膜丹赛题解

T1

测试点 $1\sim 2$

爆搜每步合并还是删除即可。

测试点 $3\sim 5$

考虑到合并后再删除和直接删除本质相同，所以先枚举删除的集合是什么然后再暴力，可以通过这档分数。

测试点 $6\sim 8$

$\lfloor\frac{x}{2}\rfloor$ 相当于 x 右移一位，由于 $\lfloor\frac{x}{2}\rfloor||\lfloor\frac{y}{2}\rfloor=\lfloor\frac{x|y}{2}\rfloor$，把合并操作拆开，设 d_i 表示 a_i 参与的合并操作次数，那么答案就是 $\lfloor\frac{a_1}{2^{d_1}}\rfloor||\lfloor\frac{a_2}{2^{d_2}}\rfloor||\cdots||\lfloor\frac{a_n}{2^{d_n}}\rfloor$，在仅考虑合并操作的前提下一组 $\{d_i\}$ 的取值合法当且仅当 $\sum_{i=1}^n\frac{1}{2^{d_i}}=1$。

加上删除操作，如果一组 $\{d_i\}$ 的取值满足 $\sum_{i=1}^n\frac{1}{2^{d_i}}\geq 1$ 那么就存在一种构造方案使得答案小于等于 $\lfloor\frac{a_1}{2^{d_1}}\rfloor||\lfloor\frac{a_2}{2^{d_2}}\rfloor||\cdots||\lfloor\frac{a_n}{2^{d_n}}\rfloor$，因为必然存在一个集合 $S\subseteq\{1,2,\ldots,n\}$ 使得 $\sum_{i\in S}\frac{1}{2^{d_i}}=1$，将 S 外的 a_i 删除然后剩下的执行合并操作就可以使得答案更小了。这说明对于最终或起来得到的某个结果，$\sum_{i=1}^n\frac{1}{2^{d_i}}$ 越大越有可能成为答案。

于是就可以在这个基础上面 dp 选取 d_i 的取值。设 $dp(i,j)$ 表示考虑选取了所有小于 i 的数的 d，并且它们或起来得到的结果为 j 的前提下 $\sum_{i=1}^n\frac{1}{2^{d_i}}$ 的最大值，转移是 $\mathcal{O}(w)$ 的，状态数是 $\mathcal{O}(n2^w)$ 的，总时间复杂度为 $\mathcal{O}(Tn2^ww)$。

测试点 $9\sim 12$

考虑优化测试点 $6\sim 8$ 的 dp，发现答案不超过 $2^{w-\lfloor\log_2 n\rfloor}$，然后状态数优化为 $\mathcal{O}(n2^{w-\lfloor\log_2 n\rfloor})=\mathcal{O}(2^w)$，时间复杂度优化为 $\mathcal{O}(Tw2^w)$。

测试点 $13\sim 15$

根据测试点 $6\sim 8$ 中的分析，可以 $\mathcal{O}(nw)$ 判断对于某个数 x，答案按位与上 x 是否等于答案，考虑贪心地从高位到低位依次确定每一位填 0 还是 1，于是就可以做到 $\mathcal{O}(Tnw^2)$。

测试点 $16\sim 17$

在 a_i 全部可以表示为 2 的整数次幂的时候有一个简单的贪心，每次选择两个在集合内的最大的数，如果它们相等就合并，否则删掉较大的那个数，这么做显然是对的。

测试点 $18\sim 20$

使用一些位运算的技巧来优化掉测试点 $13\sim 15$ 中复杂度中多出来的一个 w，对每一个数 a_i 维护一个长度为 $w+1$ 的二进制数字 b_i，其中 b_i 的第 j 位为 1 当且仅当 a_i 右移 j 位和答案按位与起来不等于它本身，不难发现如果把答案的第 t 位由 1 改成了 0，那么所有的 b_i 都需要或上 a_i 右移 t 位，然后 b_i 最低的非 1 的位数就是此时 d_i 的最优取值，使用 hash 不难 $\mathcal{O}(1)$ 由已知 2^k 推出 k 的值是什么，于是时间复杂度降低为 $\mathcal{O}(Tnw)$。

T2

测试点 $1 \sim 2$

也许可以乱暴力过掉。

测试点 $3 \sim 5$

考虑从中间开始扩展得到回文路径，并且不难发现扩展经过的一定是一个连续区间，设 $f(i,j,l,r)$ 表示是否存在一个起点为 i 终点为 j 并且经过了 $[l,r]$ 这个区间中的所有字符的回文串，然后转移就是 bfs，状态数 $\mathcal{O}(n^4)$ 转移 $\mathcal{O}(1)$，时间复杂度 $\mathcal{O}(n^4 \times 1)$。

找到所有合法的区间 $[l,r]$ 之后然后设 $g(l,r)$ 表示 $[l,r]$ 这个子串是否“配得上丹”，转移继续 bfs，状态数 $\mathcal{O}(n^2)$ 转移 $\mathcal{O}(n^2)$，时间复杂度 $\mathcal{O}(n^2 \times n^2)$。

总的时间复杂度 $\mathcal{O}(n^4+q)$。

测试点 $6 \sim 9$

考虑假如一个串存在两个相等的字符中间只隔着一个字符，那么这个串就一定“配得上丹”，否则这个串中除了长度为 1 的回文串一定只存在偶回文串，是否“配得上丹”等价于是否存在偶回文覆盖（使用若干偶回文串覆盖所有字符）。**方便起见，之后说的所有回文串指的都是偶回文串。**

每次询问把串找出来，然后一遍 manacher 求出所有极长回文串，直接判断即可。

测试点 $10 \sim 13$

考虑分治，假设当前考虑的区间为 $[l,r]$，$m=\lfloor \frac{l+r}{2} \rfloor$，此时只需要考虑所有跨过 m 的询问 $[L,R]$。

考虑 $[L,R]$ 的一个回文覆盖，一定有至少一个回文串跨过了 m，按照这个回文串将这个回文覆盖分为两半，那么相当于是要从 L 开始选择若干个回文串从左到右一直覆盖到 $L_0 \geq m$，从 R 开始选择若干个回文串从右到做一直覆盖到 $R_0 \leq m+1$，然后 L 和 R 的覆盖可以拼起来当且仅当 $L_0 \leq R \wedge R_0 \geq L$。

由此可以看出我们是希望在所有合法的 L_0 中取到最小值，在所有合法的 R_0 中取到最大值。不妨设 $\mathrm{Right}(L)$ 和 $\mathrm{Left}(R)$ 分别表示所有合法的 L_0 中的最小值和所有合法的 R_0 中的最大值，由于这两个东西本质相同，所以只用考虑求其中的某一个，不妨考虑如何求 $\mathrm{Right}(L)$，有转移方程：

$$\mathrm{Right}(i)=\min(\min_{(i,j)\text{是一个回文串并且}j\geq m} j,\min_{(i,j)\text{是一个回文串并且}j<m} \min_{k=i+1}^{j+1} \mathrm{Right}(k))$$

可以发现我们只需要知道最短的满足 $j \geq m$ 的回文串 (i,j) 和最长的满足 $j<m$ 的回文串 (i,j)，这个可以在回文树上面倍增处理得到，转移使用线段树或者树状数组维护区间最小值，不难做到 $\mathcal{O}(n \log^2 n)$。

测试点 $14 \sim 15$

不难发现当一次询问 l,r 满足 $r-l \geq 3$ 的时候答案一定是 1，否则简单判一下即可。

测试点 $16 \sim 20$

考虑优化测试点 $10 \sim 13$ 的做法：

- $\mathrm{Right}(i)$ 的转移可以直接用单调栈维护做到线性。
- 如果一个回文串存在一个长度大于等于他长度一半的回文后缀，那么这个回文串显然是没有用的（可以通过这个后缀平铺得到），于是以某个位置为左端点或者右端点的有用回文串数量是

$\mathcal{O}(\log n)$ 级别的，并且分治下去每次需要找的回文串长度都是递减的，于是可以把倍增找回文串换成指针，均摊下来做到 $\mathcal{O}(n \log n)$ 。

总的时间复杂度 $\mathcal{O}(n \log n)$ 。

T3

测试点 $1 \sim 3$

这档留给玄学爆搜。

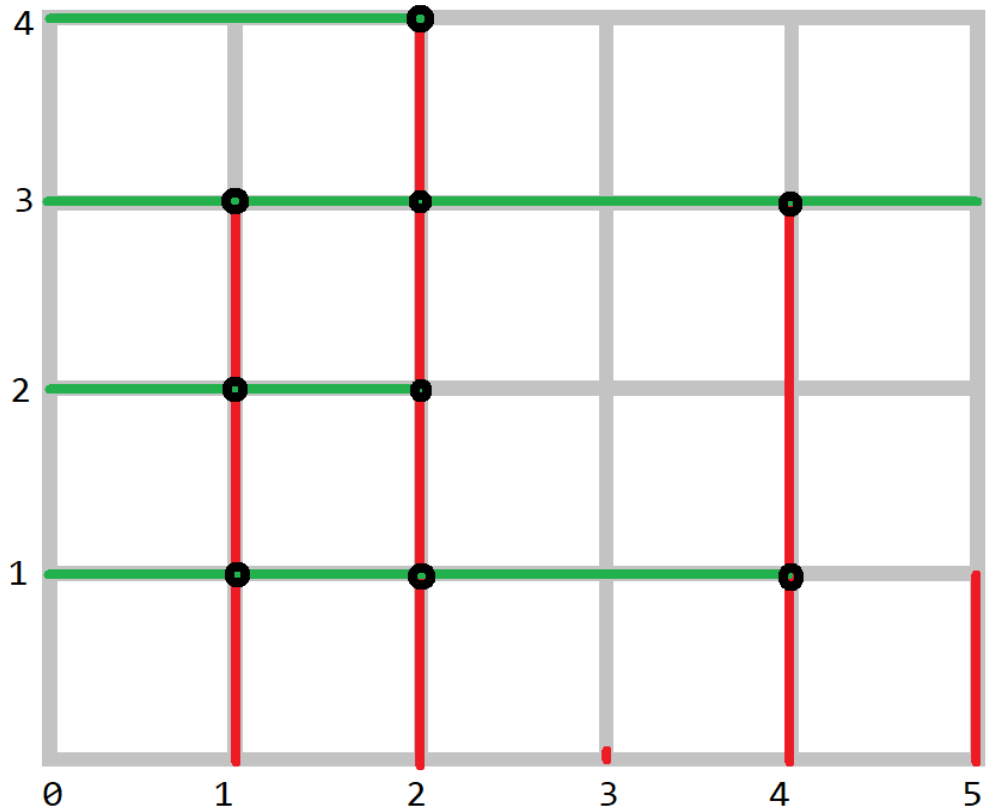
测试点 $4 \sim 6$

贪心选择环，不难发现只用考虑二元环就行了，于是问题就是在一张二分图上面做匹配，其中左边第 i 个点和右边第 j 个点有连边当且仅当 $a_i \geq j \wedge b_j \geq i$ ，匈牙利或者 dinic 都可以，时间复杂度 $\mathcal{O}(n^3)$ 或者 $\mathcal{O}(n^2 \sqrt{n})$ 。

测试点 $7 \sim 10$

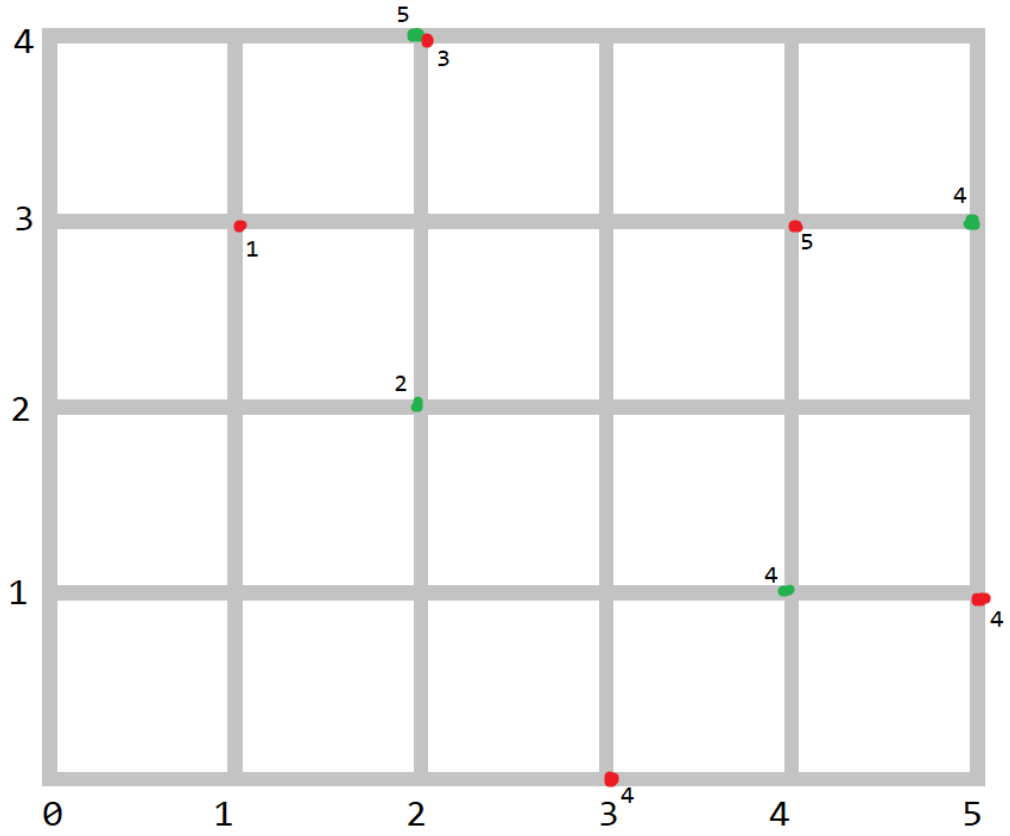
为了方便观察这个图的性质以及思考，考虑将图做一个转化：画出平面直角坐标系 xOy ，对于所有的 $i \in [1, n]$ ，从点 (i, a_i) 引一条竖线下来落到 x 轴上，称作线段 x_i ，对于所有的 $j \in [1, m]$ ，从点 (b_j, j) 引一条横线到左边落到 y 轴上，称作线段 y_j ，那么二分图左边第 i 个点和右边第 j 个点有连边当且仅当 x_i 和 y_j 有交。

举个例子，比如下图就是 $n = 5, m = 4, a = [3, 4, 0, 3, 1], b = [4, 2, 5, 2]$ 得到的图，其中涂黑的地方表示的就是一条边。

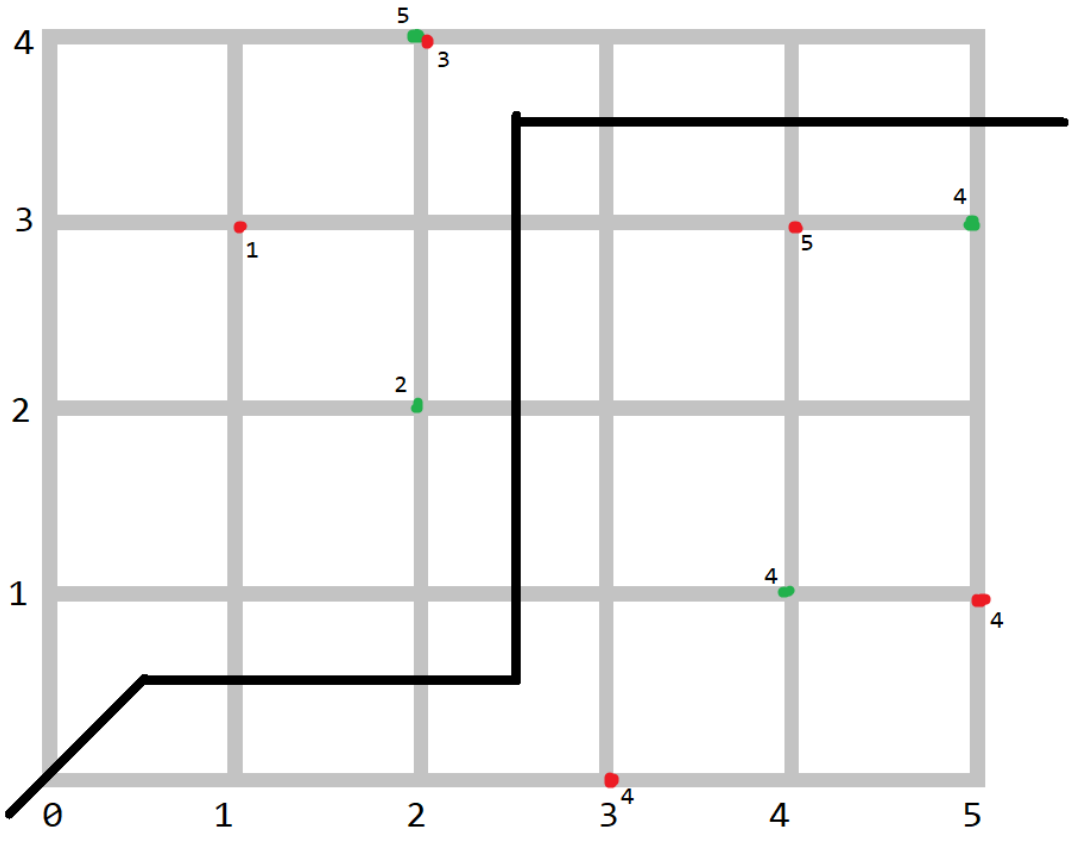


最大匹配不好求，转化成求最大独立集，发现最大独立集可以由这种方式构造出来：在 (i, a_i) 放一个点权为 c_i 的红点，在 (b_j, j) 放一个点权为 d_j 的绿点，那么在所有从左下角格子只能向右向上走走到右上角的外面的方案中，只保留路径上方的绿点点权和路径下方的红点点权的点权和的最大值。

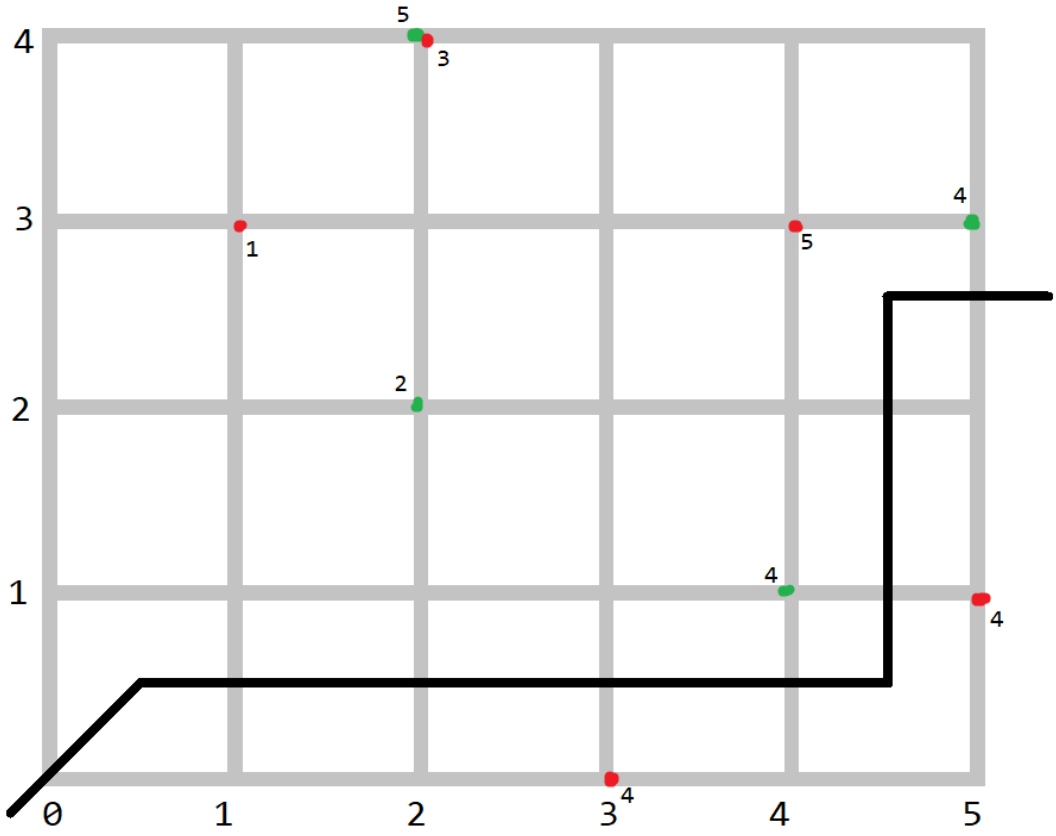
举个例子，$n=5,m=4,a=[3,4,0,3,1],b=[4,2,5,2],c=[1,3,4,5,2],d=[4,2,4,5]$，那么图是这样：



一种路径选择方案是这样（权值为 $(5+2)+(5+4+4)=20$ ）：



一种最大独立集的路径选择方案是这样（权值为 $(5+2+4+4)+(4+4)=23$ ）：



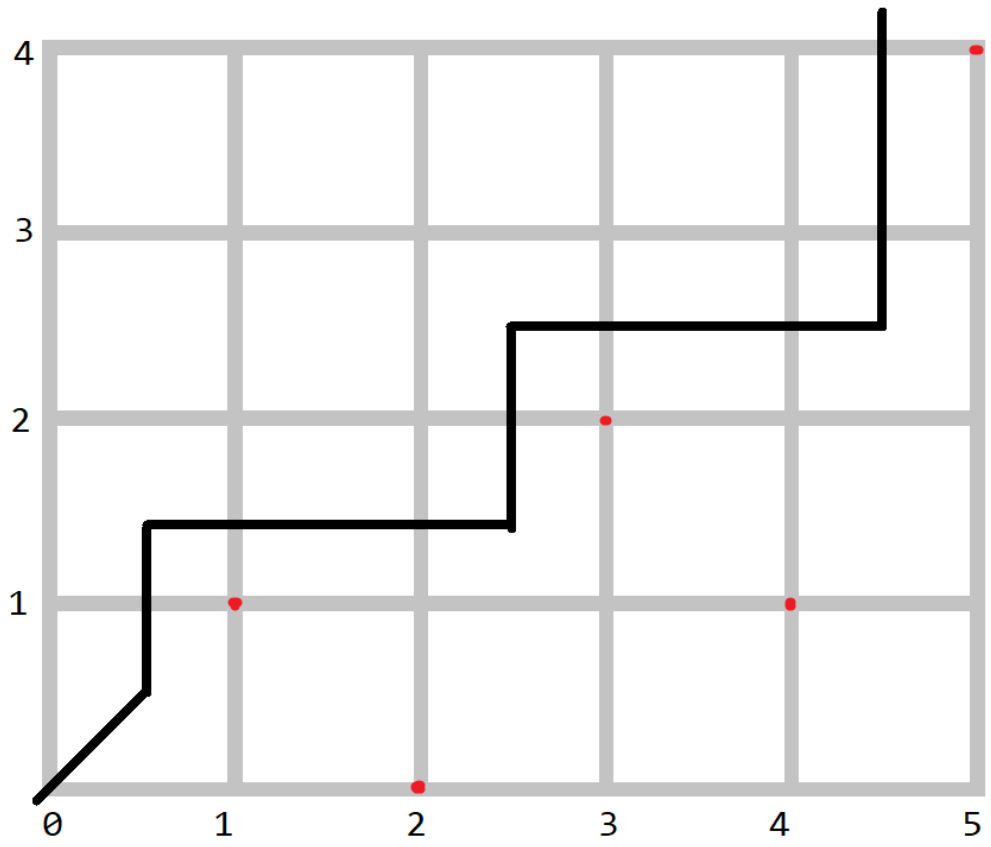
由此可以得到一种 dp 求解的方式，设 $dp(i,j)$ 表示当前路径选择到了 (i,j) 右上角的格子的位置得到的最大权值，那么转移方程：

$$dp(i,j)=\max(dp(i-1,j)+[a_i\le j]\times c_i,dp(i,j-1)+[b_j\le i]\times d_j)$$

时间复杂度 $\mathcal{O}(nm)$ 。

测试点 $11\sim13$

考虑优化 dp 的过程，发现如果钦定了一些红点被选择，那么贴着这些红点走一定是最优的，因为这样会保留尽可能多的绿点（如下图贴着红点走）。



设 $f(i)$ 表示考虑了前 $1\sim i$ 这些红点，其中最高的红点是第 i 个的最优选择方案，那么转移：

$$f(i)=\max_{j\in[0,i-1]\wedge a_j\le a_i}(f(j)+(\sum_{k=j}^{i-1}[a_k\le a_j]c_k)+(\sum_{k=a_j+1}^{a_i}[b_k<i]d_k))$$

可以通过一些处理使得 dp 复杂度降为 $\mathcal{O}(n^2+m)$。

测试点 $14\sim 15$

发现这张二分图上面右边的点是和左边的一个区间内的点有连边，简单贪心即可，时间复杂度 $\mathcal{O}(n\log n)$。（认为 n,m 同阶）

测试点 $16\sim 20$

进一步优化测试点 $11\sim 13$ 的 dp，稍微改写一下转移方程：

$$f(i)=\max_{j\in[0,i-1]\wedge a_j\le a_i}(f(j)+(\sum_{k=j}^{i-1}[a_k\le a_j]c_k)-(\sum_{k=1}^{a_j}[b_k<i]d_k))+(\sum_{k=1}^{a_i}[b_k<i]d_k)$$

把 $f(j)$ 挂在 a_j 这个位置，于是不难使用线段树优化求最大值的过程（区间加，区间减，区间最大值），总的时间复杂度 $\mathcal{O}(n\log n)$。