# 2021 CCPC 分站赛 (桂林) 题解

浙江大学命题组

2021 年 11 月 7 日

# A. A Hero Named Magnus

输出 $2x - 1$.

注意使用 32 位有符号整数会溢出.

# B. A Plus B Problem

记三个数分别为 $A, B, C$. 注意到 $C_i = (A_i + B_i + [A_j + B_j \geq 10]) \bmod 10$, 其中 $j$ 为最小的满足 $j > i$ 且 $A_j + B_j \neq 9$ 的位置 $j$.

如果没有发生修改, 那么发生变化的位数为 $0$, 否则至少为 $2$. 如果第 $i$ 位从不进位变成进位或者从进位变成不进位, 那么高位会发生变化直到最大的满足 $j < i$ 且 $A_j + B_j \neq 9$ 的位置 $j$.

因此需要使用支持插入, 删除, 求前驱和后继的数据结构维护所有 $A_i + B_i \neq 9$ 的位置 $i$. 可以使用 van Emde Boas Tree 或 std::set, 复杂度为 $O((q+n)\log\log n)$ 或 $O((q+n)\log n)$, 均可通过.

# C. AC Automaton

记 $Anc_{u,c}$ 和 $Dec_{u,c}$ 分别为结点 $u$ 的字符在 $c$ 中的祖先和后代 (均不包含 $u$) 的数量.

先考虑所有 ? 都填 $C$, 那么对于结点 $u$, 字符为 $C$ 的后代数量为 $F_u = Dec_{u,\{'C','?'\}}$.

再考虑哪些 ? 应该填 $A$.? 中不可能出现填 $C$ 为填 $A$ 祖先的情况, 因为交换后会使得答案更大. 因此对于每个填 $A$ 的 ?, 所有祖先的 ? 都会填 $A$.

# C. AC Automaton

对于结点 $u$, 考虑所有祖先的 ? 已经全填 $A$, 后代的 ? 仍然全填 $C$ 的情况下, 它从 $C$ 变成 $A$ 对答案的影响为 $G_u = Dec_{u,\{'C','?'\}} - Anc_{u,\{'A','?'\}}$.

注意到 $G$ 从祖先到后代是单调递减的, 也就是说如果 $G_u > 0$, 那么 $u$ 所有祖先的 $G$ 也大于 $0$, 那么当 $G_u > 0$ 时 $u$ 填 $A$, 答案最大.

综合考虑答案为 $\sum_{s_u='A'} F_u + \sum_{s_u='?'} \max\{G_u, 0\}$.

# C. AC Automaton

每次修改后，可能会使得 $u$ 祖先的 $F$ 加减 $1$，祖先或后代的 $G$ 加减 $1$.

对询问进行分块，块大小为 $S$，每次可以 $O(n)$ 重新统计所有 $F$ 和 $G$ 的值以及答案. 使用虚树压缩的方法，将修改涉及的至多 $S$ 个点以及它们的最近公共祖先和根作为关键点形成虚树，至多 $2S$ 个关键点.

考虑非关键点，对每对相邻关键点的链上的非关键点压缩成一块，对每个关键点不含关键点的子树压缩成一块，共计最多 $4S-1$ 块. 那么现在只需要考虑支持 $O(1)$ 块内 $F$ 和 $G$ 加减 $1$ 的操作同时计算对答案的影响，其他操作可以暴力解决在 $O(S)$ 内解决.

# C. AC Automaton

修改 $F$ 的操作比较简单，记录 $s_u =' A'$ 的结点 $u$ 数量 $cntA$. 如果块内 $F$ 加 (减)$1$，答案加 (减)$cntA$.

修改 $G$ 的操作比较复杂，记录修改前 $s_u =' ?'$ 且 $G_u = i$ 的数量 $c_i$，维护 $s_u =' ?'$ 且 $G_u > 0$ 的数量 $p$ 以及修改的总变化量 $d$. 当块内 $G$ 加 $1$ 时，依次进行 $p$ 加 $c_{i-d}$，$d$ 加 $1$，答案加 $p$ 的操作；当块内 $G$ 减 $1$ 时，依次进行答案减 $p$，$d$ 减 $1$，$p$ 减 $c_{i-d}$ 的操作.

注意到每块询问总变化量 $d$ 绝对值不超过 $S$，那么实际上只需要记录 $|i| \le S$ 的 $c_i$，也就是每块只需要 $O(S)$ 空间记录信息. 因此时间复杂度为 $O(\frac{qn}{S} + qS)$ 空间复杂度为 $O(S^2 + n)$，选择合适的 $S$ 可以使时间复杂度变为 $O(q\sqrt{n})$，空间复杂度变为 $O(n)$.

# D. Assumption is All You Need

记 $p$ 为最小的使得 $A_p \neq B_p$ 的位置 $p, q$ 为最小的满足 $q > p$ 且 $A_p \geq A_q \geq B_p$ 的位置 $q$, 交换 $A_p$ 和 $A_q$. 重复操作直到不可以操作为止. 有解当且仅当这种方法是可行解.

## 证明

记 $(i, j)$ 为交换 $A_i$ 和 $A_j$ 且 $i < j$, 答案操作序列为 $(x_1, y_1), (x_2, y_2), \cdots, (x_k, y_k)$. 那么需要证明可以调整序列使得 $(x_1, y_1) = (p, q)$. 假设有 $(x_1, y_1) \neq (p, q)$, 分 3 类情况共 6 种情况:

1. 不存在 $m$ 使得当且仅当 $i \leq m$ 时满足 $x_i = p$. 因此存在 $(p, y_i)$ 使得 $x_{i-1} > p$.

i. 如果 $x_{i-1} \neq y_i$ 且 $y_{i-1} \neq y_i$, 那么交换两个操作变为 $(p, y_i), (x_{i-1}, y_{i-1})$.

ii. 如果 $y_i = y_{i-1}$, 说明在操作 $i-1$ 前 $A_p > A_{x_{i-1}} > A_{y_i}$, 那么可以将两个操作变为 $(p, x_{i-1}), (x_{i-1}, y_{i-1})$.

## 证明 (续)

iii. 如果 $y_i = x_{i-1}$ 且在操作 $i-1$ 前 $A_p > A_{y_i}$ 那么可以将两个操作变为 $(p, y_i), (p, y_{i-1})$.
iv. 如果 $y_i = x_{i-1}$ 且在操作 $i-1$ 前 $A_p < A_{y_i}$ 那么可以将两个操作变为 $(p, y_{i-1}), (y_i, y_{i-1})$.

2. 存在 $m$, 使得当且仅当 $i \le m$ 时满足 $x_i = p$, 但不存在 $j$ 使得 $(x_j, y_j) = q$. 注意到 $q < y_i (i \le m)$ 且 $A_{y_1} \ge A_{y_2} \cdots \ge A_{y_m} = B_p$, 那么可以找到最小的满足 $A_q \ge A_{y_i}$ 的 $i$, 将操作 $(p, y_i)$ 变为 $(p, q), (q, y_i), (p, q)$.

3. 存在 $m$ 当且仅当 $i \le m$ 时满足 $x_i = p$ 且存在 $j$ 使得 $(x_j, y_j) = (p, q)$, 但 $y_1 \ne q$. 那么 $y_{j-1} > q$ 且 $A_{y_{j-1}} > A_q$, 可以将操作 $(p, y_{j-1}), (p, q)$ 变为 $(p, q), (q, y_{j-1})$.

# D. Assumption is All You Need

## 证明 (续)

因为每次交换都会减少逆序对数量，所以操作序列的数量是有限的. 因为每次调整都会使得操作序列字典序变小，因此在有限步之后无法调整，那么此时必然有 $(x_1, y_1) = (p, q)$.

因为有解时总可以选择 $(p, q)$ 作为第一步操作，且由于逆序对总是变小操作次数有限，因此有解时反复进行该操作能得到可行解.

可以发现这种方法得到的解字典序是最小的. 注意到按这种方法交换时，$p$ 的值是单调递增的，且对每个 $p$，$q$ 的值也是单调递增的，因此只需要按字典序枚举所有有序对 $(p, q)$，如果满足 $A_p \neq B_p$ 且 $A_p \geq A_q \geq B_p$ 则交换，复杂度为 $O(n^2)$.

# E. Buy and Delete

注意到鲍勃只需至多删除两次: 所有满足 $u > v$ 的边 $(u, v)$ 和所有满足 $u < v$ 的边 $(u, v)$.

因此问题转换为求有向图最小环: 如果任何一条边都大于 $p$ 答案为 $0$; 如果存在边小于等于 $p$ 且最小环大于 $p$ 答案为 $1$; 如果最小环小于等于 $p$ 答案为 $2$.

有向图最小环: 使用 Dijkstra 算法求出所有点对 $(u, v)$ 最短路 $d_{uv}$, 答案为 $\min\limits_{(u,v,p)\in E}\{d_{vu} + p\}$. 复杂度为 $O(n(m + n)\log m)$.

# F. Illuminations II

由于期望可加，可以转换为求内多边形每条边被照到的概率.

概率为这条边直线延长线和外多边形的两个交点间沿着外多边形边的距离与外多边形周长比值.

维护外多边形边长前缀和，使用双指针或者二分找到内多边形每条边直线延长线与外多边形的哪条边相交以及交点，就可以求出概率.

复杂度为 $O(n+m)$ 或 $O(n+m\log n)$ 均可通过. 注意使用 $64$ 位浮点数可能会出现精度问题，需要使用 $64$ 位有符号整数或者更高精度的浮点数.

# G. Occupy the Cities

从第二步开始，所有连续的 $1$ 长度至少为 $2$，可以同时向左右扩展. 因此只需要考虑第一步每个 $1$ 选择向左还是向右拓展.

可以直接使用动态规划求解，对每个的 $1$ 的位置记录第一步向左/向右扩展时，前缀需要时间的最小值.

也可以考虑对每段连续的 $0$，对答案贡献的值域不超过 $2$，取这些集合的最大值，那么只需要贪心检查至多两个值即可.

复杂度为 $O(n)$.

# H. Popcount Words

对询问串 $p_i$ 建立 AC 自动机，如果可以得到 $S$ 经过每个结点的次数，那么每个串 $p_i$ 的答案为对应结点 fail 子树的次数和.

记 $W_{n,0} = w(0, 2^n - 1), W_{n,1} = w(2^n, 2^{n+1} - 1)$. 可以发现对于 $n > 1, W_{n,i} = W_{n-1,i} + W_{n-1,1-i}$ 并且 $w(k2^n, (k+1)2^n - 1) = W(n, \text{popcount}(k))$，那么对于每个 $w(l, r)$，可以由 $O(\log r)$ 个 $W_{i,j}$ 拼接而成.

分解 $w(l, r)$ 的方法：先从小到大枚举 $n$，再从大到小枚举 $n$，只要满足存在整数 $k$ 使得 $l = k2^n$ 且 $l + 2^n - 1 \le r$，取出 $W(n, \text{popcount}(l))$ 并更新 $l = l + 2^n$.

# H. Popcount Words

记 $f_{u,n,i}$ 为结点 $u$ 经过 $W_{n,i}$ 后转移到的结点，那么对于 $n > 1$，$f_{u,n,i} = f_{f_{u,n-1,i},n-1,1-i}$，可以通过动态规划求出.

在使用 $f$ 转移 $S$ 的过程中记录 $c_{u,n,i}$ 为使用 $W_{n,i}$ 在 $u$ 转移的次数. 那么可通过方程

$$g_{u,n,i} = c_{u,n,i} + g_{u,n+1,i} + \sum_{f_{v,n+1,1-i}=u} g_{v,n+1,1-i}.$$

求出 $g_{u,1,0} + g_{u,1,1}$，也就是 $S$ 实际经过每个结点 $u$ 的次数. 最后在 fail 树上统计子树和就可以得到答案.

计算 $f$ 和 $g$ 的复杂度为 $O(\sum |p| \log r_{\max})$，转移 $S$ 的复杂度为 $O(n \log r_{\max})$. 总复杂度为 $O((n + \sum |p|) \log r_{\max})$.

# I. PTSD

等价于两两匹配，如果较小的为 PTSD 则对答案有贡献.

从大到小枚举每个数，维护一个初值为 $0$ 的计数器，如果它是 PTSD 且计数器大于 $0$，说明有更大的数还未匹配，那么统计进答案且计数器 $-1$. 否则计数器 $+1$. 可以发现这样答案是最优的. 复杂度 $O(n)$.

# J. Suffix Automaton

求出每个长度本质不同子串的数量，然后可以将询问转化为求给定长度的本质不同的字典序第 $k$ 大的子串，那么只需要对每个长度离线处理所有对应询问.

后缀树的方法：对 $S$ 末尾添加特殊字符后建出后缀树，那么长度为 $l$ 的本质不同子串就是深度为 $l$ 的所有结点数量 (包括边上被压缩的点但不包括特殊字符)，那么每条边相当于对长度区间进行数量加 $1$ 操作，可以通过差分数组统计每个长度本质不同字串数量.

同时可以维护每条边在枚举长度时出现的时刻和结束的时刻. 如果边按字符大小进行深度优先搜索，那么所有边的深度优先搜索序就是对应字符串字典序，枚举长度时使用数据结构按深度优先搜索序存储边就可以找到第 $k$ 大的字符串. 后缀树的每条边存储的是对应字符串的最小下标，因此可以直接输出答案. 复杂度为 $O((n+q)\log n)$.

# J. Suffix Automaton

后缀数组的方法: 求出 height 数组 (第一位添加 0 使得长度为 $n$), 长度为 $l$ 的本质不同字串数量等于 height 数组中小于 $l$ 的位置个数减去长度小于 $l$ 的后缀.

维护在所有 height 数组中小于 $l$ 且对应后缀长度大于等于 $l$ 的位置集合 $S$, 因为后缀数组是按字典序排序的, 那么可以找到集合中第 $k$ 大和第 $k+1$ 大的 $S_k, S_{k+1}$, 答案就是后缀数组区间 $[S_k, S_{k+1})$ 中长度大于等于 $l$ 对应的字符串. 由于需要用最小下标表示答案, 还需要求区间中的后缀起点的最小值.

同样使用数据结构维护位置集合以及查询第 $k$ 大, 同时还需要数据结构维护区间最小值, 复杂度也是 $O((n+q)\log n)$.

# K. Tax

记 $d_u$ 为 $1$ 到 $u$ 的最短路，那么当且仅当 $d_u + 1 = d_v$ 最短路会经过 $(u, v)$.

记 $c_i$ 为满足 $d_u = i$ 的点数，那么合法路径至多 $\sum_i \prod_{0 \le j \le i} c_j$ 条，可以发现图为每层 $3$ 个点的分层图时路径最多.

直接搜索所有路径复杂度为 $O(3^{\frac{n}{3}})$，可以通过.

# L. Wiring Engineering

考虑动态规划. 记 $f_{i,j}$ 为只连接区间 $[a,i)$ 和 $[b,j)$ 的最优解.
$g_{i,j}$ 为只连接区间 $[a,i]$ 和 $[b,j)$, 且已经安装 $i$ 号建筑的最优解,
$h_{i,j}$ 为只连接区间 $[a,i)$ 和 $[b,j]$, 且已经安装 $j$ 号塔的最优解.

那么

$$f_{i,j} = \max\{f_{i-1,j}, f_{i,j-1}, g_{i-1,j}, h_{i,j-1}\},$$

$$g_{i,j} = \max\{f_{i,j} - u_i, g_{i,j-1}, g_{i,j-1} + w_{i,j-1} - v_{j-1}, h_{i,j-1} + w_{i,j-1} - u_i\},$$

$$h_{i,j} = \max\{f_{i,j} - v_j, h_{i-1,j}, h_{i-1,j} + w_{i-1,j} - u_{i-1}, g_{i-1,j} + w_{i-1,j} - v_j\},$$

可以 $O(n^2)$ 求出所有 $f, g, h$ 的值, 答案为 $\max\{f_{b+1,d+1}, 0\}$.

# L. Wiring Engineering

注意到转移方程实际上和 $a, c$ 无关. 可以将问题转化为有向无环图最长路问题, 如根据第 $2$ 个方程的第 $4$ 项, 可以从 $h_{i,j-1}$ 向 $g_{i,j}$ 连接长度为 $w_{i,j-1} - u_i$ 的边. 共计 $3(n+1)^2$ 个点和 $12n(n+1)$ 条边, 答案为 $f_{a,c}$ 到 $f_{b+1,d+1}$ 的最长路.

考虑分治. 对某个 $m$, 先处理所有 $a \le m$ 且 $b \ge m$ 的询问. 如果询问答案不为零, 那么必定存在 $j$ 和最长路使得最长路经过 $h_{m,j}$, 因为必定存在某个塔 $j$, 在它之前的塔只会连接 $a < m$ 的建筑, 在它之后的塔只会连接 $b > m$ 的建筑. 记 $d_{u,v}$ 为 $u$ 到 $v$ 的最长路, 那么 $d_{f_{a,c},f_{b+1,d+1}} = \max\{\max_j\{d_{f_{a,c},h_{m,j}} + d_{h_{m,j},f_{b+1,d+1}}\}, 0\}$.

对每个 $j$ 求出 $h_{m,j}$ 到所有点已经所有点到它的最长路需要 $O(n^2)$, 对每个询问枚举 $j$ 需要 $O(n)$, 总复杂度为 $O(n^3 + qn)$.

# L. Wiring Engineering

然后对 $b < m$ 和 $a > m$ 的询问分成两部分分别递归下去. 在下一轮分治时对某个 $m$, 先处理 $c \leq m$ 和 $d \geq m$ 的询问, 这时候类似地需要枚举 $g_{i,m}$ 求最长路. 接着轮流按建筑和塔分治交替进行下去直到区间为空或没有询问为止.

每分两步, 需要 $O(n^3)$ 枚举每个中转点到其他点以及其他点到它的最长路, 然后会将问题分成 $4$ 个规模减半的子问题, 也就是 $T(n) = 4\,T(\frac{n}{2}) + O(n^3)$, 根据主定理 $T(n) = O(n^3)$. 而对于每个询问, 只会在某层分治中枚举中转点, 层数为 $O(\log n)$, 枚举中转点数量为 $O(n)$, 因此处理询问的复杂度为 $O(qn)$. 所以总复杂度为 $O(n^3 + qn)$.

注意到点数和边数会极大地影响复杂度的常数, 更多的点和边可能会导致程序运行时间超限.