Solutions for Pacific Northwest Region Programming Contest Division 1

中国人民大学 ACM 队

2020 年 2 月 21 日

目录

1 Alphabet

1.1 Description

我们定义一个小写字符串是“按字母表的”,当且仅当它删除掉一些字符后,可以变为“abcdefghijklmnopqrstuvwxyz” 的字母表。

给定一个长度为 n 的小写字母字符串，询问至少插入多少个字符才能使其变成 “按字母表的”。

$n <= 50$

1.2 Solution

不难看出后来添加的字母一定不会在变成 “abcdefghijklmnopqrstuvwxyz” (以下简写为 a $\sim$ z) 被删去，不然没有加入的必要。

所以为了将原字符串变成 a $\sim$ z，我们考虑先删除再添加。

不难看出删除后的字符串字典序一定严格递增，不然不能仅通过添加得到 a $\sim$ z. 而严格递增的序列一定是 a $\sim$ z 的子序列，可以经过添加变成 a $\sim$ z.

所以任务变成寻找输入串的最长上升子序列，答案就是 (a $\sim$ z $-$ 最长上升子序列长度)。

设以第 i 位为结尾的最长上升子序列长度为 f_i, 得到转移方程:

$$f_i = max\{f_j + 1\}\ (i > j\ \&\&\ \text{第}\ i\ \text{位字典序大于第}\ j\ \text{位})$$

2 Buggy Robot

2.1 Description

一个机器人要走出一个 $n \times m$ 的矩形网格迷宫，迷宫中的每个格子为障碍物或者为空。已知迷宫的布局、机器人的起始位置和迷宫的唯一出口位置。保证存在从起始位置到出口的路径。

已知一串给机器人的指令，指令由若干 'U' 、'D' 、'L' 或 'R' 组成，分别表示上、下、左、右四个方向。机器人会按照指令的顺序依次向对应方向移动一格，若碰到障碍物或迷宫边界则在原地不动。一旦机器人到达终点，后续指令都将被忽略。

给出的这串指令并不一定能使机器人到达终点，现在要求修改该指令使得机器人能走出迷宫，修改方法是每次可以花费单位代价来删除某一条指令、或者在指令串的任意位置插入一条指令。求最少花费多少代价可以使机器人按照指令能够到达迷宫出口。

$n, m \leq 50$ ，给出的指令串长度范围为 $l \in [1, 50]$

2.2 Solution

可以发现，机器人的状态由其所在位置和已经执行了多少条原指令串中的指令（包括删除的指令）所确定，这样的状态数为 $O(nml)$ 。而状态的转移只需枚举不进行指令的修改、删除下一条指令、插入一条指令这些决策即可。

将状态视作图上的节点，转移视作图上的边，则问题转化为了最短路问题。而图中的边权只可能为 0 或 1 ，因此只需对 bfs 算法稍加修改即可求出最短路。具体方法是使用一个双端队列，搜索过程中将通过边权为 0 的边扩展的状态加入到队首，通过边权为 1 的边扩展的状态加入到队尾。需要注意的是，这样修改之后一个元素可能在队列中出现两次，需要进行判断。具体实现时可以使用 'C++' 标准库中的 'std::deque' 。

总时间复杂度为 $O(nml)$ 。

3 Cameras

3.1 Description

有 n 个房子，从 1 到 n 标号一字排开。每个房子可以选择装或不装摄像头，要求任意连续 r 个房子中至少有 2 个房子有摄像头。现在有 k 间房子已经安装了摄像头，问最少还需要安装几个摄像头。

2 n 100,000，0 k n，2 r n。

3.2 Solution

不难想出如下的贪心策略：从左到右扫描每个区间，如果该区间内不足 2 个摄像头，则从右边未安装摄像头的房子开始安装起。这是因为越靠右边的摄像头越能被后面更多的区间共有。如果从右向左扫描区间，则优先安装靠左的摄像头。

下面证明该贪心策略的正确性：

1）最优子结构

设一个最优解 S 有一个子问题 S' 表示 $1 \sim t$ 房子的摄像头安装情况。如果 S' 不是最优的，则存在一个比 S" 更优的解，使得 S 更优。此时，S 就不是最优解，与假设不符。因此，若 S 最优，其子问题 S' 必然最优。最优子结构得证。

2）贪心选择性质

如果一个区间不足两个摄像头时，不选择从最右（不妨记为 y）开始安装，并安装在 x 处。则右边的所有区间不会有更多的摄像头，其中某些区间 [x, x + r - 1] 到 [y, y + r - 1] 还会减少摄像头，显然使解更劣。贪心选择性质得证。

4 Contest Strategy

4.1 Description

一 acmer 有 n 道题要做，他在看每道题后会瞬间了解需要花的固定时间 t（不同题目所花时间可能相同），他可以最开始同时了解 k 道题的内容，并选出耗时最小的题目来做（若有多到题目耗时最小，随机选择一道），每做完一道再看一道，并选耗时最小的来做，直到题目全部做完，现求对于 n! 种看题顺序，所花时间的和为多少。

其中 $n, k \leq 300$，任意的 $t \leq 10^6$

4.2 Solution

首先由于有花时间相同的题目，不同的看题序列造成的时间序列是可能相同的，故而规定若耗时相同，保证先看到的先做，建立看题序列与所耗时间一一对应的关系。

本题中最重要而显然的性质是在一道题目被做时，同时看的 k-1 道题目耗时都大于等于该题，进而可推知最后的 k-1 道题的耗时就是所有题目中耗时最长的 k-1 道题。

其余的题目的贡献，考虑枚举第 i 道题第 j 个被做出，那么此时看题序列一定推进到了第 j+k-1 个位置，那么统计满足这前 j+k-1 个元素使得第 i 第在此时被做的排列方案即可。

分类讨论。若做该题前做过耗时更大的题，那么它一定刚看到就被做了，前面至少有 k-1 个耗时更长的题，可由之前的结论推出。否则一定是最后一个看到的题“耗时更长”（可以认为耗时严格相同的情况下，先看到的“耗时短”，后看到的“耗时长”）且恰好达到共有 k-1 个“耗时更长”的题目。

此时情况已讨论完全，枚举并列式计算即可。

5 Enclosure

5.1 Description

一共有 n 个点，你拥有其中的前 k 个，并从剩下的 n-k 个点中选择一个加入你的 k 个点中，使得最小的包含这 k+1 个点的多边形最大。求最大的面积，保留一位小数。

$3 \leq k < n \leq 10^5$

$|x_i|, |y_i| \leq 10^9$

5.2 Solution

考虑“最小包含这 k+1 个点的多边形”的含义，显然是指这些点的凸包。

接下来考虑 k+1 个点的凸包（记为 A）和 k 个点的凸包（记为 B）的关系，可以发现以下性质：

1. A 上的点（新加进来的点除外）也一定在 B 上；
2. 在 B 上且不在 A 上的点，必然是 B 上连续的一段；
3. 若新加入的点不在 B 的内部，那么这个点一定在 A 上。

综上，我们可以先求出 k 个点的凸包，然后根据以上性质快速维护 k+1 个点的凸包及其面积。

6 Illumination

6.1 Description

有一个 $n \times n$ 的网格图，网格中有 l 盏灯，每盏灯可以照亮它所在的行或列，照亮的范围至多是灯的两侧各 r 个格子（共 2r+1 个）。

判断是否存在一种方案，使得不存在任意两盏同行或同列的灯照亮同一个格子（注意不同行且不同列的灯可以照亮同一个格子，满足要求）。

$n \leq 1000, r \leq 1000$

6.2 Solution

注意到灯只有两种状态，即照亮行/列二选一。

问题转化为一个 2-SAT 问题，命题为“第 i 盏灯照亮所在列”，则限制关系转化为某两个命题不能同时成立/不成立。

接下来按照 2-SAT 的建图方法，拆点连边即可，最后运行一次 Tarjan 算法，计算出每个点所在的强连通分量，判断是否出现矛盾情况。

7 Maximum Islands

7.1 Description

给出一张 $N \times M$ 的网格图，L 表示陆地，W 表示水，C 表示不确定，为陆地或水其中一种。两块陆地联通当且仅当从一块陆地开始可以通过向周围上下左右四个格子移动互相到达。确定全部的 C 使得陆地不联通的块数最多。

$1 \leq N, M \leq 40$

7.2 Solution

不难发现每一块陆地联通块越小越好，那么我们先使已给出的陆地周围的 C 确定为 W。

对于剩下的 C，我们要在里面取一些不相邻的格子，并让数量最大化。

这显然是一个最大独立集。又因为是网格图，所以将相邻的 C 连边可以得到一个二分图，二分图的最大独立集等于顶点数-最大匹配数。建立一个虚拟源点和虚拟汇点，从源点向奇位置的 C 格子连边，奇位置的 C 格子向相邻偶位置的 C 格子连边，偶位置的 C 格子向汇点连边，流量上限都为 1，求最大流即可。

8 Paint

8.1 Description

有一段长度为 n 的围栏标号 $1 \sim n$，有 k 位画家将会在 $[L_i, R_i]$ 的围栏画画，你要选择一个画家的集合，使得集合中画家的绘画区域两两不相交，在此基础上使得未被画的围栏数量最小。

8.2 Solution

题目等价于使得被画围栏数量最多。将 [Li, Ri] 看作线段，按右端点排序，则可以用 dp 解决：

设 f_i 表示选择线段 i 的最大被画围栏数，则 $f_i = max(f_j + 1)(R_j < L_i)$，寻找 j 的过程即在区间 $[1, L_i)$ 上找 f 最大值，可以用线段树/树状数组优化为 O(logn)，总复杂度 O（nlogn）.

注意到我们按 R 顺序遍历，但判定条件为 L。如果我们将每一条线段拆成 L_i 和 R_i 两个点，排序后依次遍历，则对每条线段 i，在遍历到 L_i 时就可以更新 f_i 了，我们可以存一个当前最优值 $dmax = max(f_i), R_i < K$, 在遍历到 R_i 时更新一下 dmax，这样遍历 L_i 时就可以直接用 dmax 更新 f_i，复杂度降至 O(n)，加上排序仍为 O(nlogn).

9 Postman

9.1 Description

有一名邮递员要在一个坐标轴上从邮局送信到 n 个地点，每个地点有 m_i 封信件。

邮局在位置 X = 0 处，邮递员每移动 1 个单位长度需要 1 个单位时间。邮递员一次至多带 k 封信。求邮递员从出发到送完信返回邮局至多需要的时间。

$n \leq 1000, k \leq 10^7$

$|x_i|, m_i \leq 10^7$

9.2 Solution

当邮递员到某一半轴送信时，若要去另一半轴则必须返回邮局，因而将两个半轴分开考虑。

单次送信时间只由信件中需要送达的最远距离决定，因而设计贪心策略：每次从远至近取 k 封信进行寄送，裸算法复杂度 O($\frac{nm}{k}$)，注意到可能超时，优化后变为 O(n)

10 Shopping

10.1 Description

排成一排，编号为 $1 \sim n$ 的商品，第 i 个商品的价格为 a_i。

现在有 q 名顾客，第 i 位顾客有 v_i 元，并会从第 l_i 个商品走到第 r_i 个商品。

对于经过的每个商品，每个顾客都会一直购买该商品直到没有足够的钱再次购买该商品，然后再移动到下一个商品。问每个顾客最后各自会剩下多少钱。

10.2 Solution

根据题意，每个顾客的余额将会依次对经过的每个商品的价格进行取模操作，而显然只有比当前余额要便宜的商品才会在取模操作中对余额造成影响，那么一个简单的思路就是模拟每位顾客的行动，每次找到下一个价格比当前余额小的商品，并直接用它对余额取模，最后得到答案。

这个方法的复杂度为 O(q × 找到下一个有效商品的复杂度 × 完成一个顾客所需要找的商品数)。

如果使用倍增或二分 rmq 等方法优化的话，找到下一个商品的复杂度将为 O(logn)。

而对于每个顾客，若我们找到了一个有影响的商品，则这位顾客的余额在这次操作后将不会多于原本余额的一半，因此完成一个顾客所需的次数的复杂度最大将为 O(logv)。由此可得总复杂度为 O(qlognlogv)，可以通过此题。

11 Tournament Wins

11.1 Description

你要参加一场共 2^k 人参与的单淘赛制的比赛。已知你在所有人中实力排名为第 r 名，比赛时排名靠前的一定赢（没有并列）。比赛分组完全随机。问你的胜场数的期望。

$1 \le k \le 20, 1 \le r \le 2^k$

11.2 Solution

令：p_i 表示第 i 场胜利的概率，

指示器变量 $X_i = \begin{cases} 1, & win \\ 0, & lose \end{cases}$，

随机变量 X 表示总胜场。

因此可知：$X = \sum_{i=1}^{k} X_i$，

$E[X] = \sum_{i=1}^{k} E[X_i] = \sum_{i=1}^{k} p_i$。

所以只需计算每一场获胜的概率 p_i 即可。

要计算第 i 场获胜的概率，只与同组的 $m = 2^i - 1$ 个选手的选取有关。已知：总人数（除去主角）$N = 2^k - 1$，

比主角弱的人数 $M = 2^k - r$，

因此：这 m 个人的选取方案总数为 $\binom{N}{m}$，

可以使主角获胜的方案数为 $\binom{M}{m}$，

所以 $p_i = \binom{M}{m} / \binom{N}{m} = \prod_{j=0}^{m-1} \frac{M-j}{N-j}$。

总时间复杂度为 $O(2^k)$。

12 Windy Path

12.1 Description

给定一个包含 n 个二维平面上整点的点集，以及一个长度为 n-2 的只包含字符'L' 和'R' 的字符串，请构造长度为 n 的点列，满足：

- 每个点都从给出的整点集中选出；
- 每个点仅出现一次；
- 对于第 i 个字符，若为'L'，则第 i，i+1，i+2 个点成逆时针，反之，成顺时针

保证不存在三点共线的情况，$n <= 50$, $1 <= x_i, y_i <= 1000$

12.2 Solution

考虑一种将问题缩小的办法：假设第 i 个点已经确定，若第 i+1 个点可以满足之后的点无论怎么选都在 i, i+1 点的左（或右）侧，那么之后就无需考虑之前的情况，从而使问题缩小。

这种点存在的条件为第 i 个点在剩余点集中的凸包上，而选出的第 i+1 个点必然在剩余点集的凸包上，因此我们只需要保证第 1 个点在凸包上即可（为了方便可以直接选择最左下的点）。

那么问题的具体做法为：选取点集中的最左下的点作为第一个点，对于第 i+1 个点，视字符串中第 i 个位置的字符为'L' 还是'R'，选取剩余点集中最右 or 最左的点即可。

时间复杂度: $O(n^2)$