Codeforces Round #703 (Div. 2)

比赛链接

C2. Guessing the Greatest (hard version)

标签

交互题、二分、次大值、最大值

题意

交互题。给定一固定的数列，每次可询问一区间的次大值的位置，要求在 20 次询问内找到该数列的最大值的位置$2\le n\le 10^5,1\le l<r\le n$

题解

先对整个序列区间进行询问找到整个序列的次大值的位置 pos然后询问 $[1,pos]$得到该区间的次大值位置 $pos1$若 $pos1==pos$则说明最大值位置在 pos 左侧，否则在 pos 右侧。假设最大值位置在 pos 右侧，则需要找到最小的 r 使得 $ask(pos,r)==pos$此时的 r 即为最大值位置。所以简单说来就是用两次询问找到最大值在次大值左边还是右边，然后二分找最大值位置，最多需进行 $2+\lceil\log_2^{1e5}\rceil=2+17=19$ 次询问。

代码

```
#include<bits/stdc++.h>
using namespace std;
const int N=1e5+5;
int ask(int l,int r){
    printf("? %d %d\n",l,r);
    fflush(stdout);
    int pos;
    scanf("%d",&pos);
    return pos;
}
int main(){
    int n;
    scanf("%d",&n);
    int pos=ask(1,n);
    if(pos==1){
        int l=pos+1,r=n;
        int mid;
        int pos1;
        while(1){
```

```
            mid=l+r>>1;
            pos1=ask(pos,mid);
            if(l==mid) break;
            if(pos==pos1) r=mid;
            else l=mid;
        }
        if(pos1==pos) printf("! %d",l);
        else printf("! %d",r);
        return 0;
    }
    int pos1=ask(1,pos);
    if(pos1==pos){
        int l=1,r=pos-1;
        int mid;
        while(1){
            mid=(l+r+1)>>1;
            pos1=ask(mid,pos);
            if(r==mid) break;
            if(pos1==pos) l=mid;
            else r=mid;
        }
        if(pos1==pos) printf("! %d",r);
        else printf("! %d",l);
        return 0;
    }
    int l=pos+1,r=n;
    int mid;
    while(1){
        mid=l+r>>1;
        pos1=ask(pos,mid);
        if(l==mid) break;
        if(pos1==pos) r=mid;
        else l=mid;
    }
    if(pos1==pos) printf("! %d",l);
    else printf("! %d",r);
    return 0;
}
```

D. Max Median

标签

二分答案、中位数、前缀

题意

给定长度为 n 的数列和 k，要找到长度至少为 k 且具有最大中位数的子区间，$1\le k\le n\le 2\cdot10^5,1\le a_i\le n$。

题解

先考虑这样一种情况：数列中全为 -1 或 1。要使某区间的中位数为 1，只需该区间的和为正。对于本题，二分答案，假设为 x，则把大于等于 x 的数变成 1，小于 x 的数变成 -1，然后求出前缀和 $sum[i]$ 以及前缀和的前缀最小值 $minsum[i]$，对于每个位置 i，若 $sum[i]-minsum[i-k]>0$ 则存在中位数大于等于 x 的区间。时间复杂度 $O(n\log n)$。

代码

```
#include<bits/stdc++.h>
using namespace std;
const int N=2e5+5;
typedef long long ll;
const ll INF=1e18;
ll sum[N],minsum[N],a[N],b[N];
int n,k;
bool check(ll x){
    for(int i=1;i<=n;i++){
        b[i]=a[i]>=x?1:-1;
        sum[i]=sum[i-1]+b[i];
        minsum[i]=min(minsum[i-1],sum[i]);
    }
    for(int i=k;i<=n;i++){
        if(sum[i]-minsum[i-k]>0)
            return true;
    }
    return false;
}
int main(){
    scanf("%d%d",&n,&k);
    for(int i=1;i<=n;i++){
        scanf("%lld",&a[i]);
    }
    ll l=1,r=n;
    ll mid;
    while(l<r){
        mid=l+r+1>>1;
        if(check(mid)) l=mid;
        else r=mid-1;
    }
    printf("%lld",l);
    return 0;
```

```
}
```

E. Paired Payment

标签

最短路，构造，虚点

题意

给定一个无向图，不保证连通，每条边有花费 w每次走必须一次性走两条边 $a\rightarrow b\rightarrow c$花费为 $(w_{ab}+w_{bc})^2$求从 1 号点走到其他每个点的最小花费，若不能到达则输出 -1。

题解

一看题目是跟最短路有关的，但是不能直接套用 dijkstra 模板。注意到边权 $w_i\leq 50$ 考虑添加虚点。对每个点 v添加虚点 $v(w)=v*51+w$其中 w 是与之相连的边的权值。对于边 (u,v,w)连边 $u\rightarrow v(w)$权值取 0，再连边 $u(was)\rightarrow v$权值取 $(was+w)^2$这样一来，若 u 是作为三个点 q,u,v 的中间结点，则会走 $(q,u(was),0),(u(was),v,(was+w)^2)$ 这两条边。当 v 作为中间结点时同理。按这种方式建图，然后从 1 号点跑 dijkstra 即可得到答案。

代码

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    ios_base::sync_with_stdio(0);
    cin.tie(NULL);
    cout.tie(NULL);
    int n,m;
    cin>>n>>m;
    vector<vector<pair<int,int>>> G(n*51);
    auto addedge=[&](int u,int v,int w){
        G[u*51].push_back({v*51+w,0});
        for(int was=1;was<=50;was++){
            G[u*51+was].push_back({v*51,(was+w)*(was+w)});
        }
    };
    for(int i=0;i<m;i++){
        int u,v,w;
        cin>>u>>v>>w;
        --u,--v;
        addedge(u,v,w);
```

```
        addedge(v,u,w);
    }
    set<pair<int,int>>st;
    int inf=1e9+7;
    vector<int> d(G.size(),inf);
    d[0]=0;
    st.insert({d[0],0});
    while(st.size()){
        auto f=*st.begin();
        st.erase(st.begin());
        for(auto i:G[f.second]){
            if(d[i.first]>d[f.second]+i.second){
                st.erase({d[i.first],i.first});
                d[i.first]=d[f.second]+i.second;
                st.insert({d[i.first],i.first});
            }
        }
    }
    for(int i=0;i<n;++i){
        int ans=d[i*51];
        if(ans==inf) cout<<"-1 ";
        else cout<<ans<<' ';
    }
    return 0;
}
```

F. Pairs of Paths