IOI2020 中国国家集训队第一阶段作业 试题准备

江苏省常州高级中学 徐翊轩

November 18, 2019

Contents

1 Codeforces 573E Bear and Bowling

1.1 题目大意

题目来源：https://codeforces.com/contest/573/problem/E

给定一个长度为 N 的整数序列 $a_1, a_2, \ldots, a_N$ ，序列中可能包含负数。要求选定一个可以为空的子序列 $b_1, b_2, \ldots, b_M$ ，最大化函数 $\sum_{i=1}^{M} i \cdot b_i$ ，求出此时该函数的值。

1.2 数据范围

对于所有的数据，保证 $1 \leq N \leq 10^5, |a_i| \leq 10^7$ 。

时间限制 6s ，空间限制 256MB 。

1.3 算法标签

贪心、分块、凸包、动态规划、平衡树

1.4 解题过程

1.4.1 贪心思路

由于这是一个在序列上的最优化问题，采用贪心的策略往往能够得到不错的结果。

考虑依次向已经选择的子序列中加入一个数 a_i 的过程，如果在 i 的左侧已经有 $k_i - 1$ 个数被选取，并且 i 的右侧被选取的数之和为 suf_i ，那么，加入 a_i 将会带来 $k_i \cdot a_i + suf_i$ 的贡献，不妨记加入 a_i 的贡献为 Q_i 。

一个自然的贪心算法就出现了：我们每次选择 Q_i 最大的 a_i 加入答案，同时更新其余尚未加入答案的 Q_j ，直到所有元素都被加入答案，取贡献总和的历史最大值作为答案。

在下一节中，我们将证明这个算法的正确性。

1.4.2 正确性证明

为了证明上述算法的正确性，我们首先需要证明如下引理。

引理 1：如果 $a_i > a_j$ ，并且 $i < j$ ，那么 a_i 一定比 a_j 先被选取。

证明：考虑上述引理首次不满足位置，即使得二元组 (i, j) 字典序最小的不满足的位置。如果 a_i, a_j 之间没有元素被选中，那么 $k_i = k_j, suf_i = suf_j$ ，因此 $Q_i = k_i \cdot a_i + suf_i > Q_j = k_j \cdot a_j + suf_j$ ，a_i 会先被选取。对于 a_i, a_j 之间每一个已经选中的元素 a_x ，其对 Q_i 的贡献为 a_x ，对 Q_j 的贡献为 a_j 。又因为 (i, j) 是引理首次不满足位置，而 $i < x$ ，因此必然有 $a_x \geq a_i$ ，从而 $a_x > a_j$ ，Q_i 的增加量始终大于 Q_j 的增加量，所以 $Q_i > Q_j$ ，引理得证。

定理 1：每次选择 Q_i 最大的 a_i 加入答案，同时更新其余尚未加入答案的 Q_j ，得到的每一个大小的序列都是对应大小的最优解。

证明：假设定理不成立，那么，必然存在一个最早的时刻，使得我们选择了某个元素 a_i ，此后，对于某个大小 s ，我们无法通过选择更多的元素来得到 s 的最优解，无论我们在选择更多元素时使用的策略如何。令集合 A 表示该时刻之前已经选取的元素集合，那么，集合 $A \cup \{a_j\}$ 不是任何一个大小为 s 的最优解的子集。但是，存在一个集合 B $(A \cap B = \emptyset)$ ，使得 $|A \cup B| = s$ ，并且，$|A \cup B|$ 是一个大小为 s 的最优解，注意 B 应当不为空。

情况 1：B 中包含至少一个 a_i 左侧的元素

令 a_j 表示 B 中最靠右侧的一个使得 $j < i$ 的元素。由于我们的贪心策略会首先选取 a_i ，由引理 1 ，我们知道 $a_i \geq a_j$ 。我们将证明将 $A \cup B$ 中的 a_j 替换为 a_i 不会使得方案变得更劣，从而选择 a_i 是合理的。

在选择 a_i 时，a_i 具有最大的 Q_i ，因此 $Q_i \geq Q_j$ 。考虑将 B 中除 a_j 以外的元素加入 A 中，并比较此时 Q_i, Q_j 的大小。对于每一个 a_x $(x > i)$ ，其对 Q_i, Q_j 的贡献均为 a_x ，对于每一个 a_x $(x < j)$ ，其对 Q_i 的贡献均为 a_i ，对 Q_j 的贡献均为 a_j ，并且，由 a_j 的选择方式，不存在 $j < x < i$ 的 a_x 。因此，Q_i 的增加量始终大于 Q_j 的增加量，$Q_i > Q_j$ ，将 $A \cup B$ 中的 a_j 替换为 a_i 不会使得方案变得更劣。

情况 2：B 中的所有元素都在 a_i 的右侧

令 a_j 表示 B 中最靠左侧的一个元素，由于 B 中的所有元素都在 a_i 的右侧，有 $j > i$ 。我们将证明将 $A \cup B$ 中的 a_j 替换为 a_i 不会使得方案变得更劣，从而选择 a_i 是合理的。

在选择 a_i 时，a_i 具有最大的 Q_i ，因此 $Q_i \geq Q_j$ 。同样考虑将 B 中除 a_j 以外的元素加入 A 中，并比较此时 Q_i, Q_j 的大小。对于每一个 a_x $(x > j)$ ，其对 Q_i, Q_j 的贡献均为 a_x ，并且，由 a_j 的选择方式，不存在 $x < j$ 的 a_x 。因此，Q_i 的增加量始终大于 Q_j 的增加量，$Q_i > Q_j$ ，将 $A \cup B$ 中的 a_j 替换为 a_i 不会使得方案变得更劣。

综上所述，假设不成立，从而定理 1 得证。

定理 1 的成立直接保证了上述贪策略的正确性，从而也导出了一个直接应用上述贪心策略的 $O(N^2)$ 算法，在下一节中，我们会讨论如何快速实现这个算法。

1.4.3 分块解法 $O(N\sqrt{N}LogN)$

考虑向答案中加入 a_i 对 Q 数组的影响，对于 $j < i$ ，a_i 对 Q_j 的贡献为 a_i ，而对于 $j > i$ ，a_i 对 Q_j 的贡献为 a_j ，因此，我们需要一个能够支持区间加，区间加对应 a_i ，以及询问全局最大值的数据结构。

考虑用分块维护这个结构，我们将每 $O(\sqrt{N})$ 个元素分为一块，在向答案中加入 a_i 时，对于 a_i 所在块左侧的块，进行整块加常数操作；对于 a_i 所在块右侧的块，进行整块加对应 a_i 操作，由此，我们需要在改变 k 的情况下维护若干一次函数 $k \cdot a_i + b_i$ 的最大值，可以通过在凸包上二分解决；对于 a_i 所在的块，可以暴力重建凸包。

每次向答案中加入 a_i 后，我们都需要重构一个大小为 $O(\sqrt{N})$ 的凸包，并且，在 $O(\sqrt{N})$ 个凸包上进行二分，这两部分的总时间复杂度均为 $O(N\sqrt{N}LogN)$ 。

参考程序：https://codeforces.com/contest/573/submission/63051778

1.4.4 算法优化 $O(N\sqrt{N}+NLogN)$

我们可以将上述算法的 $O(LogN)$ 因子优化掉。

首先，由于 $k\cdot a_i+b_i$ 中的系数 k 是不断增加的，我们可以用单调指针代替二分，其均摊复杂度是 $O(1)$ 的。其次，在重构凸包时，我们不需要每次都对点集进行排序，注意到 a_i 是始终不变的，我们只需要在刚开始对 a_i 进行一次排序即可。

由此，上述算法的时间复杂度被优化至了 $O(N\sqrt{N}+NLogN)$ 。

参考程序：https://codeforces.com/contest/573/submission/63051927

1.4.5 平衡树解法 $O(NLogN)$

考虑一个 $O(N^2)$ 的动态规划解法，记 $dp_{i,j}$ 表示在序列的前 i 个元素中选择恰好 j 个，可以产生的最大贡献，转移时，考虑是否选取 a_i ，则有

$$dp_{i,j}=\max\{dp_{i-1,j},dp_{i-1,j-1}+j\cdot a_i\}$$

其中 $dp_{i-1,j}$ 表示不取 a_i ，$dp_{i-1,j-1}+j\cdot a_i$ 表示选取 a_i 。

不妨记 $S_{i,j}$ 表示一种使得贡献取到 $dp_{i,j}$ 的选取方案集合。

注意到由定理 1 ，$S_{i,j+1}$ 可以通过向 $S_{i,j}$ 中加入一个元素得到，存在最优解 $S_{i,j}\subset S_{i,j+1}$ ，因此，如果 $a_i\in S_{i,j}$ ，则有 $a_i\in S_{i,j+1}$ 。

这意味着**使得 $dp_{i-1,j-1}+j\cdot a_i\geq dp_{i-1,j}$ 的 j 是一段连续的后缀**。

考虑用平衡树维护 dp 数组每一行的差分数组，在从 dp_{i-1} 计算 dp_i 时，可以首先二分得出转移的分界位置，然后对分界位置后方的元素整体 $+a_i$ ，并在分界位置插入一个新的元素。

时间复杂度 $O(NLogN)$ ，以下代码采用旋转式 Treap 实现了该算法。

参考程序：https://codeforces.com/contest/573/submission/63053271

1.5 数据生成

1.5.1 随机数据

测试点 $1\sim25$ 是随机生成的，其中：

测试点 $1\sim10$ 首先随机生成了一个 $10\%\sim90\%$ 之间的比例 r ，此后，输入的每个数 a_i 会以 r 的概率为正，$1-r$ 的概率为负，其绝对值将在 $0\sim10^7$ 内等概率随机。本部分数据可以用于区分完全错误的算法，以及时间复杂度或常数过高的算法。

测试点 $11\sim21$ 同样首先随机生成了一个 $10\%\sim90\%$ 之间的比例 r ，此后，输入的每个数 a_i 会以 r 的概率为正，$1-r$ 的概率为负，其绝对值将在 $0\sim\frac{10^7}{N}(N-i+1)$ 内等概率随机。本部分数据中，各个数值可能的范围线性递减，相比于测试点 $1\sim10$ ，对正确的决策要求更高。

测试点 $21\sim25$ 同样首先随机生成了一个 $10\%\sim90\%$ 之间的比例 r ，此后，输入的每个数 a_i 会以 r 的概率为正，$1-r$ 的概率为负，其绝对值将在 $0\sim\frac{10^7}{i}$ 内等概率随

机。本部分数据中，各个数值可能的范围以反比例函数递减，相比于测试点 $11 \sim 21$，在另一个角度对决策的正确性做出了要求。

测试点 $1 \sim 25$ 主要可以区分完全错误的算法，以及无法在时空限制内通过的算法。

1.5.2 边界情况

测试点 $26 \sim 30$ 是根据一些边界情况生成的，其中：

测试点 $26 \sim 28$ 首先随机生成了一个 $0 \sim N$ 之间的数 r，此后，输入的每个数 a_i 为负，当且仅当 $i \leq r$，其绝对值将在 $0 \sim 10^7$ 内等概率随机。本部分数据可以用于将答案卡至很大，并且保证动态规划的两种转移的个数均取到 $O(N^2)$ 级别。

测试点 29 中，所有的 a_i 均为 -10^7，该数据可以检测选手是否考虑了不选任何数的情况。

测试点 30 中，所有的 a_i 均为 10^7，该数据将答案卡至最大，检查溢出问题。

测试点 $26 \sim 30$ 主要包含了包括答案溢出、转移次数在内的一些边界情况。

1.5.3 针对性构造

测试点 $31 \sim 35$ 分别针对了 Codeforces 上的若干错误解法进行了构造。

错误解法 1：https://codeforces.com/contest/573/submission/27837319

这是一份明显错误的贪心代码，即使很小的数据也可以让它出错，但它通过了 Codeforces 上所有的测试数据。测试点 31 是针对该算法的 $N = 10$ 的构造。

错误解法 2：https://codeforces.com/contest/573/submission/27838890

错误解法 3：https://codeforces.com/contest/573/submission/51127450

这是两份贪心的代码，思想与上面一份代码并不相同，其中一份还包含了多次贪心的策略，可以通过构造许多相近的数字可以使它们提前停止，导致与正确答案相差很大。测试点 $32, 33$ 分别是针对该算法的 $N = 1000, N = 100$ 的构造，测试点 32 同样卡掉了许多已经通过 Codeforces 上所有的测试数据的错误代码。

错误解法 4：https://codeforces.com/contest/573/submission/20924159

错误解法 5：https://codeforces.com/contest/573/submission/12773249

错误解法 6：https://codeforces.com/contest/573/submission/12856587

以上三份代码是测试点 $31 \sim 33$ 无法卡掉的代码，其使用的策略分别为贪心、以及随机化 + 卡时。其中错误解法 4 同样可以通过 Codeforces 上所有的测试数据。测试点 $34, 35$ 中，$N = 10^5$，正数出现较少，大约为 1%，并且正数的绝对值与负数差距很大，贪心和随机化难以得到最优解，可以将上述做法全部卡掉。

经过本部分的构造后，几乎所有的在 Codeforces 的错误代码，以及那些通过了 Codeforces 测试数据的错误贪心，包括许多红名用户的错误解法均无法通过本题的数据。值得一提的是，本场比赛当时通过本题的唯一一个提交同样是无法通过测试点 $31 \sim 35$ 的错误代码：

提交链接：https://codeforces.com/contest/573/submission/12759251

1.6 试题总结

本题由一个贪心的思想出发，并通过说理证明了它的正确性，随后，便自然地导出了出题人期望的分块 + 凸包的解法。是一道兼具思维和代码难度的好题。

笔者没有拘泥于出题人的解法，而是继续思考，得出了一个复杂度更优的平衡树解法，从而可以在时限内通过十倍于原题数据规模的数据。

在数据生成方面，笔者进行了精心的构造，**卡掉了 Codeforces 上包括许多已经通过的贪心、卡时解法在内的几乎全部错误解法**。

2 Codeforces 704E Iron Man

题目来源：https://codeforces.com/contest/704/problem/E

2.1 题目大意

给定一棵 N 个节点的树，以及树上的 M 条路径，第 i 条路径 (v_i, u_i) 具有属性 t_i, c_i ，表示一个在时刻 t_i 出现在节点 v_i ，以 c_i 条边每秒的速度沿最短路走向 u_i ，并在到达 u_i 后消失的物体。如果两个物体在某一时刻处在相同的位置，则会发生爆炸，这里的位置也可以是某一条边上。

要求求出第一次爆炸的时刻，或者判断不会发生爆炸。

2.2 数据范围

对于所有的数据，保证 $1 \leq N, M \leq 10^5$ 。

保证 $0 \leq t_i \leq 10^4, 1 \leq c_i \leq 10^4, 1 \leq v_i, u_i \leq N$ 。

时间限制 5s ，空间限制 256MB 。

2.3 算法标签

树链剖分、扫描线、平衡树、实数处理

2.4 解题过程

2.4.1 链上解法

首先，我们可以考虑一下如何在链上解决这个问题。

引理 1：在第一次爆炸前，除物体出现和消失的时刻外，物体位置的相对顺序不变。

证明：由于物体的移动是连续的，因此不存在在不产生爆炸，且没有物体消失和出现的情况下改变物体位置相对顺序的情况。

那么，由引理 1，可以考虑用平衡树维护物体位置的相对顺序，通过扫描线算法依次处理物体出现和消失的事件，直到第一次爆炸产生。

具体来说，维护一棵按照物体坐标排序物体的平衡树 T，以及变量 $goal$，代表在当前计算下，第一次爆炸出现的时刻。初始时 $goal = +\infty$，并会随着我们的计算不断更新。

对于物体 x 出现的事件，在 T 中插入 x，并根据 x 与前驱和后继两对相邻关系更新 $goal$；对于物体 x 消失的事件，从 T 中删除 x，并根据 x 原有的前驱与后继这一对相邻关系更新 $goal$。直到处理完所有的事件，或者下一个事件的时刻超过 $goal$，结束计算，此时的 $goal$ 即为答案。

该链上算法可以在 $O(MLogM)$ 的时间内得出第一次爆炸出现的时刻。

2.4.2 树链剖分

回到原问题，既然我们已经有了链上的解法，不难用树链剖分将其拓展至树上。

对原树进行轻重链剖分，那么，每一条路径应当经过至多 $O(LogN)$ 条重链、$O(LogN)$ 条轻边。将一条路径拆分为 $O(LogN)$ 段，在每一条重链和轻边上分别进行上面的扫描线算法，并将所得答案取最小值即可。

时间复杂度 $O(N + MLog^2N)$，可以在时限内通过本题。

注意在拆分路径时，我们可能会遇到一些实数精度的问题。考虑到本题中，涉及除法的地方不多，分子和分母的乘积始终在六十四位整型范围内，以下参考程序通过手写有理数类避免了实数运算，同时确保了所造数据的正确性。

参考程序：https://codeforces.com/contest/704/submission/63060009

2.5 数据生成

2.5.1 边界数据

测试点 $1 \sim 10$ 是 Codeforces 中原有的十个较小的数据。

本部分数据涵盖了在节点处爆炸、不产生爆炸、答案超过 10^4 等多种边界情况，可以对程序实现细节的正确性作出检查。

2.5.2 随机数据

测试点 $11 \sim 50$ 是随机生成的，其中：

测试点 $11 \sim 20$ 中，树的形态是随机生成的；

测试点 $21 \sim 30$ 中，树的形态是一条链；

测试点 $31 \sim 40$ 中，树的形态是一条 $\frac{N}{2}$ 个点的链 + 一个菊花图；

测试点 $41 \sim 50$ 中，树的形态是在一条 $\frac{N}{2}$ 个点的链随机挂点生成的。

询问的生成方式根据测试点编号的个位数按照一定方式随机生成。

实际上，测试点 $11 \sim 50$ 已经具备了卡掉多数错误算法的能力。

2.5.3 溢出构造

测试点 $51 \sim 55$ 对应了 t_i, c_i 较大的情况，可以检查可能出现的整型溢出问题。

例如我校另一位集训队选手的代码：

https://codeforces.com/contest/704/submission/55959881

若在程序内的判断语句使用乘法，则可能由于不注意溢出事项而出错。

2.6 试题总结

本题是一道解法自然的数据结构题，考察选手树链剖分、扫描线等基本算法，难度偏向于实现方面。在数据生成方面，笔者沿用了 Codeforces 的若干数据，并生成了形式多样的随机数据。

3 AtCoder Regular Contest 103D Robot Arms

题目来源：https://atcoder.jp/contests/arc103/tasks/arc103_b

3.1 题目大意

给定平面上 N 个点 (X_i, Y_i)，要求构造长度大小为 M 的整数数组 d_i，以及 N 个长度为 M 的，仅由 L, R, U, D 组成的字符串 w_i，需要满足如下条件：

$1 \le M \le 40, 1 \le d_i \le 10^{12}$。

对于 $i = 1, 2, \ldots, N$，令 $(x_{i,0}, y_{i,0}) = (0, 0)$；

若 $w_{i,j} = L$，$(x_{i,j}, y_{i,j}) = (x_{i,j-1} - d_j, y_{i,j-1})$；

若 $w_{i,j} = R$，$(x_{i,j}, y_{i,j}) = (x_{i,j-1} + d_j, y_{i,j-1})$；

若 $w_{i,j} = D$，$(x_{i,j}, y_{i,j}) = (x_{i,j-1}, y_{i,j-1} - d_j)$；

若 $w_{i,j} = U$，$(x_{i,j}, y_{i,j}) = (x_{i,j-1}, y_{i,j-1} + d_j)$。

$(x_{i,N}, y_{i,N}) = (X_i, Y_i)$。

给出任意一组构造，或者判断无解。

3.2 数据范围

对于所有数据，保证输入的数均为整数，且 $1 \le N \le 1000$，$|X_i|, |Y_i| \le 10^9$。

时间限制 2s，空间限制 1024MB。

3.3 算法标签

二分图思想、构造算法

3.4 解题过程

3.4.1 判断无解

首先，我们可以初步考虑无解的情况。

引理 1：若将各个点 (x,y) 按照 $x+y$ 的奇偶性黑白染色，则对于确定的 d_i 数组，无论选择何种 w_i ，至多只能到达一种颜色的点。

证明：由染色的方式，若 d_i 为偶数，则无论走向 L,R,U,D 中的哪一个方向，到达的点均与起始点同色；而若 d_i 为奇数，则无论走向 L,R,U,D 中的哪一个方向，到达的点均与起始点异色。

引理 1 给出了一个无解的充分条件，即存在颜色不同的点。

通过下一节的构造算法，我们可以看到，若引理 1 不能判断输入无解，我们都可以通过该算法构造出一组解，因此这个条件同样是必要的。

3.4.2 构造算法

由于题目对 M 的限制很紧，大约只有 $1\times log_2(|X_i|+|Y_i|)$ ，可能的构造方式实际上不多。为了在限制内完成构造，可以想到采用 $d=\{1,2,4,8,\dots\}$ 的构造方式。

记 D_i 表示 $\{2^0,2^1,\dots,2^i\}$ ，E_i 表示在 D_i 中多加入一个 2^0 得到的集合。

记 S_i 表示对于所有的 w 的设置方案，D_i 可以达到的点集。

记 T_i 表示对于所有的 w 的设置方案，E_i 可以达到的点集。

定理 1：

$(x,y)\in S_i$ ，当且仅当 $|x|+|y|\equiv 1\ (mod\ 2)$ 且 $|x|+|y|\le 2^{i+1}$ ；

$(x,y)\in T_i$ ，当且仅当 $|x|+|y|\equiv 0\ (mod\ 2)$ 且 $|x|+|y|\le 2^{i+1}$ 。

证明：由引理 1 的证明，定理 1 的必要性是显然的，考虑证明其充分性。

考虑归纳法，验证得 $i\le 1$ 时，定理 1 成立，因此，我们只需要在定理 1 对 $i-1$ 成立的条件下证明定理 1 对 i $(i\ge 2)$ 成立。

由于问题在四个象限是对称的，不失一般性地，考虑 $x,y\ge 0$ 的情况。

因为 $x,y\ge 0$ ，$x+y\equiv 1\ (mod\ 2)$ 且 $x+y\le 2^{i+1}$ ，$x\ge 2^i$ 与 $y\ge 2^i$ 中至少有一者成立，不失一般性地，令 $x\ge 2^i$ 。

那么，$x-2^i\ge 0$ ，并且 $x-2^i+y\equiv 1\ (mod\ 2), x-2^i+y\le 2^i$ 。

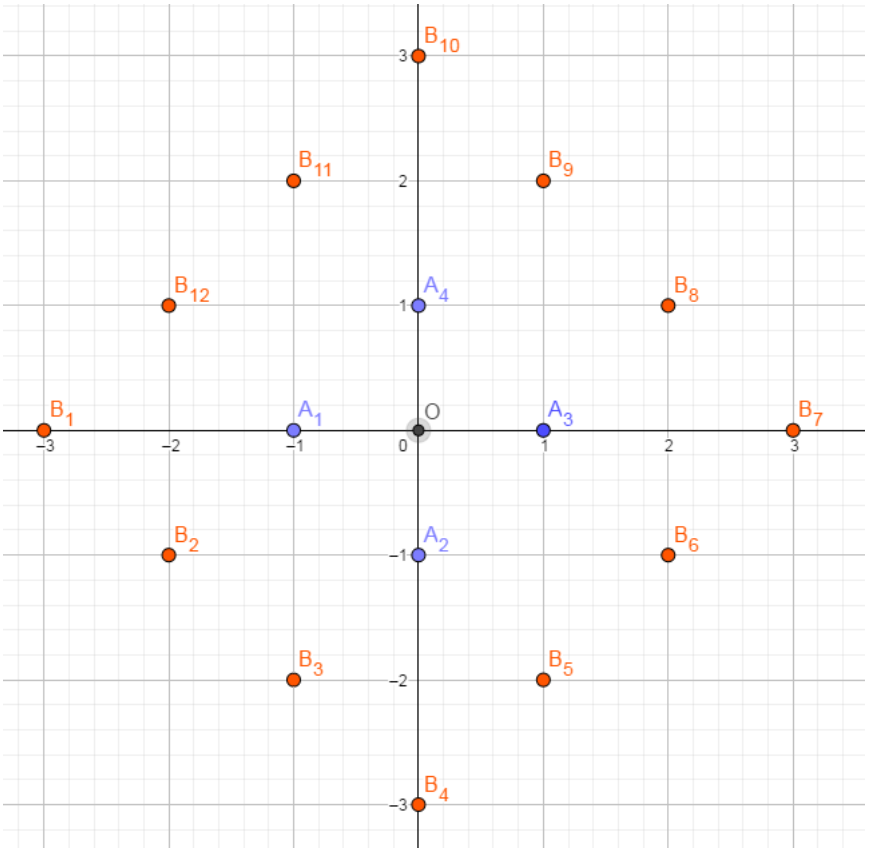
因此 $(x-2^i,y)\in S_{i-1}$ ，而 $D_i=D_{i-1}\cup\{2^i\}$ ，从而 $(x,y)\in S_i$ 。

同样地，因为 $x,y\ge 0$ ，$x+y\equiv 0\ (mod\ 2)$ 且 $x+y\le 2^{i+1}$ ，$x\ge 2^i$ 与 $y\ge 2^i$ 中至少有一者成立，不失一般性地，令 $x\ge 2^i$ 。

那么，$x-2^i \geq 0$ ，并且 $x-2^i+y \equiv 0\ (mod\ 2), x-2^i+y \leq 2^i$ 。

因此 $(x-2^i, y) \in T_{i-1}$ ，而 $E_i = E_{i-1} \cup \{2^i\}$ ，从而 $(x, y) \in T_i$ 。

读者可以通过下图辅助理解以上证明过程。



由定理 1 的证明，对于不能判断无解的输入，我们也可以得到一个可行的构造。

时间复杂度 $O(N \times M)$ ，其中 $M = 40$ 。

参考程序：https://atcoder.jp/contests/arc103/submissions/8009289

3.5 数据生成

由于本题不存在针对随机数据的特殊解法，在保证点的颜色[1]的情况下，随机数据的强度是足够的。本题的测试数据中：

测试点 $1 \sim 3$ 包含了 $N = 1, 2$ 的特殊情况。

测试点 $4 \sim 15$ 中，$N = 10^3$ ，其中：

测试点 $4 \sim 5$ 的输入完全随机生成，对应了无解的情况。

测试点 $6 \sim 10$ 的输入在保证输入点的颜色为黑的前提下随机生成。

测试点 $11 \sim 15$ 的输入在保证输入点的颜色为白的前提下随机生成。

[1]指引理 1 中“将各个点 (x, y) 按照 $x+y$ 的奇偶性黑白染色”中的颜色

3.6 试题总结

本题是一道有趣的构造题，难度不高，但构思精巧。

本题的数据采用随机的方式生成，但由于题目本身的性质，其强度是有保证的。

参考文献

[1] Codeforces Round #318 [RussianCodeCup Thanks-Round] Editorial
https://codeforces.com/blog/entry/20040

[2] Codeforces Round #366 Editorial
https://codeforces.com/blog/entry/46450

[3] AtCoder Regular Contest 103 解説
https://img.atcoder.jp/arc103/editorial.pdf