G

> 题目大意：有 n 个菜肴，每个菜肴有甜度a_i和咸度b_i，他可以按照任意顺序吃菜肴，如果当前吃的总菜肴的总甜度超过X或者总咸度超过Y，他就不会吃任何菜肴，输出他能够吃的最多菜肴

知识点

- 背包优化（交换dp键值）

内容

安排顺序其实没什么用，最终还是决定要吃什么。

首先考虑满足咸度$\leq x$，甜度 $\leq y$的情况下吃的尽可能多。

考虑朴素搜索，即每个食物吃或不吃，我们需要维护的状态是 考虑前 i 个食物，已经吃的咸度 j，甜度 k 这三个状态，容易发现这已经足够作出 吃或不吃的决策，记忆化一下，即$dp[i][j][k]$表示考虑前 i 个食物，我吃的咸度为 j，甜度为 k 的最多食物数。

考虑其复杂度，其两个状态都是 $o(10^4)$的数量级，时间空间都不太行。但注意到食物数量的取值只有 $O(n)$，我们可以交换值和状态的意义，比如设 $dp[i][j][k]$表示考虑前 i 个食物，我吃了 j 个食物，且咸度是 k 的最小甜度数。

记得最后还能再吃一个，这样状态数即为 $O(n^2x)$，总的时间复杂度就是 $O(n^2x)$

E

> 题目大意：n 个点排成一行，选择每个点都有一个代价。给 m 个区间， 要求每个区间里至少选一个点，求最小总代价。

知识点

- 思维
- 单调队列dp

内容

假设$dp[i]$为i点必选时且前面取法合法的最小代价，那么$dp[i]$可以从上一个合法区间内的最小值更新过来，即$dp[i]=min(dp[x,y]+a[i])$，此时我们只需要考虑前一个合法区间是哪，不难想到，合法区间中一定不存在一段完整的区间，所以我们可以记录此时的左端点 l，根据右端点排序，每次遍历到一个右端点后，$l=max(l,l_{now})$，然后记录每个右端点的合法区间即可。因为左端点一定是递增的，所以我们可以直接使用单调队列优化dp，队头存的是当前合法区间的最小值，每次更新队头队尾即可。

K

每次应该从价格最便宜的商店购买货物，并卖给价格最贵的商店。用双指针模拟这一贪心策略即可，具体实现详见参考代码。

复杂度 O(nlogn)，主要是排序的复杂度。