训练联盟周赛第一场

UCF Local Programming Contest 2014 试题解析

CSUFT ACM集训队

Vowel Count

Description

给出一个字符串，规定元音字母为”aeiou”，问字符串中元音字母是否多于辅音

Vowel Count

Solution

按题意模拟即可，注意输出格式

Soccer Standings

Description

球赛中胜利得3分，平局得1分，失败得0分，现在知道一个球队的比赛次数和得分情况，问所有可能的赢、平和输的组合。

Soccer Standings

Solution

设x为该球队胜利的场数，y为该球队平局的场数，z为该球队失败的场数，P为该球队总得分，G为球赛总场数。可得：

x+y+z=G

3*x+y=P

将z看作常数，求出x,y的解：

x=(P-G+z)/2;

y=(3*G-P-3*z)/2;

然后遍历z的值（依照题目，需要从大往小的方向遍历），找出满足题目要求的解即可。

Jumping Frog

Description

一只青蛙要跳到距离起点C个格子的终点处，道路上有的地方可以跳，有的地方不能跳，青蛙每次最多跳过c个格子，问青蛙跳到终点的最小跳跃次数。

Jumping Frog

Solution

设 dp[i] 为青蛙跳到下标i的位置的最小次数。
状态转移方程：

str[i] == '.', dp[i] = min(dp[i-1], dp[i-2], ... , dp[i-d-1]) + 1;

str[i] == '#', dp[i] = inf;

初始状态：dp[0] = 0;
所求答案即为dp[c-1]。

Fujiyama Thursday

Description

有c辆车，每辆车可以坐4个人，每辆车到达目的地要di分钟（1<=i<=c）。有4*c 个人，每个人可以自由搭配乘车去目的地，但达到目的地后需吃完饭，每个人吃饭的时间不同。问如何安排使得总时间最少。

Fujiyama Thursday

Solution

贪心，让吃饭时间最长的四个人乘最快的车，吃饭时间次长的四个人乘第二快的车，依次乘车。用一个变量记录其中用的最大时间即可。

Chain Email

Description

有n个老人，每个老人可以给他的好友群发邮件。问以老人s为起点发邮件，朋友圈是否会进入邮件接收的死循环。

Chain Email

Solution

根据题意，需要求出从起点出发能到达的环，以及从这些环出发能到达的点。
首先重新构图，将起点所不能到达的点删掉，同时记录新图中所有点的入度。
然后根据新图拓扑排序，记录排序后入度不为0的所有点，即为答案。

Faster Microwaving

Description

给出建议烹饪时间MM：SS和表示较低和较高建议烹饪时间的百分比p，设定烹饪时间需要按按钮，按按钮和移动到另一个按钮都需要花费时间。

时间（MM：SS）有两种表示方式 MM：SS 和 MM-1:SS+60，问应该按下的按钮序列，使得按按钮花费的时间最少。

Faster Microwaving

Solution

根据所给的具体时间，依次向两边枚举满足条件范围的时间，然后分别计算某一具体时间两种表示方式的按键次数，更新最小次数和时间表示方式。最后输出时间表示方式。

Dirty Plates

Description

有两面都干净的盘子，一面干净的盘子，两面都不干净的盘子，三种盘子各有c，s，d个，这些盘子堆叠在桌子上成一列，盘子干净的一面或桌子接触到另一个盘子脏的一面或会变脏，干净的桌面接触到盘子脏的一面也会变脏。

Louie每次只吃最上面盘子的干净面，每次吃完可以重新堆叠，问最多能吃几次。

Dirty Plates

Solution

设两面干净的盘子有x个,一面干净的盘子有y个，两面都不干净的盘子有z个，总共干净的盘子的面数为2*x+y个

对于普通情况，人可以吃的顿数最多为(2*x+y)/2+1顿。（x=y=0时需要特判）

而对于特殊情况（就是没有一面干净的盘子的情况）

若z=0，则吃的最多顿数为x+1;

若z>0，则干净的面数需要减少2(包括桌面)，则吃的顿数最多为x；

Chocolate Fix

Description

食物和形状有九种组合方式（“SV”, “SS”, “SC”, “RV”, “RS”, “RC”, “TV”, “TS”, “TC”），这九种字符串要放置在3*3的网格上，给出几个线索，分别表示网格的几个部分，求出满足这些线索的解决方案。

Chocolate Fix

Solution

食物和形状有的九种组合方式总共有9！种方案。枚举每一种方案，判断是否满足所有线索，满足的即为答案。

把0-8这9个数对应为所求答案的九宫格，每个数代表上述九个字符串的一种，枚举0-8的所有排列，即可对应字符串的所有摆放方式。

与线索比较是否相等时，枚举九宫格的左上角位置，一一比较就行了。

Shopping Spree

Description

有n个商品，价值分别为$I_1, I_2 \ldots I_n$，现在要从中选出一个子集，要求如下：

如果选择了I_k，则在1~k个商品中最多选择$\lfloor\frac{1+k}{2}\rfloor$个商品。

问如何选择，能让子集的总价值最大，输出价值最大的总和。

Shopping Spree

Solution

注意，题目中的条件是对所有元素都成立的，不存在只要选最末元素便能得到最大子集从而价值最多的解法，而是得从头到尾按照条件依次迭代，在这过程中，不难发现最优子问题。

设 $F(i,j)$ 表示 在 前i件商品的范围内 只能买j件商品的条件下 所能获到的最大价值。

对于第 i 个商品：

- 不买 最大价值为$F(i-1)(j)$，即延续前i件商品范围内所能获得的最大价值。
- 买 最大价值为$F(i-1)(j-1)+V_i$，即在前$i-1$件商品范围内只能买$j-1$件商品的条件上，再买当前的i商品，同时，第i件商品产生的价值可能比已选的某件商品产生的价值小，故需要和$F(i-1)(j)$比较。

Factorial Products

Description

给出三组数字:A、B和C,分别将每组中各数字的阶乘相乘，得到ProFact(A)、ProFact(B)、ProFact(C)。求ProFact(A)、ProFact(B)、ProFact(C)的最大值。

Factorial Products

Solution

利用自然对数将式子中每个值的阶乘相乘化为每个值阶乘后的自然对数相加，同时每个数的阶乘的自然对数同样可以利用自然对数化为每个数的自然对数相加，这样只要将每个数阶乘后的自然对数打表打出来，然后对于每组，只要将所有数阶乘的自然对数相加就可以得到ProFact值的自然对数，这就避免值过大而不能比较。最后将每个列表的ProFact值的自然对数进行比较就行。

举个例子：

$$2! * 4! * 7! = 241902 \rightarrow \ln 2! + \ln 4! + \ln 7! \rightarrow (\ln 1 + \ln 2) + (\ln 1 + \cdots + \ln 4) + (\ln 1 + \cdots \ln 7)$$

Super Lucky Palindromes

Description

定义超级幸运回文数字满足：① 由4和7组成的正整数（幸运数字）； ② 数字的位数是幸运数字 ③ 4的个数和7的个数至少有一个是幸运数字。

求第k小的超级幸运回文数字。

Super Lucky Palindromes

Solution

由题意知，在10的18次方内，回文串的长度只能是4，7，44，47，74，77，444这7种，同时题目也要求4或7的个数为幸运数字，故处在后面阶段的回文串中4或7的个数为该阶段前面的回文串的长度，若该阶段的长度为偶数，则只可以去前面阶段的偶数长度，若为奇数，则没有限制。

然后运用组合数来计算上面阶段中每个阶段的回文串个数，（举例44位回文串的临界值：由于44是个偶数，在其前面为偶数的长度为4，所以其回文串个数为（C[22][22]+C[22][2]）*2,因为这是回文串，所以总长度乘以2，4或7的需要个数也除以2，即可得出该阶段回文串的个数（奇数情况很简单，减去最中间的那个再除以2即可）），然后就可以得到每个阶段的起始值。

Super Lucky Palindromes

Solution

首先判断输入的是哪一种阶段的，如果处于‘444’这个阶段，那么由计算得，其第1-157位，288-444位都是‘4‘（可通过组合数计算），然后从最外层开始向内缩短范围，如果其算出来的数大于等于了所给的k，则未确定值范围的最外层确定为‘4’；如果小于，则确定为‘7’。然后向内压缩未确定值的范围，直到无法压缩为止。

此外，最后要对奇数个数的回文串阶段做个特判，因为最中间是4还是7是未知的，如果4的个数符合幸运数，则该处填7，若7的个数符合幸运数，则该处填写4。若两者皆不符合，如果4的个数加1等于幸运数，则该处填4，若7的个数加1等于幸运数，则该处填7。

Under Construction Forever

Description

给定一个连通图，进行重构操作，重构操作即：

如果点A是点B的叶子结点（点A只连接点B），那么可以对点B进行重构（把A拆除，与B合并），成本为点B的点权；如果点B有多个叶子结点，对B进行重构时叶子节点，成本为点B的成本。

然后一直重构，直到不能重构为止。

求剩余点的最小数量，实现最小数量所需的最小成本和以最小数量实现最小成本的重构方案数。

Under Construction Forever

Solution

首先把所有点分为两种类型，叶子结点和分支结点。如果存在环，分支结点可以分为环上结点和非环上结点。

求最小数量很简单，就是如果图存在环，那就是环上所有结点数量；不存在环就是1。

求最小成本，如果无环，就是所有非叶子节点的成本和；如果有环，即为环上所有带分支的结点（连接了非环上结点的结点）和其分支结点中的非叶子节点的成本和。

求实现最小数量最低成本的方案数，因为无论怎样重构，只要重构到不能重构都会达成最小数量最低成本。如果无环，就是以所有分支结点分别做树的根结点，接下来每一层里有n个分支结点（叶子结点不能算）就乘C_n^1，然后相加即为总方法数；

如果有环，就是以环上结点为根结点，求它下面的所有分支结点的组合数，也即是每一层分支结点数量为n就乘以C_n^1，但是最后结果不能相加，因为环上的结点都是可以随意挑选一个开始，可以随时换别的环上结点，所以需要结果相乘并且乘上选择到每一个最底层分支结点（连接叶子结点的分支结点）的组合数，举个例子：如果结果存在环，环上为3个结点a,b,c，这三个结点分别直接或间接连接1，2，1个叶子结点，在所有结果都相乘之后，还需乘$C_4^1C_3^2C_1^1$。

以上过程可以递归实现。

Under Construction Forever

Solution

答案分三种情况：

1.当边数只有1的时候，这时只有两个结点和一条边，最小数量肯定为1。最小成本就是两个结点中成本低的那个。如果两栋结点成本相同，那么有2种方法可以达到最小数量最低成本，如果不同就只有1种方法即把成本高的合并给成本低的。

2.当边数为n-1时，根据题意可以知道这是一个树，不存在环。最小数量为1。最小成本为所有非叶子结点的成本和。方法数为以每个分支结点为根结点求的组合方案数之和。

3.当边数不为n-1时，由题意可知肯定存在环。最小数量为环上结点数。最小成本为所有带分支结点的环上结点与其分支结点的成本和（即所有非叶子结点除了环上没有带分支的结点的成本和）。方法数为以环上结点为根结点，它下面的所有分支结点的组合数之积并且乘上选择到每一个最底层分支结点（连接叶子结点的分支结点）的组合方案数。

THANKS

Algorithm&Share