Solution

YDSP 2024 云斗初赛模拟 S 组

tder

云斗出题组

2024 年 9 月 16 日

Part 1

- BACAD BDCAC BADDC

Part 2

- BAABAB ABBBDCC ABABAD

Part 3

- CAABD ABDBA

1st ∼ 2nd

- 【1】NOI 以及相关活动的历史.
- 其全称为 ZheJiang Olympiad in Informatics.

- 【1】代数（初中部分）.
- 通过消元法解得:

$$\begin{cases} a = 3.2 \\ b = 4.2 \\ c = -3 \end{cases}$$

3rd

- 【6】树的重心、直径.
- A 选项中，很多树都有不止一条直径，菊花就是一个很好的例子. B 选项中，树的直径必然穿过树的一个重心，但不一定穿过所有的重心. C 选项中，有性质，树的重心如果不唯一，则至多有两个，且这两个重心相邻. 并且，树中所有点到重心的距离和是最小的. 故重心两边的边数应是相同的，而两重心中间还有一条边，故边数应为奇数，即偶数条边的树不可能有两个重心. D 选项中，考虑菊花，其直径数为：

$$\binom{2^n-2}{2} = 2^{2n} - 5 \times 2^n + 6$$

当 $n \geq 3$ 时比 2^{2n-2} 更优.

4th ∼ 5th

- 【3】插入排序.
- 根据题意，该排序算法基于比较，且最优复杂度为 $\Theta(n)$. 故选项中仅有插入排序符合题意.

- 【6】线段树，【6】单源最短路：Dijkstra.
- 每次成功松弛后将原本插入堆的操作变为单点修改，而原本在堆中取最短路长度最小的点的操作则是查询全局最小值. 使用线段树维护，复杂度为 $\Theta(m \log n) = \Theta(n \log n)$.

6th ∼ 7th

- 【5】优先队列（priority_queue），【5】多重映射（multimap），【5】算法模板库中的常用函数.
- A 选项应为 prev_permutation，求按字典序排序的上一个排列. C 选项应为 nth_element，线性查找序列中的第 k 小值. D 选项应为 unordered_multimap，无序多重映射.

- 【6】圆排列.
- 用所有元素的圆排列数除以各个重复元素的全排列数之积，有：

$$ans = \frac{(8-1)!}{3! \cdot 2!} = 420$$

8th ∼ 9th

- 【6】时间复杂度分析.
- A 选项中，当出队时 f 为空，则时间复杂度为 $O(n)$. B 选项和 C 选项中，每次入队操作仅需要向 b 中压入一个数，时间复杂度为 $O(1)$. 而出队操作中，从 b 压入 f 的数最多有 $O(n)$ 个，均摊下来复杂度为 $O(n)$. 故程序的时间复杂度为 $O(n)$. D 选项中，每个元素可能会被来回压入 b 或压入 f 多次，故无法保证复杂度.

- 【5】稀疏图，【6】欧拉图
- 真正有用的点只有 $\mathcal{O}(n)$ 个，使用 Hierholzer 算法可以做到 $\mathcal{O}(n)$ 的复杂度.

10th

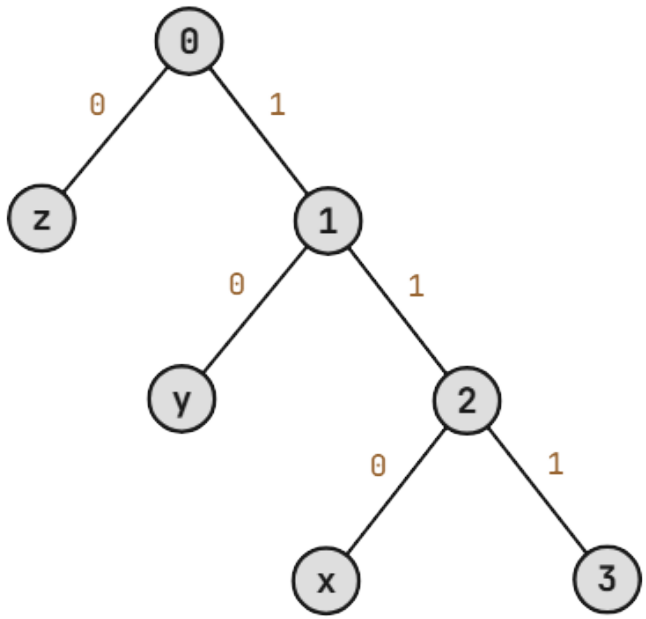
- 【1】NOI 以及相关活动的规则，【4】组合，【7】模运算意义下的逆元.
- A 选项中，$fact_i$ 显然可以线性递推. 根据费马小定理，可以使用快速幂 $O(\log V)$ 求出 $invf_V = fact_V^{M-2} \bmod M$，而又有递推 $invf_i \leftarrow invf_{i+1} \times (i+1) \bmod M$，故可以 $O(V + \log V)$ 的复杂度预处理出逆元. B 选项中，由递推式，有 $invf_{300} \equiv invf_{301} \times 301 \pmod M$. C 选项中，有：

$$\binom{n}{m} = \frac{n!}{m!(n-m)!} = fact_n \cdot invf_m \cdot invf_{n-m}$$

将 $n = 2000$ 且 $m = 500$ 带入即可. D 选项中，在参与 CCF 软件能力认证考试时，程序的长度不得超过 64 KB，直接打表会导致程序过长无法提交.

11th

- 【4】哈夫曼树的定义和构造、哈夫曼编码.
- 观察到 $2^i + 2^{i+1} < 2^{i+2}$，构造出的哈夫曼树的一部分如下图所示，以此类推. 故字符 z $\sim$ a 的哈夫曼编码长度分别为 $1 \sim 26$.



11th

所求即为：

$$\begin{aligned}
\text{ans} &= \sum_{i=1}^{26}(26-i+1)\cdot 2^i \\
&= \sum_{i=2}^{26}\sum_{j=1}^{i} 2^j \\
&= \sum_{i=2}^{26}(2^{i+1}-2) \\
&= \sum_{i=2}^{26} 2^{i+1} - 50 \\
&= 2^{28} - 8 - 50 \\
&= 2^{28} - 58
\end{aligned}$$

12th

- 【3】贪心法.
- 仅需使每行、每列、每对角线都恰有一个点没被选，此时是最优的. 因此不妨构造：

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

13th ∼ 14th

- 【0】计算理论基础.
- A 选项中，尽管 01 背包确实不是 P 问题，但是 NP-Hard 问题与 P 问题不冲突. B 选项中，如果一个问题是 NP 问题，则可以在多项式时间内验证解的正确性. C 选项中，DP 做法的时间复杂度为 $\mathcal{O}(n \cdot V)$，其中 V 为值域，与输入规模无关，因此非多项式做法. D 选项中，由于哈密顿回路问题是 NP 问题，而所有 NP 问题都可以在多项式时间内规约到 NP-Hard 问题，故正确.

- 【5】迭代器（iterator），【5】集合（set）.
- 去重排序后，有 $s = \{10, 20, 30, 50\}$. 而初始时 begin() 指向第一个元素 10，右移两个元素后指向第三个元素 30.

15th

- 【1】Linux 操作系统的基本概念及其常见操作.
- 在 Linux 系统中，用户 root 拥有最高权限.

16th

- 【1】模拟法，【3】递推法.
- 程序实现了求满足前 i 位的最小值为 b_i、最大值为 c_i 的排列 a_n 数.

16th (1st ∼ 3rd)

- 【2】函数定义.
- 由于 main() 函数是 int 类型的，故需要有一个 int 类型的返回值，修改后返回值类型变为 void().

- 【0】阅读理解题意.
- 若 $num = 0$，在第 44 行 $ans \leftarrow ans \times num$ 即 $ans \leftarrow 0$，输出亦为 0，程序的行为不变.

- 【1】模拟法.
- 根据题意手玩即可. 由于 $b_1 \neq c_1$，输出为 0.

16th (4th ∼ 6th)

- 【1】模拟法.
- 根据题意手玩即可. 满足条件的排列为 $\{1, 3, 2\}$.

- 【1】模拟法.
- 根据题意手玩即可.

- 【0】阅读理解题意.
- 必定有 $a_1 = 5$, $a_2 = 1$, $a_3 = 10$, 而剩余 7 个数无限制. 故答案为 $7! = 5040$.

17th

- 【3】计数排序，【6】字典树（Trie 树），【6】欧拉道路和欧拉回路.
- 程序实现了一种线性求解 P1127 词链 的方法，虽然由于码量原因少了很多东西. namespace A 实际上实现了一棵 Trie 树，可以用一遍 DFS 给字符串们按字典序排序. namespace B 实现了不太寻常的计数排序. namespace C 即为求解字典序最小的欧拉路径.

17th (1st ∼ 3rd)

- 【1】整数型：int、long long，【1】实数型：float、double.
- 虽然有 int INF $= 1e18$，但由于是浮点数会溢出为 2147483647，程序行为不变.

- 【1】数组与数组下标.
- 注意当 $c =$ z 时会越界，可能出现运行错误. 实际上，若第 8 行改为 $M = 26 + 5$ 则是正确的，因为字典树字符对应下标不影响答案.

- 【5】元组（tuple）.
- 元组 tuple 不能直接使用访问 first 或 operator[]，需要使用 std::get 类进行访问. 另外补充 auto [x, y] = v 为 C++17 及以后的写法.

17th (4th ∼ 6th)

- 【6】欧拉道路和欧拉回路.
- 当图不连通但满足度数约束时，例如输入为 2\naa\nbb，第 77 行的 t.size() == m 语句为假，assert() 语句使程序异常退出.

- 【1】模拟法.
- 根据题意手玩即可.

- 【6】字典树（Trie 树）.
- 观察 DFS 过程，当且仅当 $f_u \neq 0$ 时才对 tot 有贡献. 再观察每次执行 insert() 仅可能改变一个 f_p. 若存在 $s_i = s_j$，其最终得到的 p 也一定相同，此时 $tot < n$. 反之，$tot = n$.

17th (7th)

- 【6】时间复杂度分析.
- namespace A 建 Trie 树的复杂度为树上节点个数，确定字典序大小复杂度为节点个数乘字符集大小，即 $\mathcal{O}(k \cdot m)$. namespace B 为计数排序，调用的总复杂度为边数级别，即 $\mathcal{O}(n)$. namespace C 为欧拉路径，使用当前弧优化，复杂度为 $\mathcal{O}(n)$. 故总复杂度 $\mathcal{O}(k \cdot n)$.

18th

- 【6】字符串哈希函数构造.
- 程序实现了求解最长回文子串的长度.

18th (1st ∼ 2nd)

- 【0】阅读理解题意.
- 删去第 33 行后，程序无法继续向后扩展，发生死循环，有 $(l, r) \equiv (1, 0)$.

- 【8】概率的基本概念.
- 所求即为，随机一个长度为 1103 的小写字母组成的字符串是回文串的概率. 有:

$$ans = \frac{26^{552}}{26^{1103}} = \frac{1}{26^{551}} \approx 2.24 \times 10^{-780}$$

18th (3rd ∼ 4th)

- 【6】时间复杂度分析.
- 由于 r 单调向后递增，故复杂度为 $O(n)$.

- 【8】概率的基本概念.
- A 选项中，将 ULL 改为 signed int 可能导致溢出，但这份代码采用自然溢出 Hash，影响不大. B 选项中，由于 64 并非素数，用它当作 base 极有可能导致哈希冲突. C 选项中，由于字符集仅有小写字母 a 和 b，长为 10^7 的随机串的最大回文子串的长度在极大多数情况下很大，因此将 res_2 的初值设为 3 影响不大. D 选项中，由于实际长度后的字符数组均为空，对计算无影响. 每个选项的具体概率在此略去，请读者自行计算.

18th (5th ∼ 6th)

- 【1】模拟法.
- 根据题意手玩即可.

- 【4】排列，【7】容斥原理.
- 所求即为有多少个长为 5 的字符串的最长回文子串的长度为 4. 考虑一个容斥，令 $f(x)$ 表示最长回文子串的长度至少为 x 的方案数，则有：

$$\begin{aligned} ans &= f(4) - f(5) \\ &= (2 \times 26^3 - 26) - 26 \\ &= 35100 \end{aligned}$$

19th

- 【3】栈.
- 程序的时间复杂度为 $\mathcal{O}(n)$.

19th (1st ∼ 3rd)

- 【0】阅读理解题意.
- *res* 维护的是最近的可行的位置. 由题意，倒序压入，因此初始时将 $n+1$ 压入栈中.

- 【0】阅读理解题意.
- 当存在值为 a_i+1 的元素时，若 a_i 可行，则 pos_{a_i+1} 可行. 故此处应判断 pos_{a_i+1} 是否为空.

- 【0】阅读理解题意.
- 由于 $a_x \geq 1$，当某个 $a_i = 1$ 时无需入栈，这是因为不存在 $a_j = 0$，其没有贡献.

19th (4th ∼ 5th)

- 【0】阅读理解题意.
- 应向 pos_{a_i} 中压入 i，表示下标为 i 处有值为 a_i 的元素.

- 【0】阅读理解题意.
- 从最近一个未被删除的 $res.\text{top}()$ 的前一个元素到 i 都是好的，故有 $res.\text{top}() - i$ 的贡献.

20th

- 【4】高精度的加法，【4】高精度的减法，【4】高精度的乘法，【4】高精度整数除以单精度整数的商和余数.
- 程序实现了高精度整数 BigNum 类，每 4 位一节. 括号序列转化成整数序列的方法是将每个左括号替换为其深度，称转化后的整数序列为整数串.

20th (1st ∼ 3rd)

- 【4】高精度的减法.
- 当 $x < 0$ 时发生退位，当前位加上 *base* 而下一位减去 1.
- 【4】动态规划的基本思路.
- 有转移 $f_{i,j} \leftarrow \sum_{k=1}^{j+1} f_{i-1,k}$.
- 【4】杨辉三角.
- $c_{i,j}$ 表示从 i 个物品中选 j 个物品的方案数，由杨辉三角递推，即 $c_{i,j} \leftarrow c_{i-1,j} + c_{i-1,j-1}$.

20th (4th ∼ 5th)

- 【4】组合.
- 从剩下 $(n \times 2) - (i + 1)$ 次操作中选择 $n - cnt + 1$ 次作为 R，方案数为 $c_{(n\times 2)-(i+1),n-cnt+1}$. 注意此处字符串的下标从 0 开始.

- 【0】阅读理解题意.
- 目前整数串的长度为 $n - i + 1$，结尾是第 j 层，方案数即为 $f_{n-i+1,j}$.

Thank you!