蒟蒻的福利?教你用set维护斜率优化凸包

斜率优化题目大家肯定都做得不少了，有一些题目查询插入点的x坐标和查询斜率都不单调，这样就需要维护动态凸包并二分斜率。(例如bzoj1492)

常规的做法是cdq分治或手写平衡树维护凸包，然而如果我不愿意写分治，也懒得打平衡树，怎么办呢？

没关系，今天我告诉你怎么用一个set维护这种凸包。

首先orzLH，没什么特殊意义，只是单纯的orz。

我们定义f[i]表示在第i天能拥有的金券组数，按照第i天的比例。

那么，我们要把前面的金券在今天卖出获得最多的钱，并在今天进行买入。

所以，f[i]=max((f[j]\*a[i]+f[j]/rate[j]\*b[i])/(a[i]+rate[i]\*b[i]))。

除下去的东西是一个常数，扔掉。

然后我们就有:

t=max(a[i]\*(f[j])+b[i]\*(f[j]/rate[j]))。

如果我们把f[j]看做x，f[j]/rate[j]看做y，我们有:

t=a\*x+b\*y，两边同时除以b，得到:

t/b=(a/b)x+y

y=(t/b)-(a/b)\*x

好的，现在我们有一条斜率为-(a/b)的直线，要找一个点使之截距最大。

这样我们维护一个右上1/4凸壳即可。

怎么维护?

我们考虑不用斜率优化，单纯水平序维护凸包，那么点是按照x坐标单增在平衡树上排列的。

现在我们在每个点维护他与后面点连线斜率，我们会发现:这个斜率是单降的。

所以，我们可以通过适当地转换cmp函数，来通过一个set完成两种比较。

我们定义:

插入就是正常凸包插入，最后再维护一下斜率就行了。注意弹出左边后迭代器会失效，所以需要重新lower\_bound一下。(可能原来你的迭代器是原来的end，结果弹出左边后end改变了，两个end不同，然后你去弹出右边，访问无效迭代器，就直接RE了)

查询的话就更改一下比较函数，然后特判一下边界防止RE就好。

所以整体代码:

Bzoj1492:

不得不说STL的set跑的还是挺快的。

这里是回档后的世界，无论你做什么，你都一定会这样做。

而我努力改变命运，只是为了防止那一切重现。

(来自某中二病晚期患者(不))