Algorithm

chenRenning

算法模板整理

更新：2024 年 8 月 29 日

目录

代码基本框架:

```
1 #include <bits/stdc++.h>
2
3 using namespace std ;
4
```

```
void solve()
{
  // do something
}

int main()
{
  cin.tie(nullptr)->ios::sync_wtih_stdio(false) ;
  //关闭同步流-> 加快cin

  int t = 1 ;
  // cin >> t ; 多测的时候将注释去掉
  while ( t-- ) {
    solve() ;
  }
  return 0 ;
}
```

1 基础算法

1.1 排序

常用排序算法

- quick_sort $O(nlogn)$
- merge_sort $O(nlogn)$

```
// quick_sort
const int N = 1e5 + 10 ;

int a[N] , n ;

void quick_sort(int l,int r) {
  if ( l >= r ) return ;
  int mid = (l + r) >> 1 ;
  int k = a[mid] , i = l - 1 , j = r + 1 ;
  while ( j > i ) {
```

```
    do ++ i ; while ( a[i] < k ) ;
    do -- j ; while ( a[j] > k ) ;
    if ( j > i ) {
      std::swap(a[i] , a[j]) ;
    }
  }
  quick_sort(l , j) , quick_sort(j + 1 , r) ;
}
```

```
// merge_sort
const int N = 1e5 + 10 ;

int a[N] , tmp[N] , n ;

void merge_sort(int l,int r) {
  if ( l == r ) return ;
  int mid = (l + r) >> 1 ;
  merge_sort(l , mid) , merge_sort(mid + 1 , r) ;
  int i = l , j = mid + 1 , k = 0 ;
  while ( i <= mid && j <= r ) {
    if ( a[i] < a[j] ) {
      tmp[k++] = a[i++] ;
    } else {
      tmp[k++] = a[j++] ;
    }
  }
  while (i <= mid) tmp[k++] = a[i++] ;
  while (j <= r) tmp[k++] = a[j++] ;
  for(i = 0 , j = l ; j <= r ; ++ j , ++ i) a[j] = tmp[i] ;
}
```

1.2 二分 binary search

常用函数:

lower_bound(left,right,value)

upper_bound(left,right,value)

.....(以及 stl 容器自带方法等)

建立在有序序列之中 具体用法 https://oi-wiki.org/basic/binary/

```
// 边界值根据所需改变
// 整数二分
int l = 0 , r = 1e12 , ans = 0 ;
while ( r >= l ) {
  int mid = (l + r) >> 1 ; // 等同于 (l + r) / 2
  if ( check(mid) ) {
    ans = mid ;
    // r = mid - 1 or l = mid + 1 根据需求来
  } else {
    // l = mid + 1 or r = mid - 1 同理
  }
}
// ans 即所需 关键在check函数的设计
// 实数二分
const double eps = 1e-8 ; // 精度
double l = 0 , r = 1e6 , ans = 0 ;
while ( r - l >= eps )
{
  double mid = (l + r) / 2 ;
  if (check(mid)) {
    ans = mid ;
    // r = mid or l = mid
  } else {
    // l = mid or r = mid
  }
}
```

1.3 常见高精度

```
//高精度 + 高精度 O(n + m)
std::vector<int> add(std::vector<int> A,std::vector<int> B)
{
    std::vector<int> C ;
    int t = 0 ;
    for(int i = 0 , j = 0 ; i < (int)A.size() || j < (int)B.
       size() ; ++ i , ++ j)
```

```
    {
        if ( i < (int)A.size() ) t += A[i] ;
        if ( i < (int)B.size() ) t += B[i] ;
        C.push_back(t % 10) ;
        t /= 10 ;
    }
    while ( t ) C.push_back(t % 10) , t /= 10 ;
    while ( C.size() > 1 && C.back() == 0 ) C.pop_back() ;
    return C ;
}
```

```
// 高精度 - 高精度 O(n + m)
#include <bits/stdc++.h>

bool cmp(std::string a,std::string b) {
    if ( a.size() != b.size() ) return a.size() < b.size() ;
    for(int i = 0 ; i < (int)a.size() ; ++ i) if ( a[i] != b[i
       ] ) return a[i] < b[i] ;
    return false ;
}
std::vector<int> sub(std::vector<int> A,std::vector<int> B) {
    std::vector<int> C ;
    int t = 0 ;
    for(int i = 0 ; i < (int)A.size() ; ++ i)
    {
        t += A[i] ;
        if (i < (int)B.size()) t -= B[i] ;
        C.push_back((t % 10 + 10) %10) ;
        if ( t < 0 ) t = -1 ;
        else t = 0 ;
    }
    while (t) C.push_back((t % 10 + 10) % 10) , t /= 10;
    while (C.size() > 1 && C.back() == 0) C.pop_back();
    return C;
}

int main()
{
  //字符串直接比较是按照字典序，因此需要手写一个cmp函数比较二
```

```
    者数值大小
    std::string a , b ;
    std::cin >> a >> b ;
    if ( cmp(a , b) ) {
        std::cout << '-' ;
        swap(a , b) ;
    }
    std::vector<int> A , B ;
    for(int i = (int)a.size() - 1 ; i >= 0 ; -- i) A.push_back
       (a[i] - '0') ;
    for(int i = (int)b.size() - 1 ; i >= 0 ; -- i) B.push_back
       (b[i] - '0') ;
    auto C = sub(A , B) ;

    for(int i = (int)C.size() - 1 ; i >= 0 ; -- i) std::cout
       << C[i] ;

    return 0 ;
}
```

```
// 高精度 * 低精度
std::vector<int> mul(std::vector<int> A,int B) {
  std::vector<int> C ;
  int t = 0 ;
  for(int i = 0 ; i < (int)A.size() ; ++ i)
  {
      t += A[i] * B ;
      C.push_back(t % 10) ;
      t /= 10;
  }
  while ( t ) C.push_back(t % 10) , t /= 10  ;
  while ( C.size() > 1 && C.back() == 0 ) C.pop_back() ;
  return C ;
}
```

```
// 高精度 ÷ 低精度
std::vector<int> div(std::vector<int> A,int B,int &r)
{
    std::vector<int> C ;
```

```
    r = 0 ;
    for(int i = (int)A.size() - 1 ; i >= 0 ; -- i)
    {
        r = r * 10 + A[i] ;
        C.push_back(r / B) ;
        r %= B ;
    }
    reverse(C.begin(),C.end()) ;
    while ( C.size() > 1 && C.back() == 0 ) C.pop_back();
    return C ;
}
```

1.4 前缀和

```
// 一维
int n , a[N] , sum[N] ;
void solve() {
  std::cin >> n ;
  for(int i = 1 ; i <= n ; ++ i)
  {
    std::cin >> a[i] ;
    sum[i] = sum[i - 1] + a[i] ;
  }
}
// sum_i 表示前 i 个元素之和
```

```
// 二维
int n , a[N][N] , sum[N][N] ;
void solve() {
  std::cin >> n ;
  for(int i = 1 ; i <= n ; ++ i)
  {
    for(int j = 1 ; j <= n ; ++ j)
    {
      std::cin >> a[i][j] ;
      sum[i][j] = sum[i - 1][j] + sum[i][j - 1] - sum[i - 1][j
          - 1] + a[i][j] ;
    }
```

```
  }
}
// sum_{i ,j} 表示以(i , j) 为右下端点的矩阵元素之和
```

1.5 差分

```
// 一维
int n , q , a[N] , b[N] ;
void solve() {
  std::cin >> n >> q ;
  for(int i = 1 ; i <= n ; ++ i)
  {
    std::cin >> a[i] ;
  }
  while (q --)
  {
    // 每一次操作给定 l , r , v 表示在数组 a 区间[l , r] 上的
       元素加上 v
    int l , r , v ;
    std::cin >> l >> r >> v ;
    b[l] += v ;
    b[r + 1] -= v ;
  }
  for(int i = 1 ; i <= n ; ++ i)
  {
    b[i] += b[i - 1] ;
    a[i] += b[i] ;
  }
}
```

```
// 二维
#include <bits/stdc++.h>

const int N = 1010 ;

int n , m , q , a[N][N] , b[N][N] ;

void insert(int x1,int y1,int x2,int y2,int v)
```

```
{
    b[x1][y1] += v;
    b[x2 + 1][y1] -= v ;
    b[x1][y2 + 1] -= v ;
    b[x2 + 1][y2 + 1] += v ;
}

int main()
{
    std::cin >> n >> m >> q ;
    for(int i = 1 ; i <= n ; ++ i)
    {
        for(int j = 1 ; j <= m ; ++ j)
        {
            std::cin >> a[i][j] ;
        }
    }
    while ( q-- )
    {
        int x1 , y1 , x2 , y2 , v ;
        std::cin >> x1 >> y1 >> x2 >> y2 >> v ;
        insert(x1 , y1 , x2 , y2 , v) ;
    }
    for(int i = 1 ; i <= n ; ++ i)
    {
        for(int j = 1 ; j <= m ; ++ j)
        {
            b[i][j] += b[i - 1][j] + b[i][j - 1] - b[i - 1][j
               - 1] ;
            std::cout << b[i][j] + a[i][j] << ' ' ;
        }
        std::cout << '\n' ;
    }
    return 0 ;
}
```

1.6 双指针

```cpp
// 滑动窗口（同向双指针）
// 给定一个长度为 n 的整数序列，请找出最长的不包含重复的数的连续区间，输出它的长度。
#include <bits/stdc++.h>

int main()
{
    int n ;
    std::cin >> n ;
    std::vector<int> a(n) , cnt(100010 , 0) ;
    for(auto & p : a) std::cin >> p ;
    int l = 0 , r = 0 , ans = 0 ;
    while ( r < n )
    {
        ++ cnt[a[r]] ;
        while (r > l && cnt[a[r]] > 1) {
            -- cnt[a[l++]] ;
        }
        ans = std::max(ans , r - l + 1) ;
        ++ r ;
    }
    std::cout << ans << "\n" ;
    return 0 ;
}
```

```cpp
// 双向双指针
// 例题 lc1.两数之和
vector<int> twoSum(vector<int>& nums, int target) {
  vector<pair<int , int>> a;
  for(int i = 0 ; i < (int)nums.size() ; ++ i) a.push_back({nums[i] , i}) ;
  sort(a.begin(),a.end()) ;
  int l = 0 , r = (int)a.size() - 1 ;

  while (r > l)
  {
      if ( a[l].first + a[r].first == target ) {
```

```
            return {a[l].second , a[r].second} ;
        } else if ( a[l].first + a[r].first > target ) {
            --r ;
        } else {
            ++l ;
        }
    }

    return {} ;
}
```

1.7 离散化

```
//例题acwing802
#include <bits/stdc++.h>

int main()
{
    std::cin.tie(nullptr)->std::ios::sync_with_stdio(false) ;

    int n , m ;
    std::cin >> n >> m ;

    std::vector<std::pair<int , int>> a;
    std::vector<int> bin , b , sum ;
    for(int i = 1 ; i <= n ; ++ i)
    {
        int x , v ;
        std::cin >> x >> v ;
        a.push_back({x , v}) ;
        bin.push_back(x) ;
    }
    sort(a.begin(),a.end()) ;
    sort(bin.begin(),bin.end()) ;
    bin.erase(unique(bin.begin(),bin.end()),bin.end()) ;
    // unique 去重函数返回去重后的尾地址, erase(l , r) 删除下
       标
```

```
    b.resize(bin.size() , 0) ;
    sum.resize(bin.size() + 1 , 0) ;
    for(auto p : a)
    {
        int x = lower_bound(bin.begin(),bin.end(),p.first) -
           bin.begin() ;
        b[x] += p.second ;
    }

    for(int i = 0 ; i < (int)b.size() ; ++ i)
    {
        sum[i + 1] = sum[i] + b[i] ;
    }
    while (m--)
    {
        int l , r ;
        std::cin >> l >> r ;
        int L = upper_bound(bin.begin(),bin.end(),l) - bin.
           begin() - 1 ;
        int R = upper_bound(bin.begin(),bin.end(),r) - bin.
           begin() - 1 ;
        if ( bin[L] < l ) ++ L ;
        if ( L > R )
        {
            std::cout << 0 << "\n" ;
            continue ;
        }
        std::cout << sum[R + 1] - sum[L] << '\n' ;
    }
    return 0 ;
}
```

1.8 区间合并

```
//例题acwing803
std::vector<std::pair<int , int>> merge(std::vector<std::pair<
   int , int>> a)
```

```
{
    std::vector<std::pair<int , int>> C ;
    int l , r ;
    l = r = -2e9 ;
    for(auto p : a)
    {
        if ( p.first > r )
        {
            if ( r != -2e9 ) {
                C.push_back({l , r}) ;
            }
            l = p.first , r = p.second ;
        } else {
            r = std::max(p.second , r) ;
        }
    }
    C.push_back({l , r}) ;
    return C ;
}
```

1.9 应用

求逆序对的个数 (归并、线段树)

```
// O(nlogn) 归并排序过程中求得逆序对数量
i64 merge_sort(int l, int r) {
if ( l == r ) return 0 ;
int mid = (l + r) >> 1 ;
i64 x = merge_sort(l , mid) , y = merge_sort(mid + 1 , r) ,
   ans = 0 ;

int i = l , j = mid + 1 , k = 0 ;
while ( i <= mid && j <= r ) {
  if ( a[i] <= a[j] ) {
    tmp[k++] = a[i++] ;
  } else {
    ans += mid - i + 1 ;
    tmp[k++] = a[j++] ;
```

```
  }
}
while (i <= mid) tmp[k++] = a[i++] ;
while (j <= r) tmp[k++] = a[j++] ;
for(i = l , j = 0 ; i <= r ; ++ i , ++ j) a[i] = tmp[j] ;
return ans + x + y ;
```

```
// O(nlogn) 利用线段树(单点修改、区间查询)求得逆序对数量
#define ls (x << 1)
#define rs (x << 1 | 1)

int n , a[N] , sum[N << 2] , bin[N] ;
inline void update(int x) { sum[x] = sum[ls] + sum[rs] ; }
inline void modify(int pos,int l,int r, int x) {
    if ( l == r ) return sum[x] ++ , void() ;
    int mid = (l + r) >> 1 ;
    if ( pos <= mid ) modify(pos , l , mid , ls) ;
    else modify(pos , mid + 1 , r , rs) ;
    update(x) ;
}
inline int query(int A,int B,int l,int r,int x) {
    if ( A > B ) return 0 ;
    if ( A <= l && r <= B ) return sum[x] ;
    int mid = (l + r) >> 1 , ans = 0 ;
    if (A <= mid) ans += query(A , B , l , mid , ls) ;
    if ( mid < B ) ans += query(A , B , mid + 1 , r , rs) ;
    return ans ;
}
void solve() {
  std::cin >> n ;
  for(int i = 1 ; i <= n ; ++ i)
  {
    std::cin >> a[i] ;
    bin[i] = a[i] ;
  }
  std::sort(bin + 1 , bin + 1 + n) ;
  int m = std::unique(bin + 1 , bin + 1 + n) - bin - 1 ;
  i64 ans = 0 ;
  for(int i = 1 ; i <= n ; ++ i)
```

```
33    {
34        int x = std::lower_bound(bin + 1 , bin + 1 + m , a[i]) -
              bin ;
35        ans += query(x + 1 , m , 1 , n , 1) ;
36        modify(x , 1 , n , 1) ;
37    }
38    std::cout << ans << '\n' ;
39 }
```

2 数据结构

2.1 链表

```
//数组模拟单链表
//head需要初始化为 -1 表示末尾
int n , cnt , head , ne[N] , e[N] ;
inline void insert(int k,int x) {
  // 在第 k 个插入的元素后面插入一个元素 x
  e[cnt] = x , ne[cnt] = ne[k] , ne[k] = cnt++ ;
}
inline void del(int k) {
  // 删除第 k 个插入的元素后面的元素
  ne[k] = ne[ne[k]] ;
}
inline void head_insert(int x) {
  // 头部插入一个元素 x
  e[cnt] = x , ne[cnt] = head , head = cnt++ ;
}
inline void print() {
  // 输出整个链表
  for(int i = head ; i != -1 ; i = ne[i])
  {
    std::cout << e[i] << " " ;
  }
}
```

```
// 数组模拟双链表 例题acwing827
#include <bits/stdc++.h>

using namespace std ;

const int N = 2e5 + 10 ;

int n , lft[N] , rgt[N] , cnt , e[N] ;
void insert(int k, int x) {
    e[cnt] = x ;
    rgt[cnt] = rgt[k] ;
```

```
12      lft[cnt] = k ;
13      lft[rgt[cnt]] = cnt ;
14      rgt[k] = cnt++ ;
15  }
16  void del(int k) {
17      lft[rgt[k]] = lft[k] ;
18      rgt[lft[k]] = rgt[k] ;
19  }
20  int main()
21  {
22      cin >> n ;
23      cnt = 2 ;
24      lft[1] = 0 , rgt[0] = 1 ;
25      for(int i = 1 ; i <= n ; ++ i)
26      {
27          string op ;
28          cin >> op ;
29          if (op == "L")
30          {
31              int x ;
32              cin >> x ;
33              insert(0 , x) ;
34          } else if (op == "R") {
35              int x ;
36              cin >> x ;
37              insert(lft[1] , x) ;
38          } else if ( op == "D") {
39              int k ;
40              cin >> k ;
41              del(k + 1) ;
42          } else if ( op == "IL" ) {
43              int k , x ;
44              cin >> k >> x ;
45              insert(lft[k + 1] , x) ;
46          } else if ( op == "IR" ) {
47              int k , x ;
48              cin >> k >> x ;
49              insert(k + 1 , x) ;
```

```
        }
    }
    for(int i = rgt[0] ; i != 1 ; i = rgt[i])
    {
        cout << e[i] << " " ;
    }
    return 0 ;
}
```

2.2 栈

```
//经典例题 : 表达式求值
#include <bits/stdc++.h>

using namespace std ;

stack<char> op ;
stack<int> num ;

void eval()
{
    auto a = num.top() ; num.pop() ;
    auto b = num.top() ; num.pop() ;
    auto c = op.top() ; op.pop() ;
    if ( c == '*' ) {
        num.push(a * b) ;
    } else if ( c == '/' ) {
        num.push(b / a) ;
    } else if ( c == '+' ) {
        num.push(a + b) ;
    } else {
        num.push(b - a) ;
    }
}

int main()
{
```

```
    unordered_map<char , int> pr {{'+',1},{'-',1},{'*',2},{'/',2}} ;
    string str ;
    cin >> str ;

    for(int i = 0 ; i < (int)str.size() ; ++ i)
    {
        if ( isdigit(str[i]) ) {
            int j = str[i] - '0' ;
            ++ i;
            while ( i < (int)str.size() && isdigit(str[i]) ) {
                j = j * 10 + str[i++] - '0' ;
            }
            -- i;
            num.push(j) ;
        } else if (str[i] == '(') {
            op.push(str[i]) ;
        } else if (str[i] == ')') {
            while ( op.top() != '(' ) eval();
            op.pop() ;
        } else {
            while ( !op.empty() && pr[op.top()] >= pr[str[i]] ) eval() ;
            op.push(str[i]) ;
        }
    }
    while ( !op.empty() ) eval() ;
    cout << num.top() ;
    return 0 ;
}
```

2.3 单调栈、单调队列

```
//单调栈
#include <bits/stdc++.h>

using namespace std ;
```

```

int main()
{
    int n ;
    std::cin >> n ;
    stack<int> S ;
    for(int i = 1 ; i <= n ; ++ i)
    {
        int x ;
        cin >> x;
        while ( !S.empty() && S.top() >= x ) S.pop() ;
        if ( S.empty() ) {
            cout << -1 << ' ' ;
        } else {
            cout << S.top() << ' ' ;
        }
        S.push(x) ;
    }
    return 0 ;
}
```

```
//一维单调队列
#include <bits/stdc++.h>

using namespace std ;

int main()
{
    int n , k ;
    cin >> n >> k ;
    std::vector<int> a(n) ;
    for(auto & p : a) std::cin >> p ;

    deque<int> mn , mx ;
    std::vector<int> r1 , r2 ;
    for(int i = 0 ; i < n ; ++ i)
    {
        while ( !mn.empty() && a[mn.back()] > a[i] ) mn.
            pop_back();
```

```
        while ( !mx.empty() && a[mx.back()] < a[i] ) mx.
           pop_back() ;
        mx.push_back(i) ;
        mn.push_back(i) ;
        while ( i - mn.front() + 1 > k ) mn.pop_front() ;
        while ( i - mx.front() + 1 > k ) mx.pop_front() ;
        if ( i + 1 >= k ) {
            r1.push_back(a[mn.front()]) ;
            r2.push_back(a[mx.front()]) ;
        }
    }
    for(auto p : r1) cout << p << ' ' ;
    cout << '\n' ;
    for(auto p : r2) cout << p << ' ' ;
    return 0 ;
}
```

2.4 KMP

字符串重要算法之一 (模式匹配)

```
// O(n + m)
#include <bits/stdc++.h>

using namespace std ;

const int N = 1e6 + 10 ;

int n , m , ne[N] ;
string s , t ;

int main()
{
    cin >> n >> t >> m >> s ;
    s = '#' + s , t = '#' + t ;
    for(int i = 2 , j = 0 ; i <= n ; ++ i)
    {
        while ( j && t[j + 1] != t[i] ) j = ne[j] ;
```

```
        if ( t[j + 1] == t[i] ) ++ j ;
        ne[i] = j ;
    }
    for(int i = 1 , j = 0 ; i <= m ; ++ i)
    {
        while ( j && t[j + 1] != s[i] ) j = ne[j] ;
        if ( t[j + 1] == s[i] ) ++ j ;
        if ( j == n ) {
            cout << i - n << ' ' ;
            j = ne[j] ;
        }
    }
    return 0 ;
}
```

2.5 字典树 Trie

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

using namespace std ;

const int N = 2e5 + 10 ;

int son[N][26] , n , cnt[N] , tot ;
void insert(string str) {
    int p = 0 ;
    for(auto c : str) {
        if ( !son[p][c - 'a'] ) son[p][c - 'a'] = ++tot ;
        p = son[p][c - 'a'] ;
    }
    ++ cnt[p] ;
}
int query(string str) {
    int p = 0 ;
    for(auto c : str) {
        if ( !son[p][c - 'a'] ) return 0 ;
        p = son[p][c - 'a'] ;
```

```
    }
    return cnt[p] ;
}
int main()
{
    cin >> n ;
    for(int i = 1 ; i <= n ; ++ i)
    {
        char op ;
        string str ;
        cin >> op >> str ;
        if ( op == 'I' ) {
            insert(str) ;
        } else {
            cout << query(str) << '\n' ;
        }
    }
    return 0 ;
}
```

2.6 并查集 DSU

```
//维护集合大小 Acwing837
#include <bits/stdc++.h>

const int N = 1e5 + 10 ;

int f[N] , size[N] , n , q ;

int find(int x) { return f[x] == x ? x : f[x] = find(f[x]) ; }
void merge(int a,int b) {
    int fa = find(a) , fb = find(b) ;
    if ( fa != fb ) {
        f[fb] = fa ;
        size[fa] += size[fb] ;
    }
}
```

```
int main()
{
    std::cin >> n >> q ;
    for(int i = 1 ; i <= n ; ++ i) f[i] = i , size[i] = 1 ;
    while (q--)
    {
        std::string op ;
        std::cin >> op ;
        if ( op == "C" )
        {
            int a , b ;
            std::cin >> a >> b ;
            merge(a , b) ;
        } else if ( op == "Q1") {
            int a , b ;
            std::cin >> a >> b ;
            if ( find(a) == find(b) ) {
                std::cout << "Yes\n" ;
            } else {
                std::cout << "No\n" ;
            }
        } else {
            int x ;
            std::cin >> x ;
            std::cout << size[find(x)] << '\n' ;
        }
    }

    return 0 ;
}
```

2.6.1 扩展域并查集

```
// 食物链 https://www.luogu.com.cn/problem/P2024
/*
A->B , B->C , C->A
因为
A->B , B->C
```

```
6  所以
7  A->C
8  如果C->A 则是假话
9  同理
10 */
11 #include <bits/stdc++.h>
12
13 using namespace std ;
14
15 const int N = 2e5 + 10 ;
16
17 #define A(x) x          // 猎人
18 #define B(x) x+50000    // 猎物
19 #define C(x) x+100000   // 天敌
20
21 int n , m , f[N] ;
22
23 int find(int x) {return f[x] == x ? x : f[x] = find(f[x]) ;}
24 void merge(int a,int b) {
25     int fa = find(a) , fb = find(b) ;
26     if ( fa != fb ) {
27         f[fb] = fa ;
28     }
29 }
30 bool same(int a,int b) { return find(a) == find(b) ; }
31
32 int main()
33 {
34     cin.tie(nullptr)->ios::sync_with_stdio(false) ;
35
36     cin >> n >> m ;
37     for(int i = 1 ; i < N ; ++ i) f[i] = i ;
38
39     int ans = 0 ;
40
41     for(int i = 1 ; i <= m ; ++ i)
42     {
43         int op , x , y ;
```

```
44          cin >> op >> x >> y ;
45
46          if ( x > n || y > n ) {
47              ++ ans ;
48              continue ;
49          }
50
51          if ( op & 1 ) {
52              // x 和 y 是同类
53              if ( same(A(x) , B(y)) || same(A(x) , C(y)) ||
                    same(B(x) , C(y)) )
54              {
55                  ++ ans ;
56                  continue ;
57              }
58              merge(A(x) , A(y)) ;
59              merge(B(x) , B(y)) ;
60              merge(C(x) , C(y)) ;
61          } else {
62              // x 吃 y
63              if ( x == y || same(A(x) , A(y)) || same(B(x) , A(
                    y)) )
64              {
65                  ++ ans ;
66                  continue ;
67              }
68              merge(A(x) , B(y)) ;
69              merge(B(x) , C(y)) ;
70              merge(C(x) , A(y)) ;
71          }
72      }
73
74      cout << ans << '\n' ;
75      return 0 ;
76  }
```

2.7 模拟散列表

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

using namespace std ;

const int inf = 0x3f3f3f3f ;

const int N = 100003 ; // 大于等于最大存储空间的第一个质数
// 拉链法
int e[N] , h[N] , idx , ne[N] ;
inline void add(int a,int b) {
    e[idx] = b , ne[idx] = h[a] , h[a] = idx++ ;
}
inline bool query(int x) {
    int hash = (x % N + N) % N ;

    for(int i = h[hash] ; i != inf ; i = ne[i])
    {
        if ( e[i] == x ) {
            return true ;
        }
    }
    return false ;
}

// const int N = 200003 ; // 大于等于2倍空间的第一个质数
// // 开放寻址法
// int h[N] ;
// inline int find(int x) {
//     int p = (x % N + N) % N ;
//     while ( h[p] != inf && h[p] != x ) {
//         ++ p ;
//         if ( p == N ) p = 0 ;
//     }
//     return p ;
// }

```

```

int main()
{
    cin.tie(nullptr)->ios::sync_with_stdio(false) ;
    memset(h , 0x3f , sizeof h) ;
    int n ;
    cin >> n ;
    for(int i = 1 ; i <= n ; ++ i){
        char op ; int x ;
        cin >> op >> x ;
        // int k = find(x) ;
        if ( op == 'I' ) {
            // h[k] = x ;
            add((x % N + N) % N , x) ;
        } else {
            // cout << (h[k] == x ? "Yes\n" : "No\n") ;
            cout << (query(x) ? "Yes\n" : "No\n") ;
        }
    }

    return 0 ;
}
```

2.8 字符串哈希

```
// acwing841
#include <bits/stdc++.h>

using namespace std ;

typedef unsigned long long ULL ;
const int N = 1e5 + 10 ;
const int mod = 1000000007 ; // 998244353 、43112609 、
   37156667
// base : 131 、 13331

ULL P[N] , h[N] ;
```

```
12
13 ULL get(int l,int r) {
14     return (h[r] - h[l - 1] * P[r - l + 1] % mod + mod ) % mod
           ;
15 }
16
17 int main()
18 {
19     cin.tie(nullptr)->ios::sync_with_stdio(false) ;
20
21     int n , q ;
22     string str ;
23     cin >> n >> q >> str ;
24
25     str = '#' + str ;
26     P[0] = 1 ;
27     for(int i = 1 ; i <= n ; ++ i){
28         P[i] = P[i - 1] * 131 % mod  ;
29         h[i] = (h[i - 1] * 131 + str[i]) % mod ;
30     }
31     while (q--)
32     {
33         int l1 , r1 , l2 , r2 ;
34         cin >> l1 >> r1 >> l2 >> r2 ;
35         cout << (get(l1 , r1) == get(l2 , r2) ? "Yes\n" : "No\
              n") ;
36     }
37
38     return 0 ;
39 }
```

2.9 莫队

2.9.1 普通莫队

```
1 // 莫队
2  #pragma GCC optimize(3)
```

```
 #pragma GCC optimize("Ofast,no-stack-protector")
nclude <bits/stdc++.h>

ing i64 = long long ;

nst int N = 3e4 + 10 , M = 1e6 + 10 , QN = 2e5 + 10  ;

t cnt[M] , dif , a[N] , n , m , ans[QN] , pos[QN] ;
ruct Q

int l , r , idx ;
q[QN] ;

id solve() {
std::cin >> n ;
for(int i = 1 ; i <= n ; ++ i)
{
  std::cin >> a[i] ;
}
std::cin >> m ;
for(int i = 1 ; i <= m ; ++ i)
{
  std::cin >> q[i].l >> q[i].r ;
  q[i].idx = i ;
}

int block_len = sqrt(m) ;
int block_num = ceil((double) m / block_len) ;
for(int i = 1 ; i <= block_num ; ++ i)
  for(int j = (i - 1) * block_len + 1 ; j <= i * block_len &&
     j <= m ; ++ j)
    pos[j] = i ;

std::sort(q + 1 , q + m + 1 , [&](const Q& x ,const Q& y) {
  return (pos[x.l] ^ pos[y.l]) ? pos[x.l] < pos[y.l] : (pos[x.
     l] & 1) ? x.r < y.r : x.r > y.r  ;
});

```

```
int l = 1 , r = 0 ;
for(int i = 1 ; i <= m ; ++ i)
{
  int pl = q[i].l , pr = q[i].r ;
  while ( l < pl ) dif -= !--cnt[a[l++]] ;
  while ( l > pl ) dif += !cnt[a[--l]]++ ;
  while ( r < pr ) dif += !cnt[a[++r]]++ ;
  while ( r > pr ) dif -= !--cnt[a[r--]] ;
  ans[q[i].idx] = dif ;
}
for(int i = 1 ; i <= m ; ++ i)
{
  std::cout << ans[i] << '\n' ;
}


t main()

  std::cin.tie(nullptr)->std::ios::sync_with_stdio(false) ;

  int t = 1 ;
  // std::cin >> t ;
  while ( t-- ) {
      solve() ;
  }

  return 0 ;
```

2.9.2 带修莫队

```
/*
带修莫队
#pragma GCC optimize(3)
#pragma GCC optimize("Ofast,no-stack-protector")
*/
#include <bits/stdc++.h>

using i64 = long long ;
```

```
9
10 const int N = 2e5 + 10 , M = 1e6 + 10 ;
11
12 int cnt[M] , n , m , dif , a[N] , pos[N] , ans[N] ;
13 int cntq , cntc ;
14 struct Q {
15   int l , r , time , idx ;
16 } q[N] ;
17 struct modify
18 {
19   int pos , color , last ;
20 } c[N];
21
22 void solve() {
23     std::cin >> n >> m ;
24     for(int i = 1 ; i <= n ; ++ i)
25     {
26       std::cin >> a[i] ;
27     }
28
29     int block_len = pow(n , 2.0 / 3.0) ;
30     int block_num = ceil((double) n / block_len) ;
31     for(int i = 1 ; i <= block_num ; ++ i)
32       for(int j = (i - 1) * block_len + 1 ; j <= i * block_len
             && j <= n ; ++ j)
33         pos[j] = i ;
34     for(int i = 1 ; i <= m ; ++ i)
35     {
36       char op ; int l , r ;
37       std::cin >> op >> l >> r ;
38       if ( op == 'Q' ) {
39         q[++cntq] = {l , r , cntc , cntq} ;
40       } else {
41         c[++cntc] = {l , r , a[l]} ;
42       }
43     }
44     std::sort(q + 1 , q + 1 + cntq , [&](const Q &x , const Q
          & y) {
```

```
45      return (pos[x.l] ^ pos[y.l]) ? pos[x.l] < pos[y.l] : (
            pos[x.r] ^ pos[y.r]) ? pos[x.r] < pos[y.r] : x.time <
            y.time ;
46    }) ;
47
48    int l = 1 , r = 0 , time = 0 ;
49    for(int i = 1 ; i <= cntq ; ++ i)
50    {
51      int ql = q[i].l , qr = q[i].r , qt = q[i].time ;
52      while ( l < ql ) dif -= !--cnt[a[l++]] ;
53      while ( l > ql ) dif += !cnt[a[--l]]++ ;
54      while ( r < qr ) dif += !cnt[a[++r]]++ ;
55      while ( r > qr ) dif -= !--cnt[a[r--]] ;
56      while ( time < qt ) {
57        ++ time ;
58        if ( ql <= c[time].pos && c[time].pos <= qr ) dif -=
              !--cnt[a[c[time].pos]] - !cnt[c[time].color]++ ;
59        std::swap(a[c[time].pos] , c[time].color) ;
60      }
61      while ( time > qt ) {
62        if ( ql <= c[time].pos && c[time].pos <= qr ) dif -=
              !--cnt[a[c[time].pos]] - !cnt[c[time].color]++ ;
63        std::swap(a[c[time].pos] , c[time].color) ;
64        -- time ;
65      }
66      ans[q[i].idx] = dif ;
67    }
68    for(int i = 1 ; i <= cntq ; ++ i)
69    {
70      std::cout << ans[i] << '\n' ;
71    }
72  }
73
74  int main()
75  {
76    std::cin.tie(nullptr)->std::ios::sync_with_stdio(false) ;
77
78    int t = 1 ;
```

```
    // std::cin >> t ;
    while ( t-- ) {
        solve() ;
    }

    return 0 ;
}
```

2.9.3 树上莫队

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

using i64 = long long ;

const int N = 2e5 + 10 ;

int n , m , a[N] , pos[N] , cnt[N] , bin[N] , ord[N] , ordn ,
   first[N] , last[N] ;
int depth[N] , fa[21][N] , vis[N] , dif , ans[N] ;
std::vector<int> edges[N] ;
struct Q
{
  int l , r , lca , idx ;
} query[N] ;
void dfs(int v,int f) // 预处理 欧拉序
{
  ord[++ordn] = v ;
  first[v] = ordn ;
  for(auto u : edges[v])
  {
    if ( u != f )
    {
      depth[u] = depth[v] + 1 ;
      fa[0][u] = v ;
      for(int i = 1 ; i <= 20 ; ++ i)
        fa[i][u] = fa[i - 1][fa[i - 1][u]] ;
      dfs(u , v) ;
    }
```

```
28    }
29    ord[++ordn] = v ;
30    last[v] = ordn ;
31  }
32  inline int get_lca(int a,int b) { // 倍增求LCA
33    if ( depth[a] < depth[b] ) std::swap(a , b) ;
34    for(int i = 20 ; i >= 0 ; -- i)
35    {
36      if (depth[fa[i][a]] >= depth[b] ) {
37        a = fa[i][a] ;
38      }
39    }
40    if ( a == b ) return a ;
41    for(int i = 20 ; i >= 0 ; -- i)
42    {
43      if (fa[i][a] != fa[i][b])
44      {
45        a = fa[i][a] ;
46        b = fa[i][b] ;
47      }
48    }
49    return fa[0][a] ;
50  }
51  inline void work(int pos) {
52    vis[pos] ? dif -= !--cnt[a[pos]] : dif += !cnt[a[pos]]++ ;
53    vis[pos] ^= 1 ;
54  }
55
56
57  void solve() {
58      std::cin >> n >> m ;
59      for(int i = 1 ; i <= n ; ++ i)
60      {
61        std::cin >> a[i] ;
62        bin[i] = a[i] ;
63      }
64      std::sort(bin + 1 , bin + 1 + n) ;
65      int mm = std::unique(bin + 1 , bin + 1 + n) - bin ;
```

```
66      for(int i = 1 ; i <= n ; ++ i) {
67        a[i] = std::lower_bound(bin + 1 , bin + 1 + mm , a[i]) -
              bin ;
68      }
69      int block_len = sqrt(m) ;
70      int block_num = ceil((double)m / block_len) ;
71      for(int i = 1 ; i <= block_num ; ++ i)
72        for(int j = (i - 1) * block_len + 1 ; j <= i * block_len
              && j <= m ; ++ j)
73          pos[j] = i ;
74      for(int i = 1 , u , v ; i < n ; ++ i)
75      {
76        std::cin >> u >> v ;
77        edges[u].push_back(v) ;
78        edges[v].push_back(u) ;
79      }
80      depth[1] = 1 , depth[0] = 0 ;
81      dfs(1 , 0) ;
82      for(int i = 1 , u , v ; i <= m ; ++ i)
83      {
84        std::cin >> u >> v ;
85        if ( first[u] > first[v] ) std::swap(u , v) ;
86        int lca = get_lca(u , v) ;
87        if ( lca == u ) {
88          query[i] = {first[u] , first[v] , 0 , i} ;
89        } else {
90          query[i] = {last[u] , first[v] , lca , i} ;
91        }
92      }
93      std::sort(query + 1 , query + 1 + m , [&](const Q&x ,
           const Q&y) {
94        return (pos[x.l] ^ pos[y.l]) ? pos[x.l] < pos[y.l] : (
             pos[x.l] & 1) ? x.r < y.r : x.r > y.r ;
95      }) ;
96
97      int l = 1 , r = 0 ;
98      for(int i = 1 ; i <= m ; ++ i)
99      {
```

```
100         int ql = query[i].l , qr = query[i].r , lca = query[i].
                lca ;
101         while ( l < ql ) work(ord[l++]) ;
102         while ( l > ql ) work(ord[--l]) ;
103         while ( r < qr ) work(ord[++r]) ;
104         while ( r > qr ) work(ord[r--]) ;
105         if ( lca ) work(lca) ;
106         ans[query[i].idx] = dif ;
107         if ( lca ) work(lca) ;
108     }
109     for(int i = 1 ; i <= m ; ++ i)
110     {
111         std::cout << ans[i] << '\n' ;
112     }
113 }
114
115 int main()
116 {
117     std::cin.tie(nullptr)->std::ios::sync_with_stdio(false) ;
118
119     int t = 1 ;
120     // std::cin >> t ;
121     while ( t-- ) {
122         solve() ;
123     }
124
125     return 0 ;
126 }
```

2.9.4 回滚莫队

```
1 /*
2 回滚莫队(模板题)AT1219[JOI2013]
3 根据莫队的性质 : 左端点在同一块中的所有查询的右端点是单调递
      增。这样
4 对于左端点在同一块的每一个询问可以以O(N)解决所有右端点, 考虑枚
      举每个块
5 总共需要枚举O(\sqrt(N)) 个块, 这部分总复杂度为 O(N\sqrt(N))
```

```
6  又对于每个块的左端点 : 假设每个块的每个左端点都从右端开始统
       计, 每次都要重新暴力统计
7  一次, 做完每个左端点复杂度为O(\sqrt(N)), 共n个左端点, 总复杂度
       O(N\sqrt(N))
8  */
9
10 #include <bits/stdc++.h>
11
12 using i64 = long long ;
13
14 const int N = 1e5 + 10 ;
15
16 int n , q , cnt1[N] , cnt2[N] , a[N] , bin[N] , pos[N] ;
17 i64 ans[N] ;
18 struct Q{
19   int l , r , idx ;
20 } query[N] ;
21
22 void solve() {
23     std::cin >> n >> q;
24     for(int i = 1 ; i <= n ; ++ i)
25     {
26       std::cin >> a[i] ;
27       bin[i] = a[i] ;
28     }
29     std::sort(bin + 1 , bin + 1 + n) ;
30     int mm = std::unique(bin + 1 , bin + 1 + n) - bin - 1 ;
31     for(int i = 1 ; i <= n ; ++ i)
32     {
33       a[i] = std::lower_bound(bin + 1 , bin + 1 + mm , a[i]) -
             bin;
34     }
35     int block_len = sqrt(n) ;
36     int block_num = ceil((double)n / block_len) ;
37     for(int i = 1 ; i <= q ; ++ i)
38     {
39       std::cin >> query[i].l >> query[i].r ;
40       query[i].idx = i ;
```

```
41      }
42      for(int i = 1 ; i <= block_num ; ++ i)
43        for(int j = (i - 1) * block_len + 1 ; j <= n && j <= i *
              block_len ; ++ j)
44          pos[j] = i ;
45
46      std::sort(query + 1 , query + 1 + q , [&](const Q&x ,
           const Q&y) {
47        return (pos[x.l] ^ pos[y.l]) ? pos[x.l] < pos[y.l] : x.r
              < y.r ;
48      });
49
50      int i = 1 ;
51      for(int k = 1 ; k <= block_num ; ++ k)
52      {
53        int l = k * block_len + 1 , r = k * block_len ;
54        i64 now = 0;
55        std::memset(cnt1 , 0 , sizeof cnt1) ;
56        for( ; pos[query[i].l] == k ; ++ i)
57        {
58          int ql = query[i].l ,qr = query[i].r ;
59          i64 tmp ;
60          if ( pos[ql] == pos[qr] ) {
61            tmp = 0 ;
62            for(int j = ql ; j <= qr ; ++ j) cnt2[a[j]] = 0 ;
63            for(int j = ql ; j <= qr ; ++ j)
64            {
65              ++ cnt2[a[j]] ;
66              tmp = std::max(tmp , (i64)cnt2[a[j]] * bin[a[j]])
                    ;
67            }
68            ans[query[i].idx] = tmp ;
69            continue ;
70          }
71          while ( r < qr ) {
72            ++r ; ++ cnt1[a[r]] ; now = std::max(now , (i64)cnt1
                 [a[r]] * bin[a[r]]) ;
73          }
```

```
74          tmp = now ;
75          while ( l > ql ) {
76            --l ; ++ cnt1[a[l]] ; now = std::max(now , (i64)cnt1
                [a[l]] * bin[a[l]]) ;
77          }
78          ans[query[i].idx] = now ;
79          while (l < k * block_len + 1 )
80          {
81            --cnt1[a[l++]] ;
82          }
83          now = tmp ;
84        }
85      }
86      for(int i = 1 ; i <= q ; ++ i)
87      {
88        std::cout << ans[i] << '\n' ;
89      }
90  }
91
92  int main()
93  {
94      std::cin.tie(nullptr)->std::ios::sync_with_stdio(false) ;
95
96      int t = 1 ;
97      // std::cin >> t ;
98      while ( t-- ) {
99          solve() ;
100     }
101
102     return 0 ;
103 }
```

2.10 应用

```
1  // 求区间众数 (模板题该题是强制在线)
2  #include <bits/stdc++.h>
3
```

```
4  using i64 = long long ;
5
6  #define all(x) x.begin(),x.end()
7
8  const int inf = 0x3f3f3f3f ;
9  const int N = 4e4 + 10 , M = 5e4 + 10 , T = 785 ;
10
11 int n , q , pos[N] , a[N] , bin[N] , cnt[N] ;
12 int f[T][T] ; // f[i][j] 表示 以第 i , j 块为端点区间的众数
13 std::vector<int> e[N] ; // 处理散块
14
15 void solve() {
16   std::cin >> n >> q ;
17   for(int i = 1 ; i <= n ; ++ i)
18   {
19     std::cin >> a[i] ;
20     bin[i] = a[i] ;
21   }
22   std::sort(bin + 1 , bin + 1 + n) ;
23   int mm = std::unique(bin + 1 , bin + 1 + n) - bin - 1 ;
24   for(int i = 1 ; i <= n ; ++ i)
25   {
26     a[i] = std::lower_bound(bin + 1 , bin + 1 + mm , a[i]) -
          bin ;
27     e[a[i]].push_back(i) ;
28   }
29
30   int block_len = n / sqrt(n * std::__lg(n)) ;
31   int block_num = ceil((double) n / block_len) ;
32   for(int i = 1 ; i <= block_num ; ++ i) for(int j = (i - 1) *
       block_len + 1 ; j <= n && j <= i * block_len ; ++ j) pos[
      j] = i ;
33   for(int i = 1 ; i <= block_num ; ++ i) {
34     int mx = 0 , val = mm ;
35     for(int j = (i - 1) * block_len + 1 ; j <= n ; ++ j)
36     {
37       ++ cnt[a[j]] ;
```

```
38      if ( mx < cnt[a[j]] || ( mx == cnt[a[j]] && val > a[j] )
           )
39      {
40        mx = cnt[a[j]] ;
41        val = a[j] ;
42      }
43      f[i][pos[j]] = val ;
44    }
45    std::memset(cnt , 0 , sizeof (int) * (mm + 1)) ;
46  }
47  int pre = 0 ;
48  int sum = 0 ;
49  for(int i = 1 ; i <= q ; ++ i)
50  {
51    int l , r ;
52    std::cin >> l >> r ;
53    l = (l + pre - 1) % n + 1;
54    r = (r + pre - 1) % n + 1;
55    if ( l > r ) std::swap(l , r) ;
56    int mx = 0 , val = mm , pl = pos[l] , pr = pos[r] ;
57    if ( pl == pr ) {
58      sum += r - l + 1 ;
59      for(int j = l ; j <= r ; ++ j)
60      {
61        int lo = std::lower_bound(all(e[a[j]]) , l) - e[a[j]].
             begin() ;
62        int hi = std::upper_bound(all(e[a[j]]) , r) - e[a[j]].
             begin() ;
63        if ( hi - lo > mx || (hi - lo == mx && val > a[j]))
64        {
65          mx = hi - lo ;
66          val = a[j] ;
67        }
68      }
69      pre = bin[val] ;
70      std::cout << (pre = bin[val]) << '\n' ;
71    } else {
72      if ( pr > pl + 1 ) {
```

```
73          val = f[pl + 1][pr - 1] ;
74          int lo = std::lower_bound(all(e[val]) , l) - e[val].
               begin() ;
75          int hi = std::upper_bound(all(e[val]) , r) - e[val].
               begin() ;
76          mx = hi - lo ;
77        }
78        for(int j = l ; j <= block_len * pl ; ++ j)
79        {
80          int lo = std::lower_bound(all(e[a[j]]) , l) - e[a[j]].
               begin() ;
81          int hi = std::upper_bound(all(e[a[j]]) , r) - e[a[j]].
               begin() ;
82          if ( hi - lo > mx || (hi - lo == mx && val > a[j]))
83          {
84            mx = hi - lo ;
85            val = a[j] ;
86          }
87        }
88        for(int j = (pr - 1) * block_len + 1 ; j <= r ; ++ j)
89        {
90          int lo = std::lower_bound(all(e[a[j]]) , l) - e[a[j]].
               begin() ;
91          int hi = std::upper_bound(all(e[a[j]]) , r) - e[a[j]].
               begin() ;
92          if ( hi - lo > mx || (hi - lo == mx && val > a[j]))
93          {
94            mx = hi - lo ;
95            val = a[j] ;
96          }
97        }
98        pre = bin[val] ;
99        std::cout << (pre = bin[val]) << '\n' ;
100     }
101
102   }
103 }
104
```

```
int main()
{
    std::cin.tie(nullptr)->std::ios::sync_with_stdio(false) ;
    int t = 1 ;
    // std::cin >> t ;
    while ( t-- ) {
        solve() ;
    }

    return 0 ;
}
```

```
//最大异或对 url: https://www.luogu.com.cn/problem/P10471
#include <bits/stdc++.h>

using namespace std ;

const int N = 3e6 + 10 ;

int n , son[N][2] , tot ;
void insert(string str)
{
    int p = 0 ;
    for(auto c : str){
        if ( !son[p][c - '0'] ) son[p][c - '0'] = ++tot ;
        p = son[p][c - '0'] ;
    }
}
int str_xor(string a,string b) {
    int ans = 0 ;
    for(int i = 0 ; i <= 30 ; ++ i)
    {
        if ( a[i] != b[i] ) {
            ans += 1 << (30 - i) ;
        }
    }
    return ans ;
}
string query(string str) {
```

```
28      int p = 0 ;
29      string ans ;
30      for(auto c : str) {
31          int t = (c - '0') ^ 1 ;
32          if ( !son[p][t] ) {
33              p = son[p][c - '0'] ;
34              ans.push_back(c) ;
35          } else {
36              p = son[p][t] ;
37              ans.push_back(char('0' + t)) ;
38          }
39      }
40      return ans ;
41  }
42
43  int main()
44  {
45      cin >> n ;
46      std::vector<string> a(n) ;
47      for(int i = 0 ; i < n ; ++ i)
48      {
49          int x ;
50          cin >> x ;
51          for(int j = 30 ; j >= 0 ; -- j){
52              a[i].push_back(char('0' + (x >> j & 1))) ;
53          }
54          insert(a[i]) ;
55      }
56
57      int ans = 0 ;
58      for(int i = 0 ; i < n ; ++ i)
59      {
60          ans = max(ans , str_xor(a[i] , query(a[i]))) ;
61      }
62      cout << ans << '\n' ;
63
64      return 0 ;
65  }
```

```cpp
// 双哈希 https://atcoder.jp/contests/abc339/tasks/abc339_f
#include <bits/stdc++.h>

using namespace std ;

const int mod1 = 1000000007 ; // 998244353 、43112609 、37156667
const int mod2 = 43112609 ;
const int base = 131 ;

typedef unsigned long long ULL ;
typedef pair<ULL , ULL> UPII ;

int main()
{
  cin.tie(nullptr)->ios::sync_with_stdio(false) ;
  int n ;
  cin >> n ;
  std::map<UPII,int> mp ;
  std::vector<UPII> hash(n);
  for(int i = 0 ; i < n ; ++ i)
  {
    string x ;
    cin >> x ;
    ULL h1 = 0 , h2 = 0 ;
    for(auto c : x) {
      h1 = (h1 * 10 + c - '0') % mod1 ;
      h2 = (h2 * 10 + c - '0') % mod2 ;
    }
    hash[i] = {h1 , h2} ;
    ++ mp[{h1 , h2}] ;
  }

  int ans = 0 ;
  for(int i = 0 ; i < n ; ++ i)
    for(int j = 0 ; j < n ; ++ j)
    {
      ULL h1 = hash[i].first * hash[j].first % mod1 ;
```

```
      ULL h2 = hash[i].second * hash[j].second % mod2 ;
      ans += mp[{h1 , h2}] ;
    }
  cout << ans << '\n' ;
  return 0 ;
}
```

https://acm.ecnu.edu.cn/problem/115/

随机化 + 哈希的经典应用例题

引入随机化的目的是为提高概率, 显然区间内所有种类数出现的次数为偶数时异或和为 0 但这并非是充要条件, 引入随机数 + 哈希即是增大这个概率减少冲突。

```
void solve()
{
  int n ;
  cin >> n ;
  map<int , int> mp ;
  map<int , int> cnt ;
  std::vector<int> a(n + 1 , 0);
  int st = 43112609 ;

  for(int i = 1 ; i <= n ; ++ i){
    cin >> a[i] ;
    if ( !mp.count(a[i]) ) {
      st += rand() ;
      mp[a[i]] = st ;
    }
    a[i] = mp[a[i]] ;
  }
  cnt[0] = 1 ;
  int pre = 0 , ans = 0 ;
  for(int i = 1 ; i <= n ; ++ i) {
    pre ^= a[i] ;
    ans += cnt[pre] ;
    cnt[pre]++ ;
  }
  cout << ans << '\n' ;
```

```
26 }
```

3 图论

3.1 树的重心

```cpp
//acwing846
#include <bits/stdc++.h>

using namespace std ;

const int N = 1e5 + 10 ;
const int inf = 0x3f3f3f3f ;

int n , sz[N] , barycenter , tot = inf;
std::vector<int> edges[N] ;

void dfs(int v,int f)
{
    sz[v] = 1 ;

    int mx = 0 ;
    for(auto u : edges[v])
    {
        if ( u != f ) {
            dfs(u , v) ;
            sz[v] += sz[u] ;
            mx = max(mx , sz[u]) ;
        }
    }
    mx = max(mx , n - sz[v]) ;
    if ( mx < tot ) {
        tot = mx ;
        barycenter = v ;
    }
}

int main()
{
```

```
    cin.tie(nullptr)->ios::sync_with_stdio(false) ;
    cin >> n ;
    for(int i = 1 ; i < n ; ++ i)
    {
        int u , v ;
        cin >> u >> v ;
        edges[u].push_back(v) ;
        edges[v].push_back(u) ;
    }
    dfs(1 , 0) ;
    cout << tot << '\n' ;
    return 0 ;
}
```

3.2 拓扑排序

```
//acwing848
#include <bits/stdc++.h>

using namespace std ;

const int N = 1e5 + 10 ;

int n , m , idg[N] ;
std::vector<int> edges[N] ;
std::bitset<N> st;
int main()
{
    cin.tie(nullptr)->ios::sync_with_stdio(false) ;

    cin >> n >> m ;
    for(int i = 1 ; i <= m ; ++ i)
    {
        int u , v ;
        cin >> u >> v ;
        edges[u].push_back(v) ;
        ++idg[v] ;
```

```
    }

    queue<int> que ;
    vector<int> ans ;
    for(int i = 1 ; i <= n ; ++ i) if ( !idg[i] ) {
        que.push(i) ;
        st[i] = 1 ;
        ans.push_back(i) ;
    }
    while ( !que.empty() ) {
        auto v = que.front() ;
        que.pop() ;
        for(auto u : edges[v])
        {
            -- idg[u] ;
            if ( !idg[u] ) {
                ans.push_back(u) ;
                que.push(u) ;
                st[u] = 1 ;
            }
        }
    }
    if ( st.count() == n ) {
        for(auto p : ans) {
            cout << p << ' ' ;
        }
    } else {
        cout << -1 << '\n' ;
    }

    return 0 ;
}
```

3.3 最短路

3.3.1 Dijstra

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

using namespace std ;

const int N = 510 , inf = 0x3f3f3f3f ;

// O(n^2)
int a[N][N] , n , m , d[N] ;
bool st[N] ;

int dijstra(int root)
{
    d[root] = 0 ;
    while ( true )
    {
        int v = -1 ;
        for(int i = 1 ; i <= n ; ++ i)
        {
            if ( !st[i] && (v == -1 || d[v] > d[i]) ) v = i ;
        }
        if ( v == -1 ) break ;
        st[v] = true ;
        for(int i = 1; i <= n ; ++ i)
        {
            d[i] = min(d[i] , d[v] + a[v][i]) ;
        }
    }
    return (d[n] == inf ? -1 : d[n]) ;
}

int main()
{
    cin.tie(nullptr)->ios::sync_with_stdio(false) ;
    cin >> n >> m ;
    memset(a , 0x3f , sizeof a) ;
    memset(d , 0x3f , sizeof d) ;
    for(int i = 1 ; i <= m ; ++ i)
    {
```

```
        int u , v , w ;
        cin >> u >> v >> w ;
        a[u][v] = min(a[u][v] , w) ;
    }
    for(int i = 1 ; i <= n ; ++ i) a[i][i] = 0 ;

    cout << dijstra(1) << '\n' ;

    return 0 ;
}
```

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

using namespace std ;

const int N = 2e5 + 10 , inf = 0x3f3f3f3f ;
// O(MlogN)
int n , m , d[N] ;
std::vector<pair<int , int>> edges[N] ;
int dijstra(int root)
{
    memset(d , 0x3f , sizeof d) ;
    d[root] = 0 ;
    priority_queue<pair<int , int> ,vector<pair<int , int>> ,
       greater<pair<int , int>>> que ;
    que.push({0 , root}) ;
    while ( !que.empty() ) {
        auto v = que.top() ;
        que.pop() ;
        if ( d[v.second] < v.first ) continue ;

        for(auto u : edges[v.second])
        {
            if ( d[u.second] > d[v.second] + u.first )
            {
                d[u.second] = d[v.second] + u.first ;
                que.push({d[u.second] , u.second}) ;
            }
        }
```

```
    }
    return (d[n] == inf ? -1 : d[n]) ;
}
int main()
{
    cin.tie(nullptr)->ios::sync_with_stdio(false) ;
    cin >> n >> m ;
    for(int i = 1 ; i <= m ; ++ i)
    {
        int u , v , w ;
        cin >> u >> v >> w ;
        edges[u].push_back({w , v}) ;
    }
    cout << dijstra(1) << '\n' ;
    return 0 ;
}
```

3.3.2 Floyd

```
// O(n^3)
#include <bits/stdc++.h>

using namespace std ;


const int N = 210 , inf = 0x3f3f3f3f ;

int n , m , q , d[N][N] ;

int main() {
    cin.tie(nullptr)->ios::sync_with_stdio(false) ;

    cin >> n >> m >> q ;
    memset(d , 0x3f , sizeof d) ;
    for(int i = 1 ; i <= m ; ++ i)
    {
        int u , v , w ;
        cin >> u >> v >> w ;
```

```
        d[u][v] = min(d[u][v] , w) ;
    }
    for(int i = 1 ; i <= n ; ++ i) d[i][i] = 0 ;

    for(int k = 1 ; k <= n ; ++ k)
        for(int i = 1 ; i <= n ; ++ i)
            for(int j = 1 ; j <= n ; ++ j)
                d[i][j] = min(d[i][j] , d[i][k] + d[k][j]) ;
    while ( q-- )
    {
        int l , r;
        cin >> l >> r ;
        if ( d[l][r] >= inf / 2 ) {
            cout << "impossible\n" ;
        } else {
            cout << d[l][r] << '\n' ;
        }
    }
    return 0 ;
}
```

3.4 最小生成树

3.4.1 Prim

```
// 朴素O(n^2)
#include <bits/stdc++.h>

using namespace std ;

const int N = 510 ;
const int inf = 0x3f3f3f3f ;
int a[N][N] , n , m , d[N] ;
bitset<N> st ;

int Prim(int root)
{
```

```
13      d[root] = 0 ;
14      int ans = 0 ;
15      while (true) {
16          int v = -1 ;
17          for(int i = 1 ; i <= n ; ++ i) {
18              if ( !st[i] && (v == -1 || d[v] > d[i] )) v = i ;
19          }
20          if (v == -1 || d[v] == inf ) break ;
21          st[v] = 1 ;
22          ans += d[v] ;
23          for(int i = 1 ; i <= n ; ++ i)
24          {
25              d[i] = min(d[i] , a[v][i]) ;
26          }
27      }
28      if ( st.count() != n ) {
29          cout << "impossible\n" ;
30          exit(0) ;
31      }
32      return ans ;
33  }
34
35  int main()
36  {
37      cin.tie(nullptr)->ios::sync_with_stdio(false) ;
38
39      cin >> n >> m ;
40      memset(a , 0x3f , sizeof a) ;
41      memset(d , 0x3f , sizeof d) ;
42      for(int i = 1 ; i <= m ; ++ i)
43      {
44          int u , v , w ;
45          cin >> u >> v >> w ;
46          a[u][v] = min(a[u][v] , w) ;
47          a[v][u] = min(a[v][u] , w) ;
48      }
49      for(int i = 1 ; i <= n ; ++ i) a[i][i] = 0 ;
50
```

```
    cout << Prim(1) << '\n' ;

    return 0 ;
}
```

```
// 堆优化Prim O(mlogn)
#include <bits/stdc++.h>

using namespace std ;

const int N = 510 ;
const int inf = 0x3f3f3f3f ;
int n , m , d[N] ;
bitset<N> st ;
std::vector<pair<int , int>> edges[N] ;

int Prim(int root) {
    priority_queue<pair<int , int> , vector<pair<int , int>> ,
        greater<pair<int , int>>> que ;
    que.push({0 , root}) ;
    d[root] = 0 ;
    int ans = 0 ;
    while ( !que.empty() ) {
        auto v = que.top() ; que.pop() ;
        if ( st[v.second] ) continue ;
        st[v.second] = 1 ;
        ans += v.first ;
        for(auto u : edges[v.second])
        {
            if ( d[u.second] > u.first ) {
                d[u.second] = u.first ;
                que.push({d[u.second] , u.second}) ;
            }
        }
    }
    if ( st.count() != n ) {
        cout << "impossible\n" ;
        exit(0) ;
    }
```

```

    return ans ;
}

int main()
{
    cin.tie(nullptr)->ios::sync_with_stdio(false) ;

    cin >> n >> m ;
    memset(d , 0x3f , sizeof d) ;
    for(int i = 1 ; i <= m ; ++ i)
    {
        int u , v , w ;
        cin >> u >> v >> w ;
        edges[u].push_back({w , v}) ;
        edges[v].push_back({w , u}) ;
    }

    cout << Prim(1) << '\n' ;

    return 0 ;
}
```

3.4.2 Kruskal

```
// 并查集的方式求最小生成树 O(mlogn)
#include <bits/stdc++.h>
#define all(x) x.begin(),x.end()
using namespace std ;

const int N = 1e5 + 10;

int f[N] , sz[N] , n , m ;
inline int find(int x) { return f[x] == x ? x : f[x] = find(f[
   x]) ; }
inline void merge(int a,int b) {
    int fa = find(a) , fb = find(b) ;
    if ( fa != fb ) {
```

```
13          f[fb] = fa;
14          sz[fa] += sz[fb] ;
15      }
16  }
17  bool same(int a,int b) { return find(a) == find(b) ; }
18  int main()
19  {
20      cin.tie(nullptr)->ios::sync_with_stdio(false) ;
21
22      cin >> n >> m ;
23      std::vector<std::array<int , 3>> a(m) ;
24      for(int i = 1 ; i <= n ; ++ i) f[i] = i , sz[i] = 1 ;
25
26      for(auto & p : a)
27      {
28          for(int i = 0 ; i < 3 ; ++ i) cin >> p[i] ;
29      }
30      sort(all(a) , [&](const array<int , 3> &x , const array<
            int , 3> & y) {
31          return x[2] < y[2] ;
32      }) ;
33
34      int ans = 0 , mx = 0 ;
35      for(auto p : a) {
36          if ( !same(p[0] , p[1]) ) {
37              ans += p[2] ;
38              merge(p[0] , p[1]) ;
39              mx = max({sz[f[p[0]]] , sz[f[p[1]]] , mx}) ;
40          }
41      }
42      if ( mx != n ) {
43          cout << "impossible\n" ;
44      } else {
45          cout << ans << '\n' ;
46      }
47      return 0 ;
48  }
```

3.5 二分图

3.5.1 染色法判定

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

using namespace std ;

const int N = 1e5 + 10 ;

int n , m , color[N] ;
std::vector<int> edges[N] ;

bool dfs(int v,int c)
{
    color[v] = c ;
    for(auto u : edges[v])
    {
        if ( !color[u] ) {
            if ( !dfs(u , 3 - c) ) return false ;
        } else {
            if ( color[u] == color[v] ) return false ;
        }
    }
    return true ;
}

int main(){
    cin.tie(nullptr)->ios::sync_with_stdio(false) ;

    cin >> n >> m ;
    for(int i = 1 ; i <= m ; ++ i)
    {
        int u , v ;
        cin >> u >> v ;
        edges[u].push_back(v) ;
        edges[v].push_back(u) ;
    }
```

```
35
36      for(int i = 1 ; i <= n ; ++ i)
37          if ( !color[i] && !dfs(i , 1) )
38              return cout << "No\n" , 0 ;
39      cout << "Yes\n";
40      return 0 ;
41  }
```

3.5.2 匈牙利算法

```
1   #include <bits/stdc++.h>
2
3   using namespace std ;
4
5   const int N = 510 ;
6   // O(nm)
7   int n1 , n2 , m , wife[N] ;
8   std::vector<int> edges[N] ;
9   bitset<N> st ;
10
11  bool find(int v) {
12      for(auto u : edges[v])
13      {
14          if ( !st[u] ) {
15              st[u] = 1 ;
16              if ( !wife[u] || find(wife[u]) ) { // 查询当前有感
                    觉的对象是否单身 ?
17                                                 // 若非单身则查
                                                      询对象是否还
                                                      有备胎可选
18                  wife[u] = v ;
19                  return true ;
20              }
21          }
22      }
23      return false ;
24  }
25
```

```cpp
int main()
{
    cin.tie(nullptr)->ios::sync_with_stdio(false) ;

    cin >> n1 >> n2 >> m ;
    for(int i = 1 ; i <= m ; ++ i)
    {
        int u , v ;
        cin >> u >> v ;
        edges[u].push_back(v) ;
    }

    int ans = 0 ;
    for(int i = 1 ; i <= n1 ; ++ i) {
        st.reset() ;
        if ( find(i) ) ++ ans;
    }
    cout << ans << '\n' ;

    return 0 ;
}
```

3.6 LCA

```cpp
// dfn序 + st表求LCA  O(nlogn + m)
#include <bits/stdc++.h>

using i64 = long long ;
const int N = 5e5 + 10 ;

int n , q , root ;
int st[21][N] ;
int dfn[N] , tot ;
std::vector<int> edges[N] ;
inline int get(int u,int v) { return dfn[u] < dfn[v] ? u : v
    ;}
inline void dfs(int v,int f)
```

```
13 {
14   st[0][dfn[v] = ++tot] = f ;
15   for(auto u : edges[v]) if ( u != f ) dfs(u , v) ;
16 }
17 inline int lca(int u,int v) {
18   if ( u == v ) return u ;
19   if ( (u = dfn[u]) > (v = dfn[v]) ) std::swap(u , v) ;
20   int d = std::__lg(v - u++) ;
21   return get(st[d][u] , st[d][v - (1 << d) + 1]) ;
22 }
23
24 void solve() {
25     std::cin >> n >> q >> root ;
26
27     for(int i = 1 , u , v ; i < n ; ++ i)
28     {
29       std::cin >> u >> v ;
30       edges[u].push_back(v) ;
31       edges[v].push_back(u) ;
32     }
33
34     dfs(root , 0) ;
35
36     for(int i = 1 ; i <= std::__lg(n) ; ++ i)
37       for(int j = 1 ; j + (1 << i) - 1 <= n ; ++ j)
38         st[i][j] = get(st[i - 1][j] , st[i - 1][j + (1 << (i -
               1))]) ;
39     for(int i = 1 , u , v ; i <= q ; ++ i)
40     {
41       std::cin >> u >> v ;
42       std::cout << lca(u , v) << '\n' ;
43     }
44 }
45
46 int main()
47 {
48     std::cin.tie(nullptr)->std::ios::sync_with_stdio(false) ;
49
```

```
    int t = 1 ;
    // std::cin >> t ;
    while ( t-- ) {
        solve() ;
    }

    return 0 ;
}
```

```
// tarjan算法求LCA (离线) O(n + m)

#include <bits/stdc++.h>

using i64 = long long ;

const int N = 5e5 + 10 ;
std::pair<int , int> query_e[N << 1] ;
int e[N << 1] , ne[N << 1] , h[N << 1] , idx ;
int query_ne[N << 1] , query_h[N << 1] , query_idx ;
inline void add(int a,int b) {
    e[idx] = b , ne[idx] = h[a] , h[a] = idx++ ;
}
inline void query_add(int a,int b,int num) {
    query_e[query_idx] = {b , num} , query_ne[query_idx] =
       query_h[a] , query_h[a] = query_idx++ ;
}
int n , q , root , f[N] , ans[N] , vis[N] ;
inline int find_set(int x) {
    return f[x] == x ? x : f[x] = find_set(f[x]) ;
}
inline void tarjan(int v)
{
    vis[v] = 1 ;
    for(int i = h[v] ; ~i ; i = ne[i])
    {
        int u = e[i] ;
        if ( !vis[u] )
        {
            tarjan(u) ;
```

```
30              f[u] = v ;
31          }
32      }
33
34      for(int i = query_h[v] ; ~i ; i = query_ne[i])
35      {
36          int u = query_e[i].first , id = query_e[i].second ;
37          if ( vis[u] )
38          {
39              ans[id] = find_set(u) ;
40          }
41      }
42  }
43  void solve() {
44      std::cin >> n >> q >> root ;
45      std::memset(h , -1 , sizeof h) ;
46      std::memset(query_h , -1 , sizeof query_h) ;
47      for(int i = 1 , u , v ; i < n ; ++ i)
48      {
49          f[i] = i ;
50          std::cin >> u >> v ;
51          add(u , v) , add(v , u) ;
52      }
53      f[n] = n ;
54      for(int i = 1 , u , v ; i <= q ; ++ i)
55      {
56          std::cin >> u >> v ;
57          query_add(u , v , i) , query_add(v , u , i) ;
58      }
59
60      tarjan(root) ;
61
62      for(int i = 1 ; i <= q ; ++ i)
63      {
64          std::cout << ans[i] << '\n' ;
65      }
66  }
67  int main()
```

```
{
    std::cin.tie(nullptr)->std::ios::sync_with_stdio(false) ;

    int t = 1 ;
    // std::cin >> t ;
    while ( t-- ) {
        solve() ;
    }

    return 0 ;
}
```

```
// 倍增法求LCA O((n + m)logn)

#include <bits/stdc++.h>

using i64 = long long ;

const int N = 5e5 + 10 ;

std::vector<int> edges[N] ;
int n , q , root , depth[N] ;
int fa[21][N] ;
void bfs()
{
    std::memset(depth , 0x3f , sizeof depth) ;
    std::queue<int> que ;
    que.push(root) ;
    depth[root] = 1 , depth[0] = 0 ;
    while ( !que.empty() )
    {
        auto v = que.front() ; que.pop() ;

        for(auto u : edges[v])
        {
            if ( depth[u] > depth[v] + 1 )
            {
                depth[u] = depth[v] + 1 ;
                fa[0][u] = v ;
```

```
28                  que.push(u) ;
29                  for(int i = 1 ; i <= 20 ; ++ i)
30                  {
31                      fa[i][u] = fa[i - 1][fa[i - 1][u]] ;
32                  }
33              }
34          }
35      }
36  }
37  inline int lca(int a, int b) {
38      if ( depth[a] < depth[b] ) std::swap(a , b) ;
39      for(int i = 20 ; i >= 0 ; -- i)
40          if ( depth[fa[i][a]] >= depth[b] )
41              a = fa[i][a] ;
42      if ( a == b ) return a ;
43      for(int i = 20 ; i >= 0 ; -- i)
44          if ( fa[i][a] != fa[i][b] )
45              a = fa[i][a] , b = fa[i][b] ;
46      return fa[0][a] ;
47  }
48
49  void solve() {
50      std::cin >> n >> q >> root ;
51
52      for(int i = 1 , u , v ; i < n ; ++ i)
53      {
54          std::cin >> u >> v ;
55          edges[u].push_back(v) ;
56          edges[v].push_back(u) ;
57      }
58
59      bfs() ;
60
61      for(int i = 1 , u , v ; i <= q ; ++ i)
62      {
63          std::cin >> u >> v ;
64          std::cout << lca(u , v) << '\n' ;
65      }
```

```
}

int main()
{
    std::cin.tie(nullptr)->std::ios::sync_with_stdio(false) ;

    int t = 1 ;
    // std::cin >> t ;
    while ( t-- ) {
        solve() ;
    }

    return 0 ;
}
```

3.6.1 次小生成树

LCA 经典综合应用, 求 (严格) 次小生成树

定义: $d1_{i,j}$ 表示从 j 节点跳跃 2^i 步之间经过的最大边权, $d2_{i,j}$ 从 j 节点跳跃 2^i 步之间经过的严格次大边权。

步骤:

- Kruskal 构建最小生成树
- 预处理出 d1[i][j],d2[i][j] 数组
- 枚举非树边添加进去 (一定形成环) 通过比较环上最大边和次大边与添加的边权相比。

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

using i64 = long long ;

const int N = 1e5 + 10 , M = 3e5 + 10 , INF = 0x3f3f3f3f ;

std::pair<int , int> e[M] ;
int ne[M] , h[N] , idx ;
inline void add(int a,int b, int w) {
```

```
    e[idx] = {b , w} , ne[idx] = h[a] , h[a] = idx++ ;
}
struct Edge
{
    int u , v , w ;
    bool used ;
    bool operator < (const Edge&x) const {
        return w < x.w ;
    }
} edges[M] ;

int n , m , f[N] , depth[N] ;
int fa[21][N] , d1[21][N] , d2[21][N] ;
inline int find_set(int x) { return f[x] == x ? x : f[x] =
   find_set(f[x]) ; }

i64 Kurskal()
{
    i64 ans = 0 ;
    for(int i = 1 ; i <= n ; ++ i) f[i] = i ;
    std::sort(edges + 1 , edges + 1 + m) ;
    for(int i = 1 ; i <= m ; ++ i)
    {
        if ( find_set(edges[i].v) != find_set(edges[i].u) )
        {
            f[f[edges[i].v]] = f[edges[i].u] ;
            ans += edges[i].w ;
            edges[i].used = true ;
        }
    }
    return ans ;
}

void build()
{
    std::memset(h , -1 , sizeof h) ;
    for(int i = 1 ; i <= m ; ++ i)
    {
```

```
47          if (edges[i].used)
48          {
49              add(edges[i].u , edges[i].v , edges[i].w) ;
50              add(edges[i].v , edges[i].u , edges[i].w) ;
51          }
52      }
53  }
54
55  void bfs()
56  {
57      std::memset(depth , 0x3f , sizeof depth) ;
58      depth[1] = 1 , depth[0] = 0 ;
59      std::queue<int> que ;
60      que.push(1) ;
61      while ( !que.empty() ) {
62          auto v = que.front() ; que.pop() ;
63
64          for(int i = h[v] ; ~i ; i = ne[i])
65          {
66              int u = e[i].first , w = e[i].second ;
67              if ( depth[u] > depth[v] + 1 )
68              {
69                  depth[u] = depth[v] + 1 ;
70                  que.push(u) ;
71                  fa[0][u] = v ;
72                  d1[0][u] = w ;
73                  d2[0][u] = -INF;
74                  for(int j = 1 ; j <= 20 ; ++ j)
75                  {
76                      int anc = fa[j - 1][u] ;
77                      fa[j][u] = fa[j - 1][anc] ;
78                      int distance[4] = {d1[j - 1][u] , d2[j -
                            1][u] , d1[j - 1][anc] , d2[j - 1][anc]}
                             ;
79                      d1[j][u] = d2[j][u] = -INF ;
80                      for(int k = 0 ; k < 4 ; ++ k)
81                      {
82                          int d = distance[k] ;
```

```
83                          if ( d > d1[j][u] )
84                          {
85                              d2[j][u] = d1[j][u] ;
86                              d1[j][u] = d ;
87                          } else if ( d != d1[j][u] && d > d2[j
                                ][u] ) {
88                              d2[j][u] = d ;
89                          }
90                      }
91                  }
92              }
93          }
94      }
95  }
96
97  int lca(int a,int b,int w) {
98      std::vector<int> distance ;
99      if ( depth[a] < depth[b] ) std::swap(a , b) ;
100     for(int i = 20 ; i >= 0 ; -- i)
101         if ( depth[fa[i][a]] >= depth[b] )
102         {
103             distance.push_back(d1[i][a]) ;
104             distance.push_back(d2[i][a]) ;
105             a = fa[i][a] ;
106         }
107     if ( a != b )
108     {
109         for(int i = 20 ; i >= 0 ; -- i)
110             if ( fa[i][a] != fa[i][b] )
111             {
112                 distance.push_back(d1[i][a]) ;
113                 distance.push_back(d2[i][a]) ;
114                 distance.push_back(d1[i][b]) ;
115                 distance.push_back(d2[i][b]) ;
116                 a = fa[i][a] , b = fa[i][b] ;
117             }
118         distance.push_back(d1[0][a]) ;
119         distance.push_back(d1[0][b]) ;
```

```
120     }
121
122     int dist1 = -INF , dist2 = -INF ;
123     for(auto p : distance)
124     {
125         if ( p > dist1 ) {
126             dist2 = dist1 ;
127             dist1 = p ;
128         } else if ( p != dist1 && p > dist2 ) {
129             dist2 = p ;
130         }
131     }
132     if ( w > dist1 ) {
133         return w - dist1 ;
134     } else if ( w > dist2 ) {
135         return w - dist2 ;
136     }
137     return INF ;
138 }
139
140 void solve() {
141     std::cin >> n >> m ;
142     for(int i = 1 ; i <= m ; ++ i)
143     {
144         std::cin >> edges[i].u >> edges[i].v >> edges[i].w ;
145         edges[i].used = false ;
146     }
147
148     i64 sum = Kurskal() ; // 最小生成树
149
150     build() ;
151
152     bfs() ;
153
154     i64 ans = 1e18 ;
155
156     for(int i = 1 ; i <= m ; ++ i)
157     {
```

```
158         if ( !edges[i].used )
159         {
160             ans = std::min(ans , lca(edges[i].u , edges[i].v ,
                    edges[i].w) + sum ) ;
161         }
162     }
163
164     std::cout << ans << '\n' ;
165 }
166
167 int main()
168 {
169     std::cin.tie(nullptr)->std::ios::sync_with_stdio(false) ;
170
171     int t = 1 ;
172     // std::cin >> t ;
173     while ( t-- ) {
174         solve() ;
175     }
176
177     return 0 ;
178 }
```

3.7 应用

```
1  // https://codeforces.com/contest/20/problem/C 求出最短路径问
       题
2  #pragma GCC optimize(3)
3  #pragma GCC optimize("Ofast,no-stack-protector")
4  #include <bits/stdc++.h>
5  #define int long long
6  #define all(x) x.begin(),x.end()
7
8  using namespace std ;
9  typedef pair<int , int> PII ;
10 const int inf = 0x3f3f3f3f ;
11 const int LLinf = 0x3f3f3f3f3f3f3f3f ;
```

```
12 const int N = 2e5 + 10 ;
13
14 int n , m ;
15 std::vector<PII> edges[N] ;
16 int d[N] , pre[N] ;
17
18 void dijstra(int root)
19 {
20   memset(d , 0x3f , sizeof d) ;
21   memset(pre , 0x3f , sizeof pre) ;
22   priority_queue<PII,vector<PII>,greater<PII>> que ;
23   d[root] = 0 ;
24   que.push({0 , root}) ;
25   while ( !que.empty() ) {
26     auto v = que.top() ; que.pop() ;
27     if ( d[v.second] < v.first ) continue ;
28
29     for(auto u : edges[v.second])
30     {
31       if ( d[u.second] > d[v.second] + u.first ) {
32         d[u.second] = d[v.second] + u.first ;
33         que.push({d[u.second] , u.second}) ;
34         pre[u.second] = v.second ;
35       }
36     }
37   }
38   if ( d[n] == LLinf ) {
39     cout << -1 << '\n' ;
40     exit(0) ;
41   }
42 }
43
44 void solve()
45 {
46   cin >> n >> m ;
47   for(int i = 1 ; i <= m ; ++ i)
48   {
49     int u , v , w ;
```

```
    cin >> u >> v >> w ;
    edges[u].push_back({w , v}) ;
    edges[v].push_back({w , u}) ;
  }

  dijstra(1) ;

  vector<int> ans(1 , n) ;
  int p = n ;
  do {
    p = pre[p] ;
    ans.push_back(p) ;
  } while ( p != 1 ) ;
  for(int i = (int)ans.size() - 1 ; i >= 0 ; -- i)
  {
    cout << ans[i] << ' ' ;
  }
  cout << '\n' ;
}

signed main()
{
  cin.tie(nullptr)->ios::sync_with_stdio(false) ;

  int t = 1 ;
  // cin >> t ;

  for(int i = 1 ; i <= t ; i++)
  {
    // cout << "case #" << i << " : " << '\n' ;
    solve() ;
  }
  return 0 ;
}
```

4 数论

4.1 质数

4.1.1 判定

试除法判定质数 $O(\sqrt{n})$

```
bool is_primes(int x) {
  if ( x < 2 ) return false ;
  for(int i = 2 ; i <= x / i ; ++ i) if ( x % i == 0 ) return
     false ;
  return true ;
}
```

4.1.2 分解

朴素 $O(\sqrt{n})$ 若预处理出素数表之后分解 $O(\sqrt{\frac{n}{lnn}})$

```
std::map<int , int> pr ;
for(int i = 2 ; i <= x / i ; ++ i)
{
  while ( x % i == 0 ) {
    ++ pr[i] ;
    x /= i ;
  }
}
if ( x > 1 ) ++ pr[x] ;
```

4.1.3 筛

常用筛法: 埃氏 ($O(nloglogn)$) 线性 ($O(n)$) , 经过测试统计埃氏 + bool 效率会快于线性, 按需使用埃氏 + bool , 线性 + bitset

```
// 埃氏
int primes[N] ;
bool st[N] ;
```

```

void get(int n) {
    for(int i = 2 ; i <= n ; ++ i)
    {
        if ( !st[i] ) {
            primes[++primes[0]] = i ;
            for(i64 j = (i64) i * i ; j <= n ; j += i)
            {
                st[j] = true ;
            }
        }
    }
}
```

```
// 线性
int primes[N] ;
bitset<N> st ;

void get(int n) {
    for(int i = 2 ; i <= n ; ++ i){
        if ( !st[i] ) {
            primes[++primes[0]] = i ;
        }
        for(int j = 1 ; primes[j] <= n / i ; ++ j)
        {
            st[primes[j] * i] = 1 ;
            if ( i % primes[j] == 0 ) break ;
        }
    }
}
```

4.2 约数

4.2.1 约数个数

算术基本定理: $N = p_1^{a_1} * p_2^{a_2} * ... * p_k^{a_k}$

约数个数: $S = (p_1 + 1) * (p_2 + 1) * ... * (p_k + 1)$

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

using namespace std ;
using i64 = long long ;

const int mod = 1e9 + 7 ;

int main()
{
    cin.tie(nullptr)->ios::sync_with_stdio(false) ;

    map<int , int> pr ;
    int n ;
    cin >> n ;

    for(int i = 1 ; i <= n ; ++ i)
    {
        int x ;
        cin >> x ;
        for(int j = 2 ; j <= x / j ; ++ j)
            while ( x % j == 0 )
            {
                ++ pr[j] ;
                x /= j ;
            }
        if ( x > 1 ) ++ pr[x] ;
    }

    i64 ans = 1 ;

    for(auto p : pr)
    {
        ans = (ans * (i64)(p.second + 1)) % mod ;
    }

    cout << ans << '\n' ;
    return 0 ;
}
```

4.2.2 约数之和

约数之和: $S = (p_1^0 + p_1^1 + ... + p_1^{a_1}) * ... * (p_k^0 + p_k^1 + ... + p_k^{a_k})$

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

using namespace std ;
using i64 = long long ;

const int mod = 1e9 + 7 ;

int main()
{
    cin.tie(nullptr)->ios::sync_with_stdio(false) ;

    map<int , int> pr ;
    int n ;
    cin >> n ;

    for(int i = 1 ; i <= n ; ++ i)
    {
        int x ;
        cin >> x ;
        for(int j = 2 ; j <= x / j ; ++ j)
            while ( x % j == 0 )
            {
                ++ pr[j] ;
                x /= j ;
            }
        if ( x > 1 ) ++ pr[x] ;
    }

    i64 ans = 1 ;

    for(auto p : pr)
    {
        i64 t = 1 ;
        for(int i = 1 ; i <= p.second ; ++ i) {
            t = t * p.first + 1 ;
```

```
            t %= mod ;
        }
        ans = (ans * t) % mod ;
    }

    cout << ans << '\n' ;
    return 0 ;
}
```

4.3 欧拉函数

$$\phi x = N * (1 - \frac{1}{p_1}) * (1 - \frac{1}{p_2}) * ... * (1 - \frac{1}{p_k})$$

```
i64 phi(int x)
{
    i64 ans = x ;
    for(int i = 2 ; i <= x / i ; ++ i)
    {
        if ( x % i == 0 )
        {
            ans = ans * (i - 1) / i ;
            while ( x % i == 0 ) {
                x /= i ;
            }
        }
    }
    if ( x > 1 ) ans = ans * (x - 1) / x ;
    return ans ;
}
```

5 经典题

https://www.luogu.com.cn/problem/solution/P10837

扫描线、前缀和、二分

简要题意：对于每一个凋零玫瑰在 t_i 时刻会绽放延续 m 时刻，询问若此时可以修改 1 个凋零玫瑰的绽放时刻请问最多有多少时刻有且仅有 1 个凋零玫瑰绽放。

思路：显然我们需要知道一个关键信息，**改变一个凋零玫瑰绽放时刻对答案的影响会是什么？**设修改的凋零玫瑰绽放时刻区间为 $[t_l, t_r]$，若在此区间内包含某些时刻有且仅有 1 个凋零玫瑰绽放则对答案贡献是 -1，若有 2 个凋零玫瑰绽放则对答案贡献是 1，知道这个后只需要预处理出所有包含 1 个和 2 个凋零玫瑰绽放的时刻区间后枚举修改的凋零玫瑰通过**前缀和**、**二分**的手段计算求得最大即可。预处理可以通过**扫描线**的思想求得。

时间复杂度 $O(nlogn)$。

```cpp
#include <bits/stdc++.h>
#define int long long

void solve() {
  int n , m ;
  std::cin >> n >> m ;
  std::map<int , int> a;
  std::vector<int> arr(n + 1 , 0);
  for(int i = 1 ; i <= n ; ++ i)
  {
    std::cin >> arr[i] ;
    a[arr[i]] ++ ;
    a[arr[i] + m] -- ;
  }

  int cnt = 0 , x1 , x2 , ok1 = 0 , ok2 = 0 ;
  // 通过扫描线的思想预处理
  std::vector<std::pair<int , int>> b1 , b2;
```

```
19
20  for(auto &p : a)
21  {
22    cnt += p.second ;
23    if ( !p.second ) {
24      if ( ok1 ) {
25        b1.push_back({x1 , p.first - 1}) ;
26        x1 = p.first ;
27      }
28      if ( ok2 ) {
29        b2.push_back({x2 , p.first - 1}) ;
30        x2 = p.first ;
31      }
32    } else if ( cnt == 1 ) {
33      if ( !ok1 ) {
34        x1 = p.first ;
35      }
36      if ( ok2 ) {
37        b2.push_back({x2 , p.first - 1}) ;
38        ok2 = 0 ;
39      }
40      ok1 = 1 ;
41    } else if ( cnt == 2 ) {
42      if ( !ok2 ) {
43        x2 = p.first ;
44      }
45      if ( ok1 ) {
46        b1.push_back({x1 , p.first - 1}) ;
47        ok1 = 0 ;
48      }
49      ok2 = 1 ;
50    } else {
51      if ( ok1 ) {
52        b1.push_back({x1 , p.first - 1}) ;
53        ok1 = 0;
54      }
55      if ( ok2 ) {
56        b2.push_back({x2 , p.first - 1}) ;
```

```
57          ok2 = 0 ;
58        }
59      }
60    }
61    // 预处理前缀和
62    std::vector<int> s1((int)b1.size() + 1 , 0) , s2((int)b2.
         size() + 1 , 0);
63    for(int i = 0 ; i < (int)b1.size() ; ++ i)
64    {
65      s1[i + 1] = s1[i] + b1[i].second - b1[i].first + 1 ;
66    }
67    for(int i = 0 ; i < (int)b2.size() ; ++ i)
68    {
69      s2[i + 1] = s2[i] + b2[i].second - b2[i].first + 1 ;
70    }
71
72    int ans = 0 ;
73    // 枚举
74    for(int i = 1 ; i <= n ; ++ i)
75    {
76      int x = arr[i] , y = arr[i] + m - 1 ; // 影响时刻区间[x ,
           y]
77
78      int l = 0 , r = b1.size() - 1 , lo = -1 , hg = -1 , r1 = 0
           , r2 = 0 ;
79      while ( r >= l ) {
80        int mid = ( l + r ) >> 1 ;
81        if ( b1[mid].first >= x ) {
82          r = mid - 1 ;
83          lo = mid ;
84        } else {
85          l = mid + 1 ;
86        }
87      }
88      l = 0 , r = b1.size() - 1 ;
89      while ( r >= l ) {
90        int mid = (l + r) >> 1 ;
91        if ( b1[mid].second <= y ) {
```

```
92          l = mid + 1 ;
93          hg = mid ;
94        } else {
95          r = mid - 1 ;
96        }
97      }
98      if ( lo != -1 ) {
99        r1 = s1[hg + 1] - s1[lo] ;
100     }
101     l = 0 , r = b2.size() - 1 , lo = -1 , hg = -1 ;
102     while ( r >= l ) {
103       int mid = ( l + r ) >> 1 ;
104       if ( b2[mid].first >= x ) {
105         r = mid - 1 ;
106         lo = mid ;
107       } else {
108         l = mid + 1 ;
109       }
110     }
111     l = 0 , r = b2.size() - 1 ;
112     while ( r >= l ) {
113       int mid = (l + r) >> 1 ;
114       if ( b2[mid].second <= y ) {
115         l = mid + 1 ;
116         hg = mid ;
117       } else {
118         r = mid - 1 ;
119       }
120     }
121     if ( lo != -1 ) {
122       r2 = s2[hg + 1] - s2[lo] ;
123     }
124     ans = std::max(ans , s1[b1.size()] - r1 + r2 + m) ;
125   }
126   std::cout << ans << '\n' ;
127 }
128
129 signed main()
```

```
130 {
131     std::cin.tie(nullptr)->std::ios::sync_with_stdio(false) ;
132 
133     int t = 1 ;
134     // std::cin >> t ;
135     while ( t-- ) {
136         solve() ;
137     }
138 
139     return 0 ;
140 }
```

6 科技积累

6.1 随机化

```cpp

unsigned int generateSeedFromTimestamp() {
  auto now = std::chrono::system_clock::now(); // 获取当前时间点
  auto timestamp = std::chrono::duration_cast<std::chrono::milliseconds>(now.time_since_epoch()); // 转换为毫秒级的时间戳

  return static_cast<unsigned int>(timestamp.count()); // 将时间戳转换为整数种子值
}

int main()
{
  unsigned int seed = generateSeedFromTimestamp(); // 生成种子

  std::mt19937 gen(seed) ; // 使用种子值初始化伪随机数生成器

  uniform_int_distribution<> rd(l , r) ; // [l , r] 闭区间

  cout << rd(gen) << '\n' ;

  return 0 ;
}
```