = 0; i < n; i++)
    p[i] = a[i].sec;
  c[p[0]] = 0;
```

```
  for (int i = 1; i < n; i++)
    (a[i].fir == a[i - 1].fir) ? c[p[i]] = c[p[i - 1]] :
        c[p[i]] = c[p[i - 1]] + 1;
  int k = 0;
  while ((1 << k) < n)
  {
    vector<pii> v(n);
    for (int i = 0; i < n; i++)
      v[i] = {{c[i], c[(i + (1 << k)) % n]}, i};
    radix(v);
    for (int i = 0; i < n; i++)
      p[i] = v[i].sec;
    c[p[0]] = 0;
    for (int i = 1; i < n; i++)
      (v[i].fir == v[i - 1].fir) ? c[p[i]] = c[p[i - 1]]
          : c[p[i]] = c[p[i - 1]] + 1;
    k++;
  }
  for (int i = 0; i < n; i++)
    cout << p[i] << " ";
  cout << endl;
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  string s;
  cin >> s;
  get_suf(s);
  return 0;
}
// problems like:
// search if a string t is equal to any substring in s
// counting how many times string t occurs as substring
   in s
// number of different substrings
// the longest common substring beetween s and t
// and many others
// can be solved using suffix array
```

8.9 z-function

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
   rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 2001
#define mod 1000000007

vector<int> z_function(string &s)
{
  int n = s.size();
  vector<int> z(n);
  z[0] = n;
  for (int i = 1, l = 0, r = 0; i < n; i++)
  {
    if (i <= r)
      z[i] = min(r - i + 1, z[i - l]);
    while (i + z[i] < n && s[z[i]] == s[i + z[i]])
      z[i]++;
    if (i + z[i] - 1 > r)
      l = i, r = i + z[i] - 1;
  }
  return z;
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  string s;
  cin >> s;
  vector<int> z = z_function(s);
}
// z-function
// calcula para cada i:
// z[i] = o tamanho de lcp(s, s.substr(i, n - i))
// lcp -> longest comom prefix
```

8.10 rabin-karp

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;
```

```
template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<pi, int>
#define fir first
#define sec second
#define DEBUG 0
#define MAXN 100001

const int p = 31;
const int mod = 1e9 + 9;

int multiplicate(int x, int y)
{
  return (x * y) % mod;
}
int subtract(int a, int b)
{
  return (a - b < 0) ? a - b + mod : a - b;
}
int sum(int a, int b)
{
  return (a + b >= mod) ? a + b - mod : a + b;
}
vector<int> rabin_karp(string s, string t)
{
  int n = s.size(), m = t.size();
  vector<int> pot(n);
  pot[0] = 1;
  for (int i = 1; i < n; i++)
    pot[i] = multiplicate(pot[i - 1], p);
  vector<int> pref(n + 1, 0);
  for (int i = 0; i < n; i++)
  {
    int val = multiplicate(pref[i], p);
    pref[i + 1] = sum(s[i], val);
  }
  int hs = 0;
  for (int i = 0; i < m; i++)
  {
    int val = multiplicate(hs, p);
    hs = sum(t[i], val);
  }
  vector<int> ans;
  for (int i = 0; i + m - 1 < n; i++)
  {
    int cur_h = subtract(pref[i + m], multiplicate(pref[
        i], pot[m]));
    if (cur_h == hs)
      ans.pb(i);
  }
  return ans;
}
signed main()
{
  string s, t;
  cin >> s >> t;
  vector<int> ans = rabin_karp(s, t);
  for (auto const &i : ans)
    cout << i << " " << i + t.size() - 1 << endl;
  return 0;
}
// rabin-karp for pattern matching
// given two string s and t, determine all occurrences
    of t in s
// 1- calcule the hash of string t
// 2- calcule the prefix hash of string s
// 3- compare every substring of s with length |t|
// 4- store all occurrences in a vector and return this
    vector
// complexity: O(|t| + |s|)
```

8.11 manacher

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

#define PI acos(-1)
#define int long long int
#define pb push_back
#define pi pair<int, int>
#define fir first
#define sec second
#define MAXN 100001
#define mod 1000000007

vector<int> d1;
vector<int> d2;

void manacher(string s)
{
  d1.resize(s.size());
  d2.resize(s.size());
  int l = 0, r = -1;
  for (int i = 0; i < s.size(); i++)
  {
```

```
    int k = (i > r) ? 1 : min(d1[l + r - i], r - i + 1);
    while (0 <= i - k && i + k < s.size() && s[i - k] ==
        s[i + k])
      k++;
    d1[i] = k;
    k = k - 1;
    if (i + k > r)
      l = i - k, r = i + k;
  }
  l = 0, r = -1;
  for (int i = 0; i < s.size(); i++)
  {
    int k = (i > r) ? 0 : min(d2[l + r - i + 1], r - i +
        1);
    while (0 <= i - k - 1 && i + k < s.size() && s[i - k
        - 1] == s[i + k])
      k++;
    d2[i] = k;
    k = k - 1;
    if (i + k > r)
      l = i - k - 1, r = i + k;
  }
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  string s;
  cin >> s;
  manacher(s);
  return 0;
}

// algoritimo de manacher

// motivacao: dada uma string s, encontre todos os pares
    (l, r) tal que, a substring s[l,r]
// e palindroma.

// para cada posicao (0 <= i < s.size()), vamos
    encontrar os valores de d1[i] e d2[i],
// sendo estes o numero de palindromos com comprimentos
    impares e com comprimentos pares
// e com i sendo a posicao central desses palindromos

// algoritimo mais facil:
// para cada posicao (0 <= i < s.size()), ele tenta
    aumentar a resposta em 1
// ate q nao seja mais possivel
// while(s[i - curr] == s[i + curr])

// complexidade O(N^2)

// algoritimo de manacher:
// para cada posicao (0 <= i < s.size()):
// seja o par (l, r) os extremos da substring palindroma
    que possui o maior r entre todas as encontradas ate
    entao
// se i > r, o fim do ultimo palindromo foi antes de i:
    iremos rodar o algoritimo mais facil mais facil e ir
    ate o limite.
// caso contrario, so precisamos rodar o algoritimo a
    partir de onde nao foi percorrido previamente.
// ao final se o r atual e maior do que o nosso antigo r
    , atualizamos o par (l, r)
// por incrivel que pareca, a complexidade e O(N)

// voltando para a motivacao:
// se temos os valores de d1[i] e d2[i]:
// a substring s[i - k, i + k] e palindroma, para todo
    (0 <= k < d1[i])
// a substring s[i - k - 1, i + k] e palindroma, para
    todo (0 <= k < d2[i])
// dai temos todos os intervalos
// note que a complexidade do algoritimo de manacher e O
    (N),
// mas como a quantidade maxima de palindromos em uma
    string e n^2,
// imprimir todos os intervalos consequentemente teria
    complexidade O(N^2) no pior caso
```

8.12 stringhashing

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<pi, int>
#define fir first
#define sec second
#define DEBUG 0
#define MAXN 5001
```

```cpp
#define mod 1000000007

int n;
vector<int> v;

int modpow(int x, int y)
{
  int z = 1;
  while (y)
  {
    if (y & 1)
      z = (z * x) % mod;
    x = (x * x) % mod;
    y >>= 1;
  }
  return z;
}
int inverse(int x)
{
  return modpow(x, mod - 2);
}
int divide(int x, int y)
{
  return (x * inverse(y)) % mod;
}
int subtract(int x, int y)
{
  return ((x + mod) - y) % mod;
}
int multiplicate(int x, int y)
{
  return (x * y) % mod;
}
int sum(int x, int y)
{
  return (x + y) % mod;
}

namespace sh
{
  const int d = 31;
  vector<int> pot;
  vector<int> pref;
  vector<int> suf;

  void calc()
  {
    pot.resize(n + 1);
    pot[0] = 1;
    for (int i = 1; i <= n; i++)
      pot[i] = multiplicate(pot[i - 1], d);
  }
  void suffix_hash()
  {
    suf.resize(n + 1);
    suf[0] = 0;
    for (int i = 0; i < n; i++)
    {
      int val = multiplicate(v[n - i - 1], pot[i]);
      suf[i + 1] = sum(suf[i], val);
    }
  }
  void prefix_hash()
  {
    pref.resize(n + 1);
    pref[0] = 0;
    for (int i = 0; i < n; i++)
    {
      int val = multiplicate(v[i], pot[i]);
      pref[i + 1] = sum(pref[i], val);
    }
  }
  int prefix(int l, int r)
  {
    return divide(subtract(pref[r + 1], pref[l]), pot[l
      ]);
  }
  int suffix(int l, int r)
  {
    return divide(subtract(suf[n - l], suf[n - r - 1]),
      pot[n - r - 1]);
  }
} // namespace sh
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  string s;
  cin >> s;
  n = s.size();
  for (auto const &i : s)
    v.pb((i - 'a') + 1);                // indexar a
      partir do 1
  sh::calc();                           // potencias de
    d
  sh::prefix_hash();                    // hashing dos
    prefixos de s
  cout << sh::prefix(0, n - 1) << endl; // resposta
    final
  return 0;
```

```
}
// string hashing
// podemos representar uma string como um valor inteiro
// seja s uma string e d o tamanho do alfabeto
// o valor de hashing de s eh igual a:
// (s[0] * pow(d, 0)) + (s[1] * pow(d, 1)) + ... (s[n - 1] * pow(d, n - 1))
// como esse valor pode ser gigantesco
// fazer isso com um modulo que for o maior possivel
// nesse caso usaremos 10^9 + 7
// logo o hashing fica:
// ((s[0] * pow(d, 0)) + (s[1] * pow(d, 1)) + ... (s[n - 1] * pow(d, n - 1))) % mod
// o hashing possui diversas aplicacoes como:
// checar substring que sao palindromas
// numeros de substrings diferentes em uma string
// etc...
```

9 Geometry

9.1 ConvexHull

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>, rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 500001
#define mod 1000000007
#define PI acos(-1)

namespace p
{
  struct pt
  {
    double x, y;
    pt operator+(pt p) { return {x + p.x, y + p.y}; } // soma de pontos
    pt operator-(pt p) { return {x - p.x, y - p.y}; } // subtracao de pontos
    pt operator*(double d) { return {x * d, y * d}; } // multiplicacao por um double
    pt operator/(double d) { return {x / d, y / d}; } // divisao por um double
  };
  double dot(pt v, pt w) // produto escalar (dot product)
  {
    return v.x * w.x + v.y * w.y;
  }
  bool is_perp(pt v, pt w) // retorna se dois vetores sao perpendiculares (angulo 90 graus)
  {
    return dot(v, w) == 0;
  }
  double cross(pt v, pt w) // produto vetorial (cross product)
  {
    return v.x * w.y - v.y * w.x;
  }
  double dist(pt a, pt b) // distancia entre 2 pontos
  {
    pt c = a - b;
    return sqrt(c.x * c.x + c.y * c.y);
  }
  double dist2(pt a, pt b) // retorna o quadrado da distancia entre dois pontos
  {
    pt c = a - b;
    return c.x * c.x + c.y * c.y;
  }
  bool is_colinear(pt a, pt b, pt c) // retorna se os pontos a, b e c sao colineares
  {
    return cross(b - a, c - a) == 0;
  }
  bool ccw(pt a, pt b, pt c) // retorna se os pontos a,b e c estao no sentido anti horario
  {
    return cross(b - a, c - b) > 0;
  }
  bool cw(pt a, pt b, pt c) // retorna se os pontos a,b e c estao no sentido horario
  {
    return cross(b - a, c - b) < 0;
  }
  double modulo(pt v) // |v| = sqrt(x2 + y2)
  {
```

```
  return sqrt(v.x * v.x + v.y * v.y);
}
double angle(pt a, pt b, pt c) // angulo entre os
    vetores ab e ac
{
  // dot(ab , ac) / |ab| * |ac|
  pt ab = b - a; // vetor ab
  pt ac = c - a; // vetor ac
  double m1 = modulo(ab);
  double m2 = modulo(ac);
  double m3 = m1 * m2;
  return (dot(ab, ac) / m3); // retorna o cos do
      angulo em graus
}
pt rotate(pt p, double a) // rotacionar o ponto p em
    relacao a origem, em a graus, no sentido anti-
    horario
{
  a = (a * PI) / 180;
  double xx = (cos(a) * p.x) + ((sin(a) * -1) * p.y);
  double yy = (sin(a) * p.x) + (cos(a) * p.y);
  pt ans = {xx, yy};
  return ans;
}
double polar(pt p) // polar angle
{
  return atan2l(p.y, p.x);
}
bool cmp(pt a, pt b) // ordenar pontos pelo polar
    angle
{
  return polar(a) < polar(b);
}
bool cmp_x(pt a, pt b) // ordenar os pontos pela
    coordenada x
{
  if (a.x != b.x)
    return a.x < b.x;
  return a.y < b.y;
}
vector<pt> convex_hull(vector<pt> v)
{
  sort(v.begin(), v.end(), cmp_x);
  pt p1 = v[0], p2 = v.back();
  vector<pt> up;
  vector<pt> down;
  up.pb(p1);
  down.pb(p1);
  for (int i = 1; i < v.size(); i++)
  {
    if (i == v.size() - 1 || cw(p1, v[i], p2))
    {
      while (up.size() >= 2 && !cw(up[up.size() - 2],
          up[up.size() - 1], v[i]))
        up.pop_back();
      up.pb(v[i]);
    }
  }
  for (int i = 1; i < v.size(); i++)
  {
    if (i == v.size() - 1 || ccw(p1, v[i], p2))
    {
      while (down.size() >= 2 && !ccw(down[down.size()
          - 2], down[down.size() - 1], v[i]))
        down.pop_back();
      down.pb(v[i]);
    }
  }
  int start = 0, limit = 0; // para por em ans no
      sentido anti-horario e a partir de start
  for (int i = 1; i < down.size(); i++)
    if ((down[i].y < down[start].y) || (down[i].y ==
        down[start].y && down[i].x < down[start].x))
      start = i;
  if (!start)
    limit = 1;
  vector<pt> ans;
  for (int i = start; i < down.size() - 1; i++)
    ans.pb(down[i]);
  for (int i = up.size() - 1; i >= limit; i--)
    ans.pb(up[i]);
  for (int i = 1; i < start; i++)
    ans.pb(down[i]);
  return ans;
 }
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n, t = 0;
  while (cin >> n)
  {
    cout << "caso " << t << ":" << endl;
    vector<p::pt> v(n);
    for (int i = 0; i < n; i++)
      cin >> v[i].x >> v[i].y;
    vector<p::pt> ans = p::convex_hull(v);
    for (auto const &i : ans)
      cout << i.x << " " << i.y << endl;
```

```
    cout << endl;
    t++;
  }
  return 0;
}
// conceitos importantes:
// 1- poligono: uma figura plana que possui no minimo 3
    lados e 3 angulos
// 2- poligono convexo: um poligono cujo todos os seus
    angulos internos sao menores do que 180 graus

// convex hull:
// dados n pontos em um plano, o objetivo e achar o
    menor poligono convexo que possui todos os n pontos
    dados
// Graham's Scan, complexidade O(n * log(n))

// ideia do algoritimo:
// 1- ache 2 pontos a e b tal que, a e o ponto mais a
    esquerda e b o ponto mais a direita do conjunto dado
// 2- a e b devem pertencer ao convex hull
// 3- desenhar uma linha ab, essa linha ira separar os
    outros pontos em 2 conjuntos s1 (superior) e s2 (
    inferior).
// 4- a e b pertencem aos dois conjuntos
// 5- agora para os conjuntos s1 e s2, achamos o convex
    hull dos dois conjuntos.
// 6- para isso, ordene todos os pontos pela cordenada x
// 7- para cada ponto, se o ponto dado pertence ao
    conjunto superior, verificamos o angulo formado pela
     linha
//    que liga o penultimo ponto e o ultimo ponto do
    convex hull superior, com a linha que conecta o
//    ultimo ponto do convex hull e o ponto atual. Se o
    angulo nao for no sentido horario,
//    removemos o ponto mais recente adicionado ao
    convex hull superior, pois o ponto atual sera capaz
//    de conter o ponto anterior, uma vez que seja
    adicionado ao convex hull.
// 8- fazer o mesmo para o conjunto inferior
// 9- ao final teremos o conjunto de pontos que formam o
     convex hull dos n pontos
```

9.2 minkowski

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 15
#define mod 1000000007

struct pt
{
  int x, y;
  bool operator<(pt ot)
  {
    if (x != ot.x)
      return x < ot.x;
    return y < ot.y;
  }
  void operator=(pt p) { x = p.x, y = p.y; }
  bool operator==(pt p) { return (x == p.x && y == p.y);
      }
  bool operator!=(pt p) { return (x != p.x || y != p.y);
      }
  pt operator+(const pt &p) { return {x + p.x, y + p.y};
      }
  pt operator-(const pt &p) { return {x - p.x, y - p.y};
      }
  pt operator*(int d) { return {x * d, y * d}; }
  pt operator/(int d) { return {x / d, y / d}; }
  int cross(pt ot) const { return x * ot.y - y * ot.x; }
  int cross(pt a, pt b) const { return (a - *this).cross
      (b - *this); }
};
enum type
{
  outside,
  inside,
  boundary
};
int cross(pt v, pt w)
{
  return v.x * w.y - v.y * w.x;
}
bool ccw(pt a, pt b, pt c)
```

```
{
  return cross(b - a, c - b) > 0;
}
void radial_sort(vector<pt> &a)
{
  pt pivot = *min_element(a.begin(), a.end());
  auto cmp = [&](pt p, pt q)
  {
    if (p == pivot || q == pivot)
      return q != pivot;
    return ccw(pivot, p, q) > 0;
  };
  sort(a.begin(), a.end(), cmp);
}
vector<pt> trata(vector<pt> p)
{
  vector<pt> ans;
  for (int i = 0; i < p.size(); i++)
  {
    while (ans.size() >= 2 && ans.back().cross(p[i], ans
        .end()[-2]) == 0)
      ans.pop_back();
    ans.pb(p[i]);
  }
  if (ans.size() > 2 && ans.back().cross(p[0], ans.end()
      [-2]) == 0)
    ans.pop_back();
  return ans;
}
void prepare(vector<pt> &p)
{
  radial_sort(p); // sort points in counter-clockwise
      order
  p = trata(p);   // and the polygon dont have 3
      colinear points
}
int sgn(int val)
{
  if (val > 0)
    return 1;
  else if (val < 0)
    return -1;
  return 0;
}
bool in_seg(pt p, pt a, pt b)
{
  // check if point p is in the line segment formed by a
      and b
  if (a.cross(b, p) == 0)
    return (p.x >= min(a.x, b.x) && p.x <= max(a.x, b.x)
        && p.y >= min(a.y, b.y) && p.y <= max(a.y, b.y)
        );
  return 0;
}
bool in_tri(pt p, pt a, pt b, pt c)
{
  // check if point p is in the triangle formed by a, b
      and c
  int a1 = abs(a.cross(b, c));
  int a2 = abs(p.cross(a, b)) + abs(p.cross(a, c)) + abs
      (p.cross(b, c));
  return a1 == a2;
}
int in_polygon(vector<pt> &poly, pt p)
{
  int n = poly.size();
  if (n == 1)
    return (p == poly[0]) ? type::boundary : type::
        outside;
  if (n == 2)
    return (in_seg(p, poly[0], poly[1])) ? type::
        boundary : type::outside;
  if (poly[0].cross(poly[1], p) != 0 && sgn(poly[0].
      cross(poly[1], p)) != sgn(poly[0].cross(poly[1],
      poly[n - 1])))
    return type::outside;
  if (poly[0].cross(p, poly[n - 1]) != 0 && sgn(poly[0].
      cross(p, poly[n - 1])) != sgn(poly[0].cross(poly
      [1], poly[n - 1])))
    return type::outside;
  int l = 2, r = n - 1;
  if (poly[0].cross(poly[l], p) > 0)
  {
    while (l < r)
    {
      int mid = (l + r) >> 1;
      (poly[0].cross(poly[mid], p) <= 0) ? r = mid : l =
          mid + 1;
    }
  }
  if (!in_tri(p, poly[0], poly[l - 1], poly[l]))
    return type::outside;
  if (in_seg(p, poly[l - 1], poly[l]))
    return type::boundary;
  if (in_seg(p, poly[0], poly[1]))
    return type::boundary;
  if (in_seg(p, poly[0], poly[n - 1]))
    return type::boundary;
  return type::inside;
}
```

```
vector<pt> minkowski(vector<pt> a, vector<pt> b)
{
  prepare(a);
  prepare(b);
  a.push_back(a[0]);
  a.push_back(a[1]);
  b.push_back(b[0]);
  b.push_back(b[1]);
  vector<pt> ans;
  int i = 0, j = 0;
  while (i < a.size() - 2 || j < b.size() - 2)
  {
    ans.pb(a[i] + b[j]);
    auto c = cross(a[i + 1] - a[i], b[j + 1] - b[j]);
    if (c >= 0)
      i++;
    if (c <= 0)
      j++;
  }
  return ans;
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  vector<pt> v;
  for (int _ = 0; _ < 3; _++)
  {
    int n;
    cin >> n;
    vector<pt> p(n);
    for (int i = 0; i < n; i++)
      cin >> p[i].x >> p[i].y;
    if (_ == 0)
      v = p;
    else
      v = minkowski(v, p);
  }
  prepare(v);
  int q;
  cin >> q;
  while (q--)
  {
    pt p;
    cin >> p.x >> p.y;
    p.x *= 3, p.y *= 3;
    // ve se o ponto (3x, 3y) esta na bora, dentro ou
       fora do poligono v
    (in_polygon(v, p) != type::outside) ? cout << "YES\n
       " : cout << "NO\n";
  }
  return 0;
}
// problema exemplo:
// https://codeforces.com/contest/87/problem/E
```

9.3 LineSweep

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
   rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 200005
#define mod 1000000007
#define PI acos(-1)

const double EPS = 1e-9;

struct pt
{
  double x, y;
};
struct seg
{
  pt p, q;
  int id;
  double get_y(double x) const
  {
    if (abs(p.x - q.x) < EPS)
      return p.y;
    return p.y + (q.y - p.y) * (x - p.x) / (q.x - p.x);
  }
};
bool intersect1d(double l1, double r1, double l2, double
    r2)
{
```

```
  if (l1 > r1)
    swap(l1, r1);
  if (l2 > r2)
    swap(l2, r2);
  return max(l1, l2) <= min(r1, r2) + EPS;
}
int vec(const pt &a, const pt &b, const pt &c)
{
  double s = (b.x - a.x) * (c.y - a.y) - (b.y - a.y) * (
      c.x - a.x);
  return abs(s) < EPS ? 0 : s > 0 ? +1
                                  : -1;
}
bool intersect(const seg &a, const seg &b)
{
  return intersect1d(a.p.x, a.q.x, b.p.x, b.q.x) &&
         intersect1d(a.p.y, a.q.y, b.p.y, b.q.y) &&
         vec(a.p, a.q, b.p) * vec(a.p, a.q, b.q) <= 0 &&
         vec(b.p, b.q, a.p) * vec(b.p, b.q, a.q) <= 0;
}
bool operator<(const seg &a, const seg &b)
{
  double x = max(min(a.p.x, a.q.x), min(b.p.x, b.q.x));
  return a.get_y(x) < b.get_y(x) - EPS;
}
struct event
{
  double x;
  int tp, id;
  event() {}
  event(double x, int tp, int id) : x(x), tp(tp), id(id)
      {}
  bool operator<(const event &e) const
  {
    if (abs(x - e.x) > EPS)
      return x < e.x;
    return tp > e.tp;
  }
};

set<seg> s;

set<seg>::iterator prev(set<seg>::iterator it)
{
  return it == s.begin() ? s.end() : --it;
}
set<seg>::iterator next(set<seg>::iterator it)
{
  return ++it;
}

pi line_sweep(vector<seg> v)
{
  vector<event> e;
  for (int i = 0; i < v.size(); i++)
  {
    e.push_back({min(v[i].p.x, v[i].q.x), 1, i});
    e.push_back({max(v[i].p.x, v[i].q.x), 0, i});
  }
  sort(e.begin(), e.end());
  for (int i = 0; i < e.size(); i++)
  {
    int id = e[i].id;
    if (e[i].tp == 1)
    {
      auto nxt = s.lower_bound(v[id]), prv = prev(nxt);
      if (nxt != s.end() && intersect(*nxt, v[id]))
        return {(*nxt).id, id};
      if (prv != s.end() && intersect(*prv, v[id]))
        return {(*prv).id, id};
      s.insert(nxt, v[id]);
    }
    else
    {
      auto where = s.lower_bound(v[id]);
      auto nxt = next(where), prv = prev(where);
      if (nxt != s.end() && prv != s.end() && intersect
          (*nxt, *prv))
        return {(*prv).id, (*nxt).id};
      s.erase(where);
    }
  }
  return {-1, -1};
}
signed main()
{
  int n;
  cin >> n;
  vector<seg> v(n);
  for (int i = 0; i < n; i++)
  {
    cin >> v[i].p.x >> v[i].p.y >> v[i].q.x >> v[i].q.y;
    v[i].id = i;
  }
  pi ans = line_sweep(v);
  if (ans.fir == -1)
  {
    cout << "NO\n";
  }
  else
  {
```

```
      cout << "YES\n";
      cout << ans.fir + 1 << " " << ans.sec + 1 << endl;
    }
    return 0;
}
// https://cp-algorithms.com/geometry/
    intersecting_segments.html
// https://acm.timus.ru/problem.aspx?space=1&num=1469
```

9.4 points and vectors

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 500001
#define mod 1000000007
#define PI acos(-1)

namespace p
{
  struct pt
  {
    double x, y;
    pt operator+(pt p) { return {x + p.x, y + p.y}; } //
        soma de pontos
    pt operator-(pt p) { return {x - p.x, y - p.y}; } //
        subtracao de pontos
    pt operator*(double d) { return {x * d, y * d}; } //
        multiplicacao por um double
    pt operator/(double d) { return {x / d, y / d}; } //
        divisao por um double
  };
  double dot(pt v, pt w) // produto escalar (dot product
      )
  {
    return v.x * w.x + v.y * w.y;
  }
bool is_perp(pt v, pt w) // retorna se dois vetores
    sao perpendiculares (angulo 90 graus)
{
  return dot(v, w) == 0;
}
double cross(pt v, pt w) // produto vetorial (cross
    product)
{
  return v.x * w.y - v.y * w.x;
}
double dist(pt a, pt b) // distancia entre 2 pontos
{
  pt c = a - b;
  return sqrt(c.x * c.x + c.y * c.y);
}
double dist2(pt a, pt b) // retorna o quadrado da
    distancia entre dois pontos
{
  pt c = a - b;
  return c.x * c.x + c.y * c.y;
}
bool is_colinear(pt a, pt b, pt c) // retorna se os
    pontos a, b e c sao colineares
{
  return cross(b - a, c - a) == 0;
}
bool ccw(pt a, pt b, pt c) // retorna se os pontos a,b
     e c estao no sentido anti horario
{
  return cross(b - a, c - b) > 0;
}
bool cw(pt a, pt b, pt c) // retorna se os pontos a,b
    e c estao no sentido horario
{
  return cross(b - a, c - b) < 0;
}
double modulo(pt v) // |v| = sqrt(x2 + y2)
{
  return sqrt(v.x * v.x + v.y * v.y);
}
double angle(pt a, pt b, pt c) // angulo entre os
    vetores ab e ac
{
  // dot(ab , ac) / |ab| * |ac|
  pt ab = b - a; // vetor ab
  pt ac = c - a; // vetor ac
  double m1 = modulo(ab);
  double m2 = modulo(ac);
  double m3 = m1 * m2;
  return (dot(ab, ac) / m3); // retorna o cos do
```

```
      angulo em graus
  }
  pt rotate(pt p, double a) // rotacionar o ponto p em
      relacao a origem, em a graus, no sentido anti-
      horario
  {
    a = (a * PI) / 180;
    double xx = (cos(a) * p.x) + ((sin(a) * -1) * p.y);
    double yy = (sin(a) * p.x) + (cos(a) * p.y);
    pt ans = {xx, yy};
    return ans;
  }
  double polar(pt p) // polar angle
  {
    return atan2l(p.y, p.x);
  }
  bool cmp(pt a, pt b) // ordenar pontos pelo polar
      angle
  {
    return polar(a) < polar(b);
  }
  bool cmp_x(pt a, pt b) // ordenar os pontos pela
      coordenada x
  {
    if (a.x != b.x)
      return a.x < b.x;
    return a.y < b.y;
  }
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  return 0;
}
```

10 Structures

10.1 persistent seg

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<pi, int>
#define fir first
#define sec second
#define MAXN 100003
#define mod 1000000007

int v[MAXN];

namespace seg
{
  struct node
  {
    int item, lazy, lazy_status, l, r;
  };

  int cnt;
  node seg[500 * MAXN];
  vector<int> roots;

  int neutral()
  {
    return 0;
  }
  int merge(int a, int b)
  {
    return a + b;
  }
  int newleaf(int vv)
  {
    int p = ++cnt;
    seg[p].l = 0;
    seg[p].r = 0;
    seg[p].lazy = 0;
    seg[p].lazy_status = 0;
    seg[p].item = vv;
    return p;
  }
  int newparent(int l, int r)
  {
    int p = ++cnt;
    seg[p].l = l;
    seg[p].r = r;
    seg[p].lazy = 0;
    seg[p].lazy_status = 0;
    seg[p].item = merge(seg[seg[p].l].item, seg[seg[p].r
        ].item);
```

```
  return p;
}
int newkid(int i, int diff, int l, int r)
{
  int p = ++cnt;
  seg[p].l = seg[i].l;
  seg[p].r = seg[i].r;
  seg[p].lazy = seg[i].lazy + diff;
  seg[p].lazy_status = 1;
  seg[p].item = seg[i].item + ((r - l + 1) * diff);
  return p;
}
void add(int i, int l, int r)
{
  if (!seg[i].lazy_status)
    return;
  if (l != r)
  {
    int mid = (l + r) >> 1;
    seg[i].l = newkid(seg[i].l, seg[i].lazy, l, mid);
    seg[i].r = newkid(seg[i].r, seg[i].lazy, mid + 1,
        r);
  }
  seg[i].lazy = 0;
  seg[i].lazy_status = 0;
}
int update(int i, int l, int r, int ql, int qr, int
    diff)
{
  if (l > r || l > qr || r < ql)
    return i;
  if (l >= ql && r <= qr)
    return newkid(i, diff, l, r);
  add(i, l, r);
  int mid = (l + r) >> 1;
  return newparent(update(seg[i].l, l, mid, ql, qr,
      diff), update(seg[i].r, mid + 1, r, ql, qr, diff
      ));
}
int query(int l, int r, int ql, int qr, int i)
{
  if (l > r || l > qr || r < ql)
    return neutral();
  if (l >= ql && r <= qr)
    return seg[i].item;
  add(i, l, r);
  int mid = (l + r) >> 1;
  return merge(query(l, mid, ql, qr, seg[i].l), query(
      mid + 1, r, ql, qr, seg[i].r));
}
```

```
int build(int l, int r)
{
  if (l == r)
    return newleaf(v[l]);
  int mid = (l + r) >> 1;
  return newparent(build(l, mid), build(mid + 1, r));
}
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n, q;
  cin >> n >> q;
  for (int i = 0; i < n; i++)
    cin >> v[i];
  int root = seg::build(0, n - 1);
  seg::roots.pb(root);
  while (q--)
  {
    char t;
    cin >> t;
    if (t == 'C')
    {
      int l, r, d;
      cin >> l >> r >> d;
      l--, r--;
      int root = seg::update(seg::roots.back(), 0, n -
          1, l, r, d);
      seg::roots.pb(root);
    }
    else if (t == 'Q')
    {
      int l, r;
      cin >> l >> r;
      l--, r--;
      cout << seg::query(0, n - 1, l, r, seg::roots.back
          ()) << endl;
    }
    else if (t == 'H')
    {
      int l, r, d;
      cin >> l >> r >> d;
      l--, r--;
      cout << seg::query(0, n - 1, l, r, seg::roots[d])
          << endl;
    }
    else
    {
      int d;
```

```
      cin >> d;
      while (seg::roots.size() > d + 1)
        seg::roots.pop_back();
    }
  }
  return 0;
}
// https://www.spoj.com/problems/TTM/
// rollback segtree to a time stamp t
```

10.2 treap

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define PI acos(-1)
#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define DEBUG 0
#define MAXN 101

namespace treap
{
  struct treap // struct
  {
    int data, priority;
    vector<treap *> kids;
    int subtree_size, sum, lazy;
  };
  int size(treap *node) // retorna o tamanho da subtree
      do no
  {
    return (node == NULL) ? 0 : node->subtree_size;
  }
  void recalc(treap *node) // recalculo das informacoes
      do no
  {
    if (node == NULL)
      return;
    node->subtree_size = 1;
    node->sum = (node->data) + (node->lazy * size(node))
        ; // lazy propagation
    for (auto const &i : node->kids)
    {
      if (i == NULL)
        continue;
      node->subtree_size += i->subtree_size;
      node->sum += ((i->sum) + (i->lazy * size(i)));
    }
  }
  void lazy_propagation(treap *node) // para aplicar o
      lazy
  {
    if (node == NULL || !(node->lazy))
      return;
    for (auto const &i : node->kids)
    {
      if (i == NULL)
        continue;
      i->lazy += node->lazy;
    }
    node->data += node->lazy;
    node->lazy = 0;
  }
  vector<treap *> split(treap *node, int n) // n =
      quantidade de elementos na subarvore da esquerda
  {
    if (node == NULL)
      return {NULL, NULL};
    lazy_propagation(node);
    if (size(node->kids[0]) >= n)
    {
      vector<treap *> left = split(node->kids[0], n);
      node->kids[0] = left[1];
      recalc(node);
      return {left[0], node};
    }
    else
    {
      vector<treap *> right = split(node->kids[1], n -
          size(node->kids[0]) - 1);
      node->kids[1] = right[0];
      recalc(node);
      return {node, right[1]};
    }
  }
  treap *merge(treap *l, treap *r) // merge entre duas
      treaps
  {
```

```
    if (l == NULL)
      return r;
    if (r == NULL)
      return l;
    lazy_propagation(l);
    lazy_propagation(r);
    if (l->priority < r->priority)
    {
      l->kids[1] = merge(l->kids[1], r);
      recalc(l);
      return l;
    }
    else
    {
      r->kids[0] = merge(l, r->kids[0]);
      recalc(r);
      return r;
    }
  }
  treap *add(treap *t, int l, int r, int k) // add pro
      lazy propagation
  {
    vector<treap *> a = split(t, l);
    vector<treap *> b = split(a[1], r - l + 1);
    b[0]->lazy += k;
    return merge(a[0], merge(b[0], b[1]));
  }
  treap *create_node(int data, int priority) // criar um
      novo no
  {
    treap *ret = new treap;
    ret->data = data;
    ret->priority = priority;
    ret->kids = {NULL, NULL};
    ret->subtree_size = 1;
    ret->sum = ret->data;
    ret->lazy = 0;
    return ret;
  }
  void print_treap(treap *t) // dfs in treap tree
  {
    if (t == NULL)
      return;
    lazy_propagation(t);
    print_treap(t->kids[0]);
    cout << t->data << " ";
    print_treap(t->kids[1]);
  }
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  srand(time(NULL)); // para as prioridades
  treap::treap *t = NULL;
  int n;
  cin >> n;
  for (int i = 0; i < n; i++)
  {
    int k;
    cin >> k;
    t = treap::merge(t, treap::create_node(k, rand()));
  }
  treap::print_treap(t);
  cout << endl;
  int q;
  cin >> q;
  while (q--)
  {
    int l, r, k; // test lazy propagation
    cin >> l >> r >> k;
    t = treap::add(t, l, r, k);
    treap::print_treap(t);
    cout << endl;
  }
  return 0;
}
```

10.3 mergesorttree

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<pi, int>
#define fir first
#define sec second
#define DEBUG 0
#define MAXN 100001
#define mod 1000000007
```

```
vector<int> seg[4 * MAXN];
int v[MAXN];

void update(int i, int l, int r, int q, int x)
{
  if (l == r)
  {
    seg[i].clear();
    seg[i].pb(x);
    return;
  }
  int mid = (l + r) >> 1;
  if (q <= mid)
    update(i << 1, l, mid, q, x);
  else
    update((i << 1) | 1, mid + 1, r, q, x);
  // a merge do c++ une os dois vectors, deixando ele
     ordenado em O(n)
  seg[i].clear();
  merge(seg[i << 1].begin(), seg[i << 1].end(), seg[(i
     << 1) | 1].begin(), seg[(i << 1) | 1].end(),
     back_inserter(seg[i]));
}
int query(int l, int r, int ql, int qr, int i, int x)
{
  int mid = (l + r) >> 1;
  if (l > r || l > qr || r < ql)
    return 0;
  if (l >= ql && r <= qr) // quantidade de elementos
     maiores do que x no range atual
    return seg[i].end() - upper_bound(seg[i].begin(),
       seg[i].end(), x);
  return query(l, mid, ql, qr, i << 1, x) + query(mid +
     1, r, ql, qr, (i << 1) | 1, x);
}
void build(int l, int r, int i)
{
  if (l == r)
  {
    seg[i].pb(v[l]);
    return;
  }
  int mid = (l + r) >> 1;
  build(l, mid, i << 1);
  build(mid + 1, r, (i << 1) | 1);
  // a merge do c++ une os dois vectors, deixando ele
     ordenado em O(n)
  merge(seg[i << 1].begin(), seg[i << 1].end(), seg[(i
     << 1) | 1].begin(), seg[(i << 1) | 1].end(),
     back_inserter(seg[i]));
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  return 0;
}
// merge sort tree
// a segment tree with ordered vectors in range nodes

// example:
// number of elements > x in a range [l, r]

// memory: O(n * log n)
// query: O(log^2 n)
```

10.4 sparsetable

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
   rb_tree_tag, tree_order_statistics_node_update>;

#define PI acos(-1)
#define pb push_back
#define int long long int
#define pi pair<int, int>
#define pii pair<int, pair<int, pi>>
#define fir first
#define sec second
#define DEBUG 0
#define MAXN 10005
#define mod 1000000007

int n;
vector<int> v;

namespace st
{
  int st[MAXN][25];
  int log[MAXN + 1];

  void init()
  {
    log[1] = 0;
```

```
    for (int i = 2; i <= MAXN; i++)
      log[i] = log[i / 2] + 1;
    for (int i = 0; i < n; i++)
      st[i][0] = v[i];
    for (int j = 1; j <= 25; j++)
      for (int i = 0; i + (1 << j) <= n; i++)
        st[i][j] = min(st[i][j - 1], st[i + (1 << (j -
            1))][j - 1]);
  }
  int query(int l, int r)
  {
    int j = log[r - l + 1];
    int minimum = min(st[l][j], st[r - (1 << j) + 1][j])
        ;
    return minimum;
  }
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
}
```

10.5 sqrt decomposition2

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define PI acos(-1)
#define pb push_back
#define int long long int
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 100001
#define mod 1000000007

int n, q;
vector<int> v;

namespace mo
{
struct query
{
  int idx, l, r, t;
};
struct update
{
  int i, x;
};

int block;
vector<query> queries;
vector<update> updates;
vector<int> ans;

bool cmp(query x, query y)
{
  if (x.l / block != y.l / block)
    return x.l / block < y.l / block;
  if (x.r / block != y.r / block)
    return x.r / block < y.r / block;
  return x.t < y.t;
}
void sqrt_decomposition()
{
  block = 2800; // (2 * n) ^ 0.666
  sort(queries.begin(), queries.end(), cmp);
  ans.resize(queries.size());
  int curr_left = 0, curr_right = 0, curr_sum = 0,
      curr_t = 0;
  for (int i = 0; i < queries.size(); i++)
  {
    int idx = queries[i].idx;
    int l = queries[i].l;
    int r = queries[i].r;
    int t = queries[i].t;
    while (curr_right <= r)
    {
      curr_sum += v[curr_right];
      curr_right++;
    }
    while (curr_left > l)
    {
      curr_left--;
      curr_sum += v[curr_left];
    }
    while (curr_right > r + 1)
    {
      curr_right--;
      curr_sum -= v[curr_right];
    }
```

```
      while (curr_left < l)
      {
        curr_sum -= v[curr_left];
        curr_left++;
      }
      while (curr_t > t)
      {
        curr_t--;
        if (l <= updates[curr_t].i && r >= updates[
            curr_t].i)
          curr_sum -= updates[curr_t].x;
        v[updates[curr_t].i] -= updates[curr_t].x;
      }
      while (curr_t < t)
      {
        if (l <= updates[curr_t].i && r >= updates[
            curr_t].i)
          curr_sum += updates[curr_t].x;
        v[updates[curr_t].i] += updates[curr_t].x;
        curr_t++;
      }
      ans[idx] = curr_sum;
    }
  }
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  cin >> n >> q;
  v.resize(n);
  for (int i = 0; i < n; i++)
    cin >> v[i];
  for (int i = 0; i < q; i++)
  {
    int type;
    cin >> type;
    if (!type)
    {
      mo::update curr;
      cin >> curr.i >> curr.x;
      mo::updates.pb(curr);
    }
    else
    {
      mo::query curr;
      cin >> curr.l >> curr.r;
      curr.r--;
      curr.idx = mo::queries.size();
      curr.t = mo::updates.size();
      mo::queries.pb(curr);
    }
  }
  mo::sqrt_decomposition();
  for (auto const &i : mo::ans)
    cout << i << endl;
}
//https://judge.yosupo.jp/problem/point_add_range_sum
```

10.6 implicit seg

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 200005
#define mod 998244353

struct implcit_seg
{
  int l, r;
  int sum, lazy;
  implcit_seg *left_child = nullptr;
  implcit_seg *right_child = nullptr;

  implcit_seg(int l, int r) : l(l), r(r)
  {
    sum = 0;
    lazy = 0;
  }
  void check_childs()
  {
    if (!left_child && l != r)
    {
      int mid = (l + r) >> 1;
      left_child = new implcit_seg(l, mid);
      right_child = new implcit_seg(mid + 1, r);
```

```
  }
}
void add(int x)
{
  sum += (r - l + 1) * x;
  if (l != r)
  {
    check_childs();
    left_child->lazy += x;
    right_child->lazy += x;
  }
  lazy = 0;
}
void upd(int ql, int qr, int x)
{
  add(lazy);
  if (l > r || l > qr || r < ql)
    return;
  if (l >= ql && r <= qr)
  {
    add(x);
    return;
  }
  check_childs();
  left_child->upd(ql, qr, x);
  right_child->upd(ql, qr, x);
  sum = left_child->sum + right_child->sum;
}
void upd(int k, int x)
{
  sum += x;
  check_childs();
  if (left_child)
  {
    if (k <= left_child->r)
      left_child->upd(k, x);
    else
      right_child->upd(k, x);
  }
}
int qry(int ql, int qr)
{
  add(lazy);
  if (l > r || l > qr || r < ql)
    return 0;
  if (l >= ql && r <= qr)
    return sum;
  check_childs();
  return left_child->qry(ql, qr) + right_child->qry(ql
      , qr);
  }
};
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n, q;
  cin >> n >> q;
  implcit_seg *s = new implcit_seg(0, n - 1);
  while (q--)
  {
    int t;
    cin >> t;
    if (t == 1)
    {
      int l, r, x;
      cin >> l >> r >> x;
      if (l == r - 1) // point update
        s->upd(l, x);
      else // range update
        s->upd(l, r - 1, x);
    }
    else
    {
      int l, r;
      cin >> l >> r;
      cout << s->qry(l, r - 1) << endl; // range sum
    }
  }
  return 0;
}
```

10.7 min queue

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<pi, int>
#define fir first
#define sec second
```

```cpp
#define MAXN 1005
#define mod 998244353

namespace min_queue
{
  deque<pi> q;
  int l, r;

  void init()
  {
    l = r = 1;
    q.clear();
  }
  void push(int v)
  {
    while (!q.empty() && v < q.back().fir)
      q.pop_back();
    q.pb({v, r});
    r++;
  }
  void pop()
  {
    if (!q.empty() && q.front().sec == l)
      q.pop_front();
    l++;
  }
  int getmin()
  {
    return q.front().fir;
  }
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n, m;
  cin >> n >> m;
  vector<int> v(n);
  for (int i = 0; i < n; i++)
    cin >> v[i];
  int l = 0, r = m - 1;
  cout << l << " " << r << endl;
  for (int i = l; i <= r; i++)
    min_queue::push(v[i]);
  cout << min_queue::getmin() << " ";
  l++, r++;
  while (r < n)
  {
    min_queue::pop();
    min_queue::push(v[r]);
    cout << min_queue::getmin() << " ";
    l++, r++;
  }
  cout << endl;
  return 0;
}
// minimum of each subarray of length m (m <= n)
```

10.8 segtree lazy

```cpp
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 200005
#define mod 998244353

struct segtree
{
  int n;
  vector<int> v;
  vector<int> seg;
  vector<int> lazy;

  segtree(int sz)
  {
    n = sz;
    seg.assign(4 * n, 0);
    lazy.assign(4 * n, 0);
    // v = vv; // for build
    // build(0, n - 1, 1); // for build
  }
  int single(int x)
  {
    return x;
  }
  int neutral()
```

```
{
  return 0;
}
int merge(int a, int b)
{
  return a + b;
}
void add(int i, int l, int r, int diff)
{
  seg[i] += (r - l + 1) * diff;
  if (l != r)
  {
    lazy[i << 1] += diff;
    lazy[(i << 1) | 1] += diff;
  }
  lazy[i] = 0;
}
void update(int i, int l, int r, int ql, int qr, int
    diff)
{
  if (lazy[i])
    add(i, l, r, lazy[i]);
  if (l > r || l > qr || r < ql)
    return;
  if (l >= ql && r <= qr)
  {
    add(i, l, r, diff);
    return;
  }
  int mid = (l + r) >> 1;
  update(i << 1, l, mid, ql, qr, diff);
  update((i << 1) | 1, mid + 1, r, ql, qr, diff);
  seg[i] = merge(seg[i << 1], seg[(i << 1) | 1]);
}
int query(int l, int r, int ql, int qr, int i)
{
  if (lazy[i])
    add(i, l, r, lazy[i]);
  if (l > r || l > qr || r < ql)
    return neutral();
  if (l >= ql && r <= qr)
    return seg[i];
  int mid = (l + r) >> 1;
  return merge(query(l, mid, ql, qr, i << 1), query(
      mid + 1, r, ql, qr, (i << 1) | 1));
}
void build(int l, int r, int i)
{
  if (l == r)
  {
    seg[i] = single(v[l]);
    return;
  }
  int mid = (l + r) >> 1;
  build(l, mid, i << 1);
  build(mid + 1, r, (i << 1) | 1);
  seg[i] = merge(seg[i << 1], seg[(i << 1) | 1]);
}
int qry(int l, int r)
{
  return query(0, n - 1, l, r, 1);
}
void upd(int l, int r, int x)
{
  update(1, 0, n - 1, l, r, x);
}
};
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n, q;
  cin >> n >> q;
  segtree s(n);
  while (q--)
  {
    int t;
    cin >> t;
    if (t == 1)
    {
      int l, r, x;
      cin >> l >> r >> x;
      s.upd(l, r, x);
    }
    else
    {
      int l, r;
      cin >> l >> r;
      cout << s.qry(l, r) << endl;
    }
  }
  return 0;
}
```

10.9 fenwick3

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
```

```
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

//#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<pi, int>
#define fir first
#define sec second
#define MAXN 200005
#define mod 1000000007

int v[MAXN];

namespace bit
{
  ordered_set<int> bit[MAXN];

  int query(int r, int a, int b)
  {
    int ret = 0, curr = r;
    for (; r >= 0; r = (r & (r + 1)) - 1)
      ret += (bit[r].order_of_key(b + 1) - bit[r].
          order_of_key(a));
    return ret;
  }
  void add(int idx, int delta)
  {
    for (; idx < MAXN; idx = idx | (idx + 1))
      bit[idx].insert(delta);
  }
  void rem(int idx, int delta)
  {
    for (; idx < MAXN; idx = idx | (idx + 1))
      bit[idx].erase(delta);
  }
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  return 0;
}
// ideia da merge sort tree na bit (fica mais rapido)
// so fazer uma bit de ordered set ou vector(se nao
    tiver update)
// add -> adiciona o numero delta na posicao idx
// rem -> remove o numero delta na posicao idx
// query -> retorna o numero de elementos tal que
    posicao <= r && (a <= num <= b)
```

10.10 treap2

```
// https://codeforces.com/contest/863/problem/D
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define PI acos(-1)
#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define DEBUG 0
#define MAXN 101

vector<int> ans;

namespace treap
{
  struct treap
  {
    int data, priority;
    vector<treap *> kids;
    int subtree_size, sum, lazy;
  };
  int size(treap *node)
  {
    return (node == NULL) ? 0 : node->subtree_size;
  }
  void recalc(treap *node)
  {
    if (node == NULL)
      return;
    node->subtree_size = 1;
    node->sum = (node->data) + (node->lazy * size(node))
        ;
    for (auto const &i : node->kids)
    {
```

```
    if (i == NULL)
      continue;
    node->subtree_size += i->subtree_size;
    node->sum += ((i->sum) + (i->lazy * size(i)));
  }
}
void lazy_propagation(treap *node)
{
  if (node == NULL || !(node->lazy))
    return;
  swap(node->kids[0], node->kids[1]);
  for (auto const &i : node->kids)
  {
    if (i == NULL)
      continue;
    i->lazy ^= 1;
  }
  node->lazy = 0;
}
vector<treap *> split(treap *node, int n)
{
  if (node == NULL)
    return {NULL, NULL};
  lazy_propagation(node);
  if (size(node->kids[0]) >= n)
  {
    vector<treap *> left = split(node->kids[0], n);
    node->kids[0] = left[1];
    recalc(node);
    return {left[0], node};
  }
  else
  {
    vector<treap *> right = split(node->kids[1], n -
        size(node->kids[0]) - 1);
    node->kids[1] = right[0];
    recalc(node);
    return {node, right[1]};
  }
}
treap *merge(treap *l, treap *r)
{
  if (l == NULL)
    return r;
  if (r == NULL)
    return l;
  lazy_propagation(l);
  lazy_propagation(r);
  if (l->priority < r->priority)
  {
    l->kids[1] = merge(l->kids[1], r);
    recalc(l);
    return l;
  }
  else
  {
    r->kids[0] = merge(l, r->kids[0]);
    recalc(r);
    return r;
  }
}
treap *create_node(int data, int priority)
{
  treap *ret = new treap;
  ret->data = data;
  ret->priority = priority;
  ret->kids = {NULL, NULL};
  ret->subtree_size = 1;
  ret->sum = ret->data;
  ret->lazy = 0;
  return ret;
}
void dfs(treap *t)
{
  if (t == NULL)
    return;
  lazy_propagation(t);
  dfs(t->kids[0]);
  ans.pb(t->data);
  dfs(t->kids[1]);
}
treap *shift(treap *t, int l, int r)
{
  vector<treap *> a = split(t, l);
  vector<treap *> b = split(a[1], r - l + 1);
  vector<treap *> c = split(b[0], r - l);
  return merge(merge(a[0], c[1]), merge(c[0], b[1]));
}
treap *reverse(treap *t, int l, int r)
{
  vector<treap *> a = split(t, l);
  vector<treap *> b = split(a[1], r - l + 1);
  b[0]->lazy ^= 1;
  return merge(a[0], merge(b[0], b[1]));
}
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
```

```
  srand(time(NULL));
  treap::treap *t = NULL;
  int n, m, q;
  cin >> n >> q >> m;
  for (int i = 0; i < n; i++)
  {
    int k;
    cin >> k;
    t = treap::merge(t, treap::create_node(k, rand()));
  }
  while (q--)
  {
    int ty, l, r;
    cin >> ty >> l >> r;
    l--, r--;
    (ty == 1) ? t = treap::shift(t, l, r) : t = treap::
       reverse(t, l, r);
  }
  treap::dfs(t);
  while (m--)
  {
    int i;
    cin >> i;
    i--;
    cout << ans[i] << " ";
  }
  cout << endl;
  return 0;
}
```

10.11 SegTree

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

#define PI acos(-1)
#define pb push_back
#define int long long int
#define mp make_pair
#define pi pair<int, int>
#define pii pair<pi, int>
#define fir first
#define sec second
#define MAXN 100001
#define MAXL 100
#define mod 1000000007

vector<int> seg;
vector<int> v;

int single(int x)
{
  return x;
}
int neutral()
{
  return 0;
}
int merge(int a, int b)
{
  return a + b;
}
void update(int i, int l, int r, int q, int x)
{
  if (l == r)
  {
    seg[i] = single(x);
    return;
  }
  int mid = (l + r) >> 1;
  if (q <= mid)
    update(i << 1, l, mid, q, x);
  else
    update((i << 1) | 1, mid + 1, r, q, x);
  seg[i] = merge(seg[i << 1], seg[(i << 1) | 1]);
}
int query(int l, int r, int ql, int qr, int i)
{
  int mid = (l + r) >> 1;
  if (l > r || l > qr || r < ql)
    return neutral();
  if (l >= ql && r <= qr)
    return seg[i];
  return merge(query(l, mid, ql, qr, i << 1), query(mid
     + 1, r, ql, qr, (i << 1) | 1));
}
void build(int l, int r, int i)
{
  if (l == r)
  {
    seg[i] = single(v[l]);
    return;
  }
  int mid = (l + r) >> 1;
  build(l, mid, i << 1);
  build(mid + 1, r, (i << 1) | 1);
  seg[i] = merge(seg[i << 1], seg[(i << 1) | 1]);
}
signed main()
{
```

```
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n, q;
  cin >> n >> q;
  v.resize(n);
  seg.resize(4 * n);
  for (int i = 0; i < n; i++)
    cin >> v[i];
  build(0, n - 1, 1);
  while (q--)
  {
    int l, r;
    int t;
    cin >> t >> l >> r;
    if (t == 2)
      cout << query(0, n - 1, l, r - 1, 1) << endl;
    else
      update(1, 0, n - 1, l, r);
  }
}
```

10.12 Segtree2

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 500001
#define mod 1000000007

struct segtree
{
  int n;
  vector<int> seg;

  int neutral()
  {
    return 0;
  }
  int merge(int a, int b)
  {
    return a + b;
  }
  void build(vector<int> &v)
  {
    n = 1;
    while (n < v.size())
      n <<= 1;
    seg.assign(n << 1, neutral());
    for (int i = 0; i < v.size(); i++)
      seg[i + n] = v[i];
    for (int i = n - 1; i; i--)
      seg[i] = merge(seg[i << 1], seg[(i << 1) | 1]);
  }
  void upd(int i, int value)
  {
    seg[i += n] += value;
    for (i >>= 1; i; i >>= 1)
      seg[i] = merge(seg[i << 1], seg[(i << 1) | 1]);
  }
  int qry(int l, int r)
  {
    int ansl = neutral(), ansr = neutral();
    for (l += n, r += n + 1; l < r; l >>= 1, r >>= 1)
    {
      if (l & 1)
        ansl = merge(ansl, seg[l++]);
      if (r & 1)
        ansr = merge(seg[--r], ansr);
    }
    return merge(ansl, ansr);
  }
};
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  return 0;
}
// iterative segtree without lazy propagation
```

10.13 sqrt decomposition

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;
```

```
template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define PI acos(-1)
#define pb push_back
#define int long long int
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 100001
#define mod 1000000007

int n, q;
vector<int> v;

namespace mo
{
  struct query
  {
    int idx, l, r;
  };

  int block;
  vector<query> queries;
  vector<int> ans;

  bool cmp(query x, query y)
  {
    if (x.l / block != y.l / block)
      return x.l / block < y.l / block;
    if (x.r != y.r)
      return x.r < y.r;
  }
  void sqrt_decomposition()
  {
    block = (int)sqrt(n);
    sort(queries.begin(), queries.end(), cmp);
    ans.resize(queries.size());
    int curr_left = 0, curr_right = 0, curr_sum = 0;
    for (int i = 0; i < queries.size(); i++)
    {
      int idx = queries[i].idx;
      int l = queries[i].l;
      int r = queries[i].r;
      while (curr_left < l)
      {
        curr_sum -= v[curr_left];
        curr_left++;
      }
      while (curr_left > l)
      {
        curr_left--;
        curr_sum += v[curr_left];
      }
      while (curr_right <= r)
      {
        curr_sum += v[curr_right];
        curr_right++;
      }
      while (curr_right > r + 1)
      {
        curr_right--;
        curr_sum -= v[curr_right];
      }
      ans[idx] = curr_sum;
    }
  }
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  cin >> n >> q;
  v.resize(n);
  for (int i = 0; i < n; i++)
    cin >> v[i];
  for (int i = 0; i < q; i++)
  {
    mo::query curr;
    cin >> curr.l >> curr.r;
    curr.r--;
    curr.idx = i;
    mo::queries.pb(curr);
  }
  mo::sqrt_decomposition();
  for (auto const &i : mo::ans)
    cout << i << endl;
}
// to test: https://judge.yosupo.jp/problem/
   static_range_sum
```

10.14 color update

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
```

```
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 500001
#define mod 1000000007

struct color_upd
{
#define left fir
#define right sec.fir
#define color sec.sec
  set<pii> ranges;
  vector<pii> erased;

  color_upd(int n) // inicialmente, todo mundo pintado
      com a cor -1
  {
    ranges.insert({0, {n - 1, -1}});
  }
  int get(int i) // qual a cor do elemento na posicao i
  {
    auto it = ranges.upper_bound({i, {1e18, 1e18}});
    if (it == ranges.begin())
      return -1;
    it--;
    return (*(it)).color;
  }
  void del(int l, int r) // apaga o intervalo [l, r]
  {
    erased.clear();
    auto it = ranges.upper_bound({l, {0, 0}});
    if (it != ranges.begin())
    {
      it--;
    }
    while (it != ranges.end())
    {
      if ((*(it)).left > r)
        break;
      else if ((*(it)).right >= l)
        erased.push_back(*it);
      it++;
    }
    if (erased.size() > 0)
    {
      int sz = erased.size();
      auto it = ranges.lower_bound({erased[0].left, {0,
          0}});
      auto it2 = ranges.lower_bound({erased[sz - 1].left
          , {0, 0}});
      pii ini = *it, fim = *it2;
      it2++;
      ranges.erase(it, it2);
      pii upd1 = {ini.left, {l - 1, ini.color}};
      pii upd2 = {r + 1, {fim.right, fim.color}};
      erased[0].left = max(erased[0].left, l);
      erased[sz - 1].right = min(erased[sz - 1].right, r
          );
      if (upd1.left <= upd1.right)
        ranges.insert(upd1);
      if (upd2.left <= upd2.right)
        ranges.insert(upd2);
    }
  }
  void upd(int a, int b, int c) // pinta o intervalo [a,
      b] com a cor c
  {
    del(a, b);
    ranges.insert({a, {b, c}});
  }
};
struct segtree
{
  vector<int> seg;
  vector<int> lazy;

  segtree(int n)
  {
    seg.resize(4 * n, 0);
    lazy.assign(4 * n, 0);
  }
  int single(int x)
  {
    return x;
  }
  int neutral()
  {
    return 0;
  }
  int merge(int a, int b)
```

```
  {
    return a + b;
  }
  void add(int i, int l, int r, int diff)
  {
    seg[i] += (r - l + 1) * diff;
    if (l != r)
    {
      lazy[i << 1] += diff;
      lazy[(i << 1) | 1] += diff;
    }
    lazy[i] = 0;
  }
  void update(int i, int l, int r, int ql, int qr, int
      diff)
  {
    if (lazy[i])
      add(i, l, r, lazy[i]);
    if (l > r || l > qr || r < ql)
      return;
    if (l >= ql && r <= qr)
    {
      add(i, l, r, diff);
      return;
    }
    int mid = (l + r) >> 1;
    update(i << 1, l, mid, ql, qr, diff);
    update((i << 1) | 1, mid + 1, r, ql, qr, diff);
    seg[i] = merge(seg[i << 1], seg[(i << 1) | 1]);
  }
  int query(int l, int r, int ql, int qr, int i)
  {
    if (lazy[i])
      add(i, l, r, lazy[i]);
    if (l > r || l > qr || r < ql)
      return neutral();
    if (l >= ql && r <= qr)
      return seg[i];
    int mid = (l + r) >> 1;
    return merge(query(l, mid, ql, qr, i << 1), query(
        mid + 1, r, ql, qr, (i << 1) | 1));
  }
};
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n, q;
  cin >> n >> q;
  color_upd c = color_upd(n);
  segtree s = segtree(n);
  for (int i = 0; i < n; i++)
    c.upd(i, i, i + 1);
  while (q--)
  {
    int t;
    cin >> t;
    if (t == 1)
    {
      int l, r, x;
      cin >> l >> r >> x;
      l--, r--;
      c.upd(l, r, x);
      for (auto const &i : c.erased)
        s.update(1, 0, n - 1, i.left, i.right, abs(x - i
            .color));
    }
    else
    {
      int l, r;
      cin >> l >> r;
      l--, r--;
      cout << s.query(0, n - 1, l, r, 1) << endl;
    }
  }
  return 0;
}
// https://codeforces.com/contest/444/problem/C
```

10.15 bit2d

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 1000001
#define mod 1000000007
```

```
// source: https://github.com/tfg50/Competitive-
    Programming/blob/master/Biblioteca/Data%20Structures
    /Bit2D.cpp
struct bit2d
{
  vector<int> ord;
  vector<vector<int>> t;
  vector<vector<int>> coord;

  bit2d(vector<pi> &pts) // recebe todos os pontos que
      vao ser inseridos pra construir, mas nao insere
      eles
  {
    sort(pts.begin(), pts.end());
    for (auto const &a : pts)
    {
      if (ord.empty() || a.fir != ord.back())
        ord.pb(a.fir);
    }
    t.resize(ord.size() + 1);
    coord.resize(t.size());
    for (auto &a : pts)
    {
      swap(a.fir, a.sec);
    }
    sort(pts.begin(), pts.end());
    for (auto &a : pts)
    {
      swap(a.fir, a.sec);
      for (int on = upper_bound(ord.begin(), ord.end(),
          a.fir) - ord.begin(); on < t.size(); on += on
          & -on)
      {
        if (coord[on].empty() || coord[on].back() != a.
            sec)
          coord[on].push_back(a.sec);
      }
    }
    for (int i = 0; i < t.size(); i++)
      t[i].assign(coord[i].size() + 1, 0);
  }
  void add(int x, int y, int v) // v[a][b] += v
  {
    for (int xx = upper_bound(ord.begin(), ord.end(), x)
        - ord.begin(); xx < t.size(); xx += xx & -xx)
    {
      for (int yy = upper_bound(coord[xx].begin(), coord
          [xx].end(), y) - coord[xx].begin(); yy < t[xx
          ].size(); yy += yy & -yy)
        t[xx][yy] += v;
    }
  }
  int qry(int x, int y) // soma de todos os v[a][b] com
      (a <= x && b <= y)
  {
    int ans = 0;
    for (int xx = upper_bound(ord.begin(), ord.end(), x)
        - ord.begin(); xx > 0; xx -= xx & -xx)
    {
      for (int yy = upper_bound(coord[xx].begin(), coord
          [xx].end(), y) - coord[xx].begin(); yy > 0; yy
          -= yy & -yy)
        ans += t[xx][yy];
    }
    return ans;
  }
  int qry2(int x1, int y1, int x2, int y2)
  {
    return qry(x2, y2) - qry(x2, y1 - 1) - qry(x1 - 1,
        y2) + qry(x1 - 1, y1 - 1);
  }
  void add2(int x1, int y1, int x2, int y2, int v)
  {
    add(x1, y1, v);
    add(x1, y2 + 1, -v);
    add(x2 + 1, y1, -v);
    add(x2 + 1, y2 + 1, v);
  }
};
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  return 0;
}
```

10.16 fenwick2

```
// fenwick com update pro range [l, r]
// complexidade O(q * log(n)) com a criacao de duas bits
    ao inves de uma
#include <bits/stdc++.h>
using namespace std;

#define PI acos(-1)
#define int long long int
#define pb push_back
#define mp make_pair
#define pi pair<string, int>
```

```
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 100001
#define MAXL 20
#define mod 998244353

int n;
vector<int> bit, bit2;

void add1(int idx, int delta)
{
  for (; idx < n; idx = idx | (idx + 1))
    bit[idx] += delta;
}
void add2(int idx, int delta)
{
  for (; idx < n; idx = idx | (idx + 1))
    bit2[idx] += delta;
}
void update_range(int val, int l, int r)
{
  add1(l, val);
  add1(r + 1, -val);
  add2(l, val * (l - 1));
  add2(r + 1, -val * r);
}
int sum1(int r)
{
  int ret = 0;
  for (; r >= 0; r = (r & (r + 1)) - 1)
    ret += bit[r];
  return ret;
}
int sum2(int r)
{
  int ret = 0;
  for (; r >= 0; r = (r & (r + 1)) - 1)
    ret += bit2[r];
  return ret;
}
int sum(int x)
{
  return (sum1(x) * x) - sum2(x);
}
int range_sum(int l, int r)
{
  return sum(r) - sum(l - 1);
}
int main()
{
  bit.assign(MAXN, 0); // inicializar sempre
  bit2.assign(MAXN, 0); // inicializar sempre
  update_range(x, l, r); // pra cada elemento em [l, r]
      += x
  range_sum(l, r);  // soma de [l, r]
}
```

10.17 fenwick

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

#define PI acos(-1)
#define int long long int
#define pb push_back
#define mp make_pair
#define pi pair<int, int>
#define fir first
#define sec second
#define MAXN 501
#define MAXL 20
#define mod 998244353

int n;
vector<int> bit;
int sum(int r)
{
  int ret = 0;
  for (; r >= 0; r = (r & (r + 1)) - 1)
    ret += bit[r];
  return ret;
}
void add(int idx, int delta)
{
  for (; idx < n; idx = idx | (idx + 1))
    bit[idx] += delta;
}
signed main()
{
  cin >> n;
  vector<int> v(n);
  bit.assign(n, 0);
  for (int i = 0; i < n; i++)
    cin >> v[i], add(i, v[i]);
  int q;
  cin >> q;
  while (q--)
  {
    char t;
```

```
cin >> t;
if (t == 'Q') // query
{
  int l, r;
  cin >> l >> r;
  cout << (sum(r) - sum(l - 1)) << endl;
}
else // update
{
  int a, b;
  cin >> a >> b;
  add(a, b - v[a]);
 }
}
return 0;
}
```