搜索达到最优化
Translate:USACO/nuggets

Beef McNuggets 麦香牛块

Hubert Chen

译 by ragazza gatti

描述

农夫布朗的奶牛们正在进行斗争，因为它们听说麦当劳正在考虑引进一种新产品：麦香牛块。奶牛们正在想尽一切办法让这种可怕的设想泡汤。奶牛们进行斗争的策略之一是“劣质的包装”。“看，”，奶牛们说，“如果你只用一次能装 3 块、6 块或者 10 块的三种包装盒包装麦香牛块，你就不可能满足一次只想买 1、2、4、5、7、8、11、14 或者 17 块麦香牛块的顾客了。劣质的包装意味着劣质的产品。”

你的任务是帮助这些奶牛。给出包装盒的种类数 N(1<=N<=10)和 N 个代表不同种类包装盒容纳麦香牛块个数的正整数(1<=i<=256)，输出顾客不能用上述包装盒(每种盒子数量无限)买到麦香牛块的最大块数。如果所有购买方案都能得到满足或者不存在不能买到块数的上限，则输出 0。不能买到的最大块数（倘它存在）不超过 2,000,000,000。

格式

PROGRAM NAME: nuggets

INPUT FORMAT:

(file nuggets.in)

第 1 行: 包装盒的种类数 N

第 2 行到 N+1 行: 每个种类包装盒容纳麦香牛块的个数

OUTPUT FORMAT:

(file nuggets.out)

输出文件只有一行数字：顾客不能用包装盒买到麦香牛块的最大块数或 0(如果所有购买方案都能得到满足或者顾客不能买到的块数没有上限)。

SAMPLE INPUT

```
3
3
6
10
```

SAMPLE OUTPUT

```
17
```

问题分析

这是一个背包问题。一般使用动态规划求解。

一种具体的实现是：用一个线性表储存所有的节点是否可以相加得到的状态，然后每次可以通过一个可以相加得到的节点，通过加上一个输入的数求出新的可以相加得到的点。复杂度是 O(N×结果)。

但是可以证明结果不会超过最大的两个数的最小公倍数（如果有的话）。参见数论。所以复杂度也是 $O(Na^2)$，完全可以接受了。

判断无限解可以按上面的方法，另外也可以算所有的数的最大公约数。如果不是 1，也就是说这些数不互质，那么一定没办法构成所有的不被这个最大公约数整除的数。当且仅当这种情况会有无限解。另外有一种不需要任何数论知识的方法是判断是不是按照每个输入的数的循环节循环，如果是的话，继续算显然不会有任何结果。

判断有没有更大的解也可以按这种方法，另外如果连续最小的数那么多个数都可以构成，也不会有更大的符合条件的解。

通过位压缩可以使程序在 32 位机上的运行速度快 32 倍（由于程序代码会变长，也可能是 16 倍或者更小的倍数）。这样可以相当快的解决这个问题。不过复杂度还是 $O(Na^2)$。

“可以证明结果不会超过最大的两个数的最小公倍数”。我来证明一下。已知，不定方程 ax + by = c (a , b > 0 且 c >= ab) 存在 一组 整数解 (x0 , y0) 求证，该不定方程存在一组非负整数解 (xn , yn) . 证明 : 由不定方程通解式得 : xn = x0 + b * t , yn = y0 - a * t (t 是整数) 令 xn , yn >= 0 解出 - (x0 / b) <= t <= (y0 / a) 因为 c >= a * b 即 a * x0 + b * y0 >= a * b 两边同除 a * b 得 : y0 / b - (- x0 / a) >= 1 所以 一定存在 整数 t 使得 - (x0 / b) <= t <= (y0 / a) . 所以 方程一定有非负整数解 . 证毕 .

MS 就是斐蜀定理

Translate:USACO/fence8

Fence Rails 栅栏的木料

Burch, Kolsad, and Schrijvers

译 by Jeru

描述

农民 John 准备建一个栅栏来围住他的牧场。他已经确定了栅栏的形状，但是他在木料方面有些问题。当地的杂货储存商扔给 John 一些木板，而 John 必须从这些木板中找出尽可能多所需的木料。

当然，John 可以切木板。因此，一个 9 英尺的木板可以切成一个 5 英尺和一个 4 英尺的木料 (当然也能切成 3 个 3 英尺的，等等)。John 有一把（完美的）梦之锯，因此他在切木料时，不会有木料的损失。

所需要的木料规格都已经给定。你不必切出更多木料，那没有用。

格式

PROGRAM NAME: fence8

INPUT FORMAT:

(file fence8.in)

第 1 行: N (1 <= N <= 50), 表示提供的木板的数目

第 2 行到第 N+1 行: N 行，每行包括一个整数，表示各个木板的长度。

第 N+2 行: R (1 <= R <= 1023), 所需木料的数目

第 N+3 行到第 N+R+1 行: R 行，每行包括一个整数(1 <= ri <= 128)表示所需木料的长度。

OUTPUT FORMAT:

(file fence8.out)

只有一行，一个数字，表示能切出的最多的所需木料的数目。当然，并不是任何时候都能切出所有所需木料。

SAMPLE INPUT

```
4
30
40
50
25
10
15
16
17
18
19
20
21
25
24
30
```

SAMPLE OUTPUT

7

分析:

一道经典的DFS-ID算法问题,分析题目是一道多背包的背包.==|||.因此模型很类似第三章的 rockers,可是本题目顺序无关.所以需要搜索.这是选择搜索的动机.但是问题的规模非常庞大,直接 DFS 必定要 TLE.所以先要使得状态缩小.方法很简单:先令背包和材料有序.接着累计背包总容量 Total.累计一个数组 numr[i]表示从 1 到 i 块材料的总费用.然后我们很容易得到.既然序列已经有序了,那么如果 numr[j]>Total,背包们肯定装不下这些材料.因此要找到一个 j'使得至少满足 numr[j']<=Total.而这个 j'就是最大可行域.总状态压缩完毕.

接着分析一下序列.因为序列已经有序了.所以我们搜索任意一个序列使其返回一个最大值,当我们计算前 k 个材料的搜索值时会返回一个不大于 k 的数字,若返回的值小于 k,那么返回值就一定是最终结果(不知这样说清楚不清楚==).因此可以 DFS-ID,这也是 DFS-ID 的动机.但是再加以分析.若我们不要确切的返回值,而是转化为判定性的问题使问题返回 true,false.分别表示.当前长度的序列是否可以放入背包组.这样问题必定是{...true,true,false...}这样的序列.易知:k 可行,k-1 必定可行.所以我们可以二分答案.

具体的实现:我们先来粗略评估一下搜索顺序.因为目标已经有序了.所以对于每个给定长度 k.我们搜索 k 个阶段.每个阶段执行完并且可以进入下一个阶段的充分必要条件是:当且仅当在本阶段可以把当前材料分割出来.这样的话,我们可以把材料由大到小搜索,让其在搜索过程中自行剪枝.这是搜索顺序的选择.

剪枝:如果按照上面的方法做可以过 3 个点(我试过==).因此要考虑强而有力的剪枝.首先:维护一个域,space 表示永远无法再利用的浪费空间.当当前背包的剩余空间小于最小的材料占用空间时,就要加入浪费空间.因为无法再用了.这样如果(space>Total-numr[k])就保证后面的都不可行了.这是一个侧重于背包剩余空间的剪枝;还有一个剪枝:当材料中出现很多相同长度的材料时,我们发现,若其中一个材料只能装在空间为 pack[i]的背包里.那么另一种只能装在大于等于它的背包里,不可能装在反而小于他的背包里.细心一些也许你就能发现:实现这个优化要对背包容量也由小到大排序.这样面对一个材料序列{...3,3,4,4,5,5,6,6...}我们倒序搜索到第一个 5 时,如只能装在 3 号背包里(假设编号越大容积越大),那么可知 1,2 号肯定无法再装另一个 5,所以搜索下一个阶段时,也就是面对下一个 5,我们从 3 号包开始搜就行了.

以上两个优化前者面向背包本身,好比对搜索树拦腰截断,使树变矮;后者面向搜索顺序,好比对每个枝杈修剪,使树变细;两者相辅相成缺一不可.这样原本一棵又高又粗地大树,被剪得又矮又细.搜索自然很快.

Translate:USACO/fence6

Fence Loops 篱笆回路

译 by Zen

描述

农夫布朗的牧场上的篱笆已经失去控制了。它们分成了 1~200 英尺长的线段。只有在线段的端点处才能连接两个线段，有时给定的一个端点上会有两个以上的篱笆。结果篱笆形成了一张网分割了布朗的牧场。布朗想将牧场恢复原样，出于这个考虑，他首先得知道牧场上哪一块区域的周长最小。布朗将他的每段篱笆从 1 到 N 进行了标号（N=线段的总数）。他知道每段篱笆的有如下属性：

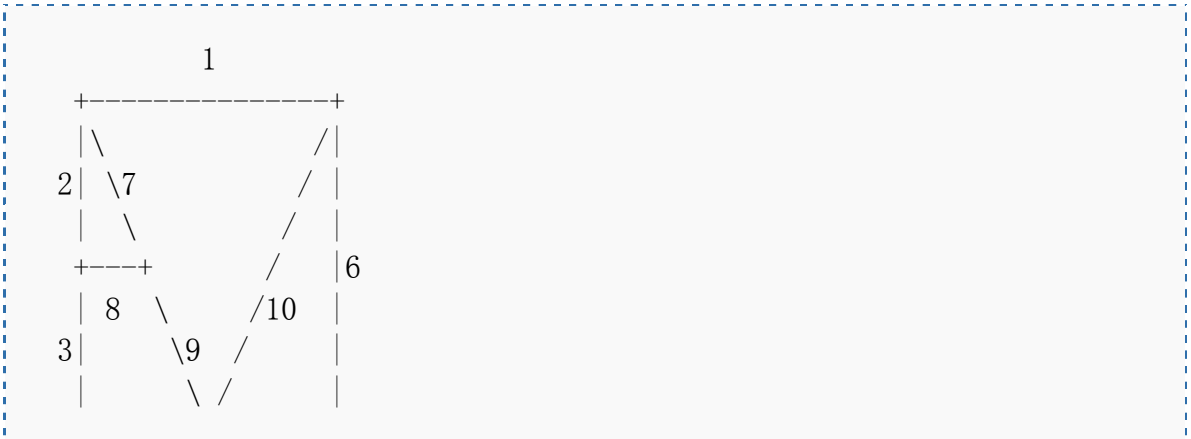
该段篱笆的长度

该段篱笆的一端所连接的另一段篱笆的标号

该段篱笆的另一端所连接的另一段篱笆的标号

幸运的是，没有篱笆连接它自身。对于一组有关篱笆如何分割牧场的数据，写一个程序来计算出所有分割出的区域中最小的周长。

例如，标号 1~10 的篱笆由下图的形式组成（下面的数字是篱笆的标号）：



```
  +-------+-------+
      4       5
```

上图中周长最小的区域是由 2，7，8 号篱笆形成的。

格式

PROGRAM NAME: fence6

INPUT FORMAT:

(file fence6.in)

第 1 行: N (1 <= N <= 100)

第 2 行到第 3*N+1 行: 每三行为一组，共 N 组信息:

每组信息的第 1 行有 4 个整数: s，这段篱笆的标号(1 <= s <= N); Ls，这段篱笆的长度 (1 <= Ls <= 255); N1s (1 <= N1s <= 8) 与本段篱笆的一端所相邻的篱笆的数量; N2s 与本段篱笆的另一端所相邻的篱笆的数量。(1 <= N2s <= 8).

每组信息的的第 2 行有 N1s 个整数，分别描述与本段篱笆的一端所相邻的篱笆的标号。

每组信息的的第 3 行有 N2s 个整数，分别描述与本段篱笆的另一端所相邻的篱笆的标号。

OUTPUT FORMAT:

(file fence6.out)

输出的内容为单独的一行，用一个整数来表示最小的周长。

SAMPLE INPUT

```
10
```

```
1 16 2 2
2 7
10 6
2 3 2 2
1 7
8 3
3 3 2 1
8 2
4
4 8 1 3
3
9 10 5
5 8 3 1
9 10 4
6
6 6 1 2
5
1 10
7 5 2 2
1 2
8 9
8 4 2 2
2 3
7 9
9 5 2 3
7 8
4 5 10
10 10 2 3
1 6
4 9 5
```

SAMPLE OUTPUT

```
12
```

做法一

求无向图中的最小环。算法很简单，就是做 m 遍 dijkstra——每次找到一条边，拿掉，求这条边两个顶点之间的最短路，那么如果这两点间存在最短路，则这条路径与原来的边就构成了一个环。这样所有环中最小的一个就是答案。问题是题目给出的是边连通的信息而非点

连通。也就是说我们得到的信息无法按照常规的方法（邻接矩阵，邻接表）来构图。这里就需要一个转化。由于我想不到什么好的算法，所以就用了一个复杂度为 O(n^2)的转化。首先将每条边的两个顶点都看作单独的点（也就是说假设所有边都不连通，为了方便，可以分别设第 i 条边的两个顶点编号为 i*2-1 和 i*2），然后对于两条连通的边，将连通这两条边的点并在一起。具体做法就是将其中一个点的连通情况全部赋给另一个点，并修改图中其他与该点连通的点信息使得合并成立。这里我借助了并查集，使得每次查找的时间都近似为常数，所以总的时间复杂度就是 O(n^2)。其中 n 是合并后总的点的个数经过上述转化以后，再用 m 遍 dijkstra，总的算法时间复杂度就是 O(mn^2)。

做法二

这道题给的数据是边之间的关系,首先想到的是构图,然后套经典的最短路做.但是这道题的数据输入会使按点构图很麻烦,其实这道题搜索即可,而且加剪枝后很快.由 v 条边开始搜索在它 b 方向的边,重复这个过程,直到回到 v 而且是由 a 方向回到的,为了判断环可以加个判重,一个剪枝:对于路径长度大于已知最小环的可剪.

做法三

一种比较诡异的方法。就是把边变成“点”。在把“边”的长度设定为边两端的“点”（既原来的边）的长度的和。

用由 floyd 算法改进的求最小环的算法求最小环。注意的是枚举“点”时，注意“点”连接的两“点”要是分别连接到该“点”代表的边的两个端点。

这个复杂度为 O(n^3)。超级快……

做法四

仍然是求最小环，先将边邻接矩阵转为点邻接矩阵，再 dfs 求出所有环。可以证明时间复杂度为 O(n^3)，所以能快速 AC 还是 DFS 好 代码也简单

Translate:USACO/cryptcow

Cryptcowgraphy 解密牛语

Brian Dean

译 by Jeru

描述

农民 Brown 和 John 的牛们计划协同逃出它们各自的农场。它们设计了一种加密方法用来保护它们的通讯不被他人知道。

如果一头牛有信息要加密，比如"International Olympiad in Informatics"，它会随机地把 C，O，W 三个字母插到到信息中（其中 C 在 O 前面，O 在 W 前面），然后它把 C 与 O 之间的文字和 O 与 W 之间的文字的位置换过来。这里是两个例子：

```
International Olympiad in Informatics
->
CnOIWternational Olympiad in Informatics
International Olympiad in Informatics
->
International Cin InformaticsOOlympiad W
```

为了使解密更复杂，牛们会在一条消息里多次采用这个加密方法（把上次加密的结果再进行加密）。一天夜里，John 的牛们收到了一条经过多次加密的信息。请你写一个程序判断它是不是这条信息经过加密（或没有加密）而得到的：

```
Begin the Escape execution at the Break of Dawn
```

格式

PROGRAM NAME:cryptcow

INPUT FORMAT:

(file cryptcow.in)

一行,不超过 75 个字符的加密过的信息。

OUTPUT FORMAT:

(file cryptcow.out)

一行，两个整数. 如果能解密成上面那条逃跑的信息，第一个整数应当为 1，否则为 0；如果第一个数为 1，则第二个数表示此信息被加密的次数，否则第二个数为 0。

SAMPLE INPUT

```
Begin the EscCution at the BreOape execWak of Dawn
```

SAMPLE OUTPUT

```
1 1
```

分析:

这到题目很考察思路的条理性，其实剪枝比 Fence Rails 要容易些，但是编程复杂度要大，因为对于数字来说，字符串的处理很繁杂。但是仔细想一想还是不难解决的。基本思路是枚举'C','O','W',然后迭代删掉选中的"COW"组合，继续直到没有"COW",判断，输出。

这里有三个优化:
1.因为为了使解的可行性加大，所以我们希望"COW"正好"包着"加密串，因此搜索顺序应该是先枚举'O'然后是正序的'C'和倒序的'W'.
2.搜索时必定会有一些重复，所以要用散列表判重，经过 Gjb 神牛测试显示:BKDRhash 性能最优，ELFhash 性能最次，冲突很多，所以建议使用 BKDRhash 或者 DJBhash.
3.接着要进行可行性剪枝:因为观察数据，所有的对调都不会影响子串的顺序，所以我们检查每一个被"COW"相隔开的加密串是否是目标串的子串，如果不是则直接剪枝。判断时应该先检查前缀和后缀，然后枚举中缀。