提高组高分试题 第二组 题解

2735 腿部挂件

解

T1: 很明显是一个区间异或问题，我们不妨先考虑最暴力的做法，每回暴力循环R~L 对于区间中每个数求异或，最终保留最大值可得到答案 期望得分 20.

让我们往正解的方向靠拢，我们不妨先看一个简化版的问题，对于一个不变的序列A[1....n]，我们每回求x与A中任意元素异或和的最大值。

对于序列异或问题，我们一般采用01tire树来处理（当然也有一部分需要用到线性基）。大体思路就是，我们把整个序列都加入到01trie树里面，对于X采取贪心策略，为了使异或和最大，我们将X二进制拆分后从高位到低位在Tire树上匹配：如果X当前位是1 我们便向01Tire的-0边走，反之亦然。每走一条边我们就把答案加上走这条边的价值。

为什么是正确的，我们优先确保X异或后的高位为一，每一位高位的价值都大于其后所有二进制拆分后低位的价值，所以我们只需要一个01Tire树就能解决区间不动的问题。对于本题，我们只需要加一个可持久化操作就能够实现全部操作，时间复杂度$O((n+m)log(1e9))$。期望得分100。

我们再考虑另一种做法（非正解），对于区间问题我们往往能采用离线操作来降低时间复杂度，我们不妨应用莫队+分块+Tire来实现，用莫队来解决区间移动以及不会可持久化这一的问题，时间复杂度$O(n^{0.5}mlog(1e9))$。期望得分 45~60

标准代码

C++:

```
#include <stdio.h>
const int maxn = 200101, maxd = 30;
int n, q, seg_tot, root[maxn];
struct Trie
{
    int cnt, ch[2];
} seg[maxn * (maxd + 1)];
void update(int &rt, int dep, int x)
{
    int last = rt;
    seg[rt = ++seg_tot] = seg[last];
    ++seg[rt].cnt;
    if(dep == -1)
        return;
    update(seg[rt].ch[(x >> dep) & 1], dep - 1, x);
}
int query(int rtL, int rtR, int dep, int x)
{
    if(dep == -1)
        return 0;
    int o = (x >> dep) & 1;
    if(seg[seg[rtL].ch[o ^ 1]].cnt < seg[seg[rtR].ch[o ^ 1]].cnt)
```

```
        return query(seg[rtL].ch[o ^ 1], seg[rtR].ch[o ^ 1], dep - 1, x) | 1 <<
dep;
    return query(seg[rtL].ch[o], seg[rtR].ch[o], dep - 1, x);
}
int main()
{
    scanf("%d%d", &n, &q);
    for(int i = 1; i <= n; ++i)
    {
        int x;
        scanf("%d", &x);
        root[i] = root[i - 1];
        update(root[i], maxd - 1, x);
    }
    while(q--)
    {
        int l, r, x;
        scanf("%d%d%d", &x, &l, &r);
        printf("%d\n", query(root[l], root[r + 1], maxd - 1, x));
    }
    return 0;
}
```

2736 走夜路

解

我们先朴素的考虑一下怎么使总花费最小，最好的办法是在电价低的地方多充电这样在电价高的地方我可以少加或不加电来降低消费。这样我们要考虑的问题就是在装满电的情况下所能达的城市中有没有电价比当前城市低的城市，这样我们就可以到下一站充低价电来节省花费。以下均用**“下一站”**来代替**“电价小于当前站且顺序大于当前站并离当前站最近的充电站”**。

一种做法：

对于当前位置的下一站，我们可以用一个单调栈来解决。首先倒序处理，依次将充电的单价加入到单调栈中保证栈顶到底单调递减。栈中元素及我们所求下一站（当前位置的下一站即为本次更新后单调栈栈顶）。

之后，从第一站开始统计答案，当前站和下一站之间有许多被忽略的站（因为电价高于当前站），如果充的电不用充满能够到下一站，就可以加刚好到下一站的电再在下一站加便宜电，如果即使充满也到不了下一站，就充满电（因为当前站和下一站之间站的电价均高于当前站，冲满电来减少在当前站和下一站之间加高价电）。

边界条件，如果当前站到下一个站的距离的耗电量大于电池容量无解。

因此本题有O(n)做法，而O(nlog n) 算法可能会因常数问题被卡。

标准代码

C++：

```
#include<stdio.h>
#include<cstring>
#include<algorithm>
using namespace std;
```

```
#define N 501005
#define ll long long
int n,m,i,r,now,d[N],p[N],q[N],nxt[N];
ll dis,ans,sum[N];
int main()
{
    scanf("%d%d",&n,&m);
    for(i=1;i<=n;i++)scanf("%d%d",&d[i],&p[i]),sum[i]=sum[i-1]+1ll*d[i];
    for(i=n;i;i--){
        while(r&&p[q[r]]>p[i])r--;
        nxt[i]=q[r];q[++r]=i;
        if(!nxt[i])nxt[i]=n+1;
    }
    ans=0;now=0;
    for(i=1;i<=n;i++){
        now-=d[i-1];
        if(now<0){puts("-1");return 0;}
        dis=min(sum[nxt[i]-1]-sum[i-1],1ll*m);
        if(dis>now)ans+=1ll*(dis-now)*p[i],now=dis;
    }
    printf("%lld\n",ans);
    return 0;
}
```

2737 宝石专家

解

不妨先考虑答案的组成，对于数列中Ai，如果在序列中有和其相等的树Ak，我们不妨从第二个点开始保存k<i 且 Ak = Ai 的点k 与 i的距离。我们可以采取一个巧妙的离线做法来完成剩下的操作。

将所有询问读入并按右端点排序，对于数列中 Ai 如果存在 Ak = Ai 且 k <i，我们不妨保留离Ai最近的点 k 并将它们的距离保存在dis 数组里，（ dis[i]表示距离长度，如果不存在Ak = Ai 且 k<i,则dis[i] = 0 ）。

从1到n更新答案，对于dis[i]>0的点，采用树状数组或线段树来维护区间最小值，每次将点i-dis[i] 用 dis[i]更新（线段树&树状数组单点修改）。在i = “询问的右端点”时更新答案并查询区间最小值，所得及为答案。

正确性分析：当前询问右端点一定大于等于 i ，在询问时限制左端点，可以保证左右端点都在询问区间中。

时间复杂度，O((m+n)log(n));

标准代码

C++：

```
#include <cstdio>
#include <iostream>
#include <algorithm>
#include <cstring>
using namespace std;

const int maxn = 5e5+100;
```

```
int n,m;
int dis[maxn],res[maxn],ans[maxn];

struct index
{
    int val,idx;
}T[maxn];
bool operator < (index x,index y)
{
    return x.val < y.val || x.val ==y.val && x.idx < y.idx;
}

struct query
{
    int l,r,idx;
}F[maxn];
bool operator <(query x,query y)
{
    return x.r < y.r;
}

int low(int x)
{
    return x&(-x);
}

void reduce(int idx,int val)
{

    for(;idx; idx-=low(idx))
        res[idx] = min(res[idx],val);
}

int get(int idx)
{
    int val = 1e9+100;
    for(;idx<=n;idx+=low(idx))
        val = min(val,res[idx]);//,cout<<idx<<endl;
    return val;
}


int main()
{
    scanf("%d%d",&n,&m);
    for(int i=1;i<=n;i++) scanf("%d",&T[i].val),T[i].idx = i;
    sort(T+1,T+1+n);
    for(int i=2;i<=n;i++)
    {
        if(T[i-1].val==T[i].val)
            dis[T[i].idx] = T[i].idx -T[i-1].idx;
    }
    for(int i=1;i<=m;i++)
    {
        scanf("%d%d",&F[i].l,&F[i].r);
        F[i].idx = i;
    }
    sort(F+1,F+1+m);
```

```
    F[m+1].r = n;F[m+1].l=1;
    memset(res,0x3f,sizeof(res));
    for(int i=1,j=1;i<=n;i++)
    {
        if(dis[i])
            reduce(i-dis[i],dis[i]);
        for(;F[j].r<=i&&j<=m;j++)
            ans[F[j].idx] = get(F[j].l);
    }
    for(int i=1;i<=m;i++)
    {
        if(ans[i] < 1e9) printf("%d\n", ans[i]);
        else puts("-1");
    }
    return 0;
}
```