# CDQ分治

Cdq分治主要思想，就是每次将一个区间分成两半，每次用前一半来更新后一半。

Int cdq(int l,int r){

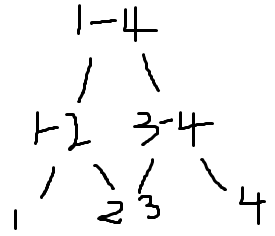
Cdq(l,mid);

Update(mid+1,r);

Cdq(mid+1,r);

}

通常这个区间是指操作时间。



假设区间为(1,4),每个点都可以从编号比他小的地方转移得到，那么我首先处理1,然后用1更新2,这时候1，2都是完整的答案，那么我用1-2来更新3和4,再处理3，用3更新4.over

cdq时间复杂度为logn\*（左半边更新右半边的复杂度）

## HDU 3842

本题是利用cdq分治  实现斜率优化的一个题目

斜率优化之前做的几个题都是斜率单调，并且插入点时由于点在某一维单调，所以仅仅操作队首和队尾就能完成优化了

但是本题显然不是

参考了两个东西

从《Cash》谈一类分治**[算法](http://lib.csdn.net/base/datastructure" \o "算法与数据结构知识库" \t "_blank)**的应用

(Day1)cdq分治相关

本题的题意是

一个公司获得了一个厂房n(10^5)天的使用权  
和一笔启动资金C(10^9)，准备在n天里租借机器生产来获得收益  
可以租借的机器有M(10^5)个，每个机器有四个值，D,P,R,G (D<=n, P,R,G都是10^9)  
表明你可以再第D天花费P费用（首先手里必须有那么多钱）  
租借这个机器，从D+1天开始该机器每天产生G的收益，在你不需要机器时  
可以卖掉这个机器，一次获得R的钱  
需要注意的是：  
厂房里只能停留一台机器  
不能再购买和卖出机器的那天操作机器，但是可以再同一天卖掉一台机器再买一台  
在第n+1天，必须卖掉手里的机器  
问n+1天后能获得的最大资金

根据这个题意

我们可以得到一个dp转移方程

首先要想的问题是是否有场地就要放机器

最开始的时候肯定不是这样，因为怕买到坑的机器很可能会亏钱，但是假设你买到了一台好的机器，在下一个机器进来之前，你肯定是一直运转下去的

然后得把所有的时间都离散化，就是每个机器的D

然后用f[i]表示在D[i]时刻卖掉手里的机器手里最多多少钱

f[i] = max(f[i - 1], f[j] - P[j] + R[j] + G[j] \* (D[i] - D[j]  - 1))

其中f[j] >= P[j]

可以看出是O(n^2)的，显然不行啊

令h[j] = f[j] + R[j]- P[j] - G[j] \* (D[j] + 1)

式子就变成  f[i] = h[j] + D[i] \* G[j]

即h[j] = -D[i] \* G[j] + f[i]

对于这个， 可以抽象成一个二维空间

由(G[j],h[j])作为点集, -D[i]为斜率

然后求使得截距最大的那个，就是f[i]最大了

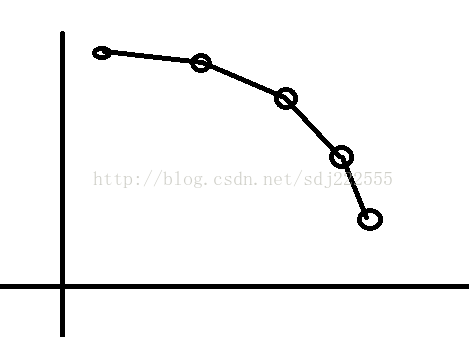
观察这些点集

可以发现，一点都不单调啊，就需要按照G[j]这一维做个排序

使得他至少在一维上单调，方便我们做插入和删除操作

然后由于斜率是单调递减的，并且是负数

可以画一画，最后要维护的最优点的点集，在图上形成的是一条上凸的线



然后每插一个点都是维护这个图形

然后这个排序，你在普通的DP方法里肯定是不能每次都去排序的

这就需要cdq分治了

对于一个区间l,r

你先更新了l,mid

然后对左边这部分的区间的点集， 进行排序，形成上面这个图形

然后更新右边区间的时候，

由于斜率是递减的，你可以发现，我们只需要扫一遍这个图形即可完成所有右边区间值的更新

然后就这样递归着去更新

完成cdq分治

总体复杂度，应该是nlognlogn

因为每个子区间中都有这个排序

需要留意的是，在斜率之间做比较的时候，如果用乘法来比较的话，会溢出long long