单调队列优化 DP

一.　什么是单调（双端）队列

单调队列，顾名思义，就是一个元素单调的队列，那么就能保证队首的元素是最小（最大）的，从而满足动态规划的最优性问题的需求。

单调队列，又名双端队列。双端队列，就是说它不同于一般的队列只能在队首删除、队尾插入，它能够在队首、队尾同时进行**删除**。

【单调队列的性质】

一般，在动态规划的过程中，单调队列中每个元素一般存储的是两个值：

1. 在原数列中的位置（下标）
2. 他在动态规划中的状态值

而单调队列则保证这两个值同时**单调**。利用单调队列的性质可以直接解决一些问题。

例1. 志愿者选拔（fzu1894）

[问题描述]

世博会马上就要开幕了，福州大学组织了一次志愿者选拔活动。参加志愿者选拔的同学们排队接受面试官们的面试。参加面试的同学们按照先来先面试并且先结束的原则接受面试官们的考查。面试中每个人的人品是主要考查对象之一。（提高人品的方法有扶老奶奶过街，不闯红灯等）作为主面试官的 John 想知道当前正在接受面试的同学队伍中人品值最高的是多少。于是他请你帮忙编写一个程序来计算。

[输入格式]

输入数据第一行为一整数 T，表示有 T 组输入数据。

每组数据第一行为"START"，表示面试开始

接下来的数据中有三种情况：

	输入	含义
1	C NAME RP_VALUE	名字为 NAME 的人品值为 RP_VALUE 的同学加入面试队伍。(名字长度不大于 5，0 <= RP_VALUE <= 1,000,000,000)
2	G	排在面试队伍最前面的同学面试结束离开考场。
3	Q	主面试官 John 想知道当前正在接受面试的队伍中人品最高的值是多少。

最后一行为"END"，表示所有的面试结束，面试的同学们可以依次离开了。

所有参加面试的同学总人数不超过 1,000,000

[输出格式]

对于每个询问 Q,输出当前正在接受面试的队伍中人品最高的值,如果当前没有人正在接受面试则输出-1。

Sample Input

```
2
START
C Tiny 1000000000
C Lina 0
Q
G
Q
END
START
Q
C ccQ 200
C cxw 100
Q
G
Q
C wzc 500
Q
END
```

Sample Output

```
1000000000
0
```

```
-1
200
100
500
```

[分析]

本题中的数据自然形成一个队列。对于“G”操作，只需直接删除队首元素。对于“C”操作，是在队尾插入元素。由于题意是求队列中的最大值，因此如果待插入的元素 a[i]小于 que[tail]，那么当 a[i]之前的元素全部离开后，a[i]有可能成为队列中的最大值。这样，a[i]就需要入队候选。如果 a[i]大于等于 que[tail]，那么对于后续询问，a[i]都要比 que[tail]大，都会在询问中胜出。这样，队列中就没必要再保留 que[tail]，可以从队尾删除。进一步，a[i]一直与队尾元素比较，不断删除小于等于 a[i]的队尾元素，直到 que[tail]>a[i]。同理，队列中所有元素都应该满足“单调递减”的性质，这样就形成一个“单调递减队列”。显然，队首元素的值最大。这样，对于“Q”操作，可以直接读取队首元素的值就可以了。

这就是单调队列的简单应用。

例 2 队列的最小值 zjbti 1119

[问题描述]

一个含有 *n* 项的数列(*n*<=2000000)，求出每一项前面的第 *m* 个数到它这个区间内的最小值。

[分析]

这道题目，我们很容易想到线段树、或者 st 算法之类的 RMQ 问题的解法。但庞大的数据范围让这些对数级的算法没有生存的空间。我们先尝试用动态规划的方

法。用 $f(i)$ 代表第 i 个数对应的答案，$a[i]$ 表示第 i 个数，很容易写出状态转移方程：

$$f(i)=\min_{j=i-m+1}^{i}(a[j])$$

这个方程，直接求解的复杂度是 $O(nm)$ 的，甚至比线段树还差。这时候，单调队列就发挥了他的作用：

我们维护这样一个队列：队列中的每个元素有两个域{*position*,*value*}，分别代表他在原队列中的位置和 $a[i]$，我们随时保持这个队列中的元素两个域都**单调递增**。

那计算 $f(i)$ 的时候，只要在队首不断删除，直到队首的 *position* 大于等于 $i-m+1$，那此时队首的 *value* 必定是 $f(i)$ 的不二人选，因为队列是**单调**的！

我们看看怎样将 $a[i]$ 插入到队列中供别人决策：首先，要保证 *position* 单调递增，由于我们动态规划的过程总是由小到大（反之亦然），所以肯定在队尾插入。又因为要保证队列的 *value* 单调递增，所以将队尾元素不断删除，直到队尾元素小于 $a[i]$。

【时间效率分析】

很明显的一点，由于每个元素最多出队一次、进队一次，所以时间复杂度是 $O(n)$。用单调队列完美的解决了这一题。

【为什么要这么做】

我们来分析为什么要这样在队尾插入：为什么前面那些比 $a[i]$ 大的数就这样无情的被枪毙了？我们来反问自己：他们活着有什么意义？！由于 $i-m+1$ 是随着 i 单调递增的，所以对于 $\forall j<i, a[j]>a[i]$，在计算任意一个状态 $f(x),x>=i$ 的时候，

j 都不会比 i 优，所以 j 被枪毙是“罪有应得”。

我们再来分析为什么能够在队首不断删除，一句话：$i-m+1$是随着 i 单调递增的！

【一些总结】

对于这样一类动态规划问题，我们可以运用单调队列来解决：

$f(x)=\underset{i=bound[x]}{\overset{x-1}{opt}}(const[i])$ 其中 bound[x]随着 x **单调不降**，而 const[i]则是可以根据 i 在常数时间内确定的**唯一的常数**。这类问题，一般用单调队列在很优美的时间内解决。

例 3 股票交易(hdu3401)

[问题描述]

某人预测未来 T 天的股票交易。在第 i 天，你可以用 APi 元买入一股股票，也可以卖出一股股票收入 BPi 元。

另外还有一些限制。第 i 天最多只能买入 ASi 股股票，也最多只卖出 BSi 股的股票。两次交易的时间必须相隔 W 天。即如果第 i 天作了交易，下一次交易必须在 i+W+1 天及以后。而且，任何时候只能最多持有 MaxP 数量的股票。

在第一个交易日之前，假如你有足够多的钱，但没有股票。

现在问题是：经过 T 个交易日后，最多可以赚多少钱？

Input

The first line is an integer t, the case number.
The first line of each case are three integers T , MaxP , W .
(0 <= W < T <= 2000, 1 <= MaxP <= 2000) .
The next T lines each has four integers APi , BPi , ASi , BSi(1<=BPi<=APi<=1000,1<=ASi,BSi<=MaxP), which are mentioned above.

Output

The most money lxhgww can earn.

Sample Input

```
1
5 2 0
2 1 1 1
2 1 1 1
3 2 1 1
4 3 1 1
5 4 1 1
```

Sample Output

```
3
```

Source

[分析]

用 dp[i][j]表示第 i 天股票数为 j 的最多赚多少钱，dp[i][j]可以从以下三个地方得到：

dp[i][j]=max{不买不卖,买入操作,卖出操作}

如果不买不卖：

dp[i][j] = max{ dp[i][j], dp[i-1][j] }

如果买入：

dp[i][j] = max{ dp[r][k]-(j-k)*AP[i] } 0<r<i-w, 0<=j-k<=AS[i]

如果卖出：

dp[i][j] = max{ dp[r][k]+(k-j)*BP[i]} 0<r<i-w, 0<=k-j<=BS[i]

这样的时间复杂度为 O(T^2*MaxP^2)，需要优化。

对于买股票的时候，

dp[i][j] = max{ dp[r][k]-(j-k)*AP[i] }

dp[i][j] + j*AP[i] = max { dp[r][k] + k*AP[i] }

令 f(k) = dp[i][k]+k*AP[i]

f(j) = max{ f(k) } (j-AS[i]<=k<=j)

f(j)满足位置与值均单调的性质，可以用单调队列维护。

所以 dp[i][j]=f(j) - j*AP[i]

卖股票的情况：

dp[i][j] = max{dp[r][k]+(k-j)*BP[i] }

dp[i][j]+j*BP[i]=max(dp[r][k]+k*BP[i])

令 g(k)=dp[i][k]+k*AP[i]

g(j)=max{g(k)} (j<=k<=j+BS[i])

所以 dp[i][j]=g(j) - j*BP[i]

此时的时间复杂度为 O(T*MaxT).

例 4 生产产品 Product(vijos1243)

[问题描述]

有 n 个产品，编号为 1~n。要在 m 个机器人的手中生产完成。其中，第 i 个产品在第 j 个机器人手中的生产时间给定为 $T[i, j]$。要把这些产品按照编号从小到大生产，同一个机器人连续生产的产品个数不能够超过给定的常数 l。求生产完所有产品的最短时间是多少。其中 $n<=10^5$，$m<=5$，$l<=5*10^4$。

[分析]

这道题目，很容易想到一个动态规划的算法：用 $f[i, j]$ 表示前 i 个产品，其中第 i 个产品实在第 j 个机器人上完成的，前 j 个机器人生产完成所需最短时间。

$f[i, j] = \min_{k=i-l}^{i-1}(f[k, p] + sum[i, j] - sum[k, j])$，其中 $p<>j$。$sum[x,y]$ 表示产品 1 到 x 在 y 机器上生产所需的时间和。这样的算法时间复杂度是 $O(n*m^2*l)$，很明显不能在时限内完成，需要进行优化。

我们由于 j 的取值范围十分小：1<=j<=5。我们可以从这里突破。我们尝试把每一个 j **单独考虑**，比如现在只考虑 j=1 的情况：那每一个决策 k，他为当前要计算的状态所提供的值是 $Min(f[k, p]) + sum[i,1] - sum[k,1]$，$p<>j$。我们将其进行稍微的变形：$f[i,1] = Min(f[k, p] - sum[k,1]) + sum[i,1]$，可以发现括号内的式子是根据 k 所确定的一个常量，直接对应上文的单调队列的方法，省掉了枚举决策的一维，时间复杂度降为 $O(n*m^2)$。

习题：

1. SEQUENCE(HDU3530)

3. 瑰丽华尔兹(noi2005)

参考论文

http://www.docin.com/p-49960245.html

hdu2993
hdu2480