Asia Urumqi Regional Contest Onsite 2017

Problem Analysis

Problem A. Coins

最优的策略一定是：当有至少 k 枚硬币面朝下时，则选 k 枚面朝下的硬币去抛掷（任意 k 枚都可以）；如果不足 k 枚面朝下，则在选择所有面朝下的硬币的基础上再额外选择若干面朝上的银币。

于是有动态规划，记 F[i][j]表示抛掷了 i 次后，有 j 枚硬币面朝上的概率。他们应该满足 F[i][0]+F[i][1]+...+F[i][n]=1。转移时，考虑从当前状态(i,j)出发，抛掷的 k 枚硬币的所有可能结果：分别有 0~k 枚面朝上。其中 k 枚硬币抛掷后有 l 枚面朝上的概率为 $C(k,l)/2^k$。

时间复杂度 O(nmk)。

Problem B. The Difference

为了最大化 $A_1A_2-A_3A_4$，一定是最大化 A_1 和 A_2，同时最小化 A_3 和 A_4，那么 A_1 和 A_2 当然是相对较大的 2 个数字，A_3 和 A_4 则是相对较小的两个。

Problem C. The Number Triangle

假设 S[i][j]是从(i,j)位置向上走出来的最优路径（路径权值和最大）的最小字符串（字典序意义下）。那么对于固定的 i，考虑所有 S[i][j]的字符串之间大小关系，并赋予他们排序数组 rank[i][j]。

那么 S[i-1][*]就可以利用 rank[i][*]维护出来，因为需要比较的字符串总是形如 S[i][*]的。我们还可以利用 rank[i][*]去维护出来 rank[i-1][*]，因为 S[i][j]总是对应 r[i][j]（在位置(i,j)的字符）与某个 S[i-1][*]拼接出来的字符串，所以他们的大小关系可以利用 rank[i][*]计算出来。

基于桶排序，可以做到时间复杂度 $O(n^2)$。

Problem D. Fence Building

最优策略总是在选定 n 个点后，两两连线，且满足任意三条线不交于一点。

尝试统计总的结点个数 A(n)与独立线段（包括圆弧上的 n 段小弧）的总个数 B(n)，然后利用欧拉公式就可以得到答案 Ans(n)=B(n)-A(n)+1。

任意四个点，会形成一个交点，并贡献额外的 2 条独立线段。所以 A(n)=n+C(n,4)，而 B(n)=n+2C(n,4)+C(n,2)，所以最后答案为 C(n,2)+C(n,4)+1。

Problem E. Friends

对于 p 个函数记他们形成的集合 V={0,1,...,p-1}。那么每一个 valuable circle of friends 形成了 V 的一个子集，这些集合全体形成了 V 的幂集的一个子集 E。问题变成找在 E 中同时出现次数不少于 lambda=[(p+1)/2]的所有极大集合。

这实际上是一个经典的关联式规则学习问题。利用先验算法可以很高效的解决。

具体来说，依次找出所有大小为 1, 2, 3, ...的满足在 E 中同时出现的次数不少于 lambda 的所有集合，依次记录为 L(1), L(2), L(3), ...。用 L(i)去推 L(i+1)：选取 L(i)中一个方案 U，U 的大小为 i；向 U 中尝试加入新的元素 e，形成 W=U+{e}，检查 W 的所有大小为 i 的子集是否在 L(i)中；对于筛选出来的 W，再去 E 中检验是否合法。

利用位运算（这里需要拆成 2 个 64 位整数来表示）可以加速判断一个集合是不是另外

一个集合的子集。这便可以解决这一题。

Problem F. Gathering Minerals

首先注意到这一题答案不会很大，因为所有时间都是 10 的倍数，实际上最大用时除以 10 大概不会超过 750。所以可以想办法在本地跑出来所有时间 t 可以获得的最大矿物量，并提交一个带有上述计算出来的数据的程序（也就是说可以打表）。

所以说，这一题对时间效率实际上是几乎没有限制的。

再注意到一开始的资源配置是 1 个基地，4 个工人，6 份供给和 50 份矿物。那么就有了一个比较容易估计出来的最优策略（虽然它的正确性并不容易说清楚）。

第一阶段，以建立尽可能多的基地为唯一目标。

第二阶段，考虑制造供给，和制造新的工人为目标（注意二者的关系，供给只有在制造新的工人的时候需要，他们二者又同时需要矿物）。

第三阶段，全心全意挖矿。

相邻阶段之间的分界线，以及第二阶段的分配，需要通过搜索来考虑所有可能的细节。具体来说，对于每一个当前状态，除了需要维护此刻的基地个数，工人个数，供给量还有矿物量，还需要维护之后 12 个时间点（因为所有耗时都是 10 的倍数，所以这里以每 10s 为一个时间点）会额外得到的工人数。

对于同一时刻的两个不同的状态，可以通过一些方法去对他们进行比较，并淘汰掉一定不优秀的那些状态策略。例如，如果其中有一个状态之后 12 个时间点的物资都不劣于另一个状态，则后者一定没有存在的必要，可以被淘汰掉。更强力的淘汰方法可以单独对最近的若干个时间点（比如最近的 2 个时间点）做出一些相对较优的、手动构造性的策略，从而更精细的去比较和淘汰状态。精细的处理可以使这一题几乎在 1 秒内搜出来所有情况。

最后注意，本题矿物的规模到达了 64 位无符号整数的极限，实际上注意到他们都是偶数，可以全部除以二，从而确保中间过程的计算不会超过 64 位无符号整数。

Problem G. The Mountain

题目给出的所需求面积的图形，可以被拆分位若干个梯形（包括退化成的三角形）的并，分别求每一个梯形的面积并累加就可以了。

Problem H. Count Numbers

记 F[i]表示数位和为 i 的不含 0 整数有多少个，再记 G[i]表示数位和为 i 的不含 0 整数的和。那么 F[i] = sum F[i-j]，G[i] = sum 10G[i-j]+jF[i-j]，其中 j 枚举了最低位的数字，并从 1 遍历到 9。

不妨同时维护相邻的 9 个位置的值（共 18 个），则可以构造 18 阶矩阵 A，实现它的快速转移。整个问题的答案也对应了 A 的 a^b 次幂。可以利用矩阵乘法的结合律实现它的快速计算。

Problem I. A Possible Tree

记 x(i)为结点 i 到根的路径上所有边的亦或和（这里可以忽视 x(0)的情况），则每一条信息实际上给出了 x(u) xor x(v)的值。考虑用带权并查集来维护他们的关系。

对于每一个集合，选择代表元 a，其余元素 b 记录与代表元的“亦或差”，即 x(a) xor x(b)。对于属于不同集合的 2 个元素 u 和 v，权值为 w 的信息给出了 u 和 v 所在集合分别的代表元 u’，v’之间的亦或值 x(u’) xor x(v’) = (x(u) xor x(u’)) xor (x(v) xor x(v’))。同一个集合（代表元为 c）内两个元素 a 和 b 的亦或值为 x(a) xor x(b) = (x(a) xor x(c)) xor (x(b) xor x(c))。

整个题目可以在线性时间内解决。

Problem J. Lowest Common Ancestors

不妨先以 1 号点为根，记 F(u,v)是 u 和 v 在以 1 为根的树内的 LCA。

之后考虑如果以任意结点 x 为根，那么对于每一对(u,v)，如果 u 到 v 的路径经过 x，则 LCA(u,v)=x；否则，他们的 LCA 一定是 F(u,v)，F(u,x)和 F(v,x)之一，具体是哪一个，要看谁距离 x 更近。

具体来说，依然考虑以 1 为根的有根树，记 Tx 是以 x 为根的子树，并记 Ans(x)是以 x 为根的情况下的答案。对于每一个结点 x 分成三部分来计算答案：

（i）对于 u 和 v 中一个在 Tx 内，一个在 Tx 外的情况，他们在 Ans(x)中贡献的值为 x。

（ii）如果有 u 和 v 同时在 Tx 内，他们在 Ans(x)中贡献的值就是 F(u,v)。

（iii）如果 u 和 v 都在 Tx 外，考虑从根一路走过来他们所在的位置，如果他们同时属于根到 x 的路径上某一个分叉内，则他们贡献的值依然是 F(u,v)，否则考察两个分叉结点中高度较大的那一个。

先线性的求出来所有 F(u,v)，上述的三部分可以分别利用三次 DFS 遍历来实现。在（i）和（ii）中，分别要维护子树内完全出现的询问对的 LCA 和，以及才出现一次的询问个数。而（iii）则需要先处理高度小的结点，并将对应部分的权值从父节点的编号修改为子节点的编号。

整个算法是线性的。

Problem K. Sum of the Line

考虑基于容斥原理的计算过程。如果 T 中所有位置 T(r,c)=c 总成立，那么答案为 $1^2+2^2+...+k^2$。下面考虑删去 T(r,c)=0 的位置造成的代价。

考虑若干两两不同的素数 $p_1, p_2, ..., p_u$ 并记 $x=p_1p_2...p_u$。则由容斥原理，上述答案需要减去$(-1)^{u+1}$倍的 $x^2(1^2+2^2+...+[k/x]^2)$。在 $k\le 10^8$ 下，u 不超过 9，所以只需要枚举 2^9 种可能即可。