

综合混练

人员

牛晓晨、刘新睿、隋天翼、范家郡、刘锦轩 到课

上周作业检查

上周作业链接: <https://vjudge.net/contest/720521>

Begin: 2025-05-31 08:30 CST

☆ 2025-0531 五队上课 (综合混练)

End: 2025-12-25 16:30 CST

Elapsed: 6:23:47:59

Running

Remaining: 201:08:12:00

Overview

Problem

Status

Rank (6:23:47:57)

Discuss

Setting

Clone

Update

Delete

Rank	Team	Score	Penalty	A 1 / 2	B 4 / 4	C 1 / 3	D 0 / 0	E 0 / 0
1	☆ Hacker Cracker sty0948 (隋天翼)	2	13269		3:14:10:38	5:14:18:26 (-2)		
2	☆ WangYanzhen (王彦臻)	1	384	6:24:24				
3	☆ longlong_int (刘锦轩)	1	9449		6:13:29:00			
4	☆ niuxiaochen (牛晓晨)	1	9474		6:13:54:24			
5	☆ exLucas	1	9502	(-1)	6:14:22:58			

作业

<https://vjudge.net/contest/721752> (课上讲了上周比赛的 A B C D E, 课后作业是本周比赛的 A B C D E 题)

课堂表现

今天讲的题目整体思路不是很复杂, 但是有一两道题写起来比较复杂, 同学们课上大部分没写完, 课下要再写一写。

课堂内容

CF1379C Choosing flowers

枚举最后只选哪个物品的 b 结尾, 然后选那些 a 比他大的物品即可 (用 二分 + 前缀和 维护即可)

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

using namespace std;

typedef long long LL;
const int maxn = 1e5 + 5;
struct node {
```

```

int a, b;
bool operator < (const node& p) const { return a < p.a; }
} w[maxn];
LL p[maxn];

LL get_sum(int l, int r) { return (l<=r ? p[r]-p[l-1] : 0); }

void solve() {
    int n, m; cin >> n >> m;
    for (int i = 1; i <= m; ++i) cin >> w[i].a >> w[i].b;

    sort(w+1, w+m+1);
    for (int i = 1; i <= m; ++i) p[i] = p[i-1] + w[i].a;

    LL res = 0;
    for (int i = 1; i <= m; ++i) {
        int limit = w[i].b;
        int pos = upper_bound(w+1, w+m+1, (node){limit,0}) - w; // [pos,m]
        if (i >= pos) {
            if (m-pos+1 >= n) res = max(res, get_sum(m-n+1, m));
            else res = max(res, get_sum(pos,m) + (LL)limit*(n-(m-pos+1)));
        } else {
            if (m-pos+1+1 >= n) res = max(res, w[i].a+get_sum(m-n+2,m));
            else res = max(res, w[i].a + get_sum(pos,m) + (LL)limit*(n-(m-pos+1)-1));
        }
    }

    // cout << "----- ";
    cout << res << endl;
}

int main()
{
    int T; cin >> T;
    while (T -- ) solve();
    return 0;
}

```

CF1904D1 Set To Max

对于第 i 个物品来说, 如果 $a[i] < b[i]$, 那么就往左或往右找到一个和 $b[i]$ 相同的 a 值

想更改过来的条件是: 要满足区间没有 a 值比这个 a 值小, 没有 b 值比这个 a 值大

```

#include<bits/stdc++.h>

using namespace std;

const int maxn = 2e5 + 5;
int a[maxn], b[maxn];
vector<int> vec[maxn];

```

```
int n;

void print(bool flag) { cout << (flag?"YES":"NO") << endl; }

bool check(int l, int r, int v) {
    if (l==-1 || r==-1) return false;
    for (int i = l; i <= r; ++i) {
        if (a[i] > v) return false;
        if (b[i] < v) return false;
    }
    return true;
}

void solve() {
    cin >> n;
    for (int i = 0; i <= n+2; ++i) vec[i].clear();

    for (int i = 1; i <= n; ++i) cin >> a[i], vec[a[i]].push_back(i);
    for (int i = 1; i <= n; ++i) cin >> b[i];

    for (int i = 1; i <= n; ++i) {
        if (a[i] > b[i]) return print(false);
    }

    for (int i = 1; i <= n; ++i) {
        for (int pos : vec[i]) {
            if (a[pos] == b[pos]) continue;

            int v = b[pos];
            int l = lower_bound(vec[v].begin(), vec[v].end(), pos) - vec[v].begin() - 1;
            int r = lower_bound(vec[v].begin(), vec[v].end(), pos) - vec[v].begin();

            int lpos = -1, rpos = -1;
            if (l >= 0) lpos = vec[v][l];
            if (r < (int)vec[v].size()) rpos = vec[v][r];
            if (lpos==-1 && rpos==-1) return print(false);

            if (!check(lpos,pos,v) && !check(pos,rpos,v)) return print(false);
        }
    }

    print(true);
}

int main()
{
    int T; cin >> T;
    while (T -- ) solve();
    return 0;
}
```

思路跟上面题一样, 就是找区间最大值、区间最小值的过程, 可以用 st 表实现; 往左往右找相同 a 值的过程, 可以用 vector 数组+二分 实现

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

using namespace std;

const int maxn = 2e5 + 5;
int a[maxn], b[maxn];
int fa[maxn][20], fb[maxn][20];
vector<int> vec[maxn];
int n;

void print(bool flag) { cout << (flag?"YES":"NO") << endl; }

int get_max(int l, int r) {
    for (int k = 19; k >= 0; --k) {
        if (l+(1<<k)-1 <= r) return max(fa[l][k], fa[r-(1<<k)+1][k]);
    }
    return 0;
}

int get_min(int l, int r) {
    for (int k = 19; k >= 0; --k) {
        if (l+(1<<k)-1 <= r) return min(fb[l][k], fb[r-(1<<k)+1][k]);
    }
    return 0;
}

bool check(int l, int r, int v) {
    if (l==-1 || r==-1) return false;
    return get_max(l,r)<=v && get_min(l,r)>=v;
}

void solve() {
    cin >> n;
    for (int i = 0; i <= n+2; ++i) vec[i].clear();

    for (int i = 1; i <= n; ++i) cin >> a[i], vec[a[i]].push_back(i);
    for (int i = 1; i <= n; ++i) cin >> b[i];

    for (int i = 1; i <= n; ++i) fa[i][0] = a[i], fb[i][0] = b[i];
    for (int k = 1; k < 20; ++k) {
        for (int i = 1; i+(1<<k)-1 <= n; ++i) {
            fa[i][k] = max(fa[i][k-1], fa[i+(1<<(k-1))][k-1]);
            fb[i][k] = min(fb[i][k-1], fb[i+(1<<(k-1))][k-1]);
        }
    }

    for (int i = 1; i <= n; ++i) {
        if (a[i] > b[i]) return print(false);
    }
}
```

```

for (int i = 1; i <= n; ++i) {
    for (int pos : vec[i]) {
        if (a[pos] == b[pos]) continue;

        int v = b[pos];
        int l = lower_bound(vec[v].begin(), vec[v].end(), pos) - vec[v].begin() - 1;
        int r = lower_bound(vec[v].begin(), vec[v].end(), pos) - vec[v].begin();

        int lpos = -1, rpos = -1;
        if (l >= 0) lpos = vec[v][l];
        if (r < (int)vec[v].size()) rpos = vec[v][r];
        if (lpos == -1 && rpos == -1) return print(false);

        if (!check(lpos, pos, v) && !check(pos, rpos, v)) return print(false);
    }
}

print(true);
}

int main()
{
    int T; cin >> T;
    while (T -- ) solve();
    return 0;
}

```

CF479E Riding in a Lift

设 $f[i][j]$: 移动 i 次后, 到达第 j 个位置有多少方案

那么 $f[i][j]$ 是由 $f[i-1][l] + \dots + f[i-1][r] - f[i-1][j]$ 得来的, 可以用前缀和实现 $O(1)$ 转移

l 和 r 这两个边界也可以根据题意 $O(1)$ 求

```

#include<bits/stdc++.h>

using namespace std;

typedef long long LL;
const int maxn = 5000 + 5;
const int mod = 1e9 + 7;
int a[maxn], b[maxn], p[maxn];

int get_sum(int l, int r) {
    if (l > r) return 0;
    return (p[r] - p[l-1] + mod) % mod;
}

int main()

```

```

{
    int n, st, ed, k; cin >> n >> st >> ed >> k;
    a[st] = 1; for (int i = 1; i <= n; ++i) p[i] = p[i-1] + a[i];

    for (int i = 1; i <= k; ++i) {
        for (int j = 1; j <= n; ++j) {
            if (j < ed) b[j] = (get_sum(1, (ed+j-1)/2) - a[j] + mod) % mod;
            else if (j > ed) b[j] = (get_sum((ed+j)/2+1, n) - a[j] + mod) % mod;
        }
        for (int j = 1; j <= n; ++j) a[j] = b[j], p[j] = (p[j-1] + a[j]) % mod;
    }

    cout << get_sum(1, n) << endl;
    return 0;
}

```

CF1905D Cyclic MEX

可以用 线段树上二分+线段树区间修改+线段树区间查询 维护

```

#include <bits/stdc++.h>

using namespace std;

typedef long long LL;
const int maxn = 2e6 + 5;
int w[maxn];
struct node {
    int l, r, lzy, maxx;
    LL sum;
} tr[maxn*4];

void pushup(int u) {
    tr[u].maxx = max(tr[u*2].maxx, tr[u*2+1].maxx);
    tr[u].sum = tr[u*2].sum + tr[u*2+1].sum;
}

void pushdown(int u) {
    if (tr[u].lzy != -1) {
        node &uu = tr[u], &ll = tr[u*2], &rr = tr[u*2+1];
        ll.lzy = uu.lzy, ll.maxx = uu.maxx, ll.sum = (LL)ll.lzy*(ll.r-ll.l+1);
        rr.lzy = uu.lzy, rr.maxx = uu.maxx, rr.sum = (LL)rr.lzy*(rr.r-rr.l+1);
        uu.lzy = -1;
    }
}

void build(int u, int l, int r) {
    tr[u] = {l, r, -1, 0, 0};
    if (l != r) {
        int mid = (l + r) / 2;
        build(u*2, l, mid), build(u*2+1, mid+1, r);
    }
}

```

```

    pushup(u);
}
}

void modify(int u, int l, int r, int k) {
    if (tr[u].l >= l && tr[u].r <= r) {
        tr[u].lzy = k, tr[u].maxx = k, tr[u].sum = (LL)k * (tr[u].r - tr[u].l + 1);
        return;
    }

    pushdown(u);
    int mid = (tr[u].l + tr[u].r) / 2;
    if (l <= mid) modify(u*2, l, r, k);
    if (r > mid) modify(u*2+1, l, r, k);
    pushup(u);
}

LL query(int u, int l, int r) {
    if (tr[u].l >= l && tr[u].r <= r) return tr[u].sum;

    pushdown(u);
    int mid = (tr[u].l + tr[u].r) / 2;
    LL res = 0;
    if (l <= mid) res += query(u*2, l, r);
    if (r > mid) res += query(u*2+1, l, r);
    return res;
}

int queryPos(int u, int l, int r, int value) { // 在 [l,r] 中找第一个 >= value 的位置
    if (tr[u].l == tr[u].r) return tr[u].l;

    pushdown(u);
    int mid = (tr[u].l + tr[u].r) / 2;
    if (r <= mid) return queryPos(u*2, l, r, value);
    if (l > mid) return queryPos(u*2+1, l, r, value);
    if (tr[u*2].maxx >= value) return queryPos(u*2, l, r, value);
    return queryPos(u*2+1, l, r, value);
}

void solve() {
    int n; cin >> n;
    for (int i = 1; i <= n; ++i) cin >> w[i];
    build(1, 1, 2*n);

    int minn = w[n];
    for (int i = n-1; i >= 1; --i) modify(1,i,i,minn), minn = min(minn,w[i]);

    LL res = query(1, 1, n-1);

    for (int i = 1; i <= n-1; ++i) {
        // 把 w[i] 移动到 w[n+i]
        // 再 i+1 ~ n+i-2 中找到第一个 >=w[i] 的位置 pos
        // 把 pos ~ n+i-1 全部赋值为 w[i]
    }
}

```

```
//    求 i+1 ~ n+i-1 的和

    modify(1, i, i, 0);
    int pos;
    if (query(1, n+i-2, n+i-2) >= w[i])
        pos = queryPos(1, i+1, n+i-2, w[i]);
    else pos = n+i-1;

    modify(1, pos, n+i-1, w[i]);
    res = max(res, query(1, i+1, n+i-1));
}

//  cout << "----- ";
    cout << res+n << endl;
}

int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0), cout.tie(0);

    int T; cin >> T;
    while (T -- ) solve();
    return 0;
}
```