slope trick

算法简介

一种维护凸包的技巧。

算法实现

假定一个函数 $f(x)$ 是连续函数，可以被划分为多条直线，直线的斜率单调递增/递减，则可以用最终直线 $g(x)$ 和可重集 S 来表示 $f(x)$。

例如 $f(x)=|x-1|$ 可以表示为 $(x-1,\{1,1\})$。$f(x)=|2x-4|+|3x|$ 可以表示为 $(5x-4,\{0,0,0,0,0,0,2,2,2,2\})$。

不难发现，设 $f(x)=(k_1x+b_1,S_1),g(x)=(k_2x+b_2,S_2)$ 且 $f(x),g(x)$ 的斜率都是单调性相同时，有

$$ f(x)+g(x)=((k_1+k_2)x+b_1+b_2,S1\cup S_2) $$

算法例题

例题一

CF713C

题意

给定序列 A，每次操作可以选中一个位置 i，令 $a_i\gets a_i+1$ 或 $a_i\gets a_i-1$，问使得序列严格单增的最小操作次数。

题解

令 $a_i\gets a_i-i$，于是问题转化求使得序列非递减的最小操作次数。

设 $f_i(x)$ 表示使得 $a[1\sim i]$ 非递减且 $a_i\le x$ 的最小操作次数，$g_i(x)$ 表示使得 $a[1\sim i]$ 非递减且 $a_i=x$ 的最小操作次数。

于是有 $g_i(x)=f_{i-1}(x)+|x-a_i|$，由于 f_i,f_{i-1} 斜率最大值为 0，g_i 斜率最大值为 1，因此为了得到 f_i，需要删除 g_i 的 S 的最大元素。

设 g_i 的 S 中的最大值为 k，则该操作相当于把 g_i 的最后一段改成水平线，则

$$ f_i(x)=(y=g_i(k),S-\{k\})=(y=f_{i-1}(k)+|k-a_i|,S-\{k\}) $$

不难发现 k 一定位于 $f_{i-1}(x)$ 的最后水平线那一段，于是只需要维护所有 $f_i(x)$ 的最后一段的截

距即可，同时这也是 $f_i(x)$ 的最小值。

```
int main()
{
    int n=read_int();
    priority_queue<int> q;
    LL ans=0;
    _for(i,0,n){
        int a=read_int()-i;
        q.push(a);
        if(a<q.top()){
            ans+=q.top()-a;
            q.pop();
            q.push(a);
        }
    }
    enter(ans);
    return 0;
}
```

例题二

CF1534G

题意

给定一个人物，初始点为 $(0,0)$，人物每步可以从 (x,y) 移动到 $(x+1,y)$ 或 $(x,y+1)$。

有 n 个物品，坐标为 (x_i,y_i)。人物在坐标 (x,y) 获得物品 i 的代价为 $\max(|x-x_i|,|y-y_i|)$。

人物可以在任意位置选取任意数量的物品获取，问人物获取所有物品的最小代价。

题解

首先，假定人物的轨迹已经固定，对每个物品，考虑将物品按正方形区域扩大，直到与轨迹相交，显然在交点取该物品最佳。

不难发现，这等价于当人物坐标位于 $y=x_i+y_i-x$ 直线上时取第 i 个物品，此时代价为 $|x-x_i|$。

设 $\text{dp}(k,x)$ 为人物位于坐标 $(x,k-x)$ 且已经获取所有 $x_i+y_i\le k$ 的物品的最小代价。

将所有物品按 $z_i=x_i+y_i$ 排序，于是有

$$ \text{dp}(z_i,x)=\min_{j\le z_i-z_{i-1}}\text{dp}(z_{i-1},j)+|x-x_i| $$

发现状态转移分为两步，第一步是 $\text{dp}(z_i,x)\gets \min_{j\le z_i-z_{i-1}}\text{dp}(z_{i-1},j)$。

考虑将函数 $\text{dp}(k,x)$ 分成斜率小于 0，斜率等于 0 和斜率大于 0 三部分。

不难发现这步只需要将斜率大于 0 部分整体向右偏移 z_i-z_{i-1} 个单位，然后中间空缺部分用斜率等于 0 的部分补全即可。

考虑分别用两个集合维护函数 $\text{dp}(k,x)$ 斜率小于 0 部分和斜率大于 0 部分的 S。

第二步是 $\text{dp}(z_i,x)\gets \text{dp}(z_i,x)+|x-x_i|$。也是分 x_i 位于原函数斜率小于 0，斜率等于 0 和斜率大于 0 三部分讨论。

然后更新斜率小于 0 部分和斜率大于 0 部分的 S 和斜率等于 0 部分的答案即可。

例如当 x_i 属于斜率小于 0 的部分时，则原函数斜率等于 0 的部分斜率变为 $+1$，但是斜率等于 0 的部分的左端点仍是最优解。

于是左端点取值 $+|x-x_i|$ 就是新答案，然后将原来的左端点从斜率小于 0 部分的 S 移除，加入斜率大于 0 部分的 S。

最后将两个 x_i 插入斜率小于 0 部分的 S 就完成了更新。总时间复杂度 $O(n\log n)$。

```
const int MAXN=8e5+5;
pair<int,int> a[MAXN];
int main()
{
    int n=read_int();
    _for(i,0,n){
        a[i].first=read_int();
        a[i].second=read_int();
        a[i].first+=a[i].second;
    }
    sort(a,a+n);
    LL ans=0;
    priority_queue<int> q1;
    priority_queue<int,vector<int>,greater<int> > q2;
    q1.push(a[0].second);
    q2.push(a[0].second-a[0].first);
    _for(i,1,n){
        if(a[i].second<q1.top()){
            int t=q1.top();q1.pop();
            ans+=t-a[i].second;
            q1.push(a[i].second);q1.push(a[i].second);
            q2.push(t-a[i].first);
        }
        else if(a[i].second<q2.top()+a[i].first){
            q1.push(a[i].second);
            q2.push(a[i].second-a[i].first);
        }
        else{
            int t=q2.top()+a[i].first;q2.pop();
            ans+=a[i].second-t;
            q2.push(a[i].second-a[i].first);q2.push(a[i].second-
a[i].first);
```

```
                q1.push(t);
            }
        }
        enter(ans);
        return 0;
}
```