CF Edu Round 94

D

题意：给定一个序列最大长度为3e3□问有多少个四元组(i,j,k,l)满足$i<j<k<l $并且$a_i = a_k,a_j = a_l$

题解：先求出所有满足$a_i = a_j ,i<j$的二元组，然后二维前缀和求和即可。（不开longlong 见祖宗）

```
#include <algorithm>
#include <cstdio>
#include <cstring>
#include <iostream>
#include <stack>
#include <vector>
using namespace std;
typedef long long  ll;
int read()
{
    int x = 0, flag = 1;
    char c = getchar();
    while ((c > '9' || c < '0') && c != '-')
        c = getchar();
    if (c == '-')
        flag = 0, c = getchar();
    while (c <= '9' && c >= '0')
    {
        x = (x << 3) + (x << 1) + c - '0';
        c = getchar();
    }
    return flag ? x : -x;
}
const int MAX = 3e3 + 10;
int pic[MAX];
pair<int, int> seg[MAX * MAX];
int tot;
ll dp[MAX][MAX];
inline void init(int n)
{
    tot = 0;
    for (int i = 0; i <= n;i++)
        memset(dp[i], 0, sizeof(ll) * (n + 1));
}
int main()
{
    int t = read();
    while(t--)
    {
```

```
        int n = read();
        ll ans = 0;
        for (int i = 1; i <= n;i++)
            pic[i] = read();
        for (int i = 1; i <n ;i++)
            for (int j = i + 1; j <= n;j++)
                if (pic[i] == pic[j])
                {
                    seg[++tot] = {i, j};
                    dp[i][j] = 1;
                }
        for (int i = 1; i <= n;i++)
            for (int j = 1; j <= n;j++)
                dp[i][j] += dp[i - 1][j] + dp[i][j - 1] - dp[i - 1][j - 1];
        for (int tem = 1; tem <= tot;tem++)
        {
            int i = seg[tem].first, k = seg[tem].second;
            if (k == n || k - i == 1)
                continue;
            ans += (dp[k - 1][n] - dp[i][n] - dp[k - 1][k] + dp[i][k]);
        }
        init(n);
        printf("%lld\n", ans);
    }
    return 0;
}
```

E

题意：积木大赛变种，新增操作，把一堆积木去掉任意个。数据范围5e3

题解：开始想成dp写到后面发现完全不是dp就是一个贪心，对于一个区间的积木，要么一个个的全都去掉，要么让最小值直接去掉，然后递归计算两边。由于要取区间最小值，所以我当时写了个线段树，后来发现完全不需要，这个不是个dp□整个复杂度不是n^2□因为每次贪心会使得区间长度减一，所以复杂度是$O(n)$的，所以即便是暴力取最小值，也不过是n^2的复杂度，完全可以过。（所以这题数据范围还可以加强）

```
#include <algorithm>
#include <cstdio>
#include <cstring>
#include <iostream>
#include <stack>
#include <vector>
#define ls (node << 1)
#define rs ((node << 1) | 1)
using namespace std;
const int MAX = 5e3 + 20;
int read()
```

```
{
    int x = 0, flag = 1;
    char c = getchar();
    while ((c > '9' || c < '0') && c != '-')
        c = getchar();
    if (c == '-')
        flag = 0, c = getchar();
    while (c <= '9' && c >= '0')
    {
        x = (x << 3) + (x << 1) + c - '0';
        c = getchar();
    }
    return flag ? x : -x;
}
int mmin[MAX << 2];
int pic[MAX];
int dp[MAX][MAX],n;
inline int get_min(int x,int y)
{
    if (pic[x] < pic[y])
        return x;
    else
        return y;
}
void build(int node,int l,int r)
{
    if (l == r)
    {
        mmin[node] = l;
        return;
    }
    int mid = (l + r) >> 1;
    build(ls, l, mid);
    build(rs, mid + 1, r);
    mmin[node] = get_min(mmin[ls] , mmin[rs]);
}
inline int qurry(int node, int l, int r, int L, int R)
{
    if (l == L and r == R)
        return mmin[node];
    else
    {
        int mid = (L + R) >> 1;
        if (r <= mid)
            return qurry(node << 1, l, r, L, mid);
        else if (l >= mid + 1)
            return qurry((node << 1) | 1, l, r, mid + 1, R);
        else
            return get_min(qurry(node << 1, l, mid, L, mid) , qurry((node <<
1) | 1, mid + 1, r, mid + 1, R));
    }
```

```
}
int mfs(int l,int r,int mins)
{
    if (l>r)
        return 0;
    if (dp[l][r] != -1)
        return dp[l][r];
    if (l == r and pic[l] - mins <=0)
        return 0;
    int pos = qurry(1, l, r, 1, n);
    int ann = min(r - l + 1, mfs(l, pos - 1, pic[pos]) + mfs(pos + 1, r,
pic[pos]) + pic[pos]-mins);
    dp[l][r] = ann;
    return ann;
}
int main()
{
    n = read();
    memset(dp, -1, sizeof(dp));
    for (int i = 1; i <= n;i++)
        pic[i] = read();
    build(1, 1, n);
    cout << mfs(1, n,0);
}
```