《算法艺术与信息学竞赛》

刘汝佳 黄亮 著

本页最新更新日期：2004-3-9

欢迎大家来到《算法艺术与信息学竞赛》习题页面！

更新记录

2004-3-9
本页建立

部分习题提示

1.1节习题

1.1.1
对；不对
1.1.2
提示：考虑规模增长时的时间开销，可以对比运行时间，也可以在程序中统计基本操作数目

1.2节习题

1.2.1
不一定。这要看枚举量和枚举时间
1.2.3
枚举即可。需要注意的是要保证每个问题只有一个正确答案而不是有个答案。本题有唯一解cdebeedcba。
1.2.5
注意行有很多但是列不太多。硬币翻两次等于不翻，所以所有行/列最多翻一次。枚举每列是否翻有2^9=512种情况，此时可以用O(n)的时间计算每行是否翻（注意：列确定以后每行独立）。
1.2.6
只需要枚举横坐标相邻的点
1.2.7
设S1的长度为n。注意用串匹配来做的话时间复杂度至少为O(10^n)，不是有效算法。应该枚举S1的“分段方式”。例如231241，如果分段方式为23|124|1，则124为完整的一段，前面必为123，后面必为125，经检验这是可行的。因此如果有一个完整段的情况可以用O(n^2)次枚举来判定。剩下只需要解决没有完整段的情况，即被分成两段。如2312，如果被分为2|312，则需要枚举位数。如果是3位，有方程??2 + 1 = 312，无解；如果是4位，有方程???2 + 1 = 312?，有解3122 + 1 = 3123。基本思想如此，但是需要注意有进位的情况。建议读者自己写程序并提交到ural在线题库检查自己的程序是否考虑周全。
1.2.12
首先可以证明：覆盖整个草坪当且仅当覆盖草坪的上边界。这样每个圆的作用范围是一条线段，问题转化为用最少的线段覆盖整个区间。先预处理再贪心，具体方法留给读者思考。

1.2.14
本题较难，需要先推出n=4时的两种可能最优策略（用代数方法容易推导出），然后用递归的思想把n>4的情况转化为n<=4的情况加以解决。提示：考虑四人速度为1,3,4,5和1,2,5,6的情况，最优时间分别为16和13。
1.2.17
本题答案不唯一。例如：解方程组。
1.2.23
枚举去掉的数字位置后，几乎就可以直接解方程了（需要分类讨论和少量附加枚举）。具体方式留给读者思考
1.2.24
把袋子编号为0,1,2...n-1，如果没有1717克的限制，就是第0,1,2...n-1个袋子依次取豌豆0,1,2...n-1颗，把称得的重量和0+1+2+...+(n-1)比较，重了几克就是第几个袋子是魔法豌豆！可是现在有总重量限制，因此只有当0+1+2+...+n-1<=1717时才能一次称出。记M(1)为能保证一次称出的最大n值，则解不等式得n<=59。二次称可以用以下策略：59个袋子为一组，第0组不取，第1组每个取1颗，第2组每个取2颗...只要总重量<=1717，则多了几克就是第几组的。由于每组只有59个，所以再称一次就可以了。解不等式可以求出保证二次称出的最大n，记为M(2)。这样递推出M(10)发现M(10)>10000，因此用我们刚才的策略就可以在10次内称出了。 1.2.27
如果某张纸上只有一个程序，则可以在其他顺序恢复后再单独把它插入；如果某张纸上有至少三个程序，则除了头尾之外的中间程序都只会出现一次，也可以最后处理，因此只需要保留恰好有两个程序的纸张。剩下的工作就不难了，留给读者思考。

1.3节习题

1.3.6
自上而下的读取各行，可以用本节介绍的floodfill方法来作，但是更节省空间的方法是1.4节介绍的并查集。需要注意的是要正确的处理新块开始、旧块结束、不同块合并、相同块再次合并（形成“洞”）等几种情况。
1.3.7
下确界可以简单的通过求轮廓线的并来实现。交就没有这么简单了，因为在求轮廓线交以后可能形成非矩形的区域，即出现“凹角”。先作一次floodfill后删除有三侧属于同一个区域的角，直到不存在这样的角。
1.3.24
设电路对应的函数是f(i)，其中i是一个n进制数，n为输入个数。如果f(000..0)=f(111..1)，则所有x设置为0即可。否则考虑序列：
000..000,000..001,000..011,000...111, ... 011..1, 111..1，每相邻两项只相差一个字符。显然一定存在相邻两项的f值不同，不同的字符保留为x，其他设置为相同值即可。可以二分查找，总时间复杂度为O(mlogn)，m为门的数目。

1.4节习题

1.4.1
先作标记，等到浪费空间大于一半时重构树。可以证明均摊时间复杂度不变。另外还有保持每个操作最坏情况时间复杂度不变的算法，非常巧妙。
1.4.7
设s[0]=0，s[i]=a[1]+a[2]+...+a[i]，则信息i j even等价于a[i]+...+a[j]为偶数，即s[j]-s[i-1]为偶数，即s[j]与s[i-1]同奇偶。这样，每条信息都可以变为某两个s[i]和s[j]是否同奇偶的信息。记same[i]为当前和s[i]同奇偶的s[j]集合，diff[i]为当前和s[i]不同奇偶的s[j]集合，则一条信息i j even将导致same[j]和same[i-1]合并，diff[j]和diff[i-1]合并；信息i j odd将导致same[j]和diff[i-1]合并；diff[j]和same[i-1]合并。具体细节留给读者思考。
1.4.12
和标准的Young Tableau查找算法很类似，稍微修改一下即可。

1.4.16
先按照x坐标排序，然后建立一棵静态的BST用于统计。需要记录附加信息。建议读者写写程序，要写得尽量简单，很少几行即可写完。

1.5节习题

1.5.8
先作减法，把两个权变成一个。可以进一步发现如果矩形有重叠，可以把重叠部分去掉和权和保持不变。这样问题变成了即找出k个不重叠的矩形使得权和最大。们用区域(i,j)来表示在矩阵W中的“第一行第i个格子右边所有元素加上第二行第j个格子右边的所有元素”这个区域，用d[s,i,j]来表示在这个区域中选择s个子矩阵，它们的元素总和的最小值。看作多阶段决策问题，则决策有五种：决策一：第一行第i个格子不用的情况，这种决策转移到状态d[s,i+1,j]；决策二：第二行第j个格子不用的情况，这种决策转移到状态d[s,i,j+1]；决策三：第一行从第i个格子放一个矩形，则大小L有O(n)种选择；转移到d[s,i+L,j]决策四：第二行从第j个格子放一个矩形，则大小L有O(n)种选择；转移到d[s,i,j+L]决策五：两行一起放宽度为2的矩形，也有O(n)种选择。
1.5.10
如果是求利益最大的方案，显然可以定义状态d[i]为考虑前i个订单并接受订单i的最大利润。但是第k的方案呢？这种状态表示是不行的。它的一个重要问题在于：问题不具备最优子结构！第k大方案所对应的决策的子决策不一定是第k大的。无奈之下，我们只好增加一维状态参量，用d[i,j]表示考虑前i个订单并介绍订单i的第j大利润，而状态转移时也必须考虑所有前趋状态各自的第1,2,3…j大利润（想一想，为什么不考虑第j+1,j+2…k大利润？），然后加以比较。需要注意的是可以利用堆来降低时间复杂度，请读者思考。
1.5.11
注意到命令序列长度不超过50，机器人不可能走得太远。所以可以先枚举终止位置，然后单独考虑每个机器人，让每个机器人删除的指令数都最少。用d[x,y,i]表示要让前i条指令被执行（或被删除）后机器人处于位置(x,y)所需要删除的最少指令数，请读者列出状态转移方程。
1.5.12
首先，我们直观的猜测：任意一副筷子中A和B一定是长度相邻的两只筷子。证明如下：对于某副筷子(A1,B1,C1)和另一副筷子(A2,B2,C2)，如果A1<=A2<=B1<=B2，那么交换一下筷子重新组合成(A1,A2,C1)和(B1,B2,C2)质量和会更优。对于某副筷子(A,B,C)和闲置的筷子D，如果A<=D<=B，那么交换一下重新组合成(A,D,C)质量和也会更优。
这样，我们得到了一个线性结构，只需要从左往右或从后往左递推。在本题中，由于第3根筷子比另2根长，所以我们从长筷子往短筷子递推。在递推之前，首先将N只筷子从小到大排序，Li是第i只筷子的长度。
用d[i,j]表示用i..n的单只筷子组成j副筷子的最小质量值之和，则当且仅当n-i+1>=3j的时候状态是合法的。请读者自己写出状态转移方程。
1.5.13
这道题目有一个很讨厌的条件：“只要有任务是可以完成的，那么工人不能闲着没事做”。如果工人必须按照任务序号递增的顺序，那么本题可以用简单的动态规划解决，但是本题没有规定任务完成的顺序，如果我们用d[i]代表i时刻以后还需要的最少时间，那么我们做状态转移的时候会很为难：我们似乎无法知道那么任务是已经完成了的（它们不能被重复执行），因此决策集合无法确定。即：问题有后效性！
真是这样吗？（解决后效性的通常办法是增加状态参量，即记录已经有哪些任务完成了，但是这样做状态量会大增，并不是一个好办法）我们需要避免重复选择任务，但是细心的读者一定已经发现了：根本不会重复选择任务。原因在于一个容易被忽略的条件：t<=di-ai<2ti。当完成一个任务以后，严格的时间期限已经不可能允许工作再重复选择这个任务了。接下来的任务就十分简单了，请读者自己思考。
1.5.17

提示：本题很容易想到O(n^3)的直接动态规划，但是时间复杂度还可以降低。先连接各点和圆心，把面积最大转化为正弦函数和最大，再利用正弦函数的性质进行优化。

1.5.18

铁球下落的高度和总是一定的，所以我们关心的问题只是它的水平移动距离。我们称平台i的两头为平台端点的横坐标为x[i,0]和x[i,1]，并设端点i纵坐标为y[i]。为方便处理，我们将铁球的初始位置抽象成宽度为0的平台0。

每落到一个平台，球有两种决策：向左滚和向右滚，因此我们设从平台i的第k个边缘(k=0为左边缘，k=1为右边缘)落下后还需要移动的最少水平距离为d[i,k]。设p[i,k]为从平台i的第k个边缘落下后到达的平台编号（即状态d[i,k]描述的“当前平台”，如果落下会摔碎，则p[i,k]=-1），则两种决策是：

决策一：向左滚，指标函数为d[p[i,k],0]+|x[i,k] - x[p[i,k],0]|

决策二：向右滚，指标函数为d[p[i,k],1]+|x[i,k] - x[p[i,k],1]|

状态数为O(n)，决策数为O(1)，动态规划的时间复杂度取决于状态转移的时间复杂度，即p数组的计算复杂度。我们可以从高到低依次考察平台i下面平台j，如果x[i,k]处于区间[x[j,0], x[j,1]]内，则p[i,k]=j。最坏情况下的时间复杂度为O(n^2)，比动态规划本身还要高。事实上可以用BST或者线段树来做到O(nlogn)，请读者思考。

1.5.19

本题最大的迷惑点在于佳佳的篮子，如果你在记录当前糖果堆的同时记录篮子内的糖果，那么你就上当了。佳佳很聪明，他只记录当前糖果堆的情况，从而推知哪些糖果曾被拿到篮子里，并标记编号为i的糖果总共被拿出来Ai颗。

本着将尽可能多的糖果放入口袋的原则，佳佳将能配对的糖果统统从篮子内取出，于是令Ai =Ai mod 2，则当前篮子里糖果的数量Tot=sum{Ai}就求出了。

我们用P1,P2,P3,P4分别表示当前4堆糖果剩余的糖果数量，用数组元素a[P1,P2,P3,P4]表示状态P1,P2,P3,P4相应的Tot值。显然，根据a[P1,P2,P3,P4+1]的值，我们可以推得a[P1,P2,P3,P4]的值。如果Tot≤5，那么状态P1,P2,P3,P4是合法的，记为d[P1,P2,P3,P4] = true；否则，这是导致游戏结束的非法状态，记为d[P1,P2,P3,P4] = false。根据这个布尔型状态定义，我们不难写出状态转移方程，请读者自己完成。

1.5.21

本题也是利用匹配点的稀疏性。

1.5.22

只需求原串和逆序串的LCS

1.5.23

标准的LIS问题

1.6节习题

1.6.1

按照拓扑顺序来搜索，即先搜最里层子天平，再次里层...可以用后文提到的极端法剪枝（估计天平两边的最大、最小可能取值）

1.6.7

仔细分析一下状态空间，你会发现它其实非常少，用简单的启发函数（例如只考虑档住的纵向车移动的最小步数和与Car0步数之和）就可以取得非常好的效果。

1.6.8

此题比较难。可以利用IDA*搜索，忽略湖泊后用网络流的方法求出的割点数目（事实上还不完全是割点，因为有些点被屏蔽掉了）作为h函数，加上可行性剪枝：如果可以加石头的地方都加了石头还是能相互看见，则剪枝。h函数和可行性剪枝的作用都非常明显，读者不妨一试。

1.6.16

对于大多数数据来说，预处理合并掉相同的块并事先保存好每个块右方和下方可能的块集合后速度将会非常快。

2.1节习题

2.1.2
先考虑n=2^k的情况，构造完成后任意n都能处理了。
2.1.3
先展开m^k - (m-1)^k = A(m) = a[k-1]*m^(k-1) + a[k-2]*m^(k-2) + ...，其中A(m)为m的k-1次多项式。代入m=1,2,3...n后左右相加得到n^k = a[k-1]*s(k-1,n) + a[k-2]*s(k-2,n) + ... 其中a[k]为A(m)的k次项系数。可用此法从s(1,n), s(2,n) ... s(k-1,n)计算s(k,n)。
2.1.6
解方程组。
2.1.9
列出方程组以后发现可以直接递推计算而不需高斯消元。
2.1.10
解方程组。
2.1.11
解方程组。注意特殊情况的处理。建议读者自己分析。
2.1.14
n<=50时可以用递推法得到解；n>=50时不管m为几一定有解。

2.2节习题

2.2.6
提示：2^k步以后的情形很有规律。利用这个规律可以用二分法逐步用2^k逼近n。

2.3节习题

2.3.3
类似15数码问题的处理方法，参见书p190页

2.4节习题

下载习题3,4,5,6,8,10,11的提示

2.5节习题

下载习题1,3,6,7,8,9,11,12,13,14,15,17,18,19,20,21,26的提示