题解

2024 年 7 月 27 日

1. shark
2. housekeep
3. electric
4. speed

1. shark
2. housekeep
3. electric
4. speed

shark

- 把两个序列都升序排列可以得到最小距离。(交换逆序对可以改进结果)

shark

- 把两个序列都升序排列可以得到最小距离。(交换逆序对可以改进结果)
- 把序列 a 的元素替换为它对应的元素在 b 中的位置。

shark

- 把两个序列都升序排列可以得到最小距离。(交换逆序对可以改进结果)
- 把序列 a 的元素替换为它对应的元素在 b 中的位置。
- 把排列拆成若干个循环。总的交换次数为元素个数 - 环的个数。

1 shark

2 housekeep

3 electric

4 speed

housekeep

- 火车的使用是以 $x_i + y_i$ 为循环的。

housekeep

- 火车的使用是以 $x_i + y_i$ 为循环的。
- Sub1,2：暴力算出每个时间是休息还是工作。

housekeep

- 火车的使用是以 $x_i + y_i$ 为循环的。
- Sub1,2：暴力算出每个时间是休息还是工作。
- Sub3：x,y 都很小，考虑对模 1 到 60 分类。

housekeep

- 火车的使用是以 $x_i + y_i$ 为循环的。
- Sub1,2：暴力算出每个时间是休息还是工作。
- Sub3：x,y 都很小，考虑对模 1 到 60 分类。
- 前缀和算一下即可。

housekeep

- 正解根号分治。

housekeep

- 正解根号分治。
- 设置一个阈值 B，如果 $x_i + y_i$ 大于 B，可以暴力，单次操作数 $\frac{n}{x_i+y_i}$。

housekeep

- 正解根号分治。
- 设置一个阈值 B，如果 $x_i + y_i$ 大于 B，可以暴力，单次操作数 $\frac{n}{x_i+y_i}$。
- 如果 $x_i + y_i$ 小于 B，那也可以暴力。

housekeep

- 正解根号分治。
- 设置一个阈值 B，如果 $x_i + y_i$ 大于 B，可以暴力，单次操作数 $\frac{n}{x_i+y_i}$。
- 如果 $x_i + y_i$ 小于 B，那也可以暴力。
- 考虑对模 1 到 B 分类，这样一个车是一种区间加，操作数 $x_i + y_i$。

housekeep

- 正解根号分治。
- 设置一个阈值 B，如果 $x_i + y_i$ 大于 B，可以暴力，单次操作数 $\frac{n}{x_i+y_i}$。
- 如果 $x_i + y_i$ 小于 B，那也可以暴力。
- 考虑对模 1 到 B 分类，这样一个车是一种区间加，操作数 $x_i + y_i$。
- 最后也是前缀和。

housekeep

- 正解根号分治。
- 设置一个阈值 B，如果 $x_i + y_i$ 大于 B，可以暴力，单次操作数 $\frac{n}{x_i+y_i}$。
- 如果 $x_i + y_i$ 小于 B，那也可以暴力。
- 考虑对模 1 到 B 分类，这样一个车是一种区间加，操作数 $x_i + y_i$。
- 最后也是前缀和。
- 总复杂度 $O(nB + \frac{n^2}{B})$，取 $B = \sqrt{n}$，得到总复杂度 $O(n\sqrt{n})$。

1. shark
2. housekeep
3. electric
4. speed

electric

- 给出一个长度为 n 的非负整数数列 a，下标编号从 1 到 n。
- 定义一个数列 a 的代价为 $\min\limits_{i \neq j} a_i | a_j$，其中 $|$ 表示运算。
- q 个询问，每个询问给出两个整数 l, r $(l < r)$，求数列 $a_l, a_{l+1}, \ldots, a_r$ 的最小代价。

electric

- 小清新数据结构。

electric

- 小清新数据结构。
- 先考虑全局的代价怎么求。

electric

- 小清新数据结构。
- 先考虑全局的代价怎么求。
- 对这种位运算的问题可以考虑 01trie。

electric

- 小清新数据结构。
- 先考虑全局的代价怎么求。
- 对这种位运算的问题可以考虑 01trie。
- 对于 trie 上的一个节点 p，如果左儿子有至少两个数，那么肯定往左走，这一位填 0。

electric

- 小清新数据结构。
- 先考虑全局的代价怎么求。
- 对这种位运算的问题可以考虑 01trie。
- 对于 trie 上的一个节点 p，如果左儿子有至少两个数，那么肯定往左走，这一位填 0。
- 如果左子树没有数，肯定走右边。

electric

- 小清新数据结构。
- 先考虑全局的代价怎么求。
- 对这种位运算的问题可以考虑 01trie。
- 对于 trie 上的一个节点 p，如果左儿子有至少两个数，那么肯定往左走，这一位填 0。
- 如果左子树没有数，肯定走右边。
- 如果左边恰有一个数，那就把左边这个数的这位变成 1 后插入右边。

electric

- 小清新数据结构。
- 先考虑全局的代价怎么求。
- 对这种位运算的问题可以考虑 01trie。
- 对于 trie 上的一个节点 p，如果左儿子有至少两个数，那么肯定往左走，这一位填 0。
- 如果左子树没有数，肯定走右边。
- 如果左边恰有一个数，那就把左边这个数的这位变成 1 后插入右边。
- 只会新插入 log 个数，复杂度 log^2。

electric

- 现在在区间 [l,r] 上怎么做？

electric

- 现在在区间 [l,r] 上怎么做？
- 发现我们的做法只有数量的判断，没有取 maxmin 这种操作。

electric

- 现在在区间 [l,r] 上怎么做？
- 发现我们的做法只有数量的判断，没有取 maxmin 这种操作。
- 所以在区间上只需要建立可持久化数据结构。

electric

- 现在在区间 [l,r] 上怎么做？
- 发现我们的做法只有数量的判断，没有取 maxmin 这种操作。
- 所以在区间上只需要建立可持久化数据结构。
- 建可持久化 01trie。判断子树内数的个数可以作差。

electric

- 现在在区间 [l,r] 上怎么做？
- 发现我们的做法只有数量的判断，没有取 maxmin 这种操作。
- 所以在区间上只需要建立可持久化数据结构。
- 建可持久化 01trie。判断子树内数的个数可以作差。
- 具体实现，把左边的数插入右边不好维护，可以用 vector 记下需要走的子树集合。

electric

- 现在在区间 [l,r] 上怎么做？
- 发现我们的做法只有数量的判断，没有取 maxmin 这种操作。
- 所以在区间上只需要建立可持久化数据结构。
- 建可持久化 01trie。判断子树内数的个数可以作差。
- 具体实现，把左边的数插入右边不好维护，可以用 vector 记下需要走的子树集合。
- 复杂度不变。

1. shark
2. housekeep
3. electric
4. speed

speed

- 一共有 n 个候选人，第 i 个候选人本来会得到 a_i 票（这些选票是无法被改变的）。
- 而你所在的小团体一共有 m 个人（这些人的选票不被 a_i 包含），你可以自由控制这些人的选票。具体来说，这 m 个人每人都必须投恰好 k 票给不同的人，最终票数前 p 大的同学可以被选举为三好学生，平票任意排列，由于一些原因，p 的值并没有被提前确定，只知道 $1 \le p \le n$。
- 现在要求出，对于这 m 个人所有投票方案以及 p 的取值，三好学生的集合有多少种可能，答案对 998244353 取模。

speed

- 核心思路：找到一个集合合法的充要条件，对着这个 dp，然后优化 dp 复杂度。

speed

- 核心思路：找到一个集合合法的充要条件，对着这个 dp，然后优化 dp 复杂度。
- 钦定当选的集合，判断是否合法。

speed

- 核心思路：找到一个集合合法的充要条件，对着这个 dp，然后优化 dp 复杂度。
- 钦定当选的集合，判断是否合法。
- 分两种情况考虑：

speed

- 核心思路：找到一个集合合法的充要条件，对着这个 dp，然后优化 dp 复杂度。
- 钦定当选的集合，判断是否合法。
- 分两种情况考虑：
  - $p \leq k$，每个人一定只会投集合内的，对于集合内的，一定从当前票数从小往大投。

speed

- 核心思路：找到一个集合合法的充要条件，对着这个 dp，然后优化 dp 复杂度。
- 钦定当选的集合，判断是否合法。
- 分两种情况考虑：
  - $p \leq k$，每个人一定只会投集合内的，对于集合内的，一定从当前票数从小往大投。
  - $p > k$，每个人一定集合内全部都投，但还需要投一些集合外的，对于集合外的，一定也是从当前票数从小往大投。

speed

- 核心思路：找到一个集合合法的充要条件，对着这个 dp，然后优化 dp 复杂度。
- 钦定当选的集合，判断是否合法。
- 分两种情况考虑：
  - $p \leq k$，每个人一定只会投集合内的，对于集合内的，一定从当前票数从小往大投。
  - p > k，每个人一定集合内全部都投，但还需要投一些集合外的，对于集合外的，一定也是从当前票数从小往大投。
- 时间复杂度 $O(2^n mk)$，期望得分 30 分。

speed

- 只考虑 $p \le k$ 合法相当于存在一种方案，集合内的 $\min \le$ 集合外的 $\max$，显然我们不会投集合外的人，所以集合外的 $\max$ 是确定的。

speed

- 只考虑 $p \le k$ 合法相当于存在一种方案，集合内的 min $\le$ 集合外的 max，显然我们不会投集合外的人，所以集合外的 max 是确定的。
- 结论：合法当且仅当 $min + m \ge max$ 且集合内 $\sum \max(0, max - a_i) \le mk$。

speed

- 只考虑 $p \le k$ 合法相当于存在一种方案，集合内的 min $\le$ 集合外的 max，显然我们不会投集合外的人，所以集合外的 max 是确定的。
- 结论：合法当且仅当 $min + m \ge max$ 且集合内 $\sum \max(0, max - a_i) \le mk$。
- 枚举 max，dp[i][j][s][p] 表示前 i 个数，min 是 j，$\max(0, max - a_i)$ 的和是 s，一共选了 p 个人（这里为了判断 p 和 k 的大小关系）的方案数，状态数 $O(n^3 \sum a_i)$，转移 O(1)，但因为枚举 max 要算 n 次，时间复杂度 $O(n^4 \sum a_i)$，滚动数组后空间复杂度 $O(n^2 \sum a_i)$，精细实现可能通过 sub3，期望得分 60 （或者 30）。

speed

- 可以提前将 a_i 从小到大排序，一开始默认全选，dp 时枚举每个值要不要扔掉，这样可以在 dp 的过程中枚举 max，dp 状态中的 s 改成直接记 a_i 的和。

speed

- 可以提前将 a_i 从小到大排序，一开始默认全选，dp 时枚举每个值要不要扔掉，这样可以在 dp 的过程中枚举 max，dp 状态中的 s 改成直接记 a_i 的和。
- 时间复杂度 $O(n^3 \sum a_i)$，空间复杂度 $O(n^2 \sum a_i)$，期望得分 60。

speed

- 注意到记录 j 的作用是为了防止选 a_i 过小的元素，事实上在排序之后，这个限制就变成了只能选 [max m, max] 中的 a_i，问题就转化为，每次询问一个区间，区间内有若干个 a_i，要对于所有 p, s 求出这个区间内选 p 个数和为 s 的方案数。

speed

- 注意到记录 j 的作用是为了防止选 a_i 过小的元素，事实上在排序之后，这个限制就变成了只能选 [max m, max] 中的 a_i，问题就转化为，每次询问一个区间，区间内有若干个 a_i，要对于所有 p, s 求出这个区间内选 p 个数和为 s 的方案数。
- 每次重新 dp，空间复杂度可以降至 $O(n \sum a_i)$，时间复杂度 $O(n^3 \sum a_i)$，期望得分 60 or 80。

speed

- 由于区间左右端点单调，而这个背包是可删除的，可以直接双指针维护，每次不用重新 dp，时间复杂度 $O(n^2\sum a_i)$，期望得分 100。

speed

- 由于区间左右端点单调，而这个背包是可删除的，可以直接双指针维护，每次不用重新 dp，时间复杂度 $O(n^2\sum a_i)$，期望得分 100。
- 具体实现可以参考 std。