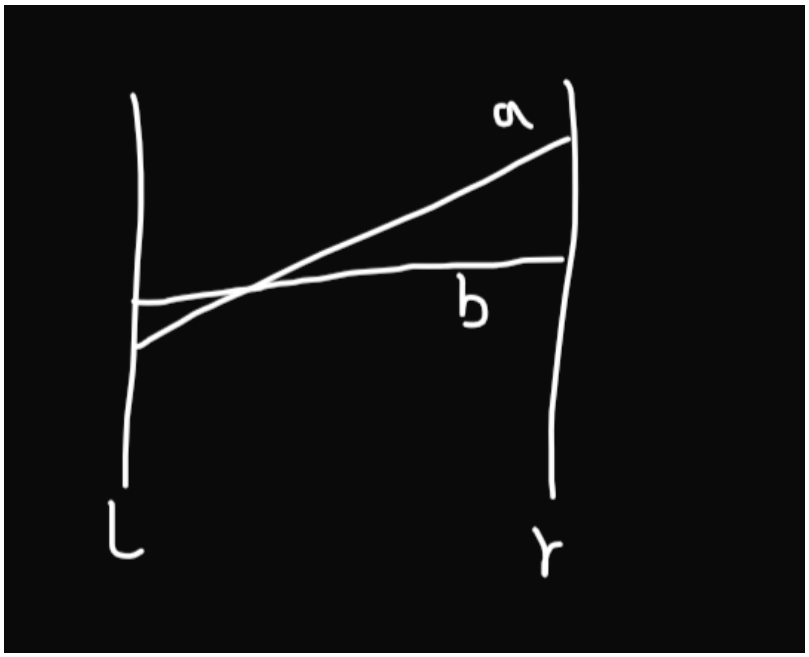
李超树

它能干什么

- 维护区间的多条直线
- 单点查询最值
- 区间查询最值

它是什么

根据用途不难看出，李超树是一种可以维护区间“ 优势线段 ”的线段树。至于优势线段，通俗地讲，从上向下看能看到覆盖长度最多的线段，如图：



其中a即为当前区间的“ 优势线段 ”。

相关操作

李超树作为线段树的变种，修改、查询和普通线段树大同小异。

杂项

在说具体操作前，先把需要的函数和结构体说一说，我们需要的结构体有两种，一个记录直线，另一个记录区间。而对于区间来说，我们又需要对区间编号，从而便于查询修改时查找优势线段。 直接上代码了：

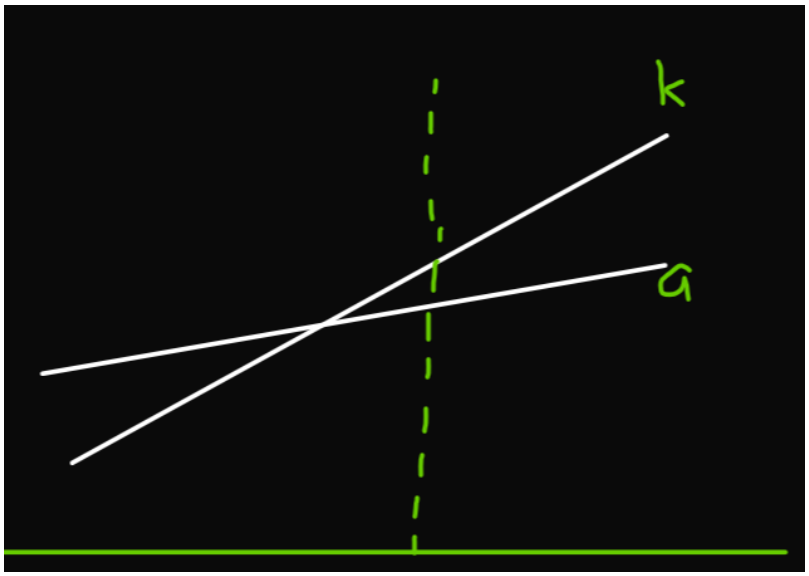
```
struct line{
    long long k,b;
    int l,r;
};
```

```
struct node{
    line t;
    int flag;
}a[MAX_N<<1];
line ca;
int get_id(int l,int r){
    return (l+r)|(l!=r);
}
long long f(line t,long long pos){
    return t.k*pos+t.b;
}
```

修改

先判断当前区间是否存在优势线段，如果没有的话直接插入就可。

假设当前已经有了优势线段a□对于待插入直线k□首先判断左右两端的纵坐标，如果都大/小于a□就可以直接判断；如果两端大小关系不一致，就判断中点位置的纵坐标。



假设虚线即为当前区间的终点，如果该处k的函数值较大，显然优势线段即为k□对于线段a□显然在中点左侧a仍有可能成为优势线段，而右侧则根本不可能，那只需要将a和k $swap$□然后处理左侧部分即可。如果中点处k的函数值较小，跳过$swap$直接向下即可。

代码：

```
void change(int l,int r,line k){
    int now=get_id(l,r);
    if(k.l<=l&&r<=k.r){
        if(!a[now].flag){//直接替换
            a[now].flag=1;
            a[now].t=k;
        }
        else if(f(k,l)>=f(a[now].t,l)&&f(k,r)>=f(a[now].t,r))//是否其中一条完全覆盖另一条
```

```
            a[now].t=k;
        else if(f(k,l)>f(a[now].t,l)||f(k,r)>f(a[now].t,r)){
            int mid=(l+r)>>1;
            if(f(k,mid)>f(a[now].t,mid))//判断中点处大小
                swap(a[now].t,k);
            if(f(k,l)>f(a[now].t,l))
                change(l,mid,k);
            else
                change(mid+1,r,k);
        }
    }
    else{
        int mid=(l+r)>>1;
        if(k.l<=mid)
            change(l,mid,k);
        if(mid<k.r)
            change(mid+1,r,k);
    }
}
```

查询

单点

这里先谈谈最大值吧，直接搬普通线段树即可。

代码：

```
long long query(int l,int r,int pos){
    int now=get_id(l,r);
    if(l==r)
        return f(a[now].t,pos);
    int mid=(l+r)>>1;
    long long ans=f(a[now].t,pos);
    if(pos<=mid)
        return max(ans,query(l,mid,pos));
    else
        return max(ans,query(mid+1,r,pos));
}
```