机器分配

- 总公司拥有高效生产设备 M 台，准备分给下属的 N 个公司。各分公司若获得这些设备，可以为国家提供一定的盈利。问：如何分配这 M 台设备才能使国家得到的盈利最大？求出最大盈利值。其中 M<=15 ， N<=10 。分配原则：每个公司有权获得任意数目的设备，但总台数不得超过总设备数 M 。
- 数据文件格式为：第一行保存两个数，第一个数是设备台数 M ，第二个数是分公司数 N 。接下来是一个 M*N 的矩阵，表明了第 I 个公司分配 J 台机器的盈利。

分析

- 用机器数来做状态，数组 F[I，J] 表示前 I 个公司分配 J 台机器的最大盈利。则状态转移方程为：
- $F[I,J]=Max\{F[I-1,K]+Value[I,J-K]\}$ $(1<=I<=N, 1<=J<=M, 0<=K<=J)$
- 初始值：$F(0,0)=0$
- 时间复杂度 $O(N*M^2)$

系统可靠性

- 一个系统由若干部件串联而成，只要有一个部件故障，系统就不能正常运行，为提高系统的可靠性，每一部件都装有备用件，一旦原部件故障，备用件就自动进入系统。显然备用件越多，系统可靠性越高，但费用也越大，那么在一定总费用限制下，系统的最高可靠性等于多少？
- 给定一些系统备用件的单价 Ck，以及当用 Mk 个此备用件时部件的正常工作概率 Pk（ Mk ），总费用上限 C 。求系统可能的最高可靠性。
- 输入文件格式：

第一行： n C

第二行： C1 P1(0) P1(1) ... P1(X1) （ 0<=X1<=[C/Ck] ）

...

第 n 行： Cn Pn(0) Pn(1) ... Pn(Xn) （ 0<=Xn<=[C/Cn] ）

分析

- 例：输入： 2 20

  3 0 .6 0.65 0.7 0.75 0.8 0.85 0.9

  5 0.7 0.75 0.8 0.8 0.9 0.95

  输出： 0.6375
- 设 F[I,money] 表示将 money 的资金用到前 I 项备用件中去的最大可靠性，则有
- F[I,money] = max{F[I-1 ,money–k*Cost[I]]*P[I,k] }
- (0<=I<=n,0<=K<= money div Cost(I))
- 初始： F[0,0]=0
- 目标： F[n,C]

快餐问题

- Peter 最近在 R 市开了一家快餐店，为了招揽顾客，该快餐店准备推出一种套餐，该套餐由 A 个汉堡， B 个薯条和 C 个饮料组成。价格便宜。为了提高产量， Peter 从著名的麦当劳公司引进了 N 条生产线。所有的生产线都可以生产汉堡，薯条和饮料，由于每条生产线每天所能提供的生产时间是有限的、不同的，而汉堡，薯条和饮料的单位生产时间又不同。这使得 Peter 很为难，不知道如何安排生产才能使一天中生产的套餐产量最大。请你编一程序，计算一天中套餐的最大生产量。为简单起见，假设汉堡、薯条和饮料的日产量不超过 100 个。
- 输入：第一行为三个不超过 100 的正整数 A 、 B 、 C 中间以一个空格分开。第二行为 3 个不超过 100 的正整数 p1,p2,p3 分别为汉堡，薯条和饮料的单位生产耗时。中间以一个空格分开。第三行为 N(0<=0<=10)，第四行为 N 个不超过 10000 的正整数，分别为各条生产流水线每天提供的生产时间，中间以一个空格分开。
- 输出：每天套餐的最大产量。

分析

- 本题是一个非常典型的资源分配问题。由于每条生产线的生产是相互独立，不互相影响的，所以此题可以以生产线为阶段用动态规划求解。
- 状态表示：用 p[i,j,k] 表示前 i 条生产线生产 j 个汉堡， k 个薯条的情况下最多可生产饮料的个数。
- 用 r[i,j,k] 表示第 i 条生产线生产 j 个汉堡， k 个薯条的情况下最多可生产饮料的个数。
- 态转移方程： p[i,j,k] = Max{p[i-1,j1,k1]+r[i,j-j1,k-k1]}

(0<=j1<=j<=100,0<=k1<=k<=100,

且 (j-j1)*p1+(k-k1)*p2<=T[i]--- 第 i 条生产线的生产时间)

- r[i,j-j1,k-k1]=(T[i]-(j-j1)*p1+(k-k1)*p2) div p3 ;
- 此算法的时间复杂度为 $O(N*100^4)$,

优化

- 在本题中，可以在动态规划方法中加入了贪心算法思想：即首先计算出每天生产套数的上限值（由 A,B,C 计算，即 min ｛ 100 div A ， 100 div B ， 100 div c ｝），接着，用贪心法计算出这 N 条流水线可以生产的套数，并与上限比较，大于则输出上限值并退出，否则再调用动态规划；同时，在运行动态规划的过程中，也可以每完成一阶段工作便与上限值进行比较，这样以来，便可望在动态规划完成前提前结束程序。其算法设计为：
- S1 ：读入数据。
- S2 ：贪心求上限并计算出一可行解，判断是否需进行下一步。
- S3 ：动态规划求解。
- S4 ：输出。

其他优化方法

- 1. 存储结构：由于每一阶段状态只与上一阶段状态有关，所以我们可以只用两个 100*100 的数组滚动实现。但考虑到滚动是有大量的赋值，可以改进为动态数组，每次交换时只需交换指针即可，这样比原来数组间赋值要快。
- 2. 减少循环次数：在计算每一阶段状态过程中无疑要用到 4 重循环，我们可以这样修改每一重循环的起始值和终数，其中 q1,q2 为 A， B 上限值。
- 原起始值　　　　　　改进后的起始值
- for i:=1 to n do　　　　　　for i:=1 to n do
- for j:=0 to tot[i] div p1 do　　　　for j:=0 to min(q1,tot[i] div p1) do
- for k:=0 to (tot[i]-p1*j) div p2 do　for k:=0 to min(q2,(tot[i]-p1*j)div p2) do
- for j1:=0 to j do　　　　　　for j1 := max(0,j-t[i] div p1) to min(j,tot[i-1] div p1) do
- for k1 := 0 to k do　　　　　for k1:=max(0,k-(t[i]-(j-j1)*p1) div p2) to min(k,(tot[i-1]-p1*j1)div p2) do

商店购物

某商店中每种商品都有一个价格。例如，一朵花的价格是 2 ICU(ICU 是信息学竞赛的货币的单位）；一个花瓶的价格是 5 ICU 。为了吸引更多的顾客，商店提供了特殊优惠价。

特殊优惠商品是把一种或几种商品分成一组。并降价销售。例如 :3 朵花的价格不是 6 而是 5 ICU ;2 个花瓶加 1 朵花是 10 ICU 不是 12 ICU 。

编一个程序，计算某个顾客所购商品应付的费用。要充分利用优惠价以使顾客付款最小。请注意，你不能变更顾客所购商品的种类及数量，即使增加某些商品会使付款总数减小也不允许你作出任何变更。假定各种商品价格用优惠价如上所述，并且某顾客购买物品为 :3 朵花和 2 个花瓶。那么顾客应付款为 14 ICU 因为 :

1 朵花加 2 个花瓶：优惠价 :10 ICU

2 朵花 正常价 : 4 ICU

商店购物

输入数据

第一个文件 INPUT． TXT 的格式为：第一行是一个数字 B （$0 \le B \le 5$），表示所购商品种类数。下面共 B 行，每行中含 3 个数 C， K， P。C 代表商品的编码（每种商品有一个唯一的编码），$1 \le C \le 999$。K 代表该种商品购买总数，$1 \le K \le 5$。P 是该种商品的正常单价（每件商品的价格），$1 \le P \le 999$。请注意，购物筐中最多可放 5*5 ＝ 25 件商品。

第二个文件 OFFER． TXT 的格式为：第一行是一个数字 S （$0 \le S \le 99$），表示共有 S 种优惠。下面共 S 行，每一行描述一种优惠商品的组合中商品的种类。下面接着是几个数字对（C， K），其中 C 代表商品编码，$1 \le C \le 9\ 99$。K 代表该种商品在此组合中的数量，$1 \le K \le 5$。本行最后一个数字 P （$1 \le P \le 9999$）代表此商品组合的优惠价。当然，优惠价要低于该组合中商品正常价之总和。

输出数据

在输出文件 OUTPUT． TXT 中写 一个数字（占一行），该数字表示顾客所购商品（输入文件指明所购商品）应付的最低货款。

分析

- 由于动态规划要满足无后效性和最优化原理，所以我们来分析此题是否满足以上两点。首先确定状态，商品不超过 5 种，而每种采购的数量又不超过 5，那么用一个五元组来表示第 i 种商品买 A_i 的最小费用。
- F（A_1，A_2，A_3，A_4，A_5）　　　　（1）
- 考虑这个状态的由来，当然，我们不用优惠商品也可以买，显然这样不是最优。于是如果我们能够使用第 i 条商品组合的话，状态就 b 变为了：
- F（A_1-S_{I1}，A_2-S_{I2}，A_3-S_{I3}，A_4-S_{I4}，A_5-S_{I5}）　（2）
- 这样的话，状态 1 的费用为状态 2 的费用加上 S_i 的费用，而状态 2 的费用必须最低（很显然，用反证法即可），同时，我们也不管状态 2 是如何来的（因为每一个优惠商品组合的使用是没有限制的），所以本题既满足无后效性，又符合最优化原理，同时还有大量重叠子问题产生，动态规划解决此题是最好不过了。

状态转移方程

- F [a, b, c, d, e] =

  $\mathrm{Min}\{F[a-S_{i1},b-S_{i2},c-S_{i3},d-S_{i4},e-S_{i5}]+SaleCost[S_i]\}$

  (0<=i<=S, 0<=a, b, c, d, e<=5)

初始条件为：

F [a, b, c, d, e] =

Cost[1]*a+Cost[2]*b+Cost[3]*c+Cost[4]*d+Cost[5]*e

金明的预算方案

- **金明今天很开心，家里购置的新房就要领钥匙了，新房里有一间金明自己专用的很宽敞的房间。更让他高兴的是，妈妈昨天对他说："你的房间需要购买哪些物品，怎么布置，你说了算，只要不超过N元钱就行"。今天一早，金明就开始做预算了，他把想买的物品分为两类：主件与附件，附件是从属于某个主件的，下表就是一些主件与附件的例子：**

主件	附件
电脑	打印机，扫描仪
书柜	图书
书桌	台灯，文具
工作椅	无

- 如果要买归类为附件的物品，必须先买该附件所属的主件。每个主件可以有 0 个、 1 个或 2 个附件。附件不再有从属于自己的附件。金明想买的东西很多，肯定会超过妈妈限定的 N 元。于是，他把每件物品规定了一个重要度，分为 5 等：用整数 1~5 表示，第 5 等最重要。他还从因特网上查到了每件物品的价格（都是 10 元的整数倍）。他希望在不超过 N 元（可以等于 N 元）的前提下，使每件物品的价格与重要度的乘积的总和最大。
- 设第 j 件物品的价格为 v[j] ，重要度为 w[j] ，共选中了 k 件物品，编号依次为 j1 ， j2 ，……， jk ，则所求的总和为：
- v[j1]*w[j1]+v[j2]*w[j2]+ …+v[jk]*w[jk] 。（其中 * 为乘号）
- 请你帮助金明设计一个满足要求的购物单。

【输入文件】

- 输入文件 budget.in 的第 1 行，为两个正整数，用一个空格隔开： n m

  （其中N（ <32000 ）表示总钱数， m（ <60 ）为希望购买物品的个数。）
- 从第 2 行到第 m+1 行，第 j 行给出了编号为 j-1 的物品的基本数据，每行有 3 个非负整数 v p q

  （其中 v 表示该物品的价格（ v<10000 ）， p 表示该物品的重要度（ 1~5 ）， q 表示该物品是主件还是附件。如果 q=0 ，表示该物品为主件，如果 q>0 ，表示该物品为附件， q 是所属主件的编号）

【输出文件】

- 输出文件 budget.out 只有一个正整数，为不超过总钱数的物品的价格与重要度乘积的总和的最大值（ <200000 ）。

假设只有主件的情况

- 给出 m 件物品和 n 元钱，每个物品有一个费用 Ci 和价值 Vi ，问买哪些东西能使得所购写的物品的价值和最大。
- 用 F[i,j] 表示在前 i 件物品中选择一些，使所花的钱数不超过 j 时所得的最大价值。则
- F[0,j]=F[i,0]=0 (边界条件)
- F[i,j]=max{F[i-1,j], F[i-1,j-Ci]+Vi}
- 此算法的复杂度为 O(nm) 。

回到原题

- 假设一件物品 i 有 t 种附件选择方案，费用分别为 Ci1..Cit，价值分别为 $V_{i1}..V_{it}$，则

$$F[i,j]=\max\begin{cases}F[i-1,j], (\text{不选物品}i)\\ F[i-1,j-C_{ik}]+V_{ik}\ (\text{选物品}i\text{的第}k\text{种附件方案})\end{cases}$$

由于每个物品不超过两个附件，所以附件的选择方案非常有限，只要手工枚举一下就可以了。当然，巧妙的做法是：为不够两件附件的物品增加费用和价值都为 0 的虚物品，使每件物品的附件数都是 2 。然后分别枚举 2 个附件选还是不选。

这个方法的复杂度仍为 O(nm)，可以很好的解决本题了