梦熊 CSP-S 模拟赛讲评

yyOI 出题团

2024 年 10 月 19 日

目录

题意

现在有 n 个敌人，第 i 个敌人的初始攻击力为正整数 a_i。初始生命值为正整数 W。
定义如下流程为一场战斗：
从第 1 个敌人开始，每个敌人依次循环进行攻击。第 i 个敌人发起攻击时，生命值 W 减去 a_i，同时 a_i 翻倍。
当 $W \le 0$ 时，本场战斗立刻结束。然后重置生命值 W 以及所有敌人的攻击力 a_i 。定义本次战斗的评分为接受敌人攻击的次数（不包括致命攻击）。
q 次询问，每次询问给出三个数 l, r, d，表示对第 $[l, r]$ 个敌人进行强化，使每个敌人的 a_i 增加 d，然后立刻进行一场战斗。输出此次战斗的评分。

询问之间相互影响。

- 设所有垃圾桶当前攻击力总和为 S。

- 设所有垃圾桶当前攻击力总和为 S。
- 考虑暴力，因为生命值 $W \leq 10^{18}$，攻击力是翻倍递增的，因此最多只会打 $\log W$ 轮。因此暴力的时间复杂度是 $O(nq \log W)$。可以获得 20 分。

- 设所有垃圾桶当前攻击力总和为 S。
- 考虑暴力，因为生命值 $W \leq 10^{18}$，攻击力是翻倍递增的，因此最多只会打 $\log W$ 轮。因此暴力的时间复杂度是 $O(nq \log W)$。可以获得 20 分。
- 假设这场战斗完整地打了 k 轮，那么这 k 轮需要消耗的生命值为 $S \times (2^0 + 2^1 + \cdots + 2^{k-1}) = S \times (2^k - 1)$。

- 设所有垃圾桶当前攻击力总和为 S。
- 考虑暴力，因为生命值 $W \le 10^{18}$，攻击力是翻倍递增的，因此最多只会打 $\log W$ 轮。因此暴力的时间复杂度是 $O(nq \log W)$。可以获得 20 分。
- 假设这场战斗完整地打了 k 轮，那么这 k 轮需要消耗的生命值为 $S \times (2^0 + 2^1 + \cdots + 2^{k-1}) = S \times (2^k - 1)$。
- 即要求出最大的 m，使得 $\sum_{i=1}^{m} a_i \le W - S \times (2^k - 1)$。

- 设所有垃圾桶当前攻击力总和为 S。
- 考虑暴力，因为生命值 $W \le 10^{18}$，攻击力是翻倍递增的，因此最多只会打 $\log W$ 轮。因此暴力的时间复杂度是 $O(nq\log W)$。可以获得 20 分。
- 假设这场战斗完整地打了 k 轮，那么这 k 轮需要消耗的生命值为 $S \times (2^0 + 2^1 + \cdots + 2^{k-1}) = S \times (2^k - 1)$。
- 即要求出最大的 m，使得 $\sum_{i=1}^{m} a_i \le W - S \times (2^k - 1)$。
- 答案即为 $k \times n + m$。

- 显然，对于每一次修改，S 只会增加 $(r_i - l_i + 1) \times d_i$。

- 显然，对于每一次修改，S 只会增加 $(r_i - l_i + 1) \times d_i$。
- 对于每一个询问，我们需要求出最大的 k。

- 显然，对于每一次修改，S 只会增加 $(r_i - l_i + 1) \times d_i$。
- 对于每一个询问，我们需要求出最大的 k。
- 发现 k 不需要每次都枚举，因为每次操作后，答案只会变小，也就是 k 是递减的。

- 显然，对于每一次修改，S 只会增加 $(r_i - l_i + 1) \times d_i$。
- 对于每一个询问，我们需要求出最大的 k。
- 发现 k 不需要每次都枚举，因为每次操作后，答案只会变小，也就是 k 是递减的。
- 对于相同的 k，显然 m 也在递减。于是用指针维护 m 的值，同时用两个差分数组维护区间加即可。

- 显然，对于每一次修改，S 只会增加 $(r_i - l_i + 1) \times d_i$。
- 对于每一个询问，我们需要求出最大的 k。
- 发现 k 不需要每次都枚举，因为每次操作后，答案只会变小，也就是 k 是递减的。
- 对于相同的 k，显然 m 也在递减。于是用指针维护 m 的值，同时用两个差分数组维护区间加即可。
- 对于不同的 k，暴力求解即可。

- 显然，对于每一次修改，S 只会增加 $(r_i - l_i + 1) \times d_i$。
- 对于每一个询问，我们需要求出最大的 k。
- 发现 k 不需要每次都枚举，因为每次操作后，答案只会变小，也就是 k 是递减的。
- 对于相同的 k，显然 m 也在递减。于是用指针维护 m 的值，同时用两个差分数组维护区间加即可。
- 对于不同的 k，暴力求解即可。
- 时间复杂度 $O(n \log W + q)$。

题意

youyou 有一个大小为 $2 \times n$ 的网格，每个格子可能是黑色或者白色。
现在 youyou 和 yy 要在这个网格上玩一个游戏：

- youyou 先选取出一个可以为空的**连通块**。
- 之后 yy 可以选择最多 m 列，将这些列上下行的格子颜色互换。

定义一个格子集合 S 为一个连通块，当且仅当 S 中任意两个点可以通过集合 S 内**边相邻**的若干个点连通。
youyou 希望最大化最终黑色格子减白色格子的数量，而 yy 希望最小化之。
现在 youyou 希望你求出：在双方都采用最优策略的情况下，最终黑色格子减白色格子的数量是多少？

- 考虑 youyou 选出的连通块左右端点被确定为 l, r。

- 考虑 youyou 选出的连通块左右端点被确定为 l, r。
- 显然，全黑列他会都去选择，全白列他只会选择一个格子，因为这些不受 yy 的影响。

- 考虑 youyou 选出的连通块左右端点被确定为 l, r。
- 显然，全黑列他会都去选择，全白列他只会选择一个格子，因为这些不受 yy 的影响。
- 考虑一黑一白的列。假如他两个格子都选择，那么贡献为 0，如果只选择一个黑色的格子，虽然贡献是 1，但是可能被 yy 操作变成 -1。

- 考虑 youyou 选出的连通块左右端点被确定为 l, r。
- 显然，全黑列他会都去选择，全白列他只会选择一个格子，因为这些不受 yy 的影响。
- 考虑一黑一白的列。假如他两个格子都选择，那么贡献为 0，如果只选择一个黑色的格子，虽然贡献是 1，但是可能被 yy 操作变成 -1。
- 于是他有两种策略：

- 考虑 youyou 选出的连通块左右端点被确定为 l, r。
- 显然，全黑列他会都去选择，全白列他只会选择一个格子，因为这些不受 yy 的影响。
- 考虑一黑一白的列。假如他两个格子都选择，那么贡献为 0，如果只选择一个黑色的格子，虽然贡献是 1，但是可能被 yy 操作变成 -1。
- 于是他有两种策略：
    - 所有的一黑一白列我们都选择两个。这样 yy 没办法操作。

- 考虑 youyou 选出的连通块左右端点被确定为 l, r。
- 显然，全黑列他会都去选择，全白列他只会选择一个格子，因为这些不受 yy 的影响。
- 考虑一黑一白的列。假如他两个格子都选择，那么贡献为 0，如果只选择一个黑色的格子，虽然贡献是 1，但是可能被 yy 操作变成 -1。
- 于是他有两种策略：
  - 所有的一黑一白列我们都选择两个。这样 yy 没办法操作。
  - 将 x 个一黑一白列选择一个格子，其余选择两个。这样 yy 可以操作。

- 发现若 youyou 选择策略二为优，当且仅当至少有 $2m$ 个一黑一白列他选择了一个格子。否则，我们可以将这些列选择两个格子，显然连通块仍连通，对答案的贡献为 0；而原来对答案的贡献为 $x-2m<0$。

- 发现若 youyou 选择策略二为优，当且仅当至少有 $2m$ 个一黑一白列他选择了一个格子。否则，我们可以将这些列选择两个格子，显然连通块仍连通，对答案的贡献为 0；而原来对答案的贡献为 $x - 2m < 0$。
- 因此，youyou 的策略二，可以视作在**不考虑操作**的情况下选出一个连通块。我们只需求这个连通块的最大权值最后减去 $2m$，这一部分可以用 dp 实现。

- 发现若 youyou 选择策略二为优，当且仅当至少有 $2m$ 个一黑一白列他选择了一个格子。否则，我们可以将这些列选择两个格子，显然连通块仍连通，对答案的贡献为 0；而原来对答案的贡献为 $x-2m<0$。
- 因此，youyou 的策略二，可以视作在**不考虑操作**的情况下选出一个连通块。我们只需求这个连通块的最大权值最后减去 $2m$，这一部分可以用 dp 实现。
- youyou 的策略一是经典最大子段和问题，也可以使用 dp 实现。

- 发现若 youyou 选择策略二为优，当且仅当至少有 $2m$ 个一黑一白列他选择了一个格子。否则，我们可以将这些列选择两个格子，显然连通块仍连通，对答案的贡献为 0；而原来对答案的贡献为 $x - 2m < 0$。
- 因此，youyou 的策略二，可以视作在**不考虑操作**的情况下选出一个连通块。我们只需求这个连通块的最大权值最后减去 $2m$，这一部分可以用 dp 实现。
- youyou 的策略一是经典最大子段和问题，也可以使用 dp 实现。
- 时间复杂度 $O(n)$。

题意

给定一个长度为 n 的非负整数序列 a，初始时所有数字均被标记为蓝色，youyou 和 yy 轮流对序列 a 进行操作，由 youyou 开始。

- 如果当前是 youyou 的回合，那么他可以至多选择连续的 c_1 个数，如果他们的和小于等于 w_1，则标记为红色。
- 如果当前是 yy 的回合，那么他可以至多选择连续的 c_2 个数，如果他们的和大于 w_2，则标记为蓝色。

定义 youyou 胜利即是在游戏任意时刻，所有数字都被标记为红色，定义 yy 胜利则是在无穷多个回合内，youyou 无法胜利。
现在给定 q 个操作，对于每个操作给定三个数 opt, x, y。

- 如果 opt 为 1，表示将 a_x 增加 y 。
- 如果 opt 为 2，表示在序列 $[x, y]$ 上进行一轮游戏。

对于每一个操作 2 ，判断 youyou 能否获得胜利。

- 首先进行特判，如果询问的区间中含有大于 w_1 的数字，那么 youyou 显然必败。

- 首先进行特判，如果询问的区间中含有大于 w_1 的数字，那么 youyou 显然必败。
- 因为所有数均为非负整数，发现 yy 选择长度正好为 c_2 的子区间是一定不劣的。

- 首先进行特判，如果询问的区间中含有大于 w_1 的数字，那么 youyou 显然必败。
- 因为所有数均为非负整数，发现 yy 选择长度正好为 c_2 的子区间是一定不劣的。
- 以下称长度为 c_2，总和大于 w_2 的子区间为“合法区间”。也即 yy 可以操作的区间。

- 首先进行特判，如果询问的区间中含有大于 w_1 的数字，那么 youyou 显然必败。
- 因为所有数均为非负整数，发现 yy 选择长度正好为 c_2 的子区间是一定不劣的。
- 以下称长度为 c_2，总和大于 w_2 的子区间为“合法区间”。也即 yy 可以操作的区间。
- 重要结论 1：对于序列中不在任何一个“合法区间”内的数，一定不会对答案产生影响。

- 首先进行特判，如果询问的区间中含有大于 w_1 的数字，那么 youyou 显然必败。
- 因为所有数均为非负整数，发现 yy 选择长度正好为 c_2 的子区间是一定不劣的。
- 以下称长度为 c_2，总和大于 w_2 的子区间为“合法区间”。也即 yy 可以操作的区间。
- 重要结论 1：对于序列中不在任何一个“合法区间”内的数，一定不会对答案产生影响。
- 对于这些数字，youyou 可以用任意多次回合将它们染红，且 yy 无法重新将它们染蓝。而由于回合数是可视作无限的，youyou 总有时间去将这些数字染色。

- 接下来我们分情况讨论。

- 接下来我们分情况讨论。
- 性质 1：对于不存在任何一个“合法区间”的序列，youyou 显然必胜。

- 接下来我们分情况讨论。
- 性质 1：对于不存在任何一个“合法区间”的序列，youyou 显然必胜。
- 性质 2：存在“合法区间”，若 youyou 可以一次性染红所有未染红的合法区间，则 youyou 必胜。

- 接下来我们分情况讨论。
- 性质 1：对于不存在任何一个“合法区间”的序列，youyou 显然必胜。
- 性质 2：存在“合法区间”，若 youyou 可以一次性染红所有未染红的合法区间，则 youyou 必胜。
- 性质 3：存在“合法区间”，若 youyou 不可以做到一次性染红所有未染红的合法区间，则 yy 必胜。

- 重要性质 2: yy 的最优策略是：尽量在整个数列的边缘进行染色。

- 重要性质 2：yy 的最优策略是：尽量在整个数列的边缘进行染色。
- 这一点很好理解，作为防守方，yy 的目的是防止 youyou 将整个序列染成红色。

- 重要性质 2: yy 的最优策略是：尽量在整个数列的边缘进行染色。
- 这一点很好理解，作为防守方，yy 的目的是防止 youyou 将整个序列染成红色。
- 设所有“合法区间”中位于最左边的左端点为 l，最右边的右端点为 r。

- 重要性质 2：yy 的最优策略是：尽量在整个数列的边缘进行染色。
- 这一点很好理解，作为防守方，yy 的目的是防止 youyou 将整个序列染成红色。
- 设所有“合法区间”中位于最左边的左端点为 l，最右边的右端点为 r。
- 如果满足性质 3，只考虑位置 l, r。若 youyou 每次把哪个点染了，yy 就可以跟着染。进而回到上述讨论。此时 youyou 必须重新将所有“合法区间”染色。此时进入循环，那么 youyou 必败。

- 重要性质 2: yy 的最优策略是：尽量在整个数列的边缘进行染色。
- 这一点很好理解，作为防守方，yy 的目的是防止 youyou 将整个序列染成红色。
- 设所有“合法区间”中位于最左边的左端点为 l，最右边的右端点为 r。
- 如果满足性质 3，只考虑位置 l, r。若 youyou 每次把哪个点染了，yy 就可以跟着染。进而回到上述讨论。此时 youyou 必须重新将所有“合法区间”染色。此时进入循环，那么 youyou 必败。
- 那么只要 youyou 无法一次染红 l 和 r，即 $r - l + 1 > c_1$ 或 $sum(l, r) > w_1$ 时，yy 必胜。

- 考虑如何求出 l, r。

- 考虑如何求出 l, r。
- 如果暴力求值，时间复杂度 $O(qn)$，不可接受，预计得分 40。

- 考虑如何求出 l, r。
- 如果暴力求值，时间复杂度 $O(qn)$，不可接受，预计得分 40。
- 可以暴力更新所有修改时受影响的区间，时间复杂度 $O(qc_2)$。预计得分 70。

- 考虑如何求出 l, r。
- 如果暴力求值，时间复杂度 $O(qn)$，不可接受，预计得分 40。
- 可以暴力更新所有修改时受影响的区间，时间复杂度 $O(qc_2)$。预计得分 70。
- trick：使用线段树，每个叶子记录一个长度为 c_2 的区间，将单点修改转为区间修改。

- 考虑如何求出 l, r。
- 如果暴力求值，时间复杂度 $O(qn)$，不可接受，预计得分 40 。
- 可以暴力更新所有修改时受影响的区间，时间复杂度 $O(qc_2)$。预计得分 70。
- trick：使用线段树，每个叶子记录一个长度为 c_2 的区间，将单点修改转为区间修改。
- 实现线段树上二分即可求出 l, r。

- 考虑如何求出 l, r。
- 如果暴力求值，时间复杂度 $O(qn)$，不可接受，预计得分 40。
- 可以暴力更新所有修改时受影响的区间，时间复杂度 $O(qc_2)$。预计得分 70。
- trick：使用线段树，每个叶子记录一个长度为 c_2 的区间，将单点修改转为区间修改。
- 实现线段树上二分即可求出 l, r。
- 时间复杂度 $O(n\log n)$。预计得分 100。

题意

题面太长，这里就不放了。

- 注意到 3×10^5 的三进制表示只有 12 位。

- 注意到 3×10^5 的三进制表示只有 12 位。
- 考虑建图。

- 注意到 3×10^5 的三进制表示只有 12 位。
- 考虑建图。
- 发现题目所述与圆方树的性质很像。

- 注意到 3×10^5 的三进制表示只有 12 位。
- 考虑建图。
- 发现题目所述与圆方树的性质很像。
- 用 tarjan 跑出圆方树。

- 思考需要在圆方树上维护什么信息。

- 思考需要在圆方树上维护什么信息。
- 对于每个点，我们维护以下三个信息：

- 思考需要在圆方树上维护什么信息。
- 对于每个点，我们维护以下三个信息：
  - sum_t 表示若有一条以节点 t 为结尾的序列 c，它的总贡献。

- 思考需要在圆方树上维护什么信息。
- 对于每个点，我们维护以下三个信息：
  - sum_t 表示若有一条以节点 t 为结尾的序列 c，它的总贡献。
  - $ssum_t$ 表示对于以节点 t 为根的整棵子树的总贡献。

- 思考需要在圆方树上维护什么信息。
- 对于每个点，我们维护以下三个信息：
  - sum_t 表示若有一条以节点 t 为结尾的序列 c，它的总贡献。
  - $ssum_t$ 表示对于以节点 t 为根的整棵子树的总贡献。
  - $sumson_t$ 表示对于节点 t 的所有儿子，它们的 $ssum$ 之和。

- 思考需要在圆方树上维护什么信息。
- 对于每个点，我们维护以下三个信息：
  - sum_t 表示若有一条以节点 t 为结尾的序列 c，它的总贡献。
  - $ssum_t$ 表示对于以节点 t 为根的整棵子树的总贡献。
  - $sumson_t$ 表示对于节点 t 的所有儿子，它们的 $ssum$ 之和。
- 维护 $sumson$ 数组是容易的，考虑如何维护 sum 数组和 $ssum$ 数组。

- 对 sum_t 有贡献的情况可以分为两种：

- 对 sum_t 有贡献的情况可以分为两种：
  - 有一条以 t 为端点的路径，贡献为 $size_t \times 2$。

- 对 sum_t 有贡献的情况可以分为两种：
  - 有一条以 t 为端点的路径，贡献为 $size_t \times 2$。
  - 有一条以 t 为 lca 的路径，贡献为 $\sum_{i=1}^{d}(size_t - size_{son_i} - 1) * size_i$，其中 d 为 t 儿子的数量。

- 对 sum_t 有贡献的情况可以分为两种：
  - 有一条以 t 为端点的路径，贡献为 $size_t \times 2$。
  - 有一条以 t 为 lca 的路径，贡献为 $\sum_{i=1}^{d}(size_t - size_{son_i} - 1) * size_i$，其中 d 为 t 儿子的数量。
- 考虑计算 $ssum_t$：

- 对 sum_t 有贡献的情况可以分为两种：
  - 有一条以 t 为端点的路径，贡献为 $size_t \times 2$。
  - 有一条以 t 为 lca 的路径，贡献为 $\sum_{i=1}^{d}(size_t - size_{son_i} - 1) * size_i$，其中 d 为 t 儿子的数量。
- 考虑计算 $ssum_t$：
  - 如果 t 为圆点，那么 t 点的所有儿子对其都有贡献，$ssum_t = (\sum_{i=1}^{d} ssum_{son_i}) + sum_t$。

- 对 sum_t 有贡献的情况可以分为两种：
  - 有一条以 t 为端点的路径，贡献为 $size_t \times 2$。
  - 有一条以 t 为 lca 的路径，贡献为 $\sum_{i=1}^{d}(size_t - size_{son_i} - 1) * size_i$，其中 d 为 t 儿子的数量。
- 考虑计算 $ssum_t$：
  - 如果 t 为圆点，那么 t 点的所有儿子对其都有贡献，$ssum_t = (\sum_{i=1}^{d} ssum_{son_i}) + sum_t$。
  - 如果 t 为方点，那么位于同一个点双内的点均无贡献，$ssum_t = (\sum_{i=1}^{d} sumson_{son_i}) + sum_t$。

- 对 sum_t 有贡献的情况可以分为两种：
  - 有一条以 t 为端点的路径，贡献为 $size_t \times 2$。
  - 有一条以 t 为 lca 的路径，贡献为 $\sum_{i=1}^{d}(size_t - size_{son_i} - 1) * size_i$，其中 d 为 t 儿子的数量。
- 考虑计算 $ssum_t$：
  - 如果 t 为圆点，那么 t 点的所有儿子对其都有贡献，$ssum_t = (\sum_{i=1}^{d} ssum_{son_i}) + sum_t$。
  - 如果 t 为方点，那么位于同一个点双内的点均无贡献，$ssum_t = (\sum_{i=1}^{d} sumson_{son_i}) + sum_t$。
- 预处理是 $O(n)$ 的，考虑计算答案。

- 由于这棵树的高度较低（本题中不会超过 23），考虑对每一个圆点暴力往上跳，计算答案。

- 由于这棵树的高度较低（本题中不会超过 23），考虑对每一个圆点暴力往上跳，计算答案。
- 对于当前跳到的点，我们考虑当前点的贡献和兄弟节点的贡献。

- 由于这棵树的高度较低（本题中不会超过 23），考虑对每一个圆点暴力往上跳，计算答案。
- 对于当前跳到的点，我们考虑当前点的贡献和兄弟节点的贡献。
- 有了预处理好的数组，这部分的计算是容易的。

- 由于这棵树的高度较低（本题中不会超过 23），考虑对每一个圆点暴力往上跳，计算答案。
- 对于当前跳到的点，我们考虑当前点的贡献和兄弟节点的贡献。
- 有了预处理好的数组，这部分的计算是容易的。
- 总时间复杂度 $O(n\log_3 n)$，可以通过本题。

- 由于这棵树的高度较低（本题中不会超过 23），考虑对每一个圆点暴力往上跳，计算答案。
- 对于当前跳到的点，我们考虑当前点的贡献和兄弟节点的贡献。
- 有了预处理好的数组，这部分的计算是容易的。
- 总时间复杂度 $O(n\log_3 n)$，可以通过本题。
- Bonus：还有一个 $O(n)$ 的换根 dp 做法，实现起来较为复杂。