动 态 规 划 2

常州市第一中学　林厚从

加工小木棍（STICK）

- 问题描述：
  - 某工厂生产一批棍状零件，每个零件都有一定的长度（Li）和重量（Wi）。
  - 现在为了加工需要，要将它们分成若干组，使每一组的零件都能排成一个长度和重量都单调递增（即若Li<=Lj，则Wi<=Wj）的序列。请问至少要分成几组？
- 问题输入：
  - 第一行为一个整数N（N<=1000），表示零件的个数。
  - 第二行有N对正整数，每对正整数表示这些零件的长度和重量，长度和重量均不超过10000。
- 问题输出：
  - 仅一行一个数，即最少分成的组数。

加工小木棍（STICK）

- 样例输入：
  - 5
  - 8 4 3 8 2 3 9 7 3 5
- 样例输出：
  - 2

- 难点：两维的单调性？
- 突破点：与初始顺序没有关系！
- 解法：
  - 先按长度非递减排序(双关键字，长度+重量)，保证一维单调。
  - 然后对排好序的序列，再求重量的LIS序列？No!
  - 求最少分成几组非递减序列，贪心法，类似题“拦截导弹”第2问。

WZK走路

CZYZ有一条小路，路是由许多的地砖铺起来的，WZK想沿着这条路从CZYZ的一边走到另一边。在路上有一些粘了泥土的脏地砖，WZK很讨厌踩在这些地砖上。由于小路的长度和WZK一次走过的距离都是正整数，我们可以把小路上WZK可能到达的点看成数轴上的一串整点：0，1，……，L（其中L是小路的长度）。坐标为0的点表示路的起点，坐标为L的点表示路的终点。WZK从小路的起点开始，不停的向终点方向走。跨一步的距离是S到T之间的任意正整数（包括S,T）。当WZK走到或走过坐标为L的点时，WZK已经走出了小路。

L<=10000, s<=t<=1000

题目给出小路的长度L，WZK跨步的距离范围S,T，路上脏地砖的位置。你的任务是确定WZK要想走过去，最少需要踩到的脏地砖数。

- 输入样例：
  - 10 2 3　　//L、S、T
  - 2 3 5 6 7　//脏地砖编号
- 输出样例：
  - 2

WZK走路

- 问题分析：
  - 状态设计：
  - f[i]表示WZK走到第i个坐标点时，最少踩到多少脏地砖。
  - 如何转移：
  - f[i] = min { f[j] + z}
  - 其中，i-t <= j <= i-s，s、t为一步跨越的距离范围

    如果第i处为脏地砖则z=1，否则z=0。
  - 答案是f[L]吗？
  - 细节：f数组要定义大一点，f[0..L+T-1]。

WZK走路

- 伪代码：
  - f[0]=0;
  - for i=1 to n do f[i]=maxlongint;

  - for i=1 to n do{
  - for j=i–t to i–s do
  - if j>=0 then f[i]=min(f[i],f[j]);
  - if i is dirty then f[i]=f[i]+1; //根据读入设计一个布尔型数组即可
  - }

  - ans=f[l];
  - for i=l+1 to l+t–1 do
  - if f[i]<ans then ans=f[i];
  - writeln(ans);

书的复制（BOOK）

- 问题描述：
  - 有M本书（编号为1，2，…，M），每本书都有一个页数（分别是P_1，P_2，…，P_M）。现在想将每本书都复制一份。
  - 将这M本书分给K个抄写员（1<=K<=M<=500），每本书只能分配给一个抄写员进行复制。每个抄写员至少被分配到一本书，而且被分配到的书必须是连续顺序的。
  - 复制工作是同时开始进行的，并且每个抄写员复制一页书的速度都是一样的。所以，复制完所有书稿所需时间取决于分配得到最多工作的那个抄写员的复制时间。
  - 试找一个最优分配方案，使分配给每一个抄写员的页数的最大值尽可能小。

书的复制（BOOK）

- 输入格式：
  - 第一行两个整数M、K；
  - 第二行M个整数，第i个整数表示第i本书的页数。
- 输出格式：
  - 共K行，每行两个正整数，第i行表示第i个人抄写的书的起始编号和终止编号。K行的起始编号应该从小到大排列，如果有多解，则尽可能让前面的人少抄写。
- 输入样例：
  - 9 3
  - 1 2 3 4 5 6 7 8 9
- 输出样例：
  - 1 5
  - 6 7
  - 8 9

书的复制（BOOK）

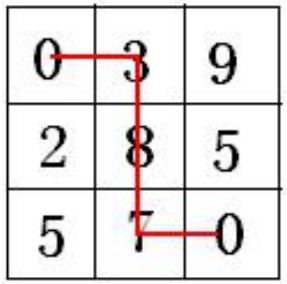
- 问题分析：
  - M本书是顺序排列的，K个抄写员选择数也是顺序且连续的。所以不管以书的编号，还是以抄写员标号作为参变量划分阶段，都符合策略的最优化原理和无后效性。考虑到K<=M，所以，假设以抄写员编号来划分阶段。
  - 状态设计：F[i][j]表示前i个抄写员复制前j本书的最小“页数最大数”，那么，第i个人可以抄写哪些书呢？穷举！
  - 转移方程：F[i][j]
    = MIN{ MAX{F[i-1][V]，T[V+1][j]} }
  - 其中：T[V+1][j]表示从第V+1本书到第j本书的页数和，1<=i<=K，i(i个人至少要人抄写1本)<=j<=M-K+i(后面还有K-i个人需要至少抄写K-i本)，i-1<=V<=j-1。
  - 边界条件：F[1][j]=T[1][j]。
  - 如何根据求得的最优值，输出具体的分配方案？

传纸条（NOIP2008，MESSAGE）

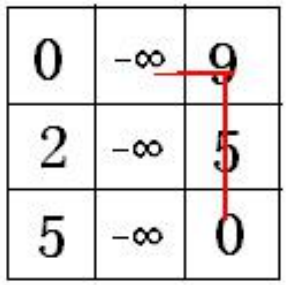
- 问题描述：
  - 小渊坐在矩阵的左上角，坐标(1,1)，小轩坐在矩阵的右下角，坐标(m,n)。从小渊传到小轩的纸条只可以向下或者向右传递，从小轩传给小渊的纸条只可以向上或者向左传递。
  - 班里每个同学都可以帮他们传递，但只会帮他们一次。
  - 每个同学愿意帮忙的好感度有高有低，可以用一个0-100的自然数来表示，数越大表示越好心。
  - 小渊和小轩希望尽可能找好心程度高的同学来帮忙传纸条，即找到来回两条传递路径，使得这两条路径上同学的好心程度之和最大。现在，请你帮助小渊和小轩找到这样的两条路径。
  - 1<=m,n<=50。

传纸条（NOIP2008，MESSAGE）

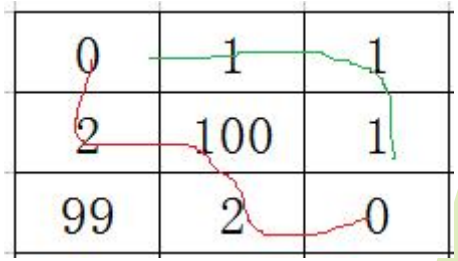
- 问题分析：
  - 贪心算法：先求出1个纸条从（1,1）到（M,N）的路线最大值，删除路径上的点值，再求出1个纸条从（M,N）到（1,1）的路线最大值，统计两次和。
  - 上述算法很容易找出反例，如下图：



第1次传递



第2次传递



  - 第1次找最优值传递后，导致第2次无法传递。
  - 可以传递2次，值也不一定最优。

传纸条（NOIP2008，MESSAGE）

- 问题分析：
  - 贪心算法错误提示我们需要同时考虑两个纸条的传递。
  - 由于小渊和小轩的路径可逆，因此，尽管出发点不同，但都可以看成同时从(1,1)出发到达(M,N)点。
  - 状态设计：设f[i_1][j_1][i_2][j_2]表示纸条1到达(i_1,j_1)位置，纸条2到达(i_2,j_2)位置的最优值。则有：

$$f[i_1][j_1][i_2][j_2]=\max\begin{cases}f[i_1-1][j_1][i_2-1][j_2]\\f[i_1][j_1-1][i_2-1][j_2]\\f[i_1-1][j_1][i_2][j_2-1]\\f[i_1][j_1-1][i_2][j_2-1]\end{cases}+C[i_1][j_1]+C[i_2][j_2]$$

  - 其中 (i_1,j_1)<> (i_2,j_2)，1<=i_1, i_2<=M, 1<=j_1 , j_2<=N。
  - 算法的时间复杂度为O(N^2*M^2)。

传纸条（NOIP2008，MESSAGE）

- 问题分析：
  - 换一种思路：每个纸条都需要走M+N步才能达到目标。
  - 因此，设F[k][i_1][i_2]表示两个纸条都走了K步，第1个纸条横坐标为i_1,第2个纸条横坐标为i_2的最优值。
  - 则两个纸条的纵坐标可以直接计算出来，分别为j_1=K-i_1，j_2=K-i_2,状态转移方程如下：

$$f[k][i_1][i_2]=\max\begin{Bmatrix} f[k-1][i_1][i_2] \\ f[k-1][i_1-1][i_2] \\ f[k-1][i_1][i_2-1] \\ f[k-1][i_1-1][i_2-1] \end{Bmatrix}+C[i_1][k-i_1]+C[i_2][k-i_2]$$

  - 其中 i_1<>i_2, 1<=i_1, i_2<=M,1<=k<=N+M。
  - 算法的时间复杂度为O((N+M)*M^2)。

经典问题3——01背包问题

- 题目描述：
  - 有一个容量为X(<10000)的背包，有n(<100)个物品，
  - 每个物品有一个体积w[i]，一个价值p[i]，
  - 在不超过背包容量的情况下选择的物品价值尽可能大？
- 样例输入：
  - 10 4
  - 7 10
  - 2 5
  - 6 4
  - 3 9
- 样例输出：
  - 19

经典问题3——01背包问题

- 为什么叫“01背包”呢？区别于“部分背包”。
- 状态设计：f[i][v]表示只考虑前i个物品，背包容量为v，所装物品的最大价值是多少？
- 状态如何转移呢？
- 只需考虑第i个物品是否被选择（选与不选，1或0）
- f[i][v] = max{ f[i-1][v], 不选择第i个物品
  f[i-1][v-w[i]] + p[i] 选择第i个物品 }
- 其中：1<=i<=N, 1<=v<=X。
- 初值：F[0][0..X]=0，表示没有选择任何物品时，任何容量的背包都是合法的，只是价值都是0。
- 程序实现：先穷举i，再穷举v，F[N][X]即答案，时间复杂度已经最优，为O(N*X)，参考程序beibao1.cpp。

经典问题3——01背包问题

- 主要代码：

```
cin >> x >> n;
for(int i = 1; i <= n; i++) cin >> w[i] >> p[i];
memset(f,0,sizeof(f));
for(int i = 1; i <= n; i++)
    for(int v = 1; v <= x; v++)
        if(v < w[i]) f[i][v] = f[i–1][v];
        else f[i][v] = max(f[i–1][v],f[i–1][v–w[i]] + p[i]);
printf("%d\n",f[n][x]);
```

- 跟踪求值的过程如下：

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	10	10	10	10
0	5	5	5	5	5	10	10	15	15
0	5	5	5	5	5	10	10	15	15
0	5	9	9	14	14	14	14	15	19

经典问题3——01背包问题

- 空间可不可以优化呢？

- 降维！连滚动数组都不需要，直接把i这一维省略，具体方法：
- f[v]表示前i个物品，背包容量为v，所装物品的最大价值。
- 先直接用一个循环穷举i（1 to N）。
- 里面的转移方程为：f[v] = max{ f[v]，f[v-w[i]] + p[i] }
- 但是，这儿的v必须从大到小穷举（X downto w[i]），为什么？

经典问题3——01背包问题

- 从大到小穷举v的原因分析：为了保证第i次循环求f[i][v]时，f[v]还是二维中的f[i–1][v]、f[v–w[i]]还是二维中的f[i–1][v–w[i]]，必须从大到小穷举，且只要到w[i]（而不需要到1），否则，前面一边求一边覆盖，就不是上一行的了，这一点非常重要！
- 代码实现：用样例跟踪求值，参考程序beibao2.cpp。

```
memset(f,0,sizeof(f));
for(int i = 1; i <= n; i++)
    for(int v = x; v >= w[i]; v--)
        f[v] = max(f[v],f[v–w[i]] + p[i]);
printf("%d\n",f[x]);
```

- 应用举例：采药（medic，noip2005）

经典问题3——01背包问题

- 思考讨论：如果要求的是“恰好把背包装满”的最大价值，有何区别？代码实现有何区别？
- 状态和转移方程都没有问题，只是初始化不一样！
- 初始化时除了F [0]:=0，其它F[1..V]:= -maxlongint。
- 为什么呢？
- 可以这样理解：初始化的F 数组事实上就是在没有任何物品可以放入背包时的合法状态。如果要求背包恰好装满，那么此时只有容量为0的背包可以在什么也不装且价值为0的情况下被“恰好装满”，其它容量的背包均没有合法的解，属于未定义的状态，应该被赋值为“负无穷大”了。
- 如果“背包不一定要装满”，那么任何容量的背包都有一个合法解“什么都不装”，这个解的价值为0，所以初始时状态的值也就全部为0了。

经典问题3——01背包问题

- 多重背包：如果刚才的问题中每种物品有k[i]个呢？
- 一种简单的解决方法：把每种物品看成k[i]个物品，转换成更多数量物品的01背包问题。

- 无限背包：如果刚才的问题中每种物品有无数个呢？
- 刚才的状态表示依然可以！
- 考虑第i个物品用了多少个（穷举k）？
- 转移方程为：f[i][j] = max { f[i-1][j-k*w[i]] + p[i]*k }

上机题

- 加工小木棍（STICK）
- 书的复制（BOOK）
- 传纸条（NOIP2008，MESSAGE）
- 接苹果（bcatch）
- 最小乘车费用（busses）
- 饥饿的牛（hunger）
- 01背包问题

谢谢大家！