1

我们可以通过向下一遍动态规划得到从每个点向上走可以得到的最大值，同理我们可以向上一遍动态规划得到从每个点向下走可以得到的最大值，这样我们可以在得到经过一个点的最大路径。

由于问题是要求不经过给定点的最大路径，但我们不难发现路径每层只会经过一个点，我们预处理出每一层，经过每个点的最大路径当中的最大值，该最大值更新的位置，以及不同于最大值位置的次大值。这样询问的时候，如果该点是更新该层的最大值的位置，则输出次大值，否则输出最大值。

时间复杂度$O(n^2+m)$

2

注意到 m<=100，每个牛棚是独立于其他存在的。
假如某个牛棚有 N 头牛，该牛棚可以消除 K 次：
将代价序列分成(K+1)份，使每份的数目尽量接近。
那么一部分代价为 1..[n/(K+1)]，另一部分为 1..[n/(k+1)]+1

计算是显然的。

这一部分可以在 O(1)的时间内算出。

现在的问题转化为，M 个牛棚，每个牛棚有 P[I]头牛要进入，总共消除 k 次，使得最终答案最小。
有:DP[i][j]=dp[i-1][j-k]+C(P[i],K)

DP 复杂度 O(mk^2）。期望得分 100.

3、

Market(market.c/cpp/pas)

给定 n 个物品，每个物品要么不选，要么选一个，第 i 个物品的价格为 c_i，价值为 v_i，出现时间为 t_i。

m 次询问，每次询问在出现时间不超过 T_i 的所有物品中选若干件，总花费不超过 M_i 的情况下，被选择物品的价值和的最大值是多少。

- 测试点 $1,2$，$n \leq 20, m \leq 10, c_i, M_i, v_i \leq 100$。
- 测试点 $3,4$，$n \leq 200, m = 1, c_i, M_i, v_i \leq 200, t_i = T_i = 1$。
- 测试点 $5,6$，$n \leq 300, m \leq 100000, c_i, M_i, v_i \leq 300$。
- 测试点 7，$n \leq 20, m \leq 100000, c_i, M_i \leq 10^9, v_i \leq 300$。
- 测试点 $8,9,10$，$n \leq 300, m \leq 100000, c_i, M_i \leq 10^9, v_i \leq 300$。

20 分做法 1

- 测试点 $1,2$，$n \leq 20, m \leq 10, c_i, M_i, v_i \leq 100$。
- 对于每个询问，暴力枚举所有选择情况即可。
- 时间复杂度 $O(m2^n)$。

20 分做法 2

- 测试点 $3,4$，$n \leq 200, m = 1, c_i, M_i, v_i \leq 200, t_i = T_i = 1$。
- 经典 01 背包模型，设 f_i 表示装了 i 块钱的最大价值，DP 即可。
- 时间复杂度 $O(nM + m)$。

60 分做法

- 测试点 $5, 6$，$n \leq 300, m \leq 100000, c_i, M_i, v_i \leq 300$。
- 考虑离线，将物品按出现时间排序，同时将询问按 T 排序。
- 按 T 从小到大依次回答每个询问，维护一个指针，将允许选择的物品放入背包。
- 时间复杂度 $O(nM + m \log m)$。

10 分做法

- 测试点 7，$n \leq 20, m \leq 100000, c_i, M_i \leq 10^9, v_i \leq 300$。
- 暴力枚举每个物品的选择情况，将其看成二维平面上坐标为（最晚出现时间，总花费）的点，权值为价值和。
- 那么对于每个询问，等价于询问某个点左下角范围内的最大权值，将纵坐标离散化后用二维前缀最值预处理即可。
- 时间复杂度 $O(n2^n + m \log m)$。

100 分做法

- 依旧考虑离线询问，即可去掉时间的限制。
- 对每个询问二分答案，需要判定用容量为 M 的背包是否可以装下 mid 的价值。
- 设 f_i 表示装了 i 价值所需的最小容量，g_i 表示 $\min(f_i, f_{i+1}, f_{i+2}, \ldots)$。
- 那么只需要检查 g_{mid} 是否不超过 M 即可。
- 时间复杂度 $O(n^2 v + m \log m)$。

4、

算法一：暴力，时间复杂度 $O(n^n)$。

算法二：总价值的计算可以理解为在选择物品本来的价值之和上减去代价，而倒数第 i 个物品会产生 $w_i * (i-1)$ 的代价，那么如果已经决定好了选择的物品，肯定是贪心地按照代价升序的顺序来选择物品。
将所有物品按照代价降序排序，则最优解一定是倒序选择一些物品。用 $f[i,j]$ 代表在前 i 个物品中选择了最后 j 个物品的最优解：

$$f[i,j] = max\{f[i-1,j], f[i-1,j-1] + v[i] - w[i] * (j-1)\}$$

5　usaco2010 gather

solution：

1、将树看成以 1 为根的一棵有根树。

2、用一个 s[N]记录该节点以下的牛的个数。

3、用一个 a[N]表示该节点为聚会地点时，奶牛们的不方便程度。

那么有：

s[i]=v[i]+s[s(i)];

a[i]+=a[son(i)]+s[son(i)]*w

a[i]+=(total - s[i])*w+(a[father(i)]-a[i]-s[v]*w)

其中 v[i] 表示该点的牛数目，total 表示总的牛数量，w 表示 father(i) 到 i 点的距离。

这样，进行两次 DFS，第一次 DFS 则要得到 s[]的值还有以这棵树来看时候的 a[]值，

这个时候 a[]值仅为 a[son(i)] + s[son(i)]* w,

不算包括父亲在内的另一部分子树的不满意度，第二次 DFS 则要计算 a[]，

需要两次 DFS 的原因是，必须要用一个节点的不满意度已经求出，才能求出这个点以下的节点的不满意度。

6、

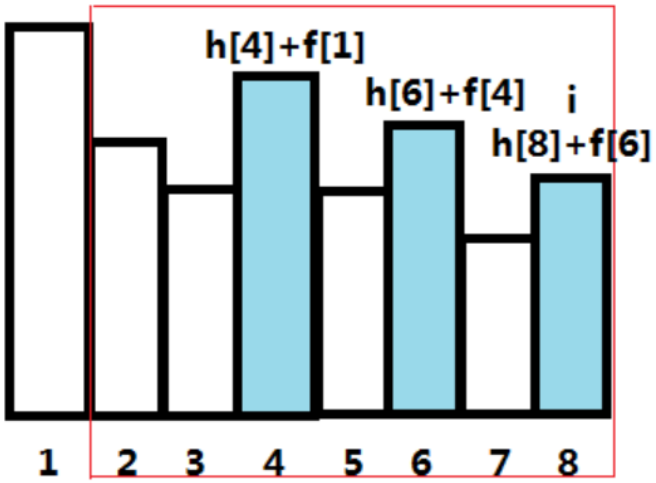
首先来想最简单的动态规划，通过第一个数的限制，我们可以得到每个位置的状态可以从哪些位置转移过来，得到如下形式的式子：

$$f(i) = max_{j=l[i]}^{i-1}(s[j+1][i] + f[j])$$

其中 s[i][j]表示从 i 到 j 的最大值，l[i]表示 i 位置最多向左走到哪里。

然后套路一波，用个单调递减的单调队列，每次将队头不合法的依次踢掉，然后我们来想想，这样我们只是得到了可以转移过来的状态集，那么如何维护答案呢？

观察下图：



现在我们的 i 在位置 8 上，红色框表示 i 向左最多到达的位置（即 2 与 8 的 p[i]是一样的）

可以看出，除了队头之外，其他的在单调队列里的元素的贡献都是从前面一个位置的 f 转移而来的（因为 f 是单调不减的），而队头的则是依赖于 l[i]的。

假设当前的队列为:a[head..tail]且$\forall head \le i < tail, a[i] < a[i+1]$

那么当前点的答案为:$\max(max_{j=head+1}^{tail}(h[a[j]] + f[a[j-1]]), h[a[head]] + f[l[i]-1])$

观察到每次我们移动一个点是有许多没有变的状态的，每个元素进一次出一次，于是我们用堆或者其他数据结构来维护队列中每个元素的贡献，对于队头特殊处理。

时间复杂度O(n log_2n)