CF1696D Permutation Graph

作者 lg_zhou

发布时间 2022-09-30 22:00
分类 题解 CF1696D

很显然，全局最大值和全局最小值之间必走一次。因为任意一次不可能跨越全局最小或全局最大。若设全局最大在全局最小前面，那么我们前面最多一次性跨越到全局最大，下一步一次跳到全局最小一定是最优的。

那我们预处理出 st 表，这里用 `pair` 可以很方便的求出最大值最小值位置。然后分治处理。

$\text{solve}(l,r)=\text{solve}(l,p1)+\text{solve}(p2,r)+1$，$p1,p2$ 为最大值最小值的位置，如果 $p2$ 小，`swap` 一下即可。

```
#include<iostream>
#include<cmath>
#include<algorithm>

#define fi first
#define se second
using namespace std;

typedef long long ll;
typedef pair<int,int> pii;

const int N = 2.5e5+5;

int T,n;
int a[N];

pii mx[N][20], mn[N][20];


void mul(){
    for (int i = 1; i <= 20; i++){
        for (int j = 1; j+(1<<i-1) <= n; j++){
            mx[j][i] = max(mx[j][i-1],mx[j+(1<<i-1)][i-1]);
            mn[j][i] = min(mn[j][i-1],mn[j+(1<<i-1)][i-1]);
        }
    }
}
pii getmx(int l, int r){
    int lg = log2(r-l+1);
    return max(mx[l][lg],mx[r-(1<<lg)+1][lg]);
}
pii getmn(int l, int r){
    int lg = log2(r-l+1);
    return min(mn[l][lg],mn[r-(1<<lg)+1][lg]);
}

int solve(int l, int r){
    if (r == l) return 0;
    int p1 = getmx(l,r).se, p2 = getmn(l,r).se;
    if (p1 > p2) swap(p1,p2);
    return solve(l,p1)+1+solve(p2,r);
}
int main(){
//  freopen("a.in","r",stdin);
    cin >> T;
    while(T--){
        cin >> n;
        for (int i = 1; i <= n; i++){
            cin >> a[i];
            mx[i][0].fi = mn[i][0].fi = a[i];
            mx[i][0].se = mn[i][0].se = i;
        }
        mul();
        cout << solve(1,n) << endl;
    }

    return 0;
}
```

作者：lg_zhou 创建时间：2022-09-30 22:00:22