Contents

1 STL

1.1 Some Data Structures of STL

```
1)Vector

vector <int> v; //Cria  o 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    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 n o 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 vari vel do tipo map que mapeia strings em int
// Em um map cada elemento  est  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 fa amos uma busca pela chave "Alana" receberemos o n mero 10 como retorno.

m.insert(make_pair("Alana", 10)); //Inserimos uma vari vel do tipo pair diretamente no map, O(log n)
M["Alana"] = 10; //Relacionando o valor 10    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++    importante apenas saber que o maior elemento sempre
      estar  na primeiro posi  o.
// Com exe  o disso, todos os outros m todos  s o semelhantes ao uso de uma queue comum, por m
      para manter a estrutura organizada, a complexidade da opera  o de inser  o    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
```

1.2 Ordered Set from GNU PBDS

```
#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  s  o menores do que 4
  cout << s.order_of_key(6) << endl;     // quantidade de elementos que  s  o menores do que 4
}
// find_by_order : O(log n), retorna (um iterator) qual o k- simo elemento do set
// order_of_key: O(log n), retorna qual a quantidade de elementos menores do que x no set
```

2 Binary Search and Ternary Search

2.1 Binary Search

```
#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 ;
}
```

2.2 Upper Bound

```
#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];
}
```

2.3 Lower Bound

```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<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];
}
```

2.4 STL Lower Bound and Upper Bound

```cpp
// 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 ;
}
```

2.5 Ternary Search

```cpp
// 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;
}
```

2.6 Some Aplications

```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 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);
```

3 Dynamic Programming

3.1 Knapsack

```
//O problema mais cl ssico de Programa o Din mica talvez seja o Knapsack.
//De maneira geral, um ladr o ir  roubar uma casa com uma mochila
//que suporta um peso s. Ele v  n objetos na casa e sabe estimar o peso pi        e  o valor vi

//de cada objeto i. Com essas informa es, qual o maior valor que o ladr o 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;
}
```

3.2 Coin Change DP

```
// dados os valores de moedas v1, v2, ... vn    possivel formar um valor m como combina o de moedas
// para isso basta montar uma dp inicializada com -1
// nesse caso a dp s  precisa de um parametro q    = valor restante ate o limite
// mas podem existir varia es do problema q precise de mais coisas
// se em achar alguma combina o v lida retorna 1, se n o retorna 0
#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;
}
```

3.3 Longest Common Subsequence

```
//Dadas duas sequ ncias s1 e s2, uma de tamanho n e outra de tamanho m, qual a maior subsequ ncia
      comum  s  duas?

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

```cpp
#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 sequ ncias 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;
}
```

3.4 Longest Increasing Subsequence

```cpp
// dada uma sequ ncia s qualquer, descobrir o tamanho da maior subsequ ncia crescente de s
// uma subsequ ncia de s    qualquer subconjunto de elementos de s.
// Para cada novo n mero, voc  tem duas opera es poss veis:
// 1 - Colocar o novo n mero no topo de uma pilha se ele n o superar o que j  est  em seu topo;
// ou
// 2 - Criar uma nova pilha    direita de todas as outras e colocar o novo n mero 1 .
// ao final do processo a nossa pilha ter  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;
}
```

3.5 Kadane

```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 op  es
// n o usar o elemento v[i]
// ou
// usamos, adicionando a maior soma poss vel que antes dele
#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

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

3.6 Max Matrix Path

```cpp
#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;
}
```

3.7 Largest Sub-Matrix 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;
}
```

3.8 Subsequences Matching

```
#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;
}
```

3.9 Digit DP

```
#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;
}
```

3.10 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;
```

```cpp
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;
}
```

3.11 Broken Profile DP

```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 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]
```

4 Common Problems

4.1 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
*/
```

4.2 Two Pointers Method

```
#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.
```

4.3 Inversion Count

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

// Solucao O(n ) 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(n ), 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 três 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;
}
```

4.4 Meet In The Middle

```
#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;
}
```

5 Graph and Trees

5.1 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);
}
```

5.2 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;
}
```

5.3 Bipartite Graph

```
// Grafo Bipartido
// 1 - nao possui ciclo de tamanho impar
// 2 - podemos colorir todos os vertices usando apenas duas cores de maneira que uma aresta nunca
      ligue dois v rtices da mesma cor
// 3 - Se quisermos checar se um grafo eh bipartido ou nao, simplesmente checamos se ele pode ser
      colorido usando duas cores.

#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 color [MAXN] ;

void colore (int x)
{
    color[x] = 0 ;

    vector <int> f ;
    f.pb(x) ;

    int pos = 0 ;

    while (pos < f.size())
    {
        int at = f[pos] ;
        pos++ ;

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

            if (color[v] == -1)
            {
                color[v] = 1 - color[at] ;
                f.pb(v) ;
            }
        }
    }
}
bool is_bipartido ()
{
    for (int i = 0 ; i < n ; i++)
    {
        if (color[i] == -1)
        {
            colore(i) ;
        }
    }

    for (int i = 0 ; i < n ; i++)
    {
        for (int j = 0 ; j < adj[i].size() ; j++)
        {
            if (color[i] == color[adj[i][j]])
            {
                return false ;
            }
        }
    }

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

    cin >> n >> m ;

    memset(color , -1 , sizeof(color)) ;

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

    (is_bipartido()) ? cout << "YES\n" : cout << "NO\n" ;

    return 0 ;
}
```

5.4 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);
}
```

5.5 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;
}
```

5.6 Kruskal

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

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

// 2 - Cada aresta liga dois v rtices x e y, checar se eles j  est o na mesma componente conexa
// (aqui, consideramos apenas as arestas j  colocadas na  rvore).

// 3 - Se x e y est o na mesma componente, ignoramos a aresta e continuamos o procedimento
// (se a us ssemos, formaramos um ciclo). Se estiverem em componentes distintas, colocamos a aresta
//na  rvore 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 op ao 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;
}
```

5.7 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 v rtice u, que ainda  n o  foi processado, que possua a menor das
      dist ncias.
// 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;
}
```

5.8 DSU

```
// union u v - une dois sets que contem u e v
// find v - acha o set que v pertence, e ve qual o maior e o menor elemento desse set
#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 parent[MAXN];
int sz[MAXN];
int maxx[MAXN];
int minn[MAXN];

int Find(int i)
{
  return parent[i] = (parent[i] == i) ? i : Find(parent[i]);
}
```

```
void Union(int x, int y)
{
  int xx = Find(x), yy = Find(y);
  if (xx != yy)
  {
    if (sz[xx] > sz[yy])
      swap(xx, yy);
    parent[xx] = yy;
    sz[yy] += sz[xx];
    minn[yy] = min(minn[xx], minn[yy]);
    maxx[yy] = max(maxx[xx], maxx[yy]);
  }
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n, q;
  cin >> n >> q;
  for (int i = 0; i < n; i++)
  {
    parent[i] = i;
    sz[i] = 1;
    maxx[i] = i;
    minn[i] = i;
  }
  while (q--)
  {
    string t;
    cin >> t;
    if (t == "union")
    {
      int a, b;
      cin >> a >> b;
      a--, b--;
      Union(a, b);
    }
    else
    {
      int a;
      cin >> a;
      a--;
      cout << minn[Find(a)] + 1 << " " << maxx[Find(a)] + 1 << " " << sz[Find(a)] << endl;
    }
  }
}
```

5.9 Euler Path

```
// caminho euleriano em um grafo
// passa por todas as arestas apenas uma unica vez e percorre todas elas
// condi  o de existencia:
// todos os v rtices possuem grau par (ciclo euleriano) come a e acaba no mesmo v rtice
// ou
// apenas 2 v rtices possuem grau impar, todos os outros possuem grau par ou == 0.
// come a num vertice de grau impar e termina num v rtice de grau impar nesse caso.
// solu  o:
// 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 tamb m    um caminho v lido

#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;
}
```

5.10 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 ;
}
```

5.11 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;
}
```

5.12 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 ;
}
```

5.13 Dinic

```
#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 502
#define mod 1000000007
#define INF 1e9

struct edge
{
  int to, from, flow, capacity, id;
};

int n, m, a, b, source, destiny;
vector<edge> adj[MAXN];
queue<int> q;
int level[MAXN];
int ptr[MAXN];

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()
{
  memset(level, -1, sizeof(level));
  level[source] = 0;
  q.push(source);
  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[destiny] != -1;
}
int dfs(int u, int flow)
{
  if (u == destiny || 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 dinic()
{
  int max_flow = 0;
  while (bfs())
  {
    memset(ptr, 0, sizeof(ptr));
    while (1)
    {
      int flow = dfs(source, INF);
      if (flow == 0)
        break;
      max_flow += flow;
    }
  }
  return max_flow;
}
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--;
    add_edge(a, b, c, i);
  }
  source = 0, destiny = n - 1;
  cout << dinic() << endl;
  vector<int> ans(m);
  for (int i = 0; i < n; i++) // fluxo em cada aresta, na ordem da entrada
    for (auto const &j : adj[i])
      if (!j.capacity)
        ans[j.id] = j.flow;
  for (auto const &i : ans)
    cout << i << endl;
  return 0;
}
```

5.14 Min Cost Flow

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

#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 true
#define MAXN 301
#define INF 1e9

int n, source, destiny;
vector<int> adj[MAXN];
int capacity[MAXN][MAXN];
int cost[MAXN][MAXN];
```

```
vector<int> dist;
vector<int> parent;
vector<bool> in_queue;

void add_edge(int a, int b, int c, int d)
{
  adj[a].pb(b); // aresta normal
  capacity[a][b] = c;
  cost[a][b] = d;
  adj[b].pb(a); // aresta do grafo residual
  capacity[b][a] = 0;
  cost[b][a] = -d;
}
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);
  in_queue.assign(MAXN, false);
  dist[s] = 0;
  queue<int> q;
  q.push(s);
  while (!q.empty())
  {
    int u = q.front();
    q.pop();
    in_queue[u] = false;
    for (auto const &v : adj[u])
    {
      if (capacity[u][v] && dist[v] > dist[u] + cost[u][v])
      {
        dist[v] = dist[u] + cost[u][v];
        parent[v] = u;
        if (!in_queue[v])
        {
          in_queue[v] = true;
          q.push(v);
        }
      }
    }
  }
  return dist[destiny] != INF; // se eu cheguei em destiny por esse caminho, ainda posso passar fluxo
}
int min_cost_flow()
{
  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"
    {
      curr_flow = min(curr_flow, capacity[parent[curr]][curr]);
      curr = parent[curr];
    }
    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
    {
      capacity[parent[curr]][curr] -= curr_flow;
      capacity[curr][parent[curr]] += curr_flow;
      curr = parent[curr];
    }
  }
  return cost; // ao final temos a resposta :)
}
signed main()
{
  int n;
  cin >> n;
  int v[n][n];
  source = 0, destiny = (2 * n) + 1;
  for (int i = 0; i < n; i++)
  {
    for (int j = 0; j < n; j++)
    {
      cin >> v[i][j];
      add_edge(i + 1, j + n + 1, 1, v[i][j]);
    }
  }
  for (int i = 1; i <= n; i++)
    add_edge(source, i, 1, 0);
  for (int i = n + 1; i <= n + n; i++)
    add_edge(i, destiny, 1, 0);
  cout << min_cost_flow() << endl;
}
```

5.15 Euler Tour

```
#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
```

5.16 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;
```

```cpp
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;
}
```

5.17 Rerooting Technique

```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 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]
```

5.18 Diameter Of A Tree

```cpp
#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;
}
```

5.19 Centroid Decomposition

```cpp
// centroid de uma  rvore ->    um n  que ao ser removido da  rvore, separaria as
//  rvores resultantes de modo com que a maior  rvore desse conjunto teria no m ximo
// (n / 2) n s, sendo n o n mero de n s da  rvore. Para qualquer  rvore com n n s,
// o centroid sempre existe.

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

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

// algortimo:
// 1) o centroid    a raiz dessa nova  rvore
// 2) achar o centroid das  rvores menores que surgiram com a remo  o 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;
}
```

5.20 HLD Vertex

```cpp
//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;
}
```

5.21 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;
}
```

6 Math

6.1 Divisors Of a Number

```
#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;
}
```

6.2 Sieve

```
#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;
}
```

6.3 Prime Factors of a Number

```
#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;
}
```

6.4 Prime Factors of a Number Using Sieve

```
#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;
}
```

6.5 Segmented Sieve

```
#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;
}
```

6.6 Modular Arithmetic

```
#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 <double, double>
#define pii pair<int, pi>
#define mp make_pair
#define fir first
#define sec second
#define MAXN 100001
#define mod 1000000007

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 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;
}
signed main()
{
  return 0;
}
```

6.7 Matrix Exponentiation

```
// https://codeforces.com/gym/102644/problem/C
// achar o n-simo termo da sequencia de fibonacci mod (10^9 + 7) em O(log(n))
// n <= 10^18
// podemos escrever a recorrencia de fibonnaci como uma exponencia o de matriz
/*
  (  fib(n)  )      (1 1) ^ (n - 1)      (fib(1) = 1)
  (fib(n - 1))   =  (1 0)             *  (fib(0) = 1)
*/
//    possivel fazer essa exponencia o em O(log(n)) com um algoritimo muito similar ao de
     exponencia o rapida
// dai calculamos o n-simo 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;
}
```

6.8 Matrix Exponentiation Trick

```
// 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;
}
```

6.9 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>;

#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;
}
// elimina o gaussiana
// para resolver sistemas com n equa es e m incognitas

// para isso iremos utilizar uma representa o usando
// matrizes, no qual uma coluna extra    adicionada,
// representando os resultados de cada equa o.

// algoritimo:
// ideia: qualquer equa o pode ser reescrita como uma combina o linear dela mesma
// 1- dividir a primeira linha(primeira equa o) 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
//    ser o 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 ser  O(n^3)
```

6.10 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 multiplica o em O(N^2)
// por m podemos melhorar
// vamos la
// 1  - todo polinomio de grau d que    representado na forma de coeficientes
//      de coeficientes possui uma representa o em forma de d - 1 pontos
// 2  - para esse conjunto de pontos, s  existe um  nico polinomio equivalente
// 3  - DFT -> transforma o da representa o de coeficientes para represnta o
//              de pontos
// 4  - com isso, para multiplicar os dois polinomios agora basta multiplicar
//      os conjuntos de pontos e com isso obtemos a representa o usando pontos
//      do polinomio resultante
// 5  - DFT(A * B) = DFT(A) * DFT(B);
// 6  - por m agora precisamos transformar a resposta obtida na multiplica o dos pontos
//      para a representa o em que usa os coeficientes
// 7  - inverseDFT -> transforma o da representa o de pontos para represnta o
//                   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    chamada de matriz de vandermonde
//      e em geral, podemos escrever a resposta como uma multiplica o de
//      matrizes
// 12 - essa multplica o 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
```

7 Data Structures

7.1 Fenwick Tree

```
#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;
}
```

7.2 Fenwick Tree With Range Update

```
// fenwick com update pro range [l, r]
// complexidade O(q * log(n)) com a cria  o 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>
```

```cpp
#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]
}
```

7.3 Fenwick Tree 2D

```cpp
#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 101
#define MAXL 20
#define mod 998244353

int n, m;
int bit[MAXN][MAXN];
int grid[MAXN][MAXN];

int sum(int x, int y)
{
  int ret = 0;
  for (int i = x; i >= 0; i = (i & (i + 1)) - 1)
    for (int j = y; j >= 0; j = (j & (j + 1)) - 1)
      ret += bit[i][j];
  return ret;
}
void add(int x, int y, int delta)
{
  for (int i = x; i < n; i = i | (i + 1))
    for (int j = y; j < m; j = j | (j + 1))
      bit[i][j] += delta;
}
signed main()
{
  cin >> n >> m;
  for (int i = 0; i < n; i++)
    for (int j = 0; j < m; j++)
      cin >> grid[i][j], add(i, j, grid[i][j]);
  int q;
  cin >> q;
  while (q--)
  {
    char t;
    cin >> t;
    if (t == 'Q') // query
    {
      int a, b;
      cin >> a >> b;
      cout << sum(a, b) << endl;
      // soma de todas as posicoes (x,y) tal que, (0 <= x <= a) e (0 <= y <= b)
    }
    else // update
    {
      int a, b, c;
      cin >> a >> b >> c;
      add(a, b, c - grid[a][b]);
    }
  }
  return 0;
}
```

7.4 Segment Tree

```cpp
#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);
  }
}
```

7.5 Minimum and Frequency With Segment Tree

```
#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<pi> seg;
vector<int> v;

pi single(int x)
{
  return {x, 1};
}
pi neutral()
{
  return {INT_MAX, 0};
}
pi merge(pi a, pi b)
{
  if (a.fir < b.fir)
    return a;
  if (a.fir > b.fir)
    return b;
  return {a.fir, a.sec + b.sec};
}
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]);
}
pi 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)
    {
      pi ans = query(0, n - 1, l, r - 1, 1);
      cout << ans.fir << " " << ans.sec << endl;
    }
    else
      update(1, 0, n - 1, l, r);
  }
}
```

7.6 Segment Tree With Lazy Propagation

```
#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 20
#define mod 1000000009

vector<int> seg(4 * MAXN);
vector<int> lazy(4 * MAXN);
vector<int> v(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] = 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]);
}
signed main()
{
  int n, q;
  cin >> n >> q;
  build(0, n - 1, 1);
  while (q--)
  {
    int t;
    cin >> t;
    if (t == 2)
    {
      int l;
      cin >> l;
      cout << query(0, n - 1, l, l, 1) << endl;
    }
    else
    {
      int l, r, v;
      cin >> l >> r >> v;
      update(1, 0, n - 1, l, r - 1, v);
    }
  }
}
```

7.7 Sparse Table

```
#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);
}
```

7.8 Mos Algorithm

```
#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
```

7.9 Mos Algorithm With Element 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 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
```

7.10 Treap

```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<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;
}
```

7.11 Treap with Cyclic Shift and Reverse Operation

```cpp
// 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>
```

```cpp
#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;
}
```

8 Strings

8.1 String Hashing

```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<pi, int>
#define fir first
#define sec second
#define DEBUG 0
#define MAXN 5001
#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...
```

8.2 String Hashing Without Division

```
#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 300001
#define mod 1000000007

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;
}

namespace sh
{
  const int d = 227;
  vector<int> hashes(MAXN);
  vector<int> pref(MAXN);
  vector<int> pot(MAXN);
  vector<int> m[MAXN];

  int get_hash(string s)
  {
    int ans = 0;
    for (int i = 0; i < s.size(); i++)
    {
      int val = multiplicate(ans, d);
      ans = sum(s[i], val);
    }
    return ans;
  }
  void prefix_hash(string s)
  {
    pref[0] = 0;
    for (int i = 0; i < s.size(); i++)
    {
      int val = multiplicate(pref[i], d);
      pref[i + 1] = sum(s[i], val);
    }
  }
  int get_substring(int l, int r)
  {
    return subtract(pref[r + 1], multiplicate(pref[l], pot[r - l + 1]));
  }
  void calc()
  {
    pot[0] = 1;
```

```
    for (int i = 1; i < MAXN; i++)
      pot[i] = multiplicate(pot[i - 1], d);
  }
}
```

8.3 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|)
```

8.4 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

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

// para cada posi o (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 posi o central desses palindromos

// algoritimo mais facil:
// para cada posi o (0 <= i < s.size()), ele tenta aumentar a resposta em 1
// at q n o seja mais possvel
// while(s[i - curr] == s[i + curr])
// complexidade O(N^2)

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

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

8.5 Aho Corasick

```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<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
```

8.6 Suffix Array

```cpp
#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 pci pair<char, int>
#define fir first
#define sec second
#define MAXN 100001
#define MAXL 20
#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] = mp(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] = mp(mp(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()
{
  string s;
  cin >> s;
  get_suf(s);
  return 0;
}
```

8.7 Suffix Array With Radix Sort

```
#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 pci pair<char, int>
#define fir first
#define sec second
#define MAXN 100001
#define MAXL 20
#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] = mp(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] = mp(mp(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()
{
  string s;
  cin >> s;
  get_suf(s);
  return 0;
}
```

8.8 LCP in Suffix Array

```
#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 pci pair<char, int>
#define fir first
#define sec second
#define MAXN 100001
#define MAXL 20
#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] = mp(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] = mp(mp(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()
  {
    string s;
    cin >> s;
    int n = s.size();
    vector<int> v = get_lcp(s);
    return 0;
  }
```

9 Geometry

9.1 Template

```
#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 <double, double>
#define pii pair<int, pi>
#define mp make_pair
#define fir first
#define sec second
#define MAXN 100001
#define mod 1000000007

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}; } // subtra o de pontos
  pt operator*(double d) { return {x * d, y * d}; } // multiplica o por um double
  pt operator/(double d) { return {x / d, y / d}; } // divis o por um double
};
double dot(pt v, pt w) // produto escalar (dot product)
{
  return v.x * w.x + v.y * w.y;
}
bool isPerp(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 orient(pt a, pt b, pt c) // se for = 0 os vetores  s o  colineares
{
  return cross(b - a, c - a);
}
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 ccw(pt a, pt b, pt c) // retorna se forma um angulo convexo ou concavo
{
  double ret = cross(b - a, c - b);
  return ret < 0;
}
double modulo(pt v) // |v| = sqrt( x  + y  )
{
  return sqrt(v.x * v.x + v.y * v.y);
}
double angle(pt a, pt b, pt c) // 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
}
signed main()
{
  //sabendo o cos p/ achar o angulo
  //double PI = acos(-1);
  //coss = acos(coss);
  //cout << (coss * 180) / PI << endl;
  return 0;
}
```

9.2 line Sweep

```
#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 200001
#define mod 1000000007

const double EPS = 1E-9; // para tratar a precisao do double e divisao por 0

struct pt
{
  double x, y;
};
struct seg
{
  pt p, q;
};
struct event
{
  double x;
  int type, id;
};

set<seg> s;                         // set de segmentos que come aram mas nao acabaram ainda
vector<set<seg>::iterator> where; // guarda os iterators de cada evento no set para que a gente
      consiga acessa-los

set<seg>::iterator prev(set<seg>::iterator it) // achar o iterator do adjacente da esquerda
{
  return it == s.begin() ? s.end() : it--;
}
set<seg>::iterator next(set<seg>::iterator it) // achar o iterator do adjacente da direita
{
  return it++;
```

```
}
double get_y(seg a, double x)
{
  if (abs(a.p.x - a.q.x) < EPS)
    return a.p.y;
  return a.p.y + (a.q.y - a.p.y) * (x - a.p.x) / (a.q.x - a.p.x);
}
bool operator<(const seg &a, const seg &b) // operator para o set de segmentos
{
  double x = max(min(a.p.x, a.q.x), min(b.p.x, b.q.x));
  return get_y(a, x) < get_y(b, x) - EPS;
}
bool cmp(event a, event b) // comparador para ordenar os eventos
{
  if (abs(a.x - b.x) > EPS)
    return a.x < b.x;
  a.type > b.type;
}
bool intersect1d(double l1, double r1, double l2, double r2) // verificar intersec o
{
  if (l1 > r1)
    swap(l1, r1);
  if (l2 > r2)
    swap(l2, r2);
  return max(l1, l2) <= min(r1, r2) + EPS;
}
int vec(pt a, pt b, pt c) // verificar intersec o
{
  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(seg a, seg b) // verificar intersec o
{
  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 line_sweep(vector<seg> v)
{
  vector<event> e;
  for (int i = 0; i < v.size(); i++) // para cada segmento
  {
    e.push_back({min(v[i].p.x, v[i].q.x), 1, i}); // evento do primeiro tipo: inicio do segmento
    e.push_back({max(v[i].p.x, v[i].q.x), 0, i}); // evento do segundo tipo: fim do segmento
  }
  sort(e.begin(), e.end(), cmp);     // ordenar os nossos eventos
  where.resize(v.size());            // tamanho do where = quantidade de eventos
  for (int i = 0; i < e.size(); i++) // para cada evento
  {
    int id = e[i].id; // id do evento atual
    if (e[i].type)    // primeiro tipo: o segmento com (ID = id) come a
    {
      auto nxt = s.lower_bound(v[id]), prv = prev(nxt); // acho os eventos adjacentes ao novo evento
      if (nxt != s.end() && intersect(*nxt, v[id]))     // um dos adjacentes se intersectam com o
            segmento atual ?
        return true;
      if (prv != s.end() && intersect(*prv, v[id]))
        return true;
      where[id] = s.insert(nxt, v[id]); // insiro o segmento no set de segmentos que come aram mas
            nao acabaram
    }
    else // segundo tipo: o segmento com (ID = id) acaba
    {
      auto nxt = next(where[id]), prv = prev(where[id]);            // acho os adjacentes do evento q
            vou remover
      if (nxt != s.end() && prv != s.end() && intersect(*nxt, *prv)) // esses novos adjacentes entre
            si se intersectam ?
        return true;
      s.erase(where[id]); // removo o segmento do set pois o segmento acabou
    }
  }
  return false;
}
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;
  (line_sweep(v)) ? cout << "YES\n" : cout << "NO\n";
  return 0;
}

// line sweep

// problema: dados n segmentos de reta em um plano, verifique se pelo menos dois desses segmentos se
      intersectam
// solu  o mais simples: iterar sobre todos os pares de segmentos existentes e verificar se algum
      deles se intersectam
// complexidade: O(n^2)

// solu  o mais eficiente: line sweep, complexidade: O(n * log n)

// algoritimo para o problema:
// 1- imagine uma reta na vertical com x = - INF (x mais a esquerda)
// 2- come ar a mover esta reta para a direita, durante o movimento essa reta ir  se encontrar com
      os segmentos.
// 3- estamos interessados na ordem relativa dos segmentos ao longo da vertical, para isso vamos criar
       um vector que
// guarda a coordenada x do inicio do segmento e a coordenada x do final do segmento
// 4- para encontrar um par de segmentos que se intersectam nesse processo, basta apenas considerar os
       segmentos
// adjacentes entre si.
// 5- se algum dos adjacentes se intersectam, return true, se eu nao achar nenhum, return false

/*
2
1 1 2 2
3 3 -1 1
NO

3
1 1 2 2
3 3 -1 1
0 0 -2 2
YES

*/
```

9.3 Convex Hull

```
#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 512
#define INF 200001
#define mod 1000000007

struct pt
{
  double x, y;
  pt operator+(pt p) { return {x + p.x, y + p.y}; }
  pt operator-(pt p) { return {x - p.x, y - p.y}; }
  pt operator*(double d) { return {x * d, y * d}; }
  pt operator/(double d) { return {x / d, y / d}; }
};
bool cmp(pt a, pt b) // ordenar os n pontos pela cordenada x
{
  return a.x < b.x || (a.x == b.x && a.y < b.y);
}
bool cw(pt a, pt b, pt c) // verificar se os pontos estao no sentido horario (clockwise)
{
  return a.x * (b.y - c.y) + b.x * (c.y - a.y) + c.x * (a.y - b.y) < 0;
}
bool ccw(pt a, pt b, pt c) // verificar se os pontos estao no sentido anti-horario (counter clockwise)
{
  return a.x * (b.y - c.y) + b.x * (c.y - a.y) + c.x * (a.y - b.y) > 0;
}
void convex_hull(vector<pt> v)
{
  sort(v.begin(), v.end(), cmp);
  pt p1 = v[0];
  pt p2 = v[v.size() - 1];
  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 printar 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]);
  for (auto const &i : ans)
    cout << i.x << " " << i.y << endl;
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n;
  int t = 0;
  while (cin >> n)
  {
    vector<pt> v(n);
    for (int i = 0; i < n; i++)
      cin >> v[i].x >> v[i].y;
    convex_hull(v);
    cout << endl;
    t++;
  }
}

// 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 s o menores do que 180 graus

// convex hull:
// dados n pontos em um plano, o objetivo    achar o menor pol gono 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    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 i r  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  ngulo formado
     pela linha
//    que liga o pen ltimo ponto e o  ltimo ponto do convex hull superior, com a linha que conecta o
//     ltimo ponto do convex hull e o ponto atual. Se o  ngulo  n o for no sentido hor rio,
//    removemos o ponto mais recente adicionado ao convex hull superior, pois o ponto atual s er
     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
```