2024年江苏省C++编程爱好者线上交流活动

AtCoder题目选讲

实 战 模 拟 讲 评　&　题 目 选 讲

主　讲：李成浩

日　期：2024.7.8

目录

CONTENTS

实战模拟讲评

T1 Sloution

- ✓ 暴力?

  根据题意，不难得到X_i的取值范围是$[10^{A_i-1}, 10^{A_i}-1]$。

  - ➢ $10^{A_1-1}+10^{A_2-1} \le X_3 \le 10^{A_1}-1+10^{A_2}-1$
  - ➢ 若$10^{A_1-1}+10^{A_2-1} > 10^{A_3}-1$ 或 $10^{A_3-1} > 10^{A_1}-1+10^{A_2}-1$，无解
- ✓ 简单的数学题：设x为A_1位数，y为A_2位数，z为A_3位数。
  - ➢ $A_3 = \max(A_1, A_2)$ 或 $\max(A_1, A_2)+1$，不妨设$A_1 \ge A_2$，则$A_3 = A_1$ 或 A_1+1。
  - ➢ 得到A_1+1位数：$x+y \ge 10^{A_1} \rightarrow x \ge 10^{A_1}-y$
    - ➢ $A_1 \ne A_2$，从$10^{A_1}-\mathrm{y}$到$10^{A_1}-1$
    - ➢ $A_1 = A_2$，从10^{A_1-1}到$10^{A_1}-1$ 减去 所有不到 A_1 位数。
  - ➢ 得到A_1位数：取补集。

T2 Sloution

- ✓ BFS?
- ✓ 最短路

问题本质上是**求解两个独立路径问题**，并通过改变最少的单元格颜色来满足两条路径的条件。

- 路径需要从给定的起点到终点，只能通过指定颜色的单元格；
- 每次都检查是否可以**以更低的成本**到达一个新单元格：
  - 从$(1,1)$到(N,N)的最小成本
  - 从$(1,N)$到$(N,1)$的最小成本
- 两者之和就是最少消费。

T3 Sloution

- ✓ 问题分析：所有电线杆等高，在第 t 次地震时，若第 t 个电线杆如果没倒下，则会倒下，可能会导致连锁反应。
- ✓ 将所有**能相互影响**的电线杆分为若干个集合，不同集合之间互不影响。
  - ➢ 同一集合内，一个电线杆向某侧倒下，该侧的电线杆都会倒下。
  - ➢ 若果要在第 t 次所有电线杆都倒下，必然要 $t-1$ 次内其他集合内的所有电线杆都已经倒下。
- ✓ 按编号 从小到大 依次考虑，设 E_i 代表第 i 个集合内所有电线杆都倒下的概率：
  - ➢ 设 pos[t] 代表电线杆 t 所在的集合，共有 M 个集合；
  - ➢ 在 $t-1$ 次内其他集合内的所有电线杆都已经倒下的概率为：

$$\prod_{i=1,i\neq \texttt{pos[t]}}^{M} E_i$$

- ✓ 考虑如何将在第 t 次地震第 pos[t] 集合全部推倒的情况：
  - ➢ 电线杆 t 左右两侧 有一侧 的电线杆已经全部倒下，此时有 $\frac{1}{2}$ 的概率。
  - ➢ 电线杆 t 左右两侧的电线杆都已经倒下，此时有 1 的概率。

- ✓ 此前该集合中所有编号小于 x 的电线杆都不能倒向 x。
- ✓ 同时，在此基础上对于那些已经被压倒的电线杆一定不会倒向 x，那些会对概率产生影响的柱子构成了随着位置远离 x，编号单调减的序列。

- ✓ 序列大小是len，概率为$\frac{1}{2}^{len}$。

T4 Sloution

- ✓ 问题分析：将从长序列通过删除若干数字变成短序列。

  如何删除某个整数 T？

  - ➢ 选择的子区间$[l,r]$，该区间包含整数$0 \sim T-1$，但不包含T。
    即比 T 小的数要么都在 T 左侧，要么都在 T 右侧。
  - ➢ 删除数字的顺序必须是**从大到小**。
- ✓ 区间动态规划：**从小到大** 向序列 A 插入数字。
  - ➢ 向序列 A 的 $M+1$ 个空隙插入数字，每个空隙可能插入多个数字。
  - ➢ 一共 $M+1$ 个空隙，`pos[i]` 代表 `i` 所在的位置。

- ✓ 状态定义：
  dp[l][r][k]，代表 已经插入$0 \sim k$个整数，并要求后续所有要插入的整数都插入到 $[0, l]$ 或 $[r, M+1]$ 的方案数。
- ✓ 状态转移：从小到大枚举插入的数字 T
  - ➢ T 在 A 中，假设 T 出现在 pos[T] 和 pos[T] + 1 的空隙间，为了扩展区间，dp[min(pos[T], l)][max(pos[T] + 1,r)][T] = dp[l][r][T - 1]。
  - ➢ T 不在 A 中，则 T 需要插入到 A中：
    - 插入到比T小的数字左侧，dp[l][r][T] = $\sum_{i=l}^{r}$ dp[i][r][T - 1]；
    - 插入到比T小的数字右侧，dp[l][r][T] = $\sum_{i=l}^{r}$ dp[l][i][T - 1]；
- ✓ 边界条件：考虑最小元素0。
  - ➢ 0在A中，dp[pos[0]][pos[0] + 1][0] = 1。
  - ➢ 0不在A中，任取一个整数都能删除0，因此0可以放在任意位置，dp[i][i][0] = 1。
- ✓ 结果：dp[1][M + 1][N - 1]。

题目选讲

Good Permutation 2 solution

You are given a positive integer N and a sequence of M positive integers $A = (A_1, A_2, \ldots, A_M)$.

Here, all elements of A are distinct integers between 1 and N, inclusive.

A permutation $P = (P_1, P_2, \ldots, P_N)$ of $(1, 2, \ldots, N)$ is called a **good permutation** when it satisfies the following condition for all integers i such that $1 \leq i \leq M$:

- No contiguous subsequence of P is a permutation of $(1, 2, \ldots, A_i)$.

Determine whether a **good permutation** exists, and if it does, find the lexicographically smallest good permutation.

Good Permutation 2 solution

- ✓ 问题目标：给定一个 N 和 M 个正整数 $A=(A_1,A_2,\cdots,A_M)$，其中 A_i 都是 1 到 N 之间不同的整数，求一个排列，满足他的任意连续子序列都不是 $(1,2,\cdots,A_i)$ 的排列。求这个**字典序最小**的满足要求的排列。
  - ➢ 容易发现$A_i=1$或$A_i=N$不存在解决方案。
- ✓ 为了求出最小的字典序，选择 **贪心思想** 。
  - ➢ 为什么可以贪心?
- ✓ 贪心思想：为了字典序最小，尽可能将$1\sim N$顺序排列。
  - ➢ 序列从前往后生成，考察第 x 位，如果不存在$A_i=x$，第 x 位就放还未放置的最小值。
  - ➢ 如果存在$A_i=x$，判断当前的最大值，若最大值不大于x，则放置$x+1$。

Between B and B solution

You are given a sequence $(X_1, X_2, \ldots, X_M)$ of length M consisting of integers between 1 and M, inclusive.

Find the number, modulo 998244353, of sequences $A = (A_1, A_2, \ldots, A_N)$ of length N consisting of integers between 1 and M, inclusive, that satisfy the following condition:

- For each $B = 1, 2, \ldots, M$, the value X_B exists between any two different occurrences of B in A (including both ends).

More formally, for each $B = 1, 2, \ldots, M$, the following condition must hold:

- For every pair of integers (l, r) such that $1 \le l < r \le N$ and $A_l = A_r = B$, there exists an integer m ($l \le m \le r$) such that $A_m = X_B$.

Between B and B solution

- ✓ 问题目标：给定一个 长为M的序列 X，求满足条件且长度为 N的序列 A 的个数。
  - ➢ 条件一：$A_i \in [1, M]$。
  - ➢ 条件二：对于任意 l 和 r，使得 $1 \le l < r \le N$ 且 $A_l = A_r = B$，存在$m \in [l, r]$ 使得$A_m = X_B$。
- ✓ 条件二翻译成人话，即A中相同的两个元素B之间有一个元素的值为X_B。
- ✓ 状态压缩动态规划：M只有10，可以考虑状态压。
  - ➢ 状态：`dp[i][j]`，表示处理到第 i 位时，j表示可以放哪些数，用二进制表示。
  - ➢ 辅助数组：`sup[i]`，用于表示哪些位置的j满足$X_j = i$。
  - ➢ 状态转移：`dp[i + 1][(j ^ (1 << k - 1)) | sup[k]] += dp[i][j]`。
  - ➢ 边界条件：初始状态下，什么数字都可以填入：`dp[0][(1 << m) - 1] = 1`。

Sum of Abs 2 solution

You are given positive integers N and L, and a sequence of positive integers $A = (A_1, A_2, \ldots, A_N)$ of length N.

For each $i = 1, 2, \ldots, N$, answer the following question:

> Determine if there exists a sequence of L non-negative integers $B = (B_1, B_2, \ldots, B_L)$ such that $\sum_{j=1}^{L-1} \sum_{k=j+1}^{L} |B_j - B_k| = A_i$. If it exists, find the minimum value of $\max(B)$ for such a sequence B.

Sum of Abs 2 solution

- ✓ 问题目标：给定一个L，对于每个A_i，构造一个长为L的非负整数序列B，使得$\sum_{j=1}^{L-1}\sum_{k=j+1}^{L}\left|B_j - B_k\right| = A_i$，如果存在此序列B，找出最小的max(B)。
  - ➢ $\sum_{j=1}^{L-1}\sum_{k=j+1}^{L}\left|B_j - B_k\right|$是 B 中元素两两间差的绝对值的和。
  - ➢ 容易发现B中元素的顺序不影响最终结果。
  - ➢ 为方便计算，假设序列B**单调不减**。
- ✓ B序列单调不减，有$\sum_{j=1}^{L-1}\sum_{k=j+1}^{L}(B_k - B_j) \to \sum_{j=1}^{L-1} j(L-j)(B_{j+1} - B_j)$ 。
  - ➢ 此时令 $C_i = B_{i+1} - B_i$， $\sum_{i=1}^{L-1} i\times(L-i)\times C_i$ 。
  - ➢ 原题变为：有$L-1$种物品，每种物品大小$i(L-i)$，价值为1，数量无限，目标为A_x的最小价值。
  - ➢ 背包问题。

Portable Gate solution

You are given a tree with N vertices numbered $1, 2, \ldots, N$. The i-th edge connects vertices u_i and v_i bidirectionally.

Initially, all vertices are painted white.

To efficiently visit all vertices of this tree, Alice has invented a magical gate. She uses one piece and one gate to travel according to the following procedure.

First, she chooses a vertex and places both the piece and the gate on that vertex. Then, she repeatedly performs the following operations until all vertices are painted black.

- Choose one of the following actions:
  1. Paint the vertex where the piece is placed black.
  2. Choose a vertex adjacent to the vertex where the piece is placed and move the piece to that vertex. The cost of this action is 1.
  3. Move the piece to the vertex where the gate is placed.
  4. Move the gate to the vertex where the piece is placed.

Note that only the second action incurs a cost.

It can be proved that it is possible to paint all vertices black in a finite number of operations. Find the minimum total cost required.

Portable Gate solution

- ✓ 问题目标：一棵N个顶点的树，你有一个传送门，从一个选定的点开始，计算遍历所有顶点至少一次的最小消费，你每次可以选择：
  - ➢ 从当前点经过树边走到相邻点，花费1，只有这一步有开销；
  - ➢ 将传送门移动到当前点；
  - ➢ 将自己传送到传送门所在的点。
- ✓ 树形DP + 换根：由于存在传送门，棋子不一定要回到起点
  - ➢ 状态设计：dp[x][i][j]
    - ➢ x是当前子树的根；
    - ➢ i代表传送门状态，0表示子树内没有传送门，1代表传送点在x处；
    - ➢ j代表是否需要返回子树根节点x，0表示不需要，1表示需要返回。

- 树形DP：由于存在传送门，棋子不一定要回到起点
  - 状态设计：dp[x][i][j]
- 状态转移：
  - dp[x][0][1] = 2(siz[x] - 1)，子树中的每个点都需要进入和返回一次。
  - dp[x][0][0] = dp[x][0][1] – far[x]，其中far[x]是子树x内部距离x最远的距离，选择这个点作为最终点。
  - dp[x][1][1] = ∑min(dp[y][1][1] + 2, dp[y][0][0] + 1)：
    - dp[y][1][1] + 2代表自己和传送门都移动到y，由于y内部走完还需要使用传送门，需要自己回到x，再将传送门移回x。
    - dp[y][0][0] + 1代表自己走到y，由于不需要返回，可以直接传送回来。
  - dp[x][1][0] = dp[x][1][1] – max(min(dp[y][1][1] + 2, dp[y][0][0] + 1) – (dp[y][1][0] + 1))。
    - dp[y][1][0] + 1代表y为此行的终点，传送门可以跟着自己移动。
    - max(min(dp[y][1][1] + 2, dp[y][0][0] + 1) – (dp[y][1][0] + 1))目标为找出最优的终点y。

- 换根DP：意为对每个根求出树形 DP 的答案，但以每个点作为根节点进行一次树形 DP 复杂度太高，因此通常使用两次DFS：
  - 第一次 DFS 指定一个点进行树形 DP；
  - 第二次 DFS 进行换根 DP，得到将根转移到每个节点的答案。
- 步骤：第二次 DFS 搜索到 u 时，首先以 u 为根的进行dp；之后枚举 u 的子节点 v，之后要将根从 u 转移到 v：
  1. 将 v 的贡献从以 u 为根的dp中移除；
  2. 将 u 作为 v 的子节点，将贡献加入到以 v 为根的dp中；
  3. 以 v 为新根节点进行第二次 DFS；
  4. 退回到以 u 为根的进行dp的状态(递归处理)。

Rectangle Concatenation solution

For positive integers h and w, let (h, w) denote a rectangle with height h and width w. In this problem, we do not consider rotating the rectangles, and the rectangles (h, w) and (w, h) are distinguished when $h \neq w$.

A sequence of rectangles $((h_1, w_1), (h_2, w_2), \ldots, (h_n, w_n))$ is called a **rectangle generation sequence** if there exists a method that successfully follows the steps below:

- Let the rectangle X be (h_1, w_1). Hereafter, let H and W respectively denote the height and width of the rectangle X at each step.
- For $i = 2, 3, \ldots, n$, perform one of the following operations. If neither can be performed, the procedure unsuccessfully terminates.
  - If the height of X is equal to h_i, attach the rectangle (h_i, w_i) horizontally to X. Formally, if $H = h_i$ at that time, replace X with the rectangle $(H, W + w_i)$.
  - If the width of X is equal to w_i, attach the rectangle (h_i, w_i) vertically to X. Formally, if $W = w_i$ at that time, replace X with the rectangle $(H + h_i, W)$.
- If the above series of operations does not fail, the procedure successfully terminates.

You are given N rectangles. The i-th rectangle has a height of H_i and a width of W_i.

Find the number of pairs of positive integers (l, r) that satisfy $1 \leq l \leq r \leq N$ and the following condition:

- The sequence of rectangles $((H_l, W_l), (H_{l+1}, W_{l+1}), \ldots, (H_r, W_r))$ is a rectangle generation sequence.

Rectangle Concatenation solution

- ✓ 问题目标：给定N个矩形，要求找出若干个区间$[l,r]$，在此区间内矩形能够拼接，拼接的条件如下：
  - ➢ 若当前矩形和下一个矩形的高相同，则高不变，宽相加，组成大矩形；
  - ➢ 若当前矩形和下一个矩形的宽相同，则宽不变，高相加，组成大矩形
- ✓ 首先进行线性扫描：将右端点r从1往n遍历，并对所有$1 \le l \le r$的左端点l，维护区间内能够生成的所有矩形。
  - ➢ 用三元组(l,h,w)代表$[l,r]$区间内能够生成矩形(h,w) 。
  - ➢ 根据合成的要求， 一个合法的(l,h,w)必然存在$h_r = h$或$w_r = w$。

- 根据分析，将所有的三元组(l,h,w)构成的集合记为S，则当每次$r \to r+1$时，需要更新S内所有的三元组，丢弃不能拼接的部分，将$[r,r]$对应的三元组(r,h_r,w_r)添加到S中，最后将结果添加cnt，为S内不同的左端点l个数。
- 大致流程：

1. 对于每个新的矩形(H,W)新建一个集合S'，并添加三元组(r,H,W)；
2. 对于原来S中的每个三元组(l,h,w)，逐一分析：
   - 若$h=H,w=W$，则将$(l,2h,w)$和$(l,h,2w)$加入S'；
   - 若$h=H,w\neq W$，则将$(l,h,w+W)$加入S'；
   - 若$h\neq H,w=W$，则将$(l,h+H,w)$加入S'；
   - 若$h\neq H,w\neq W$，则忽视。
3. 用新的S'代替S。

- ✓ 不需要维护二维的三元组，转而维护若干个二元组构成的集合 S_h 和 S_w，用 $Last_h$ 和 $Last_w$ 代表上一次操作的 H 和 W，Now_h 和 Now_w 代表当前矩形的 H 和 W：
  - ➢ S_w 代表满足 $W = Now_w$ 的情况，S_h 代表满足 $H = Now_h$ 的情况。
- ✓ 分类讨论：
  1. 若 $Now_h = Last_h, Now_w = Last_w$，则有 $(Last_h, w) \rightarrow (Now_h, w + Now_w)$，$(h, Last_w) \rightarrow (h + Now_h, Now_w)$，即 $(Last_h, Last_w)$ 都有两种转移：
     - ➢ 将原有的 $(Last_h, Last_w)$ 删除，给集合 S_h 和 S_w **添加整体加的标签**；
     - ➢ 将 $(Last_h, 2Last_w)$ 和 $(2Last_h, Last_w)$ 分别添加到 S_h 和 S_w。
  2. 若 $Now_h = Last_h, Now_w \neq Last_w$，直接清空 S_w，因为 S_w 中除了此时的 $(Last_h, Last_w)$ 外都无法进行拼接，而 $(Last_h, Last_w)$ 也在 S_h 中（$Now_h \neq Last_h, Now_w = Last_w$ 同理）：
     - ➢ 给集合 S_h **添加整体加的标签**；
     - ➢ 将 $(2Last_h, New_w)$ 添加到 S_w 中。
  3. 若 $Now_h \neq Last_h, Now_w \neq Last_w$，清空 S_h 和 S_w，只留下 $(Last_h, Now_w)$ 和 $(Now_h, Last_w)$。
- ✓ 使用Set维护。

2024年江苏省C++编程爱好者线上交流活动

感　谢　观　看

主　讲：李成浩

日　期：2024.7.8