

第 12 届 CCPC 网络预选赛题解

EPS.AC 出题组

预期难度顺序：IN < BHE < CKM < JL < AF < DG 。补题链接：<https://qoj.ac/contest/4128>

1 都是最优交易

需要注意的是，本题的区间长度是 $[\max(\lfloor 0.15n \rfloor, 1), \max(\lfloor 0.85n \rfloor, 1)]$ 内随机选取的，且区间起点是随机选取的。这就意味着区间不会过短。所以有以下可行的方法：

1.1 方法一：CDQ 分治

如果本题没有这个条件，则 CDQ 分治的大致流程是：对于左右端点在 $[l, r]$ 之间的查询，先处理跨过 $mid = \lfloor \frac{l+r}{2} \rfloor$ 的那些查询。递归分别处理在 $[l, mid], [mid+1, r]$ 的查询。

我们先看这件事怎么做。一般而言，采用的方法是得到 mid 左侧的后缀的信息，和 $mid+1$ 右侧的前缀的信息，再将二者进行合并。

左侧后缀可以维护：从 i 开始，进行最多 j 次完整买卖后，未进行买入 / 进行买入的最优收支情况 $dpl_0[i][j], dpl_1[i][j]$ 。右侧前缀可以维护：到 i 为止，不进行 / 先进行卖出操作后，进行最多 j 次完整买卖后的最优收支情况 $dpr_0[i][j], dpr_1[i][j]$ 。

这样，两侧的结果就可以合并，得到 $[l, r]$ 的结果就是

$$\max \left(\max_{x=0}^k (dpl_0[l][x] + dpr_0[r][k-x]), \max_{x=0}^{k-1} (dpl_1[l][x] + dpr_1[r][k-1-x]) \right)$$

其分别的维护可以这么做：往前的，先钦定最后一次操作是买入 / 卖出，接下来往前遍历，买入卖出操作交替进行，看 x 次行为后的最优结果。往后的，先钦定第一次操作是买入 / 卖出，接下来往后遍历，买入卖出操作交替进行，看 x 次行为后的最优结果。

这样做复杂度如何呢？将这个过程理解为一棵递归树，则每一层的总区间长度是 $\mathcal{O}(n)$ 的，前后 DP 的过程的复杂度是 $\mathcal{O}(nk)$ 。总共的递归层数 k 需要满足 $\frac{2n}{2^k} \geq 0.15n$ ，也就是 $k = \mathcal{O}(\log \frac{2}{0.15})$ ，数量级很小。同时处理的 q 次查询，每次复杂度都是 $\mathcal{O}(k)$ ，因此完全足够完成本题。你只需跳过传统 CDQ 那些不需要考虑的区间即可。

但这题还有个 512MB 的内存限制——这意味着你不能完整存下一层的所有 DP 信息。有以下几个优化方向。

- dpl_1, dpl_0 两件事和 dpr_1, dpr_0 两件事是相互独立的，所以可以分开做：先处理 dpl_0, dpr_0 的部分，再处理 dpl_1, dpr_1 的部分。这样只需存储一半的信息。
- 右端点最多只需要取到 $0.85n$ 的位置。（比较粗略的）
- 我们可以只存左侧 / 右侧的信息，同时我们可以只存“查询需要用到的”位置的信息。而在随机数据下，你可以手动模拟看右端点可能的不同数量，大概在 62000 左右波动，设定为 63000 是充分的。

在这样的优化下，空间可以达到 200MB 左右，你只需做到以上优化的一部分就足够了。如果你的代码时间效率足够优秀，也可以选择每一轮只处理一部分查询，第二轮再重新跑剩余的查询的方式，用时间换取空间的缩小。时间复杂度为 $\mathcal{O}(nk \log \frac{1}{\epsilon} + qk)$ 。

1.2 方法二：分块

因为这里的区间查询长度很大，可以考虑把块长设置成它 $B = \lfloor 0.15n \rfloor$ 。

这样，我们可以依次处理左端点在 $[0, B), [B, 2B), \dots$ 的那些查询，也利用前面提到的前后缀拆分的方法。这样我们存储的前缀信息就只需要是 $\mathcal{O}(Bk)$ 的了。

这种做法的时间复杂度为 $\mathcal{O}(\frac{n}{B}nk + qk) = \mathcal{O}(\frac{1}{\epsilon}nk + qk)$ 。

2 更好的骰子

我们先考虑构造符合要求且点数和最小的骰子。注意到有用的面值只有 $1, a_1 + 1, a_2 + 1, a_3 + 1, a_4 + 1, a_5 + 1, a_6 + 1$ 。因此可以 7^6 暴力搜索所有可能的组合，然后检查 36 种情况即可。如果最小可能点数和比给定的要求要小，差值任意分配即可。

3 过马路

首先，你到达 $y = y_c$ 前的路径一定是直线，不然变成直线没有坏处——你总能在原地等到能过去的时刻。同时，你从 $y = y_c$ 出发后，一定垂直跑向终点。

而可能发生等待吗？答案是否定的。在这种情况下，你总可以往其中一个线段端点行进进而更早往道路另一侧出发——而你一开始不应该往你等待的位置走，至少往这一时刻的端点位置走会是更优的选择。

考虑 t 时刻你能到达的横坐标区间，是 $[-\sqrt{v^2t^2 - y_c^2}, \sqrt{v^2t^2 - y_c^2}]$ （这里认为 $t \geq y_c/v$ ），而合法区间是 $(-\infty, x_1 + ut] [x_2 + ut, +\infty)$ ，最早有非空交集的时刻一定是两种情况之一：（1） $t = y_c/v$ ，使得前者区间存在。（2）某个使得端点重叠的时刻。此时 $t = y_c/v$ 的情况下交为空。假设最早的交非空的时刻下交不止是一个端点，此时往前推一个时间 ϵ ，因为函数的连续性，一定仍然有交，因此初次非空时刻一定是某个端点重合了。

综上，要么直接垂直往终点直线跑。此时可以计算出到达 $y = y_c$ 时的线段的两个端点，判断是否合法（可以转为整数判断），如果合法，直接输出 $\frac{w}{v}$ 。

否则，一定是经过某个线段交点，即 $v^2t^2 - y_c^2 = (x_1 + ut)^2$ 或 $v^2t^2 - y_c^2 = (x_2 + ut)^2$ 。需要判断的东西包括是否是一次方程、是否存在实数根。最后取这个方程的最小非负实根。最后再加上从 $y = y_c$ 跑向终点的时间即可。时间复杂度为 $\mathcal{O}(1)$ 。

4 依旧过马路

可以变换参考系，以匀速直线运动的车为参考系。此时原本的匀速直线运动仍然是匀速直线运动。

引理 1：在这个参考系下，如果从一个点可以到达另一个点，且中间直线行走没有障碍，则直接沿着这条直线走就是最优的。

证明：忽略障碍物，记完成位移 $\Delta = (\Delta x, \Delta y)$ 的最短时间为 $\tau(\Delta)$ 。时间 $t > 0$ 可行当且仅当 $(\Delta x + ut)^2 + \Delta y^2 \leq v^2t^2$ 。时间的下界刚好由匀速直线运动给出。

引理 2：只需考虑多边形的凸包。也就是说，不可能进入多边形的凸包。

证明：假设进入了多边形的凸包，考虑某一段进来了再出去的过程，这个过程的两个端点在凸包上。则这两个端点之间，应当使用直线通过是最优的，因此，不应该进入凸包内部，矛盾。

所以，多边形的问题可以直接转化为凸多边形的问题。

引理 3：存在一条最优路径，或者直接冲线，或者形如： $S \rightarrow p_{i_1} \rightarrow p_{i_2} \rightarrow \dots \rightarrow p_{i_k} \rightarrow y = w$ ，

证明：首先，只可能在多边形的端点位置转弯。因为除凸包顶点外，每个可行位置附近的可行区域都是局部凸的，你把你的路径拉直一定更优。同时，你一旦离开了凸多边形，就一定不可能再回来，因为这

种情况下，你总能找到一段弯曲路径其端点连线不与多边形相交，将其替换为线段一定更优。你到达多边形的位置一定在端点上，不然你一定在对应位置转弯了，矛盾。离开的位置也是如此总在端点上。得证。

而同时，利用上面的证明，在凸多边形上的这段路径一定是顺时针 / 逆时针贴着多边形走的。

因此，我们可以预处理凸多边形上每个点到终点的最短时间。首先，你需要判断哪些点可以直接垂直向 $y = w$ 出发到达终点直线。此时过一个极小的 Δt 后，你相当于在车的参考系下往 $(-u, v)$ 方向走了，你只需判断这个方向是否会走向多边形内部，本质上取决于多边形上相邻的那条边的倾斜角。对于剩余的点，只需从已知的这些点出发，顺时针 / 逆时针的相邻点建边进行最短路即可。当然本质是个环形图，顺时针遍历搭配逆时针遍历也足够完成这个预处理。注意这里可能存在不可达的点对，也需要基于边的倾斜角进行处理。

于是，对于每次查询，你只需找到可达的点集，计算出发点到达这些点集的时间加上这些点集到终点的最小时间即可。而因为路径不拐弯，你一定只能取切点。所以你只需找到从起点出发的两个切点即可。这里也需要判断切点是否可达。

最后还剩下直接跑到终点直线的情况。这本质上取决于在车参考系下， $(-u, v)$ 是否会经过凸多边形，也就是是否在两个切点对应的方向之间。如果不在，则直接跑向终点直线就行，输出 $\frac{w}{v}$ 。

综上，建凸包时间复杂度为 $\mathcal{O}(n)$ ，预处理最短路复杂度为 $\mathcal{O}(n)$ ，查询每次需要寻找切点，复杂度为 $\mathcal{O}(\log n)$ ，因此总复杂度是 $\mathcal{O}(n + q \log n)$ 。

5 EPS.AC

考虑某个排序的前 k 个问题后的一个边界。一个成员支持该边界，当其排序中的前 k 个问题集合恰好是放在该边界之前的集合。因此，位置 k 后的有效边界存在的充分必要条件是，三个排序中有两个具有相同的前 k 个问题集合。

考虑给出的三个排名中的每一对，记录第二个排名中每个问题的位置。从左到右扫描第一个排名，并保持所观察到的在第二个排名中的最大位置。当这个最大值等于当前扫描到的位置 k 时，它们的前 k 个问题集完全相同。对所有三对排名中的每个这样的 k 进行标记。

注意到每个标记的边界都可以同时使用。因此答案为 $1 + \text{标记位置 } k < n \text{ 的数量}$ 。时间复杂度为 $\mathcal{O}(n)$ 。

6 夜蝶飞行

假设步长 d 时，蝴蝶在时刻 k ($k \bmod 6 = 0$) 到达主轨道的 T ，且一路上都是安全的，那么把 k 拆成 $2^a \cdot 3^b \cdot c$ 的形式，其中 $a \geq 1, b \geq 1, \gcd(c, 6) = 1$ ，构造新步长 $d' = c \cdot d$ ，则在新步长下：

- 蝴蝶在时刻 $k' = 2^a \cdot 3^b$ 也会到达 $k' \cdot d' = k \cdot d = T$ 。
- 由于 $a \geq 1$ 且 $b \geq 1$ ，所以 $k' \bmod 6 = 0$ ，说明蝴蝶最终也位于主轨道上。
- 对于时刻 $i = 1, 2, 3, \dots, k'$ ，蝴蝶到原点的距离等于旧方案中时刻 $i \cdot c$ 的位置到原点的距离。更进一步地，由于 $\gcd(c, 6) = 1$ ，所以 $6|i \iff 6|i \cdot c$ ，这说明时刻 i 蝴蝶处于主轨道还是辅轨道的情况与旧方案中时刻 $i \cdot c$ 的情况完全一致。由于旧方案一路上都是安全的，新方案一路上一定也是安全的。

所以为了求出最早到达的时刻，只需枚举所有形如 $2^a \cdot 3^b$ ($a \geq 1, b \geq 1$) 的时刻 k 作为候选答案。注意到步长 d 至少是 1 才能踏入安全区，所以枚举上界是 T 。确定 k 之后，可得步长 $d = \frac{T}{k}$ ，只需分别检查：

- 主轨道安全区间是否覆盖了 T 以内所有 $6d$ 的倍数。枚举每个区间统计倍数数量，求和后判断即可。
- 辅轨道安全区间是否覆盖了 T 以内所有 d 的倍数，其中 $6d$ 倍数的位置不做要求。转化为统计 d 的倍数数量以及 $6d$ 的倍数数量，作差判断即可。

令 T 以内候选答案的数量为 cnt (1100 个左右)，总时间复杂度为 $\mathcal{O}(q(n + m) \cdot cnt)$ 。

7 前缀扩张

以下均在模 998244353 意义下计算。由于 s_i 单调不降且 $s_i \geq i$ ，可将限制改写为

$$L_i = \max \left(i, \max_{t \leq i} l_t \right), \quad R_i = \min_{t \geq i} r_t.$$

若 $L_1 \neq 1$ 或某个 $L_i > R_i$ ，答案为 0。以下令 $U_1 = 1$ ， $U_i = R_i$ ($i > 1$)。

固定一个长度为 j 的位置区间，设 $f_i(j)$ 为将 $1, \dots, i$ 放入位置后，其包围区间恰为该区间且满足前 i 个限制的方案数。加入 i 时，它或占据原区间内的 $j - i + 1$ 个空位之一，或成为新区间的一个端点。因此

$$f_i(j) = \mathbf{1}_{L_i \leq j \leq U_i} \left((j - i + 1) f_{i-1}(j) + 2 \sum_{k < j} f_{i-1}(k) \right). \quad (1)$$

最终答案为 $f_n(n)$ ，直接转移需要 $O(n^2)$ 时间。

令 $F_i(z) = \sum_j f_i(j) z^j$ ，并定义 $Q_i(z) = F_i(z) / (1 - z)^2 = \sum_j q_i(j) z^j$ 。于是 $f_i(j) = q_i(j) - 2q_i(j - 1) + q_i(j - 2)$ 。暂不施加第 i 行的区间限制，把式 (1) 写生成函数并代入 $F_{i-1} = (1 - z)^2 Q_{i-1}$ ，前缀和项抵消，得到

$$q_i^*(j) = (j - i + 1) q_{i-1}(j). \quad (2)$$

截去 $j < L_i$ 后，有效区间内的 q_i 与 q_i^* 的二阶差分相同，故只差一次函数。记 $u = q_i^*(L_i - 1)$ ， $v = q_i^*(L_i - 2)$ ， $\delta = u - v$ ，由 $q_i(L_i - 1) = q_i(L_i - 2) = 0$ 得

$$q_i(j) = (j - i + 1) q_{i-1}(j) + a_i + b_i j \quad (L_i \leq j \leq U_i), \quad a_i = -u + (L_i - 1)\delta, \quad b_i = -\delta. \quad (3)$$

又因 $j > U_i$ 时 $f_i(j) = 0$ ，该处 $q_i(j)$ 必为一次函数 $c_i + d_i j$ ；从 $q_i(U_i - 1), q_i(U_i)$ 即可确定 c_i, d_i 。初值 $q_1(j) = j$ 。固定 j ，式 (3) 乘以 $(j - i)!$ 后可以累加。设 $C_i(j) = \sum_{t < i} (a_t + b_t j)(j - t)!$ ，负下标阶乘对应的项视为 0。令 e_j 为 j 首次进入上界 U_i 的时刻，并保存 $H_j = (j - e_j + 1)! q_{e_j-1}(j) - C_{e_j}(j)$ 。在 $L_k \leq j \leq U_k$ 时便有

$$q_k(j) = (H_j + C_{k+1}(j))(j - k)!^{-1}. \quad (4)$$

而 $j < L_k$ 时 $q_k(j) = 0$ ， $j > U_k$ 时使用右侧的一次函数。每行仅需查询 $L_i - 2, L_i - 1, U_i - 1, U_i$ ，以及新进入 U_i 的位置，以求出式 (3) 的系数和下一行所需的边界值。

对时间下标做 CDQ 分治：先求左半区间的 a_t, b_t ，再将它们对右半区间所需的 $C_i(j)$ 的贡献批量加入，最后求右半区间。一次批量计算的形式为

$$\sum_{t=l}^m (a_t + b_t j)(j - t)! = \sum_{t=l}^m a_t (j - t)! + j \sum_{t=l}^m b_t (j - t)!.$$

右侧两项分别是系数序列与阶乘序列的卷积，可用 NTT 求出。由于 L_i, U_i 单调，所需的查询位置在每个分治区间内构成常数段连续值域；每层的卷积总长度为 $O(n)$ 。故总时间复杂度为 $O(n \log^2 n)$ ，空间复杂度为 $O(n)$ 。可行时 $L_n = U_n = n$ ，从二阶差分式可知答案为 $q_n(n)$ 。

8 隐藏的排列

通信题，做法很多。一种可能的做法是第一轮输出原排列及每个 p_i 分别和 p_1 交换的排列。第二轮恢复的时候 $p_2, \dots, p_n$ 只需要找对应列的众数， p_1 是剩下没有被使用的数字（或在每列都出现了的数字）。

9 整数三元组

注意到表达式右侧总是为偶数，因此 n 为奇数时无解，偶数时一种构造是 $1, -1, n/2$ 。

10 流水线作业

假设无法进行循环位移以及分段操作，对于第一道工序，它在第 j 条流水线上占据的时间段是 $[(j-1) \cdot a_1, j \cdot a_1]$ 。每条流水线和前一条流水线之间的时间差是 a_1 。从左往右依次考虑后面的每道工序，设当前相邻两条流水线的完成时间差为 T ，加入新工序 a_i 后，如果 $a_i \leq T$ ，完成时间差仍为 T ；如果 $a_i > T$ ，完成时间差变为 a_i 。因此最终相邻流水线完成时间差始终等于已处理工序中的最大值。而第一条流水线执行完毕的时刻是 $\sum a_i$ ，所以第 m 条流水线执行完毕的时刻是 $\sum a_i + (m-1) \cdot \max\{a_i\}$ 。

接下来引入循环位移以及分段操作。由于无论怎么操作， $\sum a_i$ 是定值，等价于把一个环分成不超过 k 段，最小化每段内数字之和的最大值。

二分答案，转化为能否把环分成不超过 k 段，使得每一段的总和都不超过 mid 。对于每个起点 i ，通过双指针可以预处理出以 i 为起点，顺时针能扩展到的最远位置，满足途经部分的数字之和不超过 mid ，记作 f_i ，连边 $i \rightarrow f_i$ 。只需检查 i 的 k 级后继是否覆盖了一圈即可。破环成链后 DFS 求树上 k 级祖先即可 $\mathcal{O}(n)$ 检查。

在这里也可以直接 $\mathcal{O}(n)$ 求出最少划分段数，然后与 k 进行比较。把每个连通块看作环套树，那么任选一个点走 n 步一定会落到环内，求出此时的环长就是最少段数。这是因为如果一个点 x 是某个最优方案的分割点，那么它的所有后继 $f(f(\dots f(f(x))))$ 一定也是，这说明一个环要么都是最优点，要么都不是最优点。对于长度为 L 的环 $\{p_1, p_2, \dots, p_L\}$ ，如果它不是最优环，那么最优环的长度小于 L ，不失一般性可以认为最优起点位于 (p_1, p_2) 之间，那么它的后继只能位于 (p_2, p_3) 之间，因为如果恰好是 p_2 或者 p_3 ，就会合并成一个环；如果未到 p_2 ，那么调整为当前环更优；如果超出了 p_3 ，则违反 $f(\cdot)$ 的单调性。而它的后面几步也只能每次跳到下一段间隙之中，跳了恰好 L 步才能覆盖一圈，与它的长度小于 L 矛盾。所以任取一个环一定是最优环。总时间复杂度 $\mathcal{O}(n \log ans)$ 。

11 走上正轨

从 t 开始拓扑排序 DP，计算出每个节点到 t 的最短路条数，再计算期望，根据节点是否有信号决定转移的系数即可（没信号的时候均分，有信号的时候从所有最短路中按照条数加权分配）。

12 压缩日志

假设只有一条命令，那么程序 $[i, \lfloor \frac{n}{i} \rfloor]$ 的出现次数都是 $\lfloor \frac{n}{i} \rfloor$ ，可以表示为一个二元组 (w, c) 表示有 c 种编号连续的程序的出现次数都是 w 。根据数论分块的性质，一共可以整理出 $\mathcal{O}(\sqrt{n})$ 个二元组。

对于有 m 条命令的情况，它会将所有程序划分为 $\mathcal{O}(m)$ 个连续段。对每条命令区间做差分得到每个程序被启动的次数，再与数论分块段做扫描线相交，最终会得到 $\mathcal{O}(\sqrt{n} + m)$ 个二元组。

首先特判 $n = 1$ 的情况，因为二进制编码不能为空，所以答案为 m 。

对于 $n > 1$ 的情况，最优解一定是哈夫曼编码，将所有 n 个程序视作 n 个带权节点，权值为其出现次数，即使出现次数为 0 也不能忽略。每当存在至少两个节点时，将权值最小的两个节点合并为一个节点，合并后的节点权值为这两个节点的权值之和。

针对二元组的压缩表示，首先将所有二元组按照 w 从小到大排序，记作序列 P ；再维护一个二元组队列 Q ，一开始为空。由于每次选出的最小值与次小值单调不递减，合并产生的新节点的权值一定单调不递减，这说明最小值一定来自 P 或 Q 的队首。取出权值最小的二元组 (w, c) ：

- 如果 $c \geq 2$ ，那么内部两两合并后，会产生一个新的二元组 $(2w, \lfloor \frac{c}{2} \rfloor)$ ，将其加入 Q 的队尾。
- 如果 $c \bmod 2 = 1$ ，那么还有一个节点需要与次小值合并，处理方式显然。

一共有 $\mathcal{O}(\sqrt{n} + m)$ 个二元组，每个二元组最多入队 $\mathcal{O}(\log n)$ 次，总时间复杂度 $\mathcal{O}((\sqrt{n} + m) \log n)$ 。

13 中位数为 X

不妨用中位数不超过 x 的子序列个数，减去中位数不超过 $x - \frac{1}{2}$ 的子序列个数。也就是我们把问题转化为了中位数不超过 V 的子序列个数 $F(V)$ 。

将数组进行排序。如果选取的子序列长度是奇数，则枚举子序列选取的中位数位置，不妨设为第 i 个元素，则我们在 i 前和 i 后分别需要选取相同个数的元素，因此求和结果是：

$$\sum_{x=0}^{i-1} \binom{i-1}{x} \binom{n-i}{x} = \sum_{x=0}^{i-1} \binom{i-1}{x} \binom{n-i}{n-i-x} = \binom{n-1}{n-i}$$

如果选取的子序列长度是偶数，则枚举中位数中较小的那个元素，同时，中位数中较大的那个元素需要满足和较小元素之和大于等于 $2V$ ，且较小的元素更靠后。这个可以用一个指针维护。

此时设较小元素位于 i ，较大元素的位置下界是 j 。枚举较小元素前选取的元素个数 x ，求和结果是：

$$\sum_{x=0}^{i-1} \binom{i-1}{x} \binom{n-j+1}{x+1} = \sum_{x=0}^{i-1} \binom{i-1}{x} \binom{n-j+1}{n-j-x} = \binom{n-j+i}{n-j}$$

上述两种情况相加即可得到 $F(V)$ 。用 $F(x) - F(x - \frac{1}{2})$ 即得到答案。

时间复杂度为 $\mathcal{O}(n \log n)$ 。

14 CCPC 新赛制

按题意模拟即可。