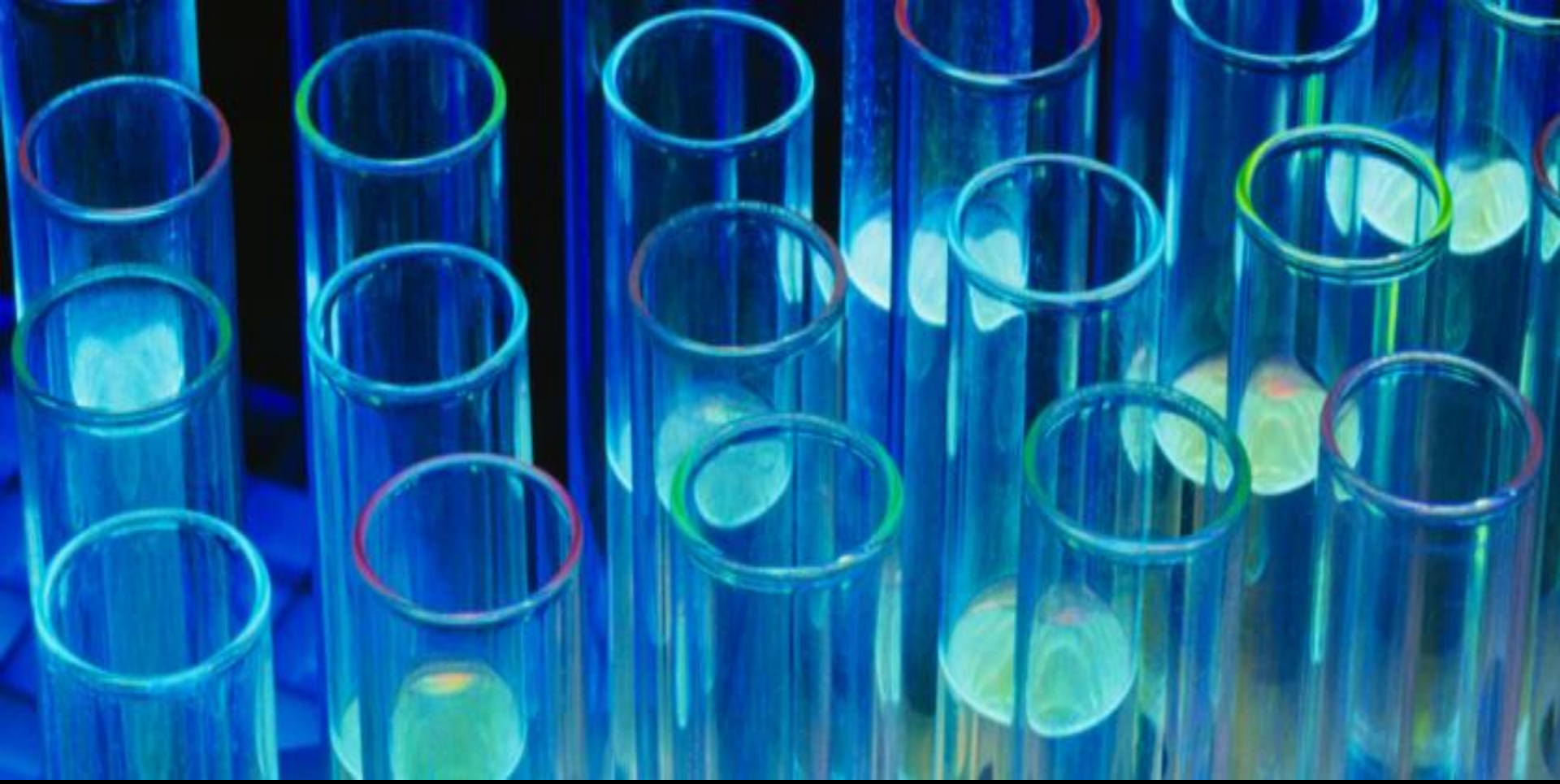


高级数据结构之线段树

BY GY（改编自CXY1999《高级数据结构之线段树》）

大纲

- 引入：RMQ问题
- 例子：洛谷P1816
- 实战：建树
- 实战：查询
- 例子：洛谷P3374
- 实战：维护
- 实战：洛谷P3372
- 实战：洛谷P3373

引入：RMQ

Range Minimum/Maximum Query

什么是RMQ

- RMQ全称为Range Minimum/Maximum Query
- 区间最值查询问题
- 就是对一个长度为N的序列A，回答Q次询问
- 每次询问RMQ（i，j）返回数列中下标在i，j之间的最小/大值。

什么是RMQ

- 例如序列：
- 233 12450 801 666 1096 7777777
- RMQ询问最小值，则
- RMQ（1,3）= 233
- RMQ（3,6）= 666
- RMQ（3,3）= 801

怎么做RMQ

- N=1000，Q=1000
- 暴力O（NQ）就行
- N=30000，Q=30000
- 我会分块！把整个数组分成若干块，每块预处理，查的时候块内暴力块间直接搞（分块是一个非常重要的思想！以后要好好学的）
- 实践证明分成sqrt(N)块是最高效的，总复杂度O(N+Qsqrt(N))　　———CXY
- 可惜我不会
- N=300000，Q=300000
- ？？？

RMQ问题

- 现在连分块都会T了，怎么办
- ……
- 线段树呀
- ……
- 什么是线段树?

例子：洛谷P1816

忠诚 https://www.luogu.org/problemnew/show/P1816

例子：洛谷P1816

- 题目描述：
  老管家是一个聪明能干的人。他为财主工作了整整10年，财主为了让自己账目更加清楚，要求管家每天记k次账。由于管家聪明能干，因而管家总是让财主十分满意。但是由于一些人的挑拨，财主还是对管家产生了怀疑。于是他决定用一种特别的方法来判断管家的忠诚，他把每次的账目按1，2，3...编号，然后不定时的问管家问题，问题是这样的：在a到b号账中最少的一笔是多少？为了让管家没时间作假他总是一次问多个问题。

- 输入格式：
  输入中第一行有两个数m,n表示有m(m<=100000)笔账,n表示有n个问题，n<=100000。
  第二行为m个数,分别是账目的钱数
  后面n行分别是n个问题,每行有2个数字说明开始结束的账目编号

例子：洛谷P1816

- RMQ裸题
- 整棵取最小值的线段树即可
- 那问题就是怎么种这棵树了

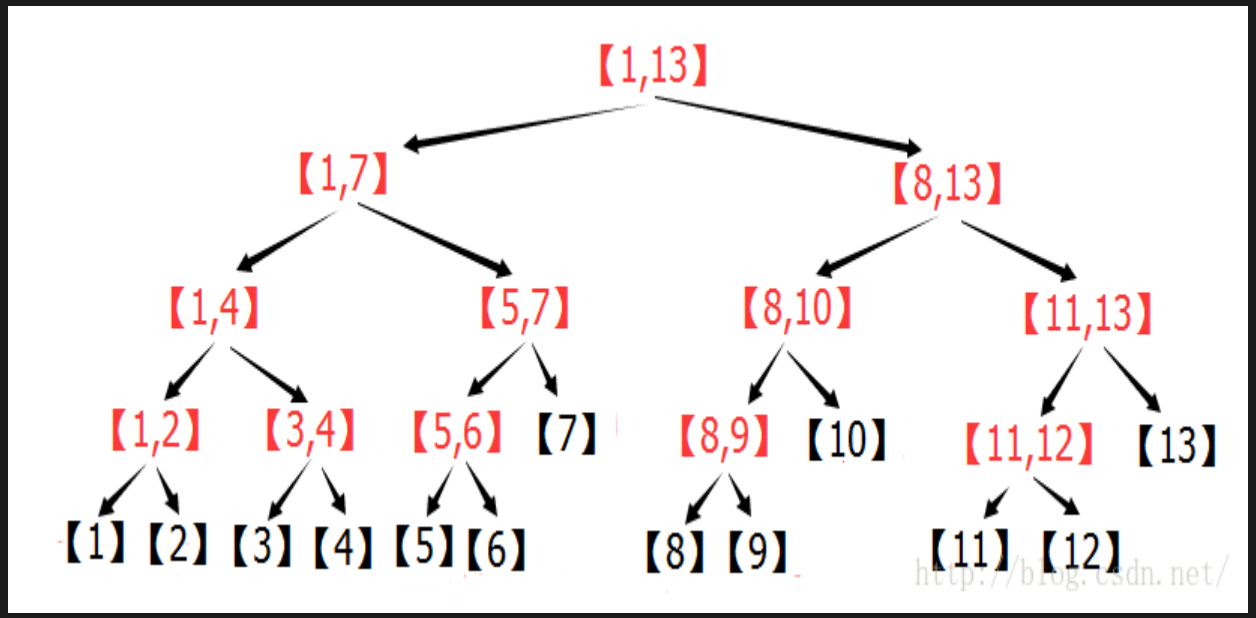
实战：建树

build

实战：建树

- 我们可以把这个序列划分为若干区间
- 而且，如果要取区间最小值
- 那么很容易由小区间的最小值的答案算出更大区间的最小值的答案
- 然后我们对区间进行二分：

实战：建树



这里是一个1到13的区间划分
黑色的都是单点，一般就是
直接对应原数组的值

实战：建树

- 从上图可以看出，如果我们可以把这个关系看做一棵树，我们也可以通过拆分合并一些节点的答案来获得我们想要的答案
- 所以我们只需维护一个简单的儿子父亲关系，对于每一个非叶子节点:
- t[now]=min(t[lson(now)] , t[rson(now)])
- 就好了
- 树的存储我们可以搞一个二叉树的数组式表示

实战：建树

- 什么是二叉树的数组式表示?
- emmmmm
- 树根记为1，第I号节点的左儿子和右儿子分别是2I和2I+1
- 这样就能用数组下标来精确表示二叉树了
- 然后乘2就能找到左儿子
- 乘2加一就能找到右儿子
- 除2就能找到爸爸
- 变成这样 t[now]=min(t[now<<1] , t[now<<1|1])

实战：建树

- 显然我们要递归着搞
- 递归函数这么声明：
- build(int l,int r,int now)
- l和r表示当前的线段区间
- now是当前在二叉树数组表示上的节点号

实战：建树

- 然后开搞
- 显然如果单点的话，直接赋值就好了
- 然后不是单点
- 建左子树
- 建右子树
- 更新节点信息
- 没了
- 就这么点

```
void build(int l,int r,int now){
if(l==r) {
  t[now]=a[l];
  return ;
}
int mid=(l+r)>>1;  //位运算加速
build(l,mid,now<<1);
build(mid+1,r,now<<1|1);
refresh(now);
}
```

实战：建树

- Refresh函数用来维护当前父子关系正常，就这么写就好了
- inline void refresh(int now){ //inline玄学加速
  t[now]=min(t[now<<1],t[now<<1|1]);
  }
- 会发现建树的时候调用a[]的顺序其实是一个一个调用的
- 可以把那句赋值换成scanf
- 至此建树彻底结束

实战：查询

query

实战：查询

- 我们刚才讲了，根据小区间的最值我们搞起来就得到了大区间的最值
- 那么这个就可以这么操作
- 把区间一点一点分开，然后搞就行了
- 因为树不超过LogN这么高所以单次区间查询可以搞到O(LogN)

实战：查询

- int query(int l, int r, int now, int LL, int RR)
- 5个参数的递归函数
- l,r,now不解释 同build
- LL RR是查询区间

实战：查询

- 如果完全包含直接返回最值
- 不然就直接劈开二分递归好了
- 就这么点

```
int query(int l, int r, int now, int LL, int RR){
    if(LL<=l&&r<=RR) return t[now];
    int mid=(l+r)>>1;
    int ans=0;
    if(LL<=mid) ans=min(ans, query(l,mid,now<<1,LL,RR)) ;
    if(mid<RR) ans=min(ans, query(mid+1,r,now<<1|1,LL,RR)) ;
return ans;
}
```

实战：查询

- 然后就可以A掉洛谷P1816 忠诚了
- 这里给出我的挫Code：
- **https://www.luogu.org/paste/kkcwnmje**
- 直接点进去就能看
- 刚学线段树的时候写的，巨丑

实战：查询

- 另外说一句
- 线段树的树的数组要开4倍空间不然会RE
- 因为除了和原数组一一对应的叶子节点外还有一些非叶子结点
- 为什么是四倍？我也不知道

例子： 洛谷P3374 【模板】树状数组 1

【模板】树状数组1 https://www.luogu.org/problemnew/show/P3374

例子： 洛谷P3374 【模板】树状数组 1

- 你问今天学的线段树为什么要打树状数组的模板题?
- 因为没从洛谷上找到更合适的题
- 反正也能用线段树做

例子： 洛谷P3374 【模板】树状数组 1

- **题目描述**
- 已知一个数列，你需要进行下面两种操作：
- 1.将某一个数加上x
- 2.求出某区间每一个数的和
- 两处不同：
- 求最值变成了求区间和
- 带修改

例子： 洛谷P3374 【模板】树状数组 1

- 改为求区间和很简单，把refresh改成下面这样即可
- inline void refresh(int now){
  t[now]=t[now<<1]+t[now<<1|1];
- }
- query中同理
- 就是把所有的min（A，B）改为A+B

例子： 洛谷P3374 【模板】树状数组 1

- 带修改才是真正重要的地方
- 不带修改的静态查询最值可以用ST表解决，区间和可以用前缀和解决，~~跑的还都比线段树快~~
- 但带修改的情况就就得靠线段树了

实战：维护

Update

实战：维护

- 这里这个Updata函数其实也蛮好写的
- 只是维护单点的直接修改
- 维护函数updata(int l,int r,int now,int x,int data)
- l,r,now同build
- x是需要维护的下标
- data是要维护的数据
- 只要一路对着区间二分找到x下标修改然后无脑refresh就好了

实战：维护

- 如果找到正确的单点就换数
- 否则取mid
- 二分区间
- Refresh
- （刚才也讲了怎么Refresh了
- 然后就没了
- 没错线段树单点维护就这点

-
```
updata(int l,int r,int now,int x,int data){
if(l==r){
    T[now]=data;
    return ;
  }
  int mid=(l+r)>>1;
  if(x<=mid)

updata(l,mid,now<<1,x,data);
  else

updta(mid+1,r,now<<1|1,x,data);
  refresh(now);
}
```

实战：维护

- 然后你就可以把洛谷P3374 过了
- 附上code
- https://www.luogu.org/paste/g42dmkz5
- 至此，恭喜你，你已经学会了线段树的最基本的内核了
- 现在让我们来点刺激的

实战：洛谷3372

线段树练习1

实战：洛谷3372

- 已知一个数列，你需要进行下面两种操作：
- 1.将某区间每一个数加上x
- 2.求出某区间每一个数的和

实战：洛谷3372

- 建树和查询没有区别
- 注意返回值要设为long long，会爆int

实战：洛谷3372

```
ll query(int l,int r,int now,int LL,int RR){
    if(LL<=l&&r<=RR){
        return t[now];
    }
    if(tag[now]) pushdown(l,r,now);
    int mid=(l+r)>>1;
    ll ans=0;
    if(LL<=mid) ans+=query(l,mid,now<<1,LL,RR);
    if(mid+1<=RR) ans+=query(mid+1,r,now<<1|1,LL,RR);
    refresh(now);
    return ans;
}
```

欸那个Pushdown是啥?

稍等，我马上就讲

像这样施展一发就行了

实战：洛谷3372

- 重头戏来啦！如何进行区间修改？
- 暴力？O（QNLogN)哇我还不如用数组……
- 这时候某个懒的要死的大佬（显然不是我）想出来了一个和他一样懒的妙招：懒惰标记！

实战：洛谷3372

- 显然我们一次更新不完所有的
- 我们考虑每个节点新增加一个标记，记录这个节点是否进行了某种修改(这种修改操作会影响其子节点)
- 对于任意区间的修改，我们先按照区间查询的方式将其划分成线段树中的节点，然后修改这些节点的信息并给这些节点标记上代表这种修改操作的标记。
- 在修改和查询的时候，如果我们到了一个节点p，并且决定考虑其子节点，那么我们就要看节点p是否被标记，如果有，就要按照标记修改其子节点的信息，并且给子节点都标上相同的标记，同时消掉节点p的标记。

手玩时间

实战：洛谷3372

- 然后我们就可以引入一个标记数组Tag[]
- 然后我们先试着写写标记下传的方法Pushdown：
- void Push_down(int l,int r,int now)
- l 和r是now对应的线段，now不用解释了吧

实战：洛谷3372

- 如果是叶子直接标记赋零，然后返回（不做这步会RE）
- 然后给左右加上Tag
- 然后给儿子按照长度加和
- 最后消除标记
- 完成下传

```
void Pushdown(int l,int r,int now)
{
   if(l==r) {Tag[now]=0;return ;}
   Tag[now<<1]+=Tag[now];
   Tag[root<<1|1]+=Tag[now];
   int mid=(l+r)>>1;
   T[now<<1]+=(mid-l+1)*Tag[now];
   T[now<<1|1]+=(r-mid)*Tag[now];
   Tag[now]=0;
}
```

实战：洛谷3372

- 每次处理当前节点的时候都要Pushdown一次，只要这个点存在Tag
- 至于修改……仿照区间查询来搞

```
void add(int l,int r,int now,int LL,int RR,int data){
    if(LL<=l&&r<=RR){
        tag[now]+=data;
        t[now]+=(r-l+1)*data;
        return;
    }
    if(tag[now]) pushdown(l,r,now);
    int mid=(l+r)>>1;
    if(LL<=mid) add(l,mid,now<<1,LL,RR,data);
    if(mid+1<=RR) add(mid+1,r,now<<1|1,LL,RR,data);
    refresh(now);
    return;
}
```

实战：洛谷3372

- 注意开long long……不然会WA
- 附上Code：
- **https://www.luogu.org/paste/l7vmxh7m**
- NOIP前打的，码风自认为还行

高级线段树：洛谷3373

线段树练习2

高级线段树：洛谷3373

- 洛谷3373：
- 如题，已知一个数列，你需要进行下面三种操作：
- 1.将某区间每一个数乘上x
- 2.将某区间每一个数加上x
- 3.求出某区间每一个数的和
- 对P取膜

高级线段树：洛谷3373

- 显然这个我们需要两个懒惰标记，一个乘法懒惰标记mulTag，一个加法懒惰标记addTag。
- 但是我们怎么处理好这两个标记之间的关系就成了难点。
- 显然加法优先级比乘法低
- 所以如果我们要对某区间进行加法操作的时候，由于加法优先级低，不会对乘法操作产生影响，故直接相加即可。
- 当对某区间执行乘法操作时，由于乘法优先级高，会对之前的加法操作产生影响，故需要在相乘时不仅对sum和mulTag相乘，也需要对addTag相乘。

高级线段树：洛谷3373

- Push_down的具体操作：
- 1.子树的sum、mulTag、addTag值分别乘上当前节点的mulTag；
- 2.当前节点的mulTag还原，即置为1；//别手残写成0
- 3.子树的addTag加上当前节点的addv值；
- 4.子树的sum加上（子树包含元素数量*当前节点的addTag）；
- 5.当前节点的addTag还原，即置为0。

高级线段树：洛谷3373

- 不多，实现起来
- 也就这点：
- 其中tag1为
- 乘法标记
- Tag2为
- 加法标记

```
void pushdown(int l,int r,int now){
    if(l==r){
        tag1[now]=1;
        tag2[now]=0;
        return;
    }
    tag1[now<<1]*=tag1[now];tag1[now<<1]%=p;
    tag1[now<<1|1]*=tag1[now];tag1[now<<1|1]%=p;
    tag2[now<<1]*=tag1[now];tag2[now<<1]%=p;
    tag2[now<<1|1]*=tag1[now];tag2[now<<1|1]%=p;
    t[now<<1]*=tag1[now];t[now<<1]%=p;
    t[now<<1|1]*=tag1[now];t[now<<1|1]%=p;
    int mid=(l+r)>>1;
    tag2[now<<1]+=tag2[now];tag2[now<<1]%=p;
    tag2[now<<1|1]+=tag2[now];tag2[now<<1|1]%=p;
    t[now<<1]+=(ll)((mid-l+1)*tag2[now])%p;t[now<<1]%=p;
    t[now<<1|1]+=(ll)((r-mid)*tag2[now])%p;t[now<<1|1]%=p;
    tag1[now]=1;
    tag2[now]=0;
}
```

高级线段树：洛谷3373

- 区间加同3372
- 区间乘的时候同区间加，只是要多修改一个AddTag。
- 注意取模。

高级线段树：洛谷3373

```
void updata2(int l,int r,int now,int LL,int RR,int data){
    if(LL<=l&&r<=RR){
        tag2[now]+=(ll)data;tag2[now]%=p;
        t[now]+=(ll)(r-l+1)*data;t[now]%=p;
        return;
    }
    if(tag1[now]!=1||tag2[now]) pushdown(l,r,now);
    int mid=(l+r)>>1;
    if(LL<=mid) updata2(l,mid,now<<1,LL,RR,data);
    if(mid<RR) updata2(mid+1,r,now<<1|1,LL,RR,data);
    refresh(now);
}
```

高级线段树：洛谷3373

```
void updata1(int l,int r,int now,int LL,int RR,int data){
    if(LL<=l&&r<=RR){
        tag1[now]*=(ll)data;tag1[now]%=p;
        tag2[now]*=(ll)data;tag2[now]%=p;
        t[now]*=(ll)data;t[now]%=p;
        return;
    }
    if(tag1[now]!=1||tag2[now]) pushdown(l,r,now);
    int mid=(l+r)>>1;
    if(LL<=mid) updata1(l,mid,now<<1,LL,RR,data);
    if(mid<RR) updata1(mid+1,r,now<<1|1,LL,RR,data);
    refresh(now);
}
```

高级线段树：洛谷3373

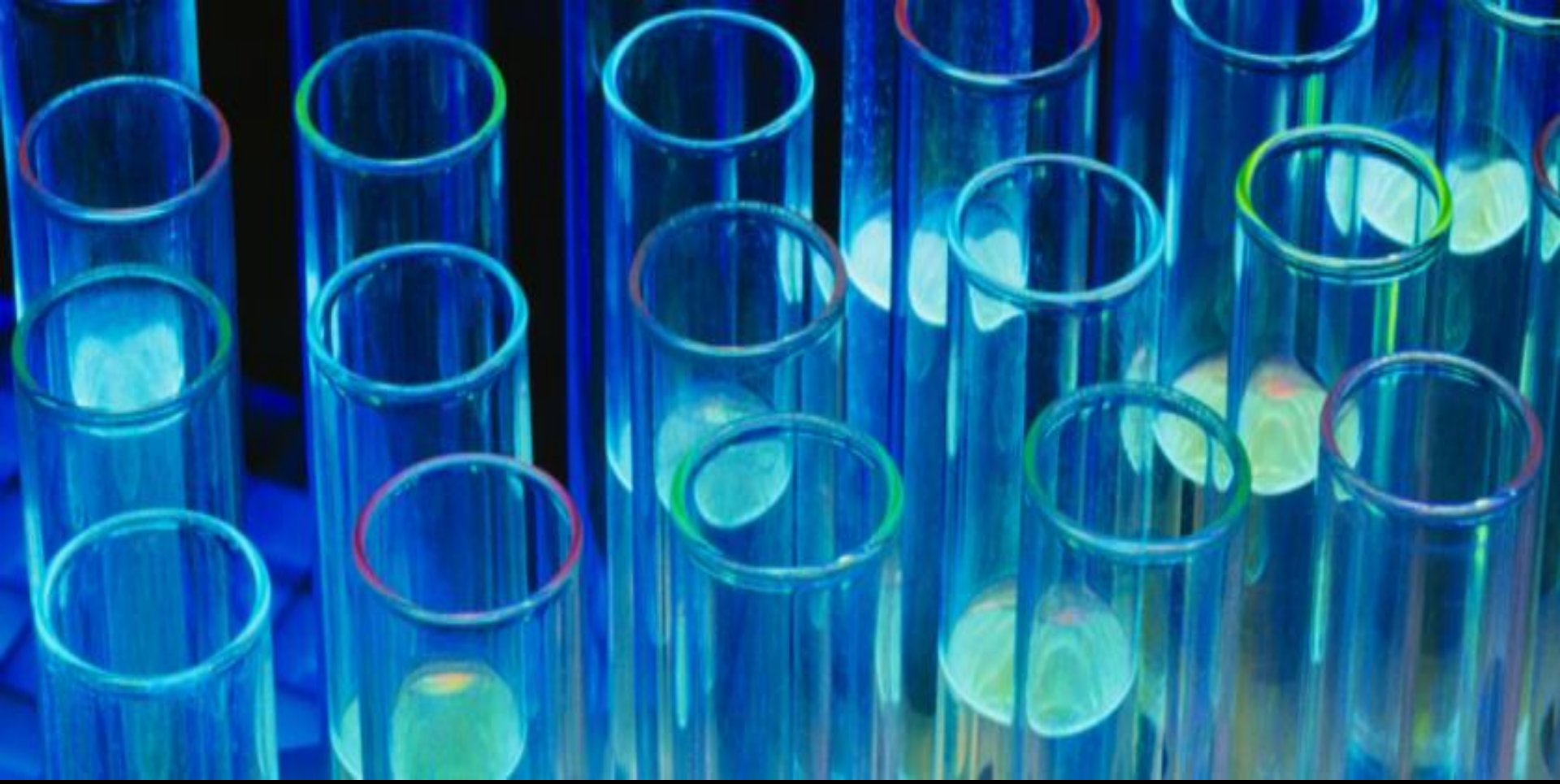
```
ll query(int l,int r,int now,int LL,int RR){
    if(LL<=l&&r<=RR){
        return t[now];
    }
    if(tag1[now]!=1||tag2[now]) pushdown(l,r,now);
    int mid=(l+r)>>1;
    ll ans=0;
    if(LL<=mid) ans=(ans+query(l,mid,now<<1,LL,RR))%p;
    if(mid<RR) ans=(ans+query(mid+1,r,now<<1|1,LL,RR))%p;
    refresh(now);
    return ans;
}
```

高级线段树：洛谷3373

- 然后就是日常贴代码时间了：
- https://www.luogu.org/paste/dvm10x06

课后练习题

- 简单题：p1198
- 难题:p2572 sp1043
- 有趣的题：p1972 p4145
- 我也不知道难不难的题:p2574



THE END

完结