倍增算法——RMQ 问题

倍增是指成倍增长的意思，比如，这次我求的是长度为 i 的信息，则下一次我可以求 i*2 的信息，再下次可以求 2*i*2 的信息，不断成倍增长。与分治类似，倍增的时间复杂度也是对数级别的。
初次了解倍增思想，我们来认识一个 RMQ 问题。

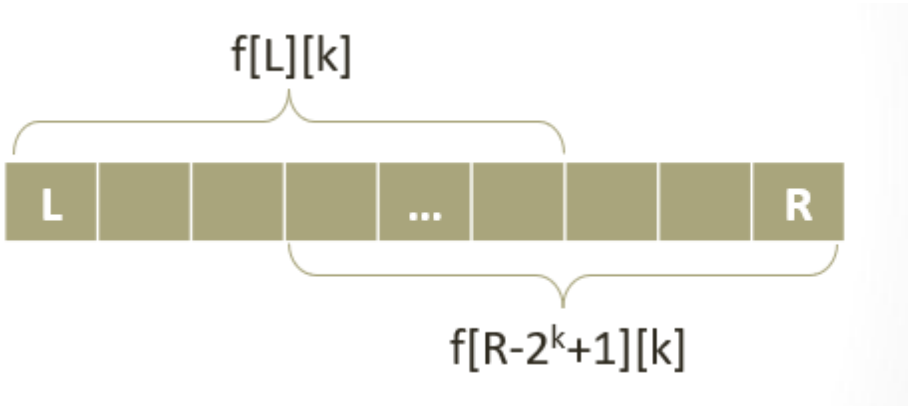
RMQ (Range Minimum/Maximum Query)问题是指：对于长度为 n 的数列 A，回答若干询问 RMQ(A,i,j)(i,j<=n)，返回数列 A 中下标在 i,j 里的最小(大）值，也就是说，RMQ 问题是指求区间最值的问题。

这个问题，我们用朴素方法做，对于每次询问的时间是 O（n），m 次询问，总时间复杂度为 O（n*m）。当然。我们也可以用线段树解决，则时间复杂度可以降为 O（m*logn）。当然，如果我们巧妙利用 ST 算法，则可以优雅地解决此问题。

Sparse_Table（ST）算法的原理实际上动态规划。我们设 F [i][j] 表示数列 A 中下标在子区间 [i ,i+2^j-1]里的数的最小值 ,也就是从 i 开始的 2^j 个数的最小值。由这个定义可知，f[i][j]=min(f[i][j-1],f[i+(1<<j-1)][j-1])，因为原序列长度为 2^j 个，则这个序列可以划分为两个长度为 2^(j-1)的子序列，这两个子序列的最小值分别为 f[i][j-1]和 f[i+(1<<j-1)][j-1]，则原序列的最小值为这两个子序列的最小值之中最小的一个。递推边界显然是 F[i,0] = A[i]，即从 i 开始长度为 1 序列的最小值，当然是它本身。这样可以在 O（nlogn）的时间复杂度内预处理 f 数组。


```
1. void initrmq(){
2.     for(int j=1;j<=log[n];j++){
3.         for(int i=1;i+(1<<j)-1<=n;i++){
4.             f[i][j]=min(f[i][j-1],f[i+(1<<j-1)][j-1]);
5.         }
6.     }
7. }
```

若我们要询问区间[L，R]的最小值，则令 k 为满足 2^k<=R-L+1 的最大整数，则以 L 开头、以 R 结尾的两个长度为 2^k 的区间合起来即覆盖了查询区间[L,R]。由于是取最小值，有些元素重复考虑了几遍也没有关系。




```
1.  int query(int a,int b){
2.      int k=log[b-a+1];
3.      return min(f[a][k],f[b-(1<<k)+1][k]);
4.  }
```

注意：因为 cmath 库中的 log2 函数效率不高，所以除了调用 log2 函数外，通常还会用 O（N）时间通过递推预处理出 1——N 这 N 种区间长度各自对应的 k 值。具体而言，就是设置变量 log[n]表示 $\log_2 n$ 的值，而 log[n]=log[n/2]+1。

洞穴里的牛

【题目描述】
洞窟里有一道长长的通道．它由 N(1≤N≤25000)段道尾相连构成，编号分别为 1 到 N．每个通道有一个阈值，其范围在[1，10^9]，如果奶牛要依次通过 i..j 的通道，那奶牛的体重指数就不能超过 i..j 通道中阈值的最小值．贝茜有 Q(1≤Q≤25000)个问题，想请教你由 i 到 j 的通道的阈值的最小值．
【输入】
第 1 行输入 N 和 Q，接下来 N 行输入每个通道的阈值，之后 Q 行每行两个整数，对应问题中的 i 和 j（i<j）．
【输出】
对于每个问题，输出其结果．
【样例输入】
10 4
75 30 100 38 50 51 52 20 81 5
1 10
3 5
6 9
8 10
【样例输出】
5
38
20
5

【分析】
本题为 RMQ 的模板题，直接套用 ST 算法即可。
【代码】

```
#include<iostream>
#include<cstdio>
using namespace std;
int n,m,x,y,a[25001],f[25001][30],log[25001];
int main(){
    scanf("%d %d",&n,&m);
    log[0]=-1;
    for(int i=1;i<=n;i++){
        scanf("%d",&a[i]);
        f[i][0]=a[i];  //
        log[i]=log[i>>1]+1; //递推预处理出 log2i
    }
    for(int j=1;j<=log[n];j++){  //先枚举区间长度
        for(int i=1;i+(1<<j)-1<=n;i++){  //枚举区间起点
            f[i][j]=min(f[i][j-1],f[i+(1<<(j-1))][j-1]);
        }
    }
    for(int i=1;i<=m;i++){
        scanf("%d %d",&x,&y);
        int k=log[y-x+1];
        printf("%d\n",min(f[x][k],f[y-(1<<k)+1][k]));
    }
}
```

均衡队形

【题目描述】
农夫约翰的 N (1 ≤ N ≤ 50,000) 头奶牛，每天挤奶时总会按同样的顺序站好。一日，农夫约翰决定为奶牛们举行一个“终极飞盘”比赛。为简化问题，他将从奶牛队列中选出一个连续区间来进行游戏。不过，参加游戏的奶牛要玩的开心的话就不能在身高上差距太大。农夫约翰制定了 Q (1 ≤ Q ≤ 200,000) 个预定的参赛组，给出它们的身高 (1 ≤ 身高 ≤ 1,000,000)。对每个参赛组，他需要你帮助确定组中最高牛和最低牛的身高差。
【输入】
第 1 行: 两个空格隔开的整数，N 和 Q。
第 2..N+1 行: 第 i+1 行包含一个整数表示第 i 头牛的身高。
第 N+2..N+Q+1 行: 两个整数 A 和 B(1 ≤ A ≤ B ≤ N)，表示一个从 A 到 B 的参赛组区间。
【输出】
第 1..Q 行: 每行包含一个整数来表示区间上最大身高差。
【样例输入】

6 3
1
7
3
4
2
5
1 5
4 6
2 2
【样例输出】
6
3
0
【分析】
对于每个组，求出最大值和最小值即可求出身高差，所以我们可以同时定义数组 f 和 g 表示 ST 表中的最大值和最小值，套用 ST 算法即可求解。
【代码】

```
#include<iostream>
#include<cstdio>
using namespace std;
const int N=50005,logN=18;
int n,q,x,y,a[N],log[N],f[N][logN],g[N][logN],ans;

int main(){
    log[0]=-1;
    scanf("%d %d",&n,&q);
    for(int i=1;i<=n;i++){
        scanf("%d",&a[i]);
        log[i]=log[i>>1]+1;
        f[i][0]=a[i];
        g[i][0]=a[i];
    }
    for(int j=1;j<=log[n];j++){
        for(int i=1;i+(1<<(j-1))<=n;i++){
            f[i][j]=max(f[i][j-1],f[i+(1<<(j-1))][j-1]);
            g[i][j]=min(g[i][j-1],g[i+(1<<(j-1))][j-1]);
        }
    }
    for(int i=1;i<=q;i++){
        scanf("%d %d",&x,&y);
        int k=log[y-x+1];
        int tall=max(f[x][k],f[y-(1<<k)+1][k]);
        int small=min(g[x][k],g[y-(1<<k)+1][k]);
```

```
27.         printf("%d\n",tall-small);
28.     }
29. }
```

选择客栈

【题目描述】

丽江河边有 n 家很有特色的客栈，客栈按照其位置顺序从 1 到 n 编号。每家客栈都按照某一种色调进行装饰（总共 k 种，用整数 0 ~ k-1 表示），且每家客栈都设有一家咖啡店，每家咖啡店均有各自的最低消费。

两位游客一起去丽江旅游，他们喜欢相同的色调，又想尝试两个不同的客栈，因此决定分别住在色调相同的两家客栈中。晚上，他们打算选择一家咖啡店喝咖啡，要求咖啡店位于两人住的两家客栈之间（包括他们住的客栈），且咖啡店的最低消费不超过 p。

他们想知道总共有多少种选择住宿的方案，保证晚上可以找到一家最低消费不超过 p 元的咖啡店小聚。

【输入】

输入共 n+1 行。

第一行三个整数 n， k， p，每两个整数之间用一个空格隔开，分别表示客栈的个数，色调的数目和能接受的最低消费的最高值；

接下来的 n 行，第 i+1 行两个整数，之间用一个空格隔开，分别表示 i 号客栈的装饰色调和 i 号客栈的咖啡店的最低消费。

【输出】

输出只有一行，一个整数，表示可选的住宿方案的总数。

【样例输入】

5 2 3
0 5
1 3
0 2
1 4
1 5

【样例输出】

3

【输入输出样例说明】

客栈编号	①	②	③	④	⑤
色调	0	1	0	1	1
最低消费	5	3	2	4	5

2 人要住同样色调的客栈，所有可选的住宿方案包括：住客栈①③，②④，②⑤，④⑤，但是若选择住 4、 5 号客栈的话， 4、 5 号客栈之间的咖啡店的最低消费是 4，而两人能承受的最低消费是 3 元，所以不满足要求。因此只有前 3 种方案可选。

【数据范围】

对于 30%的数据，有 n≤100；

对于 50%的数据，有 n≤1,000；

对于 100%的数据，有 2≤n≤200,000， 0<k≤50， 0≤p≤100， 0≤最低消费≤100。

【分析】

首先，我们先记录每一种颜色出现的位置，对于颜色 c，可能一共出现了 k 次，设第 i 次出现在 g[c][i]处，第 j 次 c 颜色出现的位置为 g[c][j]，i<j，g[c][i]<g[c][j]，那么如果查询区间 g[c][i]到 g[c][j]，如果区间的消费最小值小于等于给定限定消费 p，则 g[c][i]和 g[c][j]可以是可选方案。以 g[c][i]为左端点，对于给定的 g[c][j]符合选择方案，那么对于后续的 x>j，g[c][x]也可以是符合条件，所以，可以求出以 g[c][i]为左端点的可选方案有 k-j 个。

先预处理出区间消费的 ST 表，然后枚举每个颜色，对于颜色 c，枚举其出现的每个位置，对于当前位置 g[c][i]，以该点为左端点，枚举最近的右端点 g[c][j]，使得 g[c][i]到 g[c][j]的区间最低消费小于等于 p，则 g[c][j]为可选方案，对于后续的颜色 c 也都可选，所以方案数累加 k-j。

【代码】

```
#include<iostream>
#include<vector>
using namespace std;
int n,k,p,f[200001][20],log[200001],c,ans;
vector<int> g[50];
void initrmq(){
    for(int j=1;j<=log[n];j++){
        for(int i=1;i+(1<<j)-1<=n;i++){
            f[i][j]=min(f[i][j-1],f[i+(1<<j-1)][j-1]);
        }
    }
}
int ask(int x,int y){
    int k=log[y-x+1];
    return min(f[x][k],f[y-(1<<k)+1][k]);
}
int main(){
    cin>>n>>k>>p;
    log[0]=-1;
    for(int i=1;i<=n;i++){
        cin>>c>>f[i][0];
        log[i]=log[i>>1]+1;
        g[c].push_back(i);  //记录 c 颜色出现的位置
    }
    initrmq();
    for(int i=0;i<k;i++){  //枚举 k 种颜色
        for(int j=0;j<g[i].size()-1;j++){  //枚举第 i 种颜色出现的位置
            int L=g[i][j];  //以第 i 种颜色第 j 次出现的位置 L 为左端点
            for(int x=j+1;x<g[i].size();x++){
                int R=g[i][x];  //往后枚举最近的由端点 R
                if(ask(L,R)<=p){  //如果区间最低消费小于 p
                    ans+=g[i].size()-x;  //方案数累加
```

```
                    break;    //跳出当前循环，后面的都可以构成方案
                }
            }
        }
    }
    cout<<ans;
}
```

Magical GCD

【题目描述】
给出一个长度在 100 000 以内的正整数序列，大小不超过 10^12。
求一个连续子序列，使得在所有的连续子序列中，它们的 GCD 值乘以它们的长度最大。
【输入】
第一行一个整数 T，表示有 T 个测试样例。
接下来 T 个测试样例，对于每个测试样例，第一行一个整数 n，第二行 n 个整数，每两个整数之间用空格隔开。
【输出】
对于每个测试样例，输出一行一个整数。
【样例输入】
1
5
30 60 20 20 20
【样例输出】
80
【分析】
ST 表除了可以求最大值、最小值外，求区间最大公约数也是可以的。我们先用 ST 表求出区间最大公约数，求出的区间最大公约数如下图，而且这个区间最大公约数是满足单调递减的。

X1	X1	X1	X2	X2	X2	……	X3	X3	X3
1	2	3	4	5	6	……	i	j	k

这里，下标 1 到 3 的最大公约数是 x1，下标 1 到 4，1 到 5，1 到 6 的最大公约数 x2，依次类推，1 到 i，1 到 j，1 到 k 的最大公约数是 x3，而且 x1>x2>x3。
求解本题时，我们可以枚举一个左端点 i，对于 i，从左往右求出每个 gcd 的最大长度，然后统计答案求出最大值即可。
【代码】

```
#include<cstdio>
#include<iostream>
using namespace std;
int t,n;
long long f[100001][20],ans,log[100001];
long long gcd(long long a,long long b){
    if(a%b==0)return b;
    return gcd(b,a%b);
```

```
}
long long ask(int x,int y){
    int k=log[y-x+1];
    return gcd(f[x][k],f[y-(1<<k)+1][k]);
}
int main(){
    scanf("%d",&t);
    while(t--){
        scanf("%d",&n);
        ans=0;
        log[0]=-1;
        for(int i=1;i<=n;i++){
            scanf("%lld",&f[i][0]);
            log[i]=log[i>>1]+1;
        }
        for(int j=1;j<=log[n];j++){
            for(int i=1;i+(1<<j)-1<=n;i++){
                f[i][j]=gcd(f[i][j-1],f[i+(1<<j-1)][j-1]);
             }
        }
        for(int i=1;i<=n;i++){  //枚举每个左端点
            int x=i;  //x 表示以 i 为起点，每段 gcd 的左端点
            while(x<=n){
                int lb=x,ub=n+1;
                long long k=ask(i,x);  //求出当前段 gcd 的值
                while(ub-lb>1){   //求出当前段 gcd 的右端点，二分实现
                    int mid=(ub+lb)/2;
                    if(ask(i,mid)==k)lb=mid;
                    else ub=mid;
                }  //lb 为当前段 gcd 的有右端点
                ans=max(ans,k*(lb-i+1));  //当前段 gcd 的长度为 lb-i+1
                x=lb+1;  //x 表示下一段 gcd 的左端点
            }
        }
        printf("%lld\n",ans);
    }
}
```

降雨量

【题目描述】

我们常常会说这样的话：“X 年是自 Y 年以来降雨量最多的”。它的含义是 X 年的降雨量不超过 Y 年，且对于任意 Y＜Z＜X，Z 年的降雨量严格小于 X 年。例如 2002，2003，2004 和 2005 年的降雨量分别为 4920，5901，2832 和 3890，则可以说“2005 年是自 2003 年以来最多的”，

但不能说“2005 年是自 2002 年以来最多的”由于有些年份的降雨量未知，有的说法是可能正确也可以不正确的。

【输入】

输入仅一行包含一个正整数 n，为已知的数据。以下 n 行每行两个整数 yi 和 ri，为年份和降雨量，按照年份从小到大排列，即 yi＜yi+1。下一行包含一个正整数 m，为询问的次数。以下 m 行每行包含两个数 Y 和 X，即询问“X 年是自 Y 年以来降雨量最多的。”这句话是必真、必假还是“有可能”。

【输出】

对于每一个询问，输出 true，false 或者 maybe。

【样例输入】

```
6
2002 4920
2003 5901
2004 2832
2005 3890
2007 5609
2008 3024
5
2002 2005
2003 2005
2002 2007
2003 2007
2005 2008
```

【样例输出】

```
false
true
false
maybe
false
```

【提示】

100%的数据满足：1<=n<=50000, 1<=m<=10000, -10^9<=yi<=10^9, 1<=ri<=10^9

【分析】

本题不难，细节处理较多。详见代码注释。

【代码】

```
#include<iostream>
#include<algorithm>
using namespace std;
int n,m,Log[50001],year[50001],rain[50001],f[50001][20],x,y;
void rmq(){
    for(int j=1;j<=Log[n];j++){
        for(int i=1;i+(1<<(j-1))<=n;i++){
            f[i][j]=max(f[i][j-1],f[i+(1<<(j-1))][j-1]);
        }
    }
```

```
}
int ask(int l,int r){
    if(r<l)return -999999999;
    int k=Log[r-l+1];
    return max(f[l][k],f[r-(1<<k)+1][k]);
}
int main(){
    Log[0]=-1;
    cin>>n;
    for(int i=1;i<=n;i++){
        cin>>year[i]>>rain[i];
        Log[i]=Log[i>>1]+1;
        f[i][0]=rain[i];
    }
    rmq();
    cin>>m;
    for(int i=1;i<=m;i++){
        cin>>x>>y;
        if(x>y){
            cout<<"false"<<endl;
            continue;
        }
        int lb=lower_bound(year+1,year+n+1,x)-year;
        int ub=upper_bound(year+1,year+n+1,y)-year-1;
        if(year[lb]!=x&&year[ub]!=y){  //如果 x 没有出现和 y 也没有出现
            cout<<"maybe"<<endl;
            continue;
        }
        if(year[lb]!=x&&year[ub]==y){  //如果 x 没出现，y 出现了
            if(ask(lb,ub-1)>=rain[ub])cout<<"false"<<endl;
            else cout<<"maybe"<<endl;
            continue;
        }
        if(year[lb]==x&&year[ub]!=y){  //如果 x 出现，y 没出现
            if(ask(lb+1,ub)>=rain[lb])cout<<"false"<<endl;
            else cout<<"maybe"<<endl;
            continue;
        }
        if(year[lb]==x&&year[ub]==y){  //x 和 y 都出现了
            if(rain[lb]<rain[ub]){
                cout<<"false"<<endl;
            }else{
                if(ask(lb+1,ub-1)<rain[ub]){
                    if(ub-lb==y-x){  //中间年份没有空缺
```

```
                        cout<<"true"<<endl;
                    }else{
                        cout<<"maybe"<<endl;
                    }
                }else{
                    cout<<"false"<<endl;
                }
            }
        }
    }
}
```

与众不同

【题目描述】
A 是某公司的 CEO，每个月都会有员工把公司的盈利数据送给 A，A 是个与众不同的怪人，A 不注重盈利还是亏本，而是喜欢研究「完美序列」：一段连续的序列满足序列中的数互不相同。
A 想知道区间 [L,R][之间最长的完美序列长度。
【输入】
第一行两个整数 N,M，N 表示连续 N 个月，编号为 0 到 N-1，M 表示询问的次数；
第二行 N 个整数，第 i 个数表示该公司第 i 个月的盈利值 aii；

接下来 M 行每行两个整数 L,R，表示 A 询问的区间。
【输出】
输出 M 行，每行一个整数对应询问区间内的完美序列的最长长度。
【样例输入】
9 2
2 5 4 1 2 3 6 2 4
0 8
2 6
【样例输出】
6
5
【提示】
对于全部数据，1≤N,M≤2×10^5,0≤L≤R≤N-1,|ai| <=10^6
【分析】
本题有点难，为了求解区间里面的完美数（互不相同的最长数列），我们需要额外构建几个辅助数组。Last[x]表示数值 x 在序列中最近出现的位置，s[i]表示以 a[i]结尾构成的完美数列的开头位置，f[i]表示以 a[i]结尾构成的完美数列的长度，易得 f[i]=i-s[i]+1。那么如何求解 s[i]呢？我们可以递推求得，已知 s[i-1]表示以 a[i-1]为结尾的互不相同的最长数列的开始位置，如果 a[i]和这个数列中的任何数都不相同，那么这个最长数列的长度加 1，即 s[i]=s[i-1]。如果 a[i]和这个数列中的某个数相同呢？由 last[x]可知，上一个 a[i]出现的位置为 last[a[i]]，那么这个数列即由 last[a[i]]+1 开始，当然，这个前提是 last[a[i]]+1 是在 s[i-1]后的，即

s[i]=max(s[i-1],last[a[i]]+1)。
接下来，对于询问 x，y 区间的最长完美数列，如何求解呢？我们分两种情况讨论。
情况一：如果 s[y]<=x，那么表示整个区间的数列都不相同，最长长度为 y-x+1。
情况二：存在一个分界点 pos，使得 s[pos]>=x，即从 pos 到 y 区间的最长完美序列都包含在区间 x，y 之间，那么这时只需利用 ST 表求解出 f[pos]到 f[y]的最大值即可。此时还有一个条件不可忽略，从 x 到 pos-1 也是一个完美数列，长度为 pos-x，也需纳入最大值之中进行比较。
【代码】

```
#include<iostream>
#include<algorithm>
using namespace std;
int n,m,x,y,a[200001],s[200001],last[2000000],f[200001][20],Log[200001];
void initrmq(){
    for(int j=1;j<=Log[n];j++){
        for(int i=1;i+(1<<j)-1<=n;i++){
            f[i][j]=max(f[i][j-1],f[i+(1<<j-1)][j-1]);
        }
    }
}
int ask(int x,int y){
    int k=Log[y-x+1];
    return max(f[x][k],f[y-(1<<k)+1][k]);
}
int main(){
    scanf("%d %d",&n,&m);
    Log[0]=-1;
    for(int i=1;i<=n;i++){
        scanf("%d",&x);
        x+=1000000;  //题目 x 是绝对值，有可能出现负数，这里进行了偏移。
        s[i]=max(s[i-1],last[x]+1);  //递推求 s[i]
        f[i][0]=i-s[i]+1;  //求出 f[i]，同时放入 ST 表
        last[x]=i;  //标记本次 X 出现的位置
        Log[i]=Log[i>>1]+1;
    }
    initrmq();
    for(int i=1;i<=m;i++){
        scanf("%d %d",&x,&y);
        x++,y++;  //题目从下标 0 开始，这里进行了偏移，从 1 开始
        int ans=0;
        if(s[y]<=x)ans=y-x+1;  //情况 1 的讨论
        else{  //情况 2 的讨论
            int pos=lower_bound(s+1,s+n+1,x)-s;  //二分找出分界点
            ans=pos-x;  //分界点前半段
```

```
36.             ans=max(ans,ask(pos,y));   //分界点后半段求最大值
37.         }
38.         printf("%d\n",ans);
39.     }
40. }
```