The 2018 ACM-ICPC China Multi-Provincial Collegiate Programming Contest Analysis

The Seven Musketeers

2018 年 6 月 4 日

Problem A. Maximum Element In A Stack

- 考虑栈的结构为元素先进后出。

Problem A. Maximum Element In A Stack

- 考虑栈的结构为元素先进后出。
- 执行 push 操作时，将要 push 的元素与栈中所有元素的值取最大值。

Problem A. Maximum Element In A Stack

- 考虑栈的结构为元素先进后出。
- 执行 push 操作时，将要 push 的元素与栈中所有元素的值取最大值。
- 执行 pop 操作时，由于栈的结构，前面的元素不会受到影响，所以直接 pop 栈顶元素即可。

Problem A. Maximum Element In A Stack

- 考虑栈的结构为元素先进后出。
- 执行 push 操作时，将要 push 的元素与栈中所有元素的值取最大值。
- 执行 pop 操作时，由于栈的结构，前面的元素不会受到影响，所以直接 pop 栈顶元素即可。
- 栈内元素最大值即为栈顶元素。

Problem B. Rolling The Polygon

- 点 Q 的轨迹由 n 段圆弧构成。

Problem B. Rolling The Polygon

- 点 Q 的轨迹由 n 段圆弧构成。
- 从 $P_{i-1}P_i$ 与地面接触滚动到 P_iP_{i+1} 与地面接触期间，圆弧的半径是 $|QP_i|$，圆心角是 $\pi - \angle P_{i-1}P_iP_{i+1}$。

Problem C. Caesar Cipher

- 根据给定的明文和密文算出偏移量。

Problem C. Caesar Cipher

- 根据给定的明文和密文算出偏移量。
- 根据偏移量和给出的密文求出明文。

Problem D. Take Your Seat

- 先考虑第一问。

Problem D. Take Your Seat

- 先考虑第一问。
- 当 $n = 1$ 时概率为 $f_n = 1$，当 $n = 2$ 时概率为 $f_n = \frac{1}{2}$，以下考虑 $n \geq 3$。

Problem D. Take Your Seat

- 先考虑第一问。
- 当 $n = 1$ 时概率为 $f_n = 1$，当 $n = 2$ 时概率为 $f_n = \frac{1}{2}$，以下考虑 $n \geq 3$。
- 如果 1 坐到 1 上，那么后面所有人都会坐对，如果 1 坐到 n 上，那么 n 肯定不能坐对。

Problem D. Take Your Seat

- 先考虑第一问。
- 当 $n = 1$ 时概率为 $f_n = 1$，当 $n = 2$ 时概率为 $f_n = \frac{1}{2}$，以下考虑 $n \geq 3$。
- 如果 1 坐到 1 上，那么后面所有人都会坐对，如果 1 坐到 n 上，那么 n 肯定不能坐对。
- 如果 1 坐到 k 上，那么 $2, 3, \cdots, k-1$ 都会坐对，此时 k 相当于一开始的 1，而人数变为 $n-k$。

Problem D. Take Your Seat

- 先考虑第一问。
- 当 $n = 1$ 时概率为 $f_n = 1$，当 $n = 2$ 时概率为 $f_n = \frac{1}{2}$，以下考虑 $n \geq 3$。
- 如果 1 坐到 1 上，那么后面所有人都会坐对，如果 1 坐到 n 上，那么 n 肯定不能坐对。
- 如果 1 坐到 k 上，那么 $2, 3, \cdots, k-1$ 都会坐对，此时 k 相当于一开始的 1，而人数变为 $n - k$。
- 因此 $f_n = \frac{1 + f_2 + f_3 + \cdots + f_{n-1}}{n}$，可得 $f_n = \frac{1}{2}$。

Problem D. Take Your Seat

- 再考虑第二问。

Problem D. Take Your Seat

- 再考虑第二问。
- 等概率随机一个排列相当于等概率随机 1 的登机时刻。

Problem D. Take Your Seat

- 再考虑第二问。
- 等概率随机一个排列相当于等概率随机 1 的登机时刻。
- 如果 1 是第 i 个登机的，那么登机的前 $i-1$ 人都能坐对，相当于人数变为 $n-i+1$。

Problem D. Take Your Seat

- 再考虑第二问。
- 等概率随机一个排列相当于等概率随机 1 的登机时刻。
- 如果 1 是第 i 个登机的，那么登机的前 $i-1$ 人都能坐对，相当于人数变为 $n-i+1$。
- 因此概率为 $g_n = \frac{f_1+f_2+\cdots+f_n}{n} = \frac{n+1}{2n}$。

Problem E. 2-3-4 Tree

- 根据题意模拟。

Problem E. 2-3-4 Tree

- 根据题意模拟。
- 需要正确理解当前节点是 4-node 时的修改操作。

Problem F. Moving On

- 考虑 Floyd 算法在动态规划思想下的解释。

Problem F. Moving On

- 考虑 Floyd 算法在动态规划思想下的解释。
- $f_{k,i,j}$ 表示从 i 到 j 可以途径编号为 $1, 2, \cdots, k$ 的节点的最短路径。

Problem F. Moving On

- 考虑 Floyd 算法在动态规划思想下的解释。
- $f_{k,i,j}$ 表示从 i 到 j 可以途径编号为 $1, 2, \cdots, k$ 的节点的最短路径。
- 将所有点按照点权从小到大排序重新标号，此时 $f_{k,i,j}$ 表示从 i 到 j 可以途径点权前 k 小的节点的最短路径。

Problem F. Moving On

- 考虑 Floyd 算法在动态规划思想下的解释。
- $f_{k,i,j}$ 表示从 i 到 j 可以途径编号为 $1, 2, \cdots, k$ 的节点的最短路径。
- 将所有点按照点权从小到大排序重新标号，此时 $f_{k,i,j}$ 表示从 i 到 j 可以途径点权前 k 小的节点的最短路径。
- 对于每次询问，找出点权小于等于给定限制 w 的最大点编号回答即可。

Problem G. Factories

- 特判 $n = 2$，否则选取一个非叶节点作为根。

Problem G. Factories

- 特判 $n = 2$，否则选取一个非叶节点作为根。
- $dp_{u,i}$ 表示在以 u 为根的子树中取了 i 个叶子时 k 个工厂两两距离和的最小值。

Problem G. Factories

- 特判 $n = 2$，否则选取一个非叶节点作为根。
- $dp_{u,i}$ 表示在以 u 为根的子树中取了 i 个叶子时 k 个工厂两两距离和的最小值。
- 自底向上合并，每次将子节点 v 的信息与父节点 u 的信息进行类似背包的转移合并，如果 v 内选了 i 个叶子，那么 uv 这条边会属于 $i(k-i)$ 对选出的叶子的路径。

Problem H. Fight Against Monsters

- 一直攻击一只怪兽直到将其打倒才是有意义的。

Problem H. Fight Against Monsters

- 一直攻击一只怪兽直到将其打倒才是有意义的。
- 记 $TIME_i$ 表示打倒第 i 只怪兽需要的攻击次数，按照 $\frac{TIME_i}{ATK_i}$ 从小到大的顺序消灭怪兽是最优的。

Problem H. Fight Against Monsters

- 一直攻击一只怪兽直到将其打倒才是有意义的。
- 记 $TIME_i$ 表示打倒第 i 只怪兽需要的攻击次数，按照 $\frac{TIME_i}{ATK_i}$ 从小到大的顺序消灭怪兽是最优的。
- 证明只需要考虑交换相邻两只怪物不会使答案变优的充要条件，可以发现条件是传递的。

Problem I. Bubble Sort

- 冒泡 k 趟之后，新序列的第 i 项是原序列前 $\min(i+k, n)$ 项中未在新序列前 $i-1$ 项中出现过的最小值。

Problem I. Bubble Sort

- 冒泡 k 趟之后，新序列的第 i 项是原序列前 $\min(i+k,n)$ 项中未在新序列前 $i-1$ 项中出现过的最小值。
- 新序列的 $LIS \geq n-1$ 相当于在序列 $1,2,\cdots,n$ 选择至多一个区间循环左移或右移。

Problem I. Bubble Sort

- 冒泡 k 趟之后，新序列的第 i 项是原序列前 $\min(i+k, n)$ 项中未在新序列前 $i-1$ 项中出现过的最小值。
- 新序列的 $LIS \geq n-1$ 相当于在序列 $1, 2, \cdots, n$ 选择至多一个区间循环左移或右移。
- 于是可以枚举新序列，算出每个新序列元素在原序列中可能的位置个数，利用乘法原理计算方案数即可。

Problem J. Nested Triangles

- 直线 PQ 两侧的点分别考虑，下设所有点均在直线同侧。

Problem J. Nested Triangles

- 直线 PQ 两侧的点分别考虑，下设所有点均在直线同侧。
- 对于两个点 R 和 S，S 包含在 $\triangle PQR$ 内当且仅当 $\angle PQS < \angle PQR$ 且 $\angle QPS < \angle QPR$，也就是满足二维偏序。

Problem J. Nested Triangles

- 直线 PQ 两侧的点分别考虑，下设所有点均在直线同侧。
- 对于两个点 R 和 S，S 包含在 $\triangle PQR$ 内当且仅当 $\angle PQS < \angle PQR$ 且 $\angle QPS < \angle QPR$，也就是满足二维偏序。
- 对所有点分别关于 P 和 Q 极角排序，可以得到与每个点相关的两个角度的大小关系，按照其中一维从大到小排序（这一维相同则按照另一维从小到大排序）之后相当于对另一维求解最长下降子序列。

Problem J. Nested Triangles

- 直线 PQ 两侧的点分别考虑，下设所有点均在直线同侧。
- 对于两个点 R 和 S，S 包含在 $\triangle PQR$ 内当且仅当 $\angle PQS < \angle PQR$ 且 $\angle QPS < \angle QPR$，也就是满足二维偏序。
- 对所有点分别关于 P 和 Q 极角排序，可以得到与每个点相关的两个角度的大小关系，按照其中一维从大到小排序（这一维相同则按照另一维从小到大排序）之后相当于对另一维求解最长下降子序列。
- 要输出字典序最小的解，只需要倒着做，然后在保持解最优的前提下每次选择最小的后继。

Problem K. Vertex Covers

- 考虑折半，将点集分为大小接近的两部分 L 和 R，那么边集分为 L 内部的、R 内部的以及 L 和 R 之间的。

Problem K. Vertex Covers

- 考虑折半，将点集分为大小接近的两部分 L 和 R，那么边集分为 L 内部的、R 内部的以及 L 和 R 之间的。
- 枚举 L 的子集 S，检查是否 L 内部所有边都被覆盖。

Problem K. Vertex Covers

- 考虑折半，将点集分为大小接近的两部分 L 和 R，那么边集分为 L 内部的、R 内部的以及 L 和 R 之间的。
- 枚举 L 的子集 S，检查是否 L 内部所有边都被覆盖。
- 再枚举 R 的子集 T，检查是否 R 内部所有边都被覆盖，如果是，那么根据 L 和 R 之间的未覆盖边可以知道 L 的一个子集 T' 必须要包含于 vertex cover，那么可以在 L 内选出所有包含 T' 的可行 S，这样 $S+T$ 就是一个 vertex cover。

Problem K. Vertex Covers

- 考虑折半，将点集分为大小接近的两部分 L 和 R，那么边集分为 L 内部的、R 内部的以及 L 和 R 之间的。
- 枚举 L 的子集 S，检查是否 L 内部所有边都被覆盖。
- 再枚举 R 的子集 T，检查是否 R 内部所有边都被覆盖，如果是，那么根据 L 和 R 之间的未覆盖边可以知道 L 的一个子集 T' 必须要包含于 vertex cover，那么可以在 L 内选出所有包含 T' 的可行 S，这样 $S+T$ 就是一个 vertex cover。
- 由于乘法满足分配率，只需要对 S 做一个高维前缀和就能快速计算答案。

Problem L. Continuous Intervals

- 对于一个区间 $[L, R]$，记最大值为 max、最小值为 min、数字种类数为 cnt，那么这个区间是 continuous interval 当且仅当 $max - min + 1 = cnt$。

Problem L. Continuous Intervals

- 对于一个区间 $[L, R]$，记最大值为 max、最小值为 min、数字种类数为 cnt，那么这个区间是 continuous interval 当且仅当 $max - min + 1 = cnt$。
- 考虑从小到大枚举 R，用线段树维护每个 L 的区间 $[L, R]$ 的 $max - min - cnt$ 的值。

Problem L. Continuous Intervals

- 对于一个区间 $[L, R]$，记最大值为 max、最小值为 min、数字种类数为 cnt，那么这个区间是 continuous interval 当且仅当 $max - min + 1 = cnt$。
- 考虑从小到大枚举 R，用线段树维护每个 L 的区间 $[L, R]$ 的 $max - min - cnt$ 的值。
- 由于总有 $max - min + 1 \geq cnt$，那么只需要维护线段树上每个 L 对应的 $max - min - cnt$ 的最小值，以及有多少个 L 取到这个最小值。

Problem L. Continuous Intervals

- 对于一个区间 $[L, R]$，记最大值为 max、最小值为 min、数字种类数为 cnt，那么这个区间是 continuous interval 当且仅当 $max - min + 1 = cnt$。
- 考虑从小到大枚举 R，用线段树维护每个 L 的区间 $[L, R]$ 的 $max - min - cnt$ 的值。
- 由于总有 $max - min + 1 \geq cnt$，那么只需要维护线段树上每个 L 对应的 $max - min - cnt$ 的最小值，以及有多少个 L 取到这个最小值。
- 当 R 变大时，每个 L 对应的三个值都需要进行修改。对于 max 和 min，可以用单调栈来维护后缀 max 和 min，然后在线段树上进行区间加减操作，对于 cnt，只需要在线段树上对区间 $[last_{a_i} + 1, R]$ 进行加减操作。

Problem M. Acyclic Orientation

- 根据给出的信息，如果计算出 $\chi_G(0), \chi_G(1), \cdots, \chi_G(n+m)$，就能通过 Lagrange interpolation formula 计算出 $\chi_G(-1)$。

Problem M. Acyclic Orientation

- 根据给出的信息，如果计算出 $\chi_G(0), \chi_G(1), \cdots, \chi_G(n+m)$，就能通过 Lagrange interpolation formula 计算出 $\chi_G(-1)$。
- 也就是求出用 $0, 1, 2, \cdots, n+m$ 颜色对 $K_{n,m}$ 染色的方案数，颜色可以不用完。

Problem M. Acyclic Orientation

- 根据给出的信息，如果计算出 $\chi_G(0), \chi_G(1), \cdots, \chi_G(n+m)$，就能通过 Lagrange interpolation formula 计算出 $\chi_G(-1)$。
- 也就是求出用 $0, 1, 2, \cdots, n+m$ 颜色对 $K_{n,m}$ 染色的方案数，颜色可以不用完。
- 对于特定的 c，考虑枚举左侧 n 个点恰好用了 k 种颜色，此时右侧 m 个点每个点都有 $c-k$ 种颜色可以用，那么方案数是 $\binom{c}{k}S(n,k)(c-k)^m$，这里 $S(n,k)$ 是第二类斯特林数。

Problem M. Acyclic Orientation

- 根据给出的信息，如果计算出 $\chi_G(0), \chi_G(1), \cdots, \chi_G(n+m)$，就能通过 Lagrange interpolation formula 计算出 $\chi_G(-1)$。
- 也就是求出用 $0, 1, 2, \cdots, n+m$ 颜色对 $K_{n,m}$ 染色的方案数，颜色可以不用完。
- 对于特定的 c，考虑枚举左侧 n 个点恰好用了 k 种颜色，此时右侧 m 个点每个点都有 $c-k$ 种颜色可以用，那么方案数是 $\binom{c}{k} S(n,k)(c-k)^m$，这里 $S(n,k)$ 是第二类斯特林数。
- 这里需要对每个 c 枚举 k 进行求和，不难发现这可以写成两个多项式的乘法，用 FFT 优化即可。

Problem M. Acyclic Orientation

- 根据给出的信息，如果计算出 $\chi_G(0), \chi_G(1), \cdots, \chi_G(n+m)$，就能通过 Lagrange interpolation formula 计算出 $\chi_G(-1)$。
- 也就是求出用 $0, 1, 2, \cdots, n+m$ 颜色对 $K_{n,m}$ 染色的方案数，颜色可以不用完。
- 对于特定的 c，考虑枚举左侧 n 个点恰好用了 k 种颜色，此时右侧 m 个点每个点都有 $c-k$ 种颜色可以用，那么方案数是 $\binom{c}{k} S(n,k)(c-k)^m$，这里 $S(n,k)$ 是第二类斯特林数。
- 这里需要对每个 c 枚举 k 进行求和，不难发现这可以写成两个多项式的乘法，用 FFT 优化即可。
- 这里还需要快速计算一行的第二类斯特林数，利用容斥原理可以将 $S(n,k)$ 写成一个求和式，发现也可以写成两个多项式的乘法，同样用 FFT 优化。

Problem M. Acyclic Orientation

- 根据给出的信息，如果计算出 $\chi_G(0), \chi_G(1), \cdots, \chi_G(n+m)$，就能通过 Lagrange interpolation formula 计算出 $\chi_G(-1)$。
- 也就是求出用 $0, 1, 2, \cdots, n+m$ 颜色对 $K_{n,m}$ 染色的方案数，颜色可以不用完。
- 对于特定的 c，考虑枚举左侧 n 个点恰好用了 k 种颜色，此时右侧 m 个点每个点都有 $c-k$ 种颜色可以用，那么方案数是 $\binom{c}{k} S(n,k)(c-k)^m$，这里 $S(n,k)$ 是第二类斯特林数。
- 这里需要对每个 c 枚举 k 进行求和，不难发现这可以写成两个多项式的乘法，用 FFT 优化即可。
- 这里还需要快速计算一行的第二类斯特林数，利用容斥原理可以将 $S(n,k)$ 写成一个求和式，发现也可以写成两个多项式的乘法，同样用 FFT 优化。
- 最后，$|\chi_G(-1)| = (-1)^{n+m}\chi_G(-1)$。

Thank you!