Contents

1 Basic

1.1 vimrc

```
"close chinese typing"
set nu rnu hls ts=4 sw=4 ai cin mouse=a
inoremap {<CR> {<CR>}<Esc>O
inoremap jk <Esc>
autocmd filetype cpp nnoremap <F9> :w <bar> !g++ % -o
    %:r -std=c++14 -O2 -Wfatal-errors<CR>
autocmd filetype cpp nnoremap <F10> :w <bar> !g++ % -o
    %:r -std=c++14 -O2 -Wfatal-errors && time ./%:r<CR>
autocmd filetype cpp nnoremap <F12> :!clear<CR>
```

1.2 Python stress

```
# needed
ac.cpp
wa.cpp
gen.py
run.sh
# run.sh
set -e # 可以偵測程式 RE 時會停下來
g++ ac.cpp -o ac -std=c++14
g++ wa.cpp -o wa -std=c++14
for ((i=0;;i++))
do
    echo "$i"
    python3 gen.py > input
    ./ac < input > ac.out
    ./wa < input > wa.out
    diff ac.out wa.out || break
done

# cheatsheet
from random import *
n = randint(1, 100) # [1,100]
ch = chr(randint(ord('a'), ord('z')))
# 從集合 s 選出 4 個不同的元素
choiceSet = sample(s, 4)
# 從整數 1~n 選出 4 個不同的元素
choiceSet = sample(range(1, n+1), 4)
# 把序列 arr 順序打亂 (產生1~n的一組permutation)
shuffle(arr)

# 產生隨機一棵樹
# 每次從比自己小的節點選一個當連接自己的邊
for i in range(2, n+1):
    print(i, randint(1, i-1))
```

1.3 C++ stress

```
#include<iostream>
#include<time.h>
using namespace std;

int r(int min,int max) {
  int a = rand()%10000,b = rand()%10000,c = rand()%10;
  int d=a*100000+b*10+c;
  return d%(max-min+1)+min;
}
long long int llr(long long int min,long long int max)
    {
  long long int a=r(0,1e9-1),b=r(0,1e9-1);
  long long int c=a*1e9+b;
  return c%(max-min+1)+min;
}

int main() {
  freopen("make.txt","w",stdout);

  srand(time(0));

  int N=r(1,20000),M=r(1,20000);
  cout<<N<<" "<<M<<endl;
  while(M--) {
    cout<<(r(0,1)?"a":"b")<<" ";
    cout<<r(1,N)<<" ";
    cout<<r(1,N)<<" ";
    cout<<r(1,1000000)<<endl;
  }

  return 0;
}

#include<iostream>
#include<stdlib.h>
using namespace std;
int main() {

  int count=1;
  while(true) {
    system("生測資.exe");
    system("要測的.exe < make.txt > A.txt");
    system("暴力.exe < make.txt > B.txt");

    if(system("fc A.txt B.txt")) break;
    cout<<count++<<endl;
  }

  return 0;
}
```

2 Graph

2.1 DSU

```
struct DSU {
  static const int MAXN = 5005;
  int fa[MAXN];
  void init() {
    fill(fa, fa+MAXN, -1);
  }
  int Find(int x) {
    return ((fa[x] < 0) ? x : fa[x] = Find(fa[x])); //
        路徑壓縮
  }
  void U(int x, int y) {
    x = Find(x), y = Find(y);
    if (x == y) return;
    if (abs(fa[x]) > abs(fa[y])){ // 啟發式合併
      fa[x] += fa[y]; //size
      fa[y] = x; //boss
    }else{
      fa[y] += fa[x];
      fa[x] = y;
    }
    return;
  }
} dsu;
```

2.2 LCA

```
int dep[N], Pa[N][lgN], n, root;
void dfs(int now, int d, int pa){
  dep[now] = d;
  Pa[now][0] = pa; // important
  for (auto i : tree[now]){
    if (i != pa) dfs(i, d+1, now);
  }
}
void computePa(){
  for (int i = 0; i < lgN; i++){
    for (int k = 1; k <= n; k++){
      if (Pa[k][i] == -1) Pa[k][i+1] = -1;
      else Pa[k][i+1] = Pa[Pa[k][i]][i];
    }
  }
}
void lca_init(){
  dfs(root, 0, -1);
  computePa();
}
int lca(int a, int b){
  if (dep[b] > dep[a]) swap(a, b);
  int s = dep[a] - dep[b];
  for (int i = 0; i < lgN; i++){
    if ((s>>i)&1){
      a = Pa[a][i];
    }
  }
  if (a==b) return a;
  for (int i = lgN; i >= 0; i--){
    if (Pa[a][i] != Pa[b][i]){
      a = Pa[a][i];
      b = Pa[b][i];
    }
  }
  return Pa[a][0];
}
int dis(int a, int b){
  int c = lca(a, b);
  return dep[a]+dep[b]-(dep[c]<<1);
}
```

3 Flow

3.1 Dinic

```
// 如果是二分圖 O(ElgV), V, E <= 2e5 is ok
// O(V^2E)
// becareful long long !
#define SZ(x) (int)x.size()
struct Dinic{ // 0 base
  struct Edge{ int v,f,re; };
  int n,s,t,level[MXN];
  vector<Edge> E[MXN];
  void init(int _n, int _s, int _t){
    n = _n; s = _s; t = _t;
    for (int i=0; i<n; i++) E[i].clear();
  }
  void add_edge(int u, int v, int f){
    E[u].PB({v,f,SZ(E[v])});
    E[v].PB({u,0,SZ(E[u])-1});
  }
  bool BFS(){
    for (int i=0; i<n; i++) level[i] = -1;
    queue<int> que;
    que.push(s);
    level[s] = 0;
    while (!que.empty()){
      int u = que.front(); que.pop();
      for (auto it : E[u]){
        if (it.f > 0 && level[it.v] == -1){
          level[it.v] = level[u]+1;
          que.push(it.v);
    } } }
    return level[t] != -1;
  }
  int DFS(int u, int nf){
    if (u == t) return nf;
    int res = 0;
    for (auto &it : E[u]){
      if (it.f > 0 && level[it.v] == level[u]+1){
        int tf = DFS(it.v, min(nf,it.f));
        res += tf; nf -= tf; it.f -= tf;
        E[it.v][it.re].f += tf;
        if (nf == 0) return res;
    } }
    if (!res) level[u] = -1;
    return res;
  }
  int flow(int res=0){
    while ( BFS() )
      res += DFS(s,2147483647);
    return res;
} }flow;
```

3.2 MinCostFlow

```
// O(V^2E^2) no negative cycle
struct zkwflow{
  static const int maxN=10000;
  struct Edge{ int v,f,re; ll w;};
  int n,s,t,ptr[maxN]; bool vis[maxN]; ll dis[maxN];
  vector<Edge> E[maxN];
  void init(int _n,int _s,int _t){
    n=_n,s=_s,t=_t;
    for(int i=0;i<n;i++) E[i].clear();
  }
  void addEdge(int u,int v,int f,ll w){
    E[u].push_back({v,f,(int)E[v].size(),w});
    E[v].push_back({u,0,(int)E[u].size()-1,-w});
  }
  bool SPFA(){
    fill_n(dis,n,LLONG_MAX); fill_n(vis,n,false);
    queue<int> q; q.push(s); dis[s]=0;
    while (!q.empty()){
      int u=q.front(); q.pop(); vis[u]=false;
      for(auto &it:E[u]){
        if(it.f>0&&dis[it.v]>dis[u]+it.w){
          dis[it.v]=dis[u]+it.w;
          if(!vis[it.v]){
            vis[it.v]=true; q.push(it.v);
    } } } }
    return dis[t]!=LLONG_MAX;
  }
  int DFS(int u,int nf){
    if(u==t) return nf;
    int res=0; vis[u]=true;
    for(int &i=ptr[u];i<(int)E[u].size();i++){
      auto &it=E[u][i];
      if(it.f>0&&dis[it.v]==dis[u]+it.w&&!vis[it.v]){
        int tf=DFS(it.v,min(nf,it.f));
        res+=tf,nf-=tf,it.f-=tf;
        E[it.v][it.re].f+=tf;
        if(nf==0){ vis[u]=false; break; }
      }
    }
    return res;
  }
  pair<int,ll> flow(){
    int flow=0; ll cost=0;
    while (SPFA()){
      fill_n(ptr,n,0);
      int f=DFS(s,INT_MAX);
      flow+=f; cost+=dis[t]*f;
    }
    return{ flow,cost };
  } // reset: do nothing
} flow;
```

4 Matching

4.1 KM

```
// O(N^3)
#define ll long long
#define MXN
#define INF 1e18

struct KM{ // max weight, for min negate the weights
  int n, mx[MXN], my[MXN], pa[MXN];
  ll g[MXN][MXN], lx[MXN], ly[MXN], sy[MXN];
  bool vx[MXN], vy[MXN];
  void init(int _n) { // 1-based
    n = _n;
    for(int i=1; i<=n; i++) fill(g[i], g[i]+n+1, 0);
  }
  void addEdge(int x, int y, ll w) {g[x][y] = w;}
  void augment(int y) {
    for(int x, z; y; y = z)
      x=pa[y], z=mx[x], my[y]=x, mx[x]=y;
  }
  void bfs(int st) {
    for(int i=1; i<=n; ++i) sy[i]=INF, vx[i]=vy[i]=0;
    queue<int> q; q.push(st);
    for(;;) {
      while(q.size()) {
        int x=q.front(); q.pop(); vx[x]=1;
        for(int y=1; y<=n; ++y) if(!vy[y]){
          ll t = lx[x]+ly[y]-g[x][y];
          if(t==0){
            pa[y]=x;
            if(!my[y]){augment(y);return;}
            vy[y]=1, q.push(my[y]);
          }else if(sy[y]>t) pa[y]=x,sy[y]=t;
      } }
      ll cut = INF;
      for(int y=1; y<=n; ++y)
        if(!vy[y]&&cut>sy[y]) cut=sy[y];
      for(int j=1; j<=n; ++j){
        if(vx[j]) lx[j] -= cut;
        if(vy[j]) ly[j] += cut;
        else sy[j] -= cut;
      }
      for(int y=1; y<=n; ++y) if(!vy[y]&&sy[y]==0){
        if(!my[y]){augment(y);return;}
        vy[y]=1, q.push(my[y]);
  } } }
  ll solve(){
    fill(mx, mx+n+1, 0); fill(my, my+n+1, 0);
    fill(ly, ly+n+1, 0); fill(lx, lx+n+1, -INF);
    for(int x=1; x<=n; ++x) for(int y=1; y<=n; ++y)
      lx[x] = max(lx[x], g[x][y]);
    for(int x=1; x<=n; ++x) bfs(x);
    ll ans = 0;
    for(int y=1; y<=n; ++y) ans += g[my[y]][y];
    return ans;
} }graph;
```

5 Geometry

5.1 ConvexHull

```
int cross(PII a, PII b, PII c){
  return (b.X-a.X)*(c.Y-a.Y)-(b.Y-a.Y)*(c.X-a.X);
}

}
void convex_hull(){
    sort(v.begin(), v.end());
    int top = 0;
    for (int i = 0; i < v.size(); i++){
        while (top >= 2 && cross(hull[top-2], hull[top
            -1], v[i]) <= 0)
            hull.pop_back(), top--;
        hull.emplace_back(v[i]), top++;
    }
    for (int i = v.size()-2, t=top+1; i >= 0; i--){
        while (top >= t && cross(hull[top-2], hull[top
            -1], v[i]) <= 0)
            hull.pop_back(), top--;
        hull.emplace_back(v[i]), top++;
    }
    // hull.pop_back()
}
```

6 Tree

6.1 Trie

```
struct trie{
  trie *nxt[26];
  int ans;
  trie(){
    ans = 0;
    memset(nxt, 0, sizeof nxt);
  }
};

trie *root = new trie();

void insert(string s){
  trie *node = root;
  for (auto i : s){
    if (!node->nxt[i-'a']) node->nxt[i-'a'] = new trie
        ();
    node = node->nxt[i-'a'];
  }
  if (node->ans == 0){
    node->ans = cnt++;
    cout << "New! " << node->ans << "\n";
  }else{
    cout << "Old! " << node->ans << "\n";
  }
}

void erase(trie *&node){
  for (int i = 0; i < 26; i++){
    if (node->nxt[i])
      erase(node->nxt[i]);
  }
  delete node;
}
```

7 Math

7.1 Miller Rabin

```
// n < 4,759,123,141        3 :  2, 7, 61
// n < 1,122,004,669,633    4 :  2, 13, 23, 1662803
// n < 3,474,749,660,383          6 :  pirmes <= 13
// n < 2^64                       7 :
// 2, 325, 9375, 28178, 450775, 9780504, 1795265022
// Make sure testing integer is in range [2, n-2] if
// you want to use magic.
LL magic[]={}
bool witness(LL a,LL n,LL u,int t){
  if(!a) return 0;
  LL x=mypow(a,u,n);
  for(int i=0;i<t;i++) {
    LL nx=mul(x,x,n);
    if(nx==1&&x!=1&&x!=n-1) return 1;
    x=nx;
  }
  return x!=1;
}
bool miller_rabin(LL n) {
  int s=(magic number size)
  // iterate s times of witness on n
  if(n<2) return 0;
  if(!(n&1)) return n == 2;
  ll u=n-1; int t=0;
  // n-1 = u*2^t
  while(!(u&1)) u>>=1, t++;
  while(s--){
    LL a=magic[s]%n;
    if(witness(a,n,u,t)) return 0;
  }
  return 1;
}
```

7.2 Josephus Problem

```
int josephus(int n, int m){ //n人每m次
    int ans = 0;
    for (int i=1; i<=n; ++i)
        ans = (ans + m) % i;
    return ans;
}
```

7.3 Phi

```
ll phi(ll n){   // 計算小於n的數中與n互質的有幾個
    ll res = n, a=n;    // O(sqrtN)
    for(ll i=2;i*i<=a;i++){
        if(a%i==0){
            res = res/i*(i-1);
            while(a%i==0) a/=i;
    }   }
    if(a>1) res = res/a*(a-1);
    return res;
}
```

8 Data Structure

8.1 區間修改線段樹

```
int tree[4*N];
int tag[4*N];
#define MID (l+r)/2
#define lson root*2+1
#define rson root*2+2

void modify(int a,int b,int k,int l,int r,int root){
  tree[root]+=tag[root]*(r-l+1);
  if (l!=r){
    tag[lson]+=tag[root];
    tag[rson]+=tag[root];
  }
  tag[root] = 0;
  if (r<a || l>b) return;
  if (l>=a && r<=b){
    tree[root] += k*(r-l+1);
    if (l!=r){
      tag[lson] += k;
      tag[rson] += k;
    }
  }else{
    int mid = MID;
    modify(a, b, k, l, mid, lson);
    modify(a, b, k, mid+1, r, rson);
    tree[root] = tree[lson]+tree[rson];
  }
}

int Q(int a, int b, int l, int r, int root){
  tree[root]+=tag[root]*(r-l+1);
  if (l!=r){
    tag[lson]+=tag[root];
    tag[rson]+=tag[root];
  }
  tag[root] = 0;
  if (r<a || l>b) return 0;
  if (l>=a && r<=b){
    return tree[root];
  }else{
    int mid = MID;
    int L = Q(a, b, l, mid, lson);
    int R = Q(a, b, mid+1, r, rson);
    return L+R;
  }
}

void build(int l, int r, int root){
  if (l==r){
     tree[root] = arr[l];
     return;
  }
  int mid = MID;
  build(l, mid, lson);
  build(mid+1, r, rson);
  tree[root] = tree[lson]+tree[rson];
}
/*
 * build(0, n-1, 0);
 * modify(a-1, b-1, k, 0, n-1, 0);
 * Q(a-1, b-1, 0, n-1, 0)
 * N < 5e5, Q < 5e5
input:
4
1 2 3 4 // arr = {1, 2, 3, 4}
5
2 1 3
1 1 3 1 // [1,3] + 1
2 1 3
1 1 4 1
2 1 4

output:
```

```
6
9
17
*/
```

8.2 pbdsKth

```
總共有 n 件衣服，各自有不同的價格 (a1, a2, ..., an)。
一開始第 i 件會在編號為 i 的桶子裡，接下來會有 m 筆操作
每次操作有三種選擇：
- A x y ：將編號第 x 的衣服所在桶子裡面所有的衣服，倒到第 y 件衣服所在的桶
子 (1 ≤ x, y ≤ n)
- M x y ：第 x 件衣服的售價改為 y(1 ≤ x ≤ n, 1 ≤ y ≤ 10^9)
- Q x y ：查詢第 x 件衣服所在的桶子，裡面第 y 大的售價 (1 ≤ x ≤ n)

1 ≤ n ≤ 10^5
1 ≤ m ≤ 4 * 10^5

5 4
7 8 5 13 12
Q 4 1
A 5 4
A 3 4
Q 4 3

13
5
```

```
#include <bits/extc++.h>
#define int long long
using namespace std;
using namespace __gnu_pbds;
const int N = 1e5+10;

typedef tree<int, null_type, greater<int>, rb_tree_tag,
     tree_order_statistics_node_update> tre;
tre st[N];
int dsu[N];

int fp(int x){
  return dsu[x] == x ? x : dsu[x] = fp(dsu[x]);
}

void U(int x, int y){
  x = fp(x), y = fp(y);
  if (x == y) return;
  if (st[x].size() > st[y].size()) swap(x, y);
  dsu[x] = y;
  for (auto i : st[x]) st[y].insert(i), st[x].erase(i);

}
signed main(){
  ios::sync_with_stdio(0);
  cin.tie(0);
  int n, m;
  cin >> n >> m;
  vector <int> arr;
  arr.resize(N);
  for (int i = 1; i <= n; i++) cin >> arr[i], st[i].
      insert(arr[i]), dsu[i] = i;
  for (int i = 0; i < m; i++){
    string oper; int x, y;
    cin >> oper >> x >> y;
    if (oper == "A"){
      U(x, y);
    }else if (oper == "M"){
      int p = fp(x);
      st[p].erase(arr[x]);
      arr[x] = y;
      st[p].insert(y);
    }else{
      int p = fp(x);
      cout << *st[p].find_by_order(y-1) << "\n";
      //求k在樹中是第幾大 t.order_of_key();
      //求樹中的第k大 t.find_by_order();
    }
  }
}
```

9 Other

9.1 Suifeng0214

```
1e9+7
1e15+37

assert
他用來判斷一個條件是否成立，
如果條件成立則不會發生任何事

如果條件不成立，則會造成程式RE(Runtime Error)
通常用於debug不確定會不會錯，或者想submit時
不確定有沒有問題用的

或者想猜測資，判斷是否大於某個值，就可以assert
會造成RE的這個性質去做事

assert(x <= 5);
```

9.2 Xiang1078

```
to_string()
stoi()

std::ios::sync_with_stdio(false);
cin.tie(NULL);

#include <ext/pb_ds/assoc_container.hpp>
#include<ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
gp_hash_table<int,int>
tree<int, null_type, less<int>, rb_tree_tag,
    tree_order_statistics_node_update> rbtree;
rbtree.find_by_order

algorithm
reverse(vec.begin(),vec.end())
__gcd(x,y)

set
set_intersection
set_union
set_difference
set_symmetric_difference
set_union(A.begin(),A.end(),B.begin(),B.end(),inserter(
     C1 , C1.begin() ) )
upper_bound 大於
lower_bound 大於等於

bitset
.to_ulong()
.flip()
._Find_next(x)
._Find_first()

priority_queue<T, vector<T>, greater<T> >

struct cmp
{
    bool operator()(char* a, char* b)
    {
        return strcmp(a, b) < 0;
    }
};
set<char*, cmp> dictionary;
```

10 Empty

11 Empty

12 Empty