牛客竞赛
AC.NOWCODER.COM

NIO

2022 牛客
暑期多校训练营

Round 9 出题组

A. Car Show

题意

- 给定一个长为n的包含$1,2,\ldots,m$的序列，求有多少区间$[L,R]$包含所有$1,2,\ldots,m$。
- $m \leq n \leq 10^5$。

AC.NOWCODER.COM

A. Car Show

题解

- 枚举左端点L，合法的右端点集合一定是区间$[R,n]$，且R随着L的递增而不减。
- 在移动指针的同时维护区间内每种数字的个数以及出现数字的种类数，就可以在均摊$O(1)$的时间对每个L求出对应的R。
- 时间复杂度$O(n)$。

AC. NOWCODER. COM

B. Two Frogs

题意

- 河道里有n个荷叶排成一排，从第$i(<n)$个荷叶出发可以跳到第$(i, i+a_i]$个荷叶上，有两只青蛙从第1个荷叶出发，每一步都独立地等概率随机地跳向后边的荷叶，求两只青蛙以相同步数到达第n个荷叶的概率。

- $n \leq 8000$，保证$1 \leq a_i \leq n-i$。

AC. NOWCODER. COM

B. Two Frogs

题解

- 每步至少往后跳一个荷叶，$n-1$步之内一定能跳到第n个荷叶。
- $f_{s,i}$表示从第1个荷叶出发恰好s次跳到第i个荷叶的概率。
- 考虑向后转移，$f_{s,i}$对$f_{s+1,j}(i<j\le i+a_i)$有$f_{s,i}/a_i$的贡献，前缀和优化即可。
- 答案就是$\sum_{s=1}^{n-1} f_{s,n}^2$，时间复杂度$O(n^2)$。

AC. NOWCODER. COM

C. Global Positioning System

题意

- 有n个坐标未知的三维点，给出m对点的相对位置关系，要修改恰好一个关系使得存在一组三维点坐标满足所有关系，找出所有可以被修改的关系。
- $n \leq 5 \times 10^5$，$n-1 \leq m \leq 5 \times 10^5$。
- 保证给出的关系构成无向连通简单图，且一定有解。

AC.NOWCODER.COM

C. Global Positioning System

题解

- 不难发现，存在一组三维点坐标满足所有相对位置关系，当且仅当图上所有回路的矢量和为$\vec{0}$。
- 如果存在回路的矢量和不为$\vec{0}$，那么需要被修改的边就是所有这些回路的交集。由于保证有解，交集不会为空，也不需要考虑无法修改一条边使得所有回路的矢量和均为$\vec{0}$的情况。
- 如果所有回路的矢量和均为$\vec{0}$，也就是已经有解的情况下，有且只有图中的桥边可以被修改。

AC.NOWCODER.COM

C. Global Positioning System

题解（续）

- 要判断前述两种情况，只需要任取一棵生成树，考虑所有的非树边与树上路径构成的简单回路即可，这是因为图上所有回路都是这些简单回路的线性组合。
- 这样问题就转化为树上的路径覆盖问题，树上差分即可。这里如果取DFS树作为生成树，所有非树边都是返祖边，就不需要求LCA了。
- 时间复杂度$O(n+m)$。

AC.NOWCODER.COM

D. Half Turns

题意

- 在$n \times m$的网格图中构造一条恰好有$nm/2$次拐弯的哈密顿路径。
- $n, m \leq 1000$，且都是偶数。
- 题设条件下一定有解。

AC.NOWCODER.COM

D. Half Turns

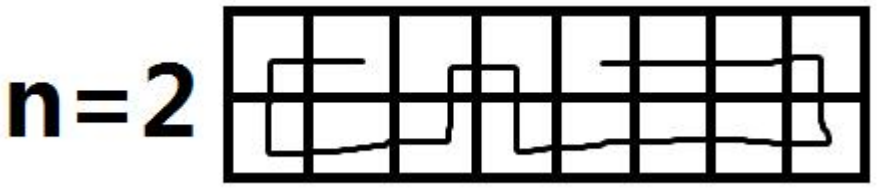
题解

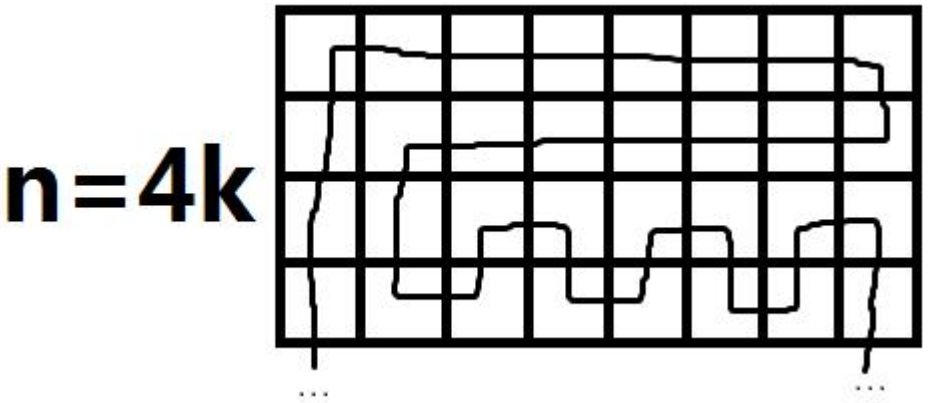
- 不妨设$n \le m$，然后分情况讨论。

- $n = 2$，前$n/2$列构造$m - 2$次拐弯，后$n/2$列构造2次拐弯。

- $n = 4k$，对于每4行，
- 前两行构造4次拐弯，后两行构造$2m - 4$次拐弯。

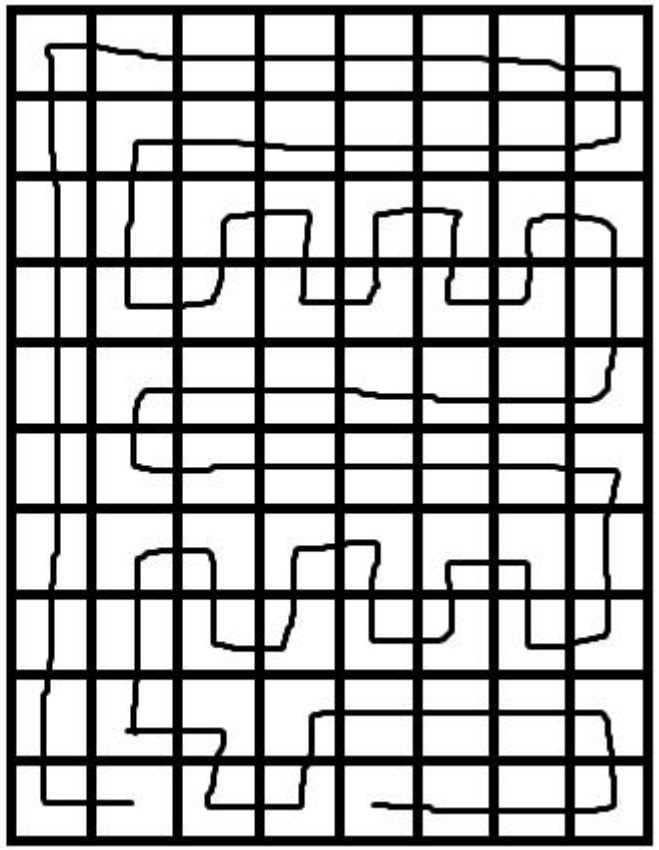
- $n = 8k + 2$和$n = 8k + 6$在前$n - 2$行的构造与$n = 4k$一致，
- 最后两行需要单独再构造一下。



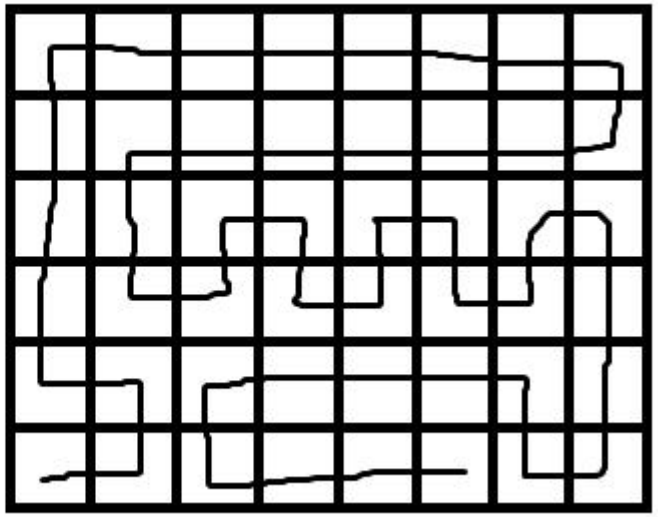


D. Half Turns

$n=8k+2$



$n=8k+6$



AC. NOWCODER. COM

E. Longest Increasing Subsequence

题意

- 构造一个$1,2,\ldots,n$的排列，使其恰好有m个不同的最长上升子序列。
- $m \leq 10^9$，要求$n \leq 100$。
- 题设条件下一定有解。

AC.NOWCODER.COM

E. Longest Increasing Subsequence

题解

- 记m的二进制表示为$b_k b_{k-1} \ldots b_0$，其中$b_k = 1$，$b_i \in \{0,1\}$。

- 先构造$2,1,4,3,6,5,\ldots,2k,2k-1$，然后低到高依次考虑$m$的二进制位。
- 如果$b_i = 1$，就在第$2i$个数之后插入一个大于$2k$的数字p_i，适当取值使得这些p_i是递增的。

- 这样原序列的LIS长度会是$k+1$，包含$p_i (i < k)$的上升序列长度可能不够。
- 只需要紧接着每个$p_i (i < k)$再插入一些递增的数字，并重新调整每个p_i的取值即可。
- 容易发现$p_i (i < k)$后边需要插入的数字个数就是m的二进制表示下第i位往上连续0的个数。

- 这样构造出来的排列长度不超过$3\lfloor \log_2 m \rfloor + 1$。

F. Matrix and GCD

题意

- 给一个$n \times m$的矩阵，满足$1,2,\ldots,nm$都恰好出现一次，对所有连续子矩阵求最大公约数之和。
- $n,m \leq 1000$。

AC.NOWCODER.COM

F. Matrix and GCD

题解

- 对于每个$d(1 \le d \le nm)$，矩阵中所有i的倍数看成1，其他数字看成0，计算全1子矩阵的个数g_d。
- 求解全1子矩阵是一个经典问题，可以做到只遍历所有1的位置进行求解，总共nm次求解的时间复杂度是$O(\sum_{d=1}^{nm} nm/d) = O(nm\log(nm))$。
- 现在g_d是满足GCD是d的倍数的子矩阵个数，记f_d为GCD恰好是d的子矩阵个数，那么有$g_d = \sum_{d|e} f_e$，通过容斥或者莫比乌斯反演可以在时间复杂度$O(nm\log(nm))$下计算出所有f_d。

AC.NOWCODER.COM

G. Magic Spells

题意

- 给定k个字符串$S_1, S_2, \ldots, S_k$，求有多少个本质不同的公共回文子串。
- $k \leq 5$，$\sum_{i=1}^{k} |S_i| \leq 3 \times 10^5$。

AC.NOWCODER.COM

G. Magic Spells

题解

- 先对每个字符串S分别求出所有本质不同回文子串，总共有$O(|S|)$个。
- 可以使用Manacher算法配合Hash求解，也可以使用回文树求解。
- 再Hash统计每个字符串的出现次数即可，时间复杂度$O(|S|\log|S|)$或$O(|S|(\log|S|+\Sigma))$。
- 也可以直接使用回文树求解，时间复杂度$O(|S|\Sigma)$。

AC.NOWCODER.COM

H. Rader Scanner

题意

- 给定二维平面上n个点(x,y)，q次询问(a,b)求所有满足$ax+by>0$的给定点的凸包面积（的两倍）。
- $n\le 10^5$，$q\le 2\times 10^5$，坐标都是$[-10^9,10^9]$的整数。
- 整点凸包面积的两倍一定是整数。

AC.NOWCODER.COM

H. Rader Scanner

题解

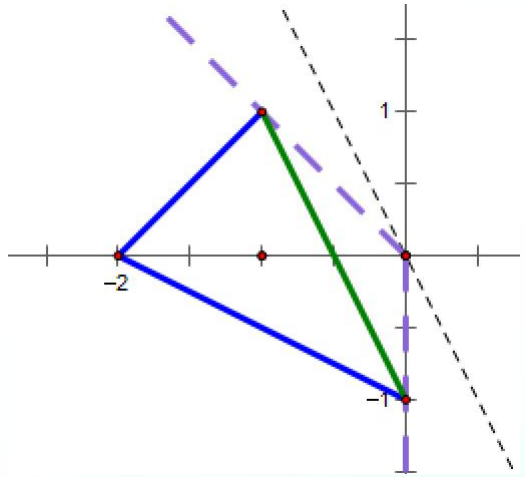
- 先去掉(0,0)以及重复点，对剩下的给定点做极角排序，极角相同则按照到原点的距离从近到远排序，得到一个新的点列$P_0, P_1, \ldots, P_{m-1}$，其中$P_i$在极角序上的下一个点就是$P_{(i+1) \bmod m}$。
- 对于每个询问(a, b)，极角在$(b, -a)$逆时针扫到$(-b, a)$的范围（不含边界）内的点就是满足$ax + by > 0$的点，在极角序上二分可以求出这个范围，得到环上区间$[l, r)$。
- 二分的时候有亿些细节需要注意，比如二分时需要忽略到原点的距离，以及$r \leq l$的区间的处理。
- 容易发现r是随着l的递增单调不减的，询问区间也可以用双指针处理出来，~~细节可能也有亿点~~。

AC. NOWCODER. COM

H. Rader Scanner

题解（续）

- 容易发现r是随着l的递增单调不减的，尝试用双指针维护滑动窗口内的点集凸包。
- 考虑分上下凸壳维护，这里的视点是原点$(0,0)$而不是水平序维护凸包时的$(*,-\infty)$。
- 上下凸壳的分界线是从视点看凸包最靠左或者最靠右的点。
- 离视点较远的一部分是上凸壳，较近的一部分是下凸壳。
- 如右图所示，蓝线构成上凸壳，绿线构成下凸壳。



AC. NOWCODER. COM

H. Rader Scanner

题解（续）

- 接下来讨论上凸壳的维护，下凸壳是类似的。
- 现在要维护滑动窗口$[l,r)$里$P_l,P_{l+1},\ldots,P_{r-1}$这些点的上凸壳，并支持push_back一个点$P_r$，或者pop_front一个点$P_l$。
- push_back操作比较容易，只需要在加入P_r之前pop_back不会在新凸壳上出现点的即可。
- pop_front操作则不太容易，似乎很难知道删掉P_l之后有哪些点会加入到新凸壳上。

H. Rader Scanner

题解（续）

- 使用两个栈stack_left和stack_right维护这个滑动窗口。
- 假设$P_m, P_{m+1}, \ldots, P_{r-1}$在stack_right里，维护这些点的上凸壳，push_back操作直接对stack_right生效。
- 假设$P_l, P_{l+1}, \ldots, P_{m-1}$在stack_left里，尝试对每个后缀求出上凸壳，具体操作是按照逆序push_front这些点，每加入一个点P_i时记录哪些点被踢出去了，将来P_i被pop_front的时候再把这些点按照顺序加回来，也就是撤销加入P_i对凸壳的影响。
- 在pop_front操作时stack_left有可能是空的，只需要先把stack_right里的所有点（不仅仅是上凸壳的点）都倒进stack_left，之后再进行pop_front操作即可。

H. Rader Scanner

题解（续）

- 现在分别维护了$P_l, P_{l+1}, \ldots, P_{m-1}$和$P_m, P_{m+1}, \ldots, P_{r-1}$的上凸壳，可以发现它们在极角序上是不交的。
- 要得到$P_l, P_{l+1}, \ldots, P_{r-1}$的上凸壳，还需要快速合并这两个上凸壳，也就是求解两个上凸壳的公切线。
- 当其中一个上凸壳为空时不需要做合并。当其中一个上凸壳只有一个点时，问题退化为点到上凸壳切线，可使用二分或三分求解。
- 接下来认为两个上凸壳都至少有两个点，直到问题退化。为了方便讨论，调整一下观看顺序使得两个凸壳的点从视点看是从左往右依次排列的。

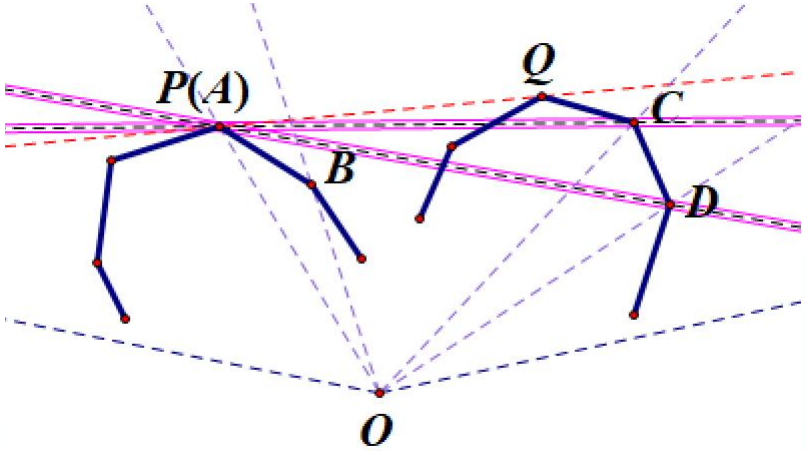
AC. NOWCODER. COM

H. Rader Scanner

题解（续）

- 在左边上凸壳选取一条边AB，在右边上凸壳选取一条边CD，从视点看A,B,C,D是从左往右排列的。

- 如果B不在直线AC或者直线AD上方，
- 那么B以及B右边的点都不可能是左边上凸壳的切点，
- 这是因为当B位于切点P左侧时一定在直线AC和直线AD上方。



- 如果C不在直线AD或者直线BD上方，
- 那么C以及C左边的点都不可能是右边上凸壳的切点，
- 这是因为当C位于切点Q右侧时一定在直线AD和直线BD上方。

AC. NOWCODER. COM

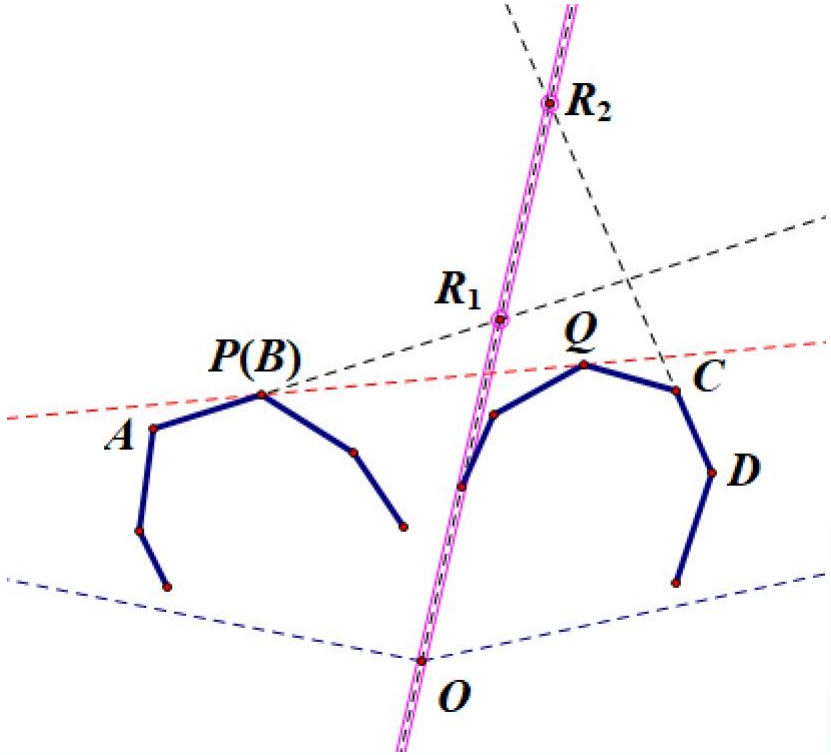
H. Rader Scanner

题解（续）

- 如非上述两种情况，AB和CD至少有一条边在合并后的凸壳上。
- 如右图所示，引一条方向向量$\vec{d}$是直线l将两个上凸壳分隔开，
- 使得左边上凸壳在直线l左侧，右边上凸壳在直线l右侧。

- 如果射线AB与直线l不相交，AB一定是合并后凸壳的边，
- 此时可以舍去左边上凸壳B左边的点，否则记交点为R_1。

- 如果射线DC与直线l不相交，CD一定是合并后凸壳的边，
- 此时可以舍去右边上凸壳C右边的点，否则记交点为R_2。



AC. NOWCODER. COM

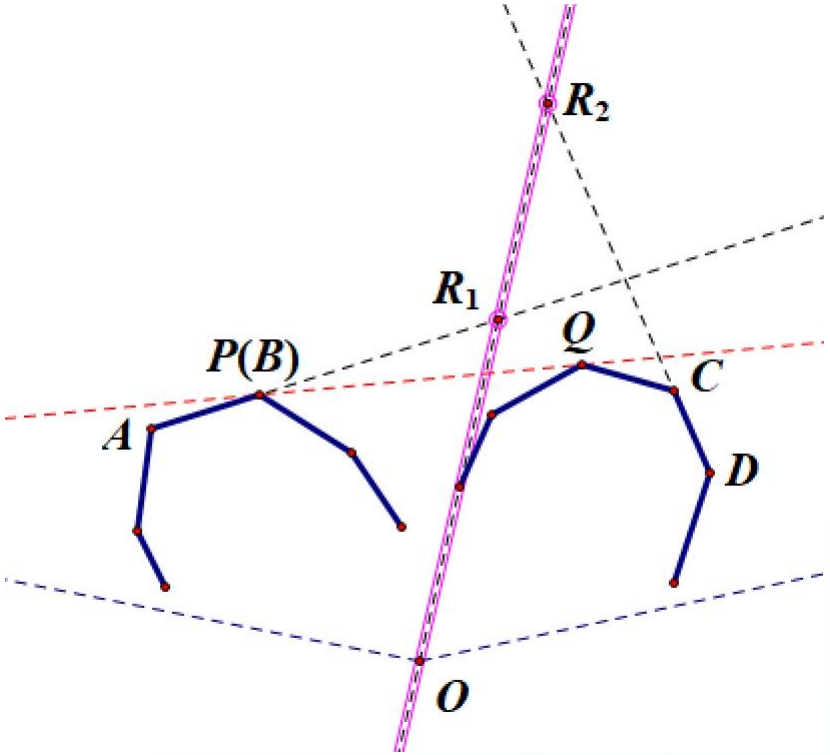
H. Rader Scanner

题解（续）

- 如果$\overrightarrow{R_1R_2} \cdot \vec{d} > 0$，$CD$一定是合并后凸壳的边，
- 此时可以舍去右边上凸壳C右边的点。

- 否则AB一定是合并后凸壳的边，
- 此时可以舍去左边上凸壳B左边的点。

- 这部分也可以做到全整数，但是需要int128支持。
- 如果实现得足够好，前边甚至可以不判是否有交。



AC. NOWCODER. COM

H. Rader Scanner

题解（续）

- 基于上述讨论，同时对两个上凸壳二分即可做到$O(\log n)$的复杂度求解两个上凸壳的公切线。
- 为了快速计算凸包面积，还需要快速截取上凸壳某一段对面积的贡献，在使用双栈维护上凸壳的同时维护上凸壳每条边的贡献的前缀和即可。
- 时间复杂度$O((n+q)\log n)$。

AC. NOWCODER. COM

I. The Great Wall II

题意

- 给定长为n的整数序列，将其分为非空的k段使得每一段的最大值之和最小，对$k = 1,2,\ldots,n$分别求解。
- $n \leq 8000$。

AC. NOWCODER. COM

I. The Great Wall II

题解

- $dp_{i,j}$表示将$a_1, a_2, \ldots, a_j$分为i段的最小代价。
- $dp_{i,j} = \min\limits_{1 \le k \le j} \{dp_{i-1,k-1} + \max\{a_k, a_{k+1}, \ldots, a_j\}\}$，直接做的复杂度是$O(n^3)$。

AC. NOWCODER. COM

I. The Great Wall II

题解（续）

- 对于固定的j，当k从j开始遍历到1时，$\max\{a_k, a_{k+1}, \ldots, a_j\}$是不减的，每一个取值对应一段$k$。
- 新增一个a_{j+1}时末尾一些段会被合并成一个max值是a_{j+1}的段，可以使用单调栈维护所有段。
- 每个数最多入栈和出栈一次，因此遍历一次整个数组时维护单调栈的复杂度是$O(n)$。

AC. NOWCODER. COM

I. The Great Wall II

题解（续）

- 假设当前的段是$[l_1, r_1], [l_2, r_2], \ldots, [l_s, r_s]$，对应的max值是$v_1, v_2, \ldots, v_s$。
- 转移式改写为$dp_{i,j} = \min\limits_{1 \le t \le s} \{\min\{dp_{i-1,l_t}, dp_{i-1,l_t+1}, \ldots, dp_{i-1,r_t}\} + v_t\}$。
- 只需要在$[l_t, r_t]$对应的栈帧上记录 $mi_t = \min\{dp_{i-1,l_t}, dp_{i-1,l_t+1}, \ldots, dp_{i-1,r_t}\}$，
- 再维护整个单调栈里$mi_t + v_t$的前缀最小值即可快速转移。
- 时间复杂度$O(n^2)$。

AC. NOWCODER. COM

J. Colourful Journey

题意

- 给一张n个点n条边的连通无向图，每条边上有一些颜色可以选用，q次询问从a出发走一条简单路到b，初始颜色任意，每经过一条边就选边上的一个颜色刷成当前颜色，最大化颜色发生改变的次数。

- $n, q \leq 2 \times 10^5$，总颜色数不超过5×10^5。

AC. NOWCODER. COM

J. Colourful Journey

题解

- 当一条边上的颜色数不少于3时，不论经过前后两条边时选择的颜色是什么，经过这条边时总可以选择和这两个颜色都不同的颜色。
- 这意味着颜色数不少于3的边都是“通配”的，可以给这些边赋予一个新的独一无二的颜色，于是每条边上的颜色数都不大于2。
- 对于一条路径，记录两端的边的所有颜色，以及两端的边取不同颜色时中间切换颜色的最大次数，合并两条路径的时候枚举四条边的颜色即可。

AC. NOWCODER. COM

J. Colourful Journey

题解（续）

- 对于树上的情况，可以使用树链剖分或者树上倍增支持快速询问，这里后者的空间复杂度相比前者要多个$O(\log n)$的因子，需要注意空间限制。

- 对于环套树的情况，有两种处理方法：
- 一种方法是找出一棵生成树之后留下一条非树边(u, v)，可行路径只有不经过非树边、$a \to u \to v \to b$以及$a \to v \to u \to b$这三种，对于后两种情况需要通过树上路径判交检验合法性。
- 另一种方法是找出环以及挂在环上的每棵树，按照a到b的路径过环和不过环两种情况处理，过环的时候枚举从环的哪一侧走，还需要预处理环上的信息。

- 时间复杂度$O((n+q)\log n)$。

AC. NOWCODER. COM

K. NIO's OAuth2 Server

题意

- 给定n个$\{1,2,\ldots,k\}$的非空子集，定义一个集合$S \subseteq \{1,2,\ldots,k\}$的度数为使用最少个数的给定集合求并能得到$S$的超集，计算有多少个$\{1,2,\ldots,k\}$的非空子集的度数分别是$1,2,\ldots,k$。

- $n \leq 10^5$，$k \leq 20$。

AC.NOWCODER.COM

K. NIO's OAuth2 Server

题解

- 记$U = \{1,2,\ldots,k\}$，设集合幂级数$f(x) = \sum_{S \subseteq U} c_S x^S$，其中$c_S = 1$当且仅当给定的$n$个集合中有$S$。
- 定义两个集合幂级数的乘积为集合并卷积，也就是$h = fg$满足$h_S = \sum_{i \cup j = S} f_i g_j$，这里$h_S$表示$h(x)$的$x^S$项的系数，其余同理。
- 两个集合幂级数的乘积可以使用快速莫比乌斯变换(FMT)完成，时间复杂度是$O(k2^k)$。

AC. NOWCODER. COM

K. NIO's OAuth2 Server

题解（续）

- 现在依次算出$f, f^2, \ldots, f^k$，如果$f_S^d > 0$，说明可以用d个给定集合求并得到S，找到最小的d记为g_S。
- 需要注意实际的方案数可能很大，由于只关心存在性，可以在每次计算乘积后将系数重新置为0和1，也可以直接在模意义下做运算。
- 为了得到每个集合S的度数，还需要求出$\min_{T \supseteq S} g_T$，再做一次高维后缀min即可。
- 时间复杂度$O(nk + k^2 2^k)$。

牛客竞赛
AC.NOWCODER.COM

NIO

THANKS !

AC.NOWCODER.COM