在确定性和非确定性查询复杂度之间

Dániel Gerbner

Alfréd Rényi Institute of Mathematics, Hungarian Academy of Sciences

P.O.B. 127, Budapest H-1364, Hungary.

gerbner@renyi.hu

2025 年 4 月 13 日

摘要

我们考虑可以通过询问某些查询来解决的问题。一个问题 P 的确定性查询复杂度 $D(P,n)$ 是在最坏情况下，为了找到大小为 n 的输入的解所需的最小查询次数，而非确定性查询复杂度 $D_0(P,n)$ 则是在已知解的情况下，为了证明它确实是解所需询问的最大查询次数（在最坏情况下）。等价地，$D(P,n)$ 是在对抗者回答查询时找到解所需的最大查询次数，而 $D_0(P,n)$ 是在对抗者选择输入时找到解所需的最大查询次数。我们在这两个值之间定义了一系列的数量，$D_k(P,n)$ 是在对手选择输入并回答查询的情况下找到解决方案所需的最大查询次数，但他可以改变输入最多 k 次。我们给出了各种问题 P 上的 $D_k(P)$ 的界限。

1 介绍

在这篇论文中，我们考虑可以通过询问某些查询来解决的问题。松散地说，一个问题由一个函数和某种允许的查询描述组成。我们给定一个函数 $f : X \to Y$。X 是输入集合，我们的目标是确定未知的 $x \in X$ 的 $f(x)$ 值。为了实现这一点，我们可以询问某些类型的查询。我们需要构建一个算法，即一种策略来描述一系列查询和答案之后的下一个查询是什么。这可以通过一个决策树表示，即一棵树，其中顶点是潜在的查询，第一个查询是根节点，查询的所有可能答案由从相应顶点出发的边表示，并且在边的另一端是我们要进行的下一个查询。

最常见的目标是找到能够最小化问题 P 的最坏情况查询复杂度 $D(P,n)$ 的算法，即算法需要询问的最大查询次数以解决大小为 n 的输入问题。

一个有用的方法是将问题视为两个玩家之间的游戏，其中一个玩家提出查询并试图尽快解决问题（我们将他称为*提问者*），另一个玩家通常被称为*对手*，他回答这些问题并试图推迟解决方案的得出。对抗者不必选择输入元素 $x \in X$，而是可以任意作答。然而，他的答案必须与至少一个输入一致。如果提问者能够找到 f 的值，并假设答案与至少一个输入一致，则游戏结束。

假设我们可以为对手找到一种算法，使得无论提问者采用什么策略，在游戏结束前至少需要提出 m 次查询。那么很容易看出 $D(P,n) \geq m$。确实，对于提问者的任何特定策略而言，对手的回答与某个输入 x 一致。这意味着特定的策略如果需要在 x 恰好是输入的情况下至少提出 m 次查询，并且这一点对每个策略都成立，因此 $D(P,n) \geq m$。

另一方面，即使是对手的最佳策略也无法迫使提问者提出超过 $D(P,n)$ 个查询。

上面我们寻找的是能够最小化最坏输入所需查询次数的算法，即在所有算法中选择一个使得最大（在所有输入中的）解决问题所需的查询次数最小的算法。也可以转而寻找最大（在所有输入中）的最小（在所有算法中）所需的查询次数。换句话说，之前我们是基于最坏输入来评估算法，并选取最佳算法，但现在我们将基于最佳算法来评估输入，然后选取最坏输入。

一个更典型的解决方法是，我们可以猜测输入 x，但仍然需要证明 f 的值确实是 $f(x)$。我们对 x 的了解仅帮助我们选择要问的问题。这被称为问题的*非确定性*版本。在决策树中，这意味着我们需要寻找的是从根到叶子的最短路径，而不是最长路径。其长度称为问题的*非确定性查询复杂度*或*证书复杂度*，我们用 $D_0(P,n)$ 表示。请注意，非确定性复杂度通常以非对称方式定义。特别地，如果 f 只有两个可能的值 0 或 1，则对于每一个 x 与 $f(x) = 1$，该事实的证明可能是简短的，而对于某些 y 与 $f(y) = 0$，该事实的证明可能是冗长的。

如果我们把问题视为一个游戏，变化在于对手必须立即选择输入 x。之前我们寻找的是在两名玩家都最优策略下所需的查询次数。现在我们仍然有相同的假设，提问者采取最优策略，而对手则必须选择能产生最多查询次数的输入。我们可以观察提问者的最优玩法，就好像他知道（因为他猜对了）输入一样。

可以这样看待确定性版本中的对抗策略。与其说对手不必选择输入，我们可以说他必须选择一个，但他可以*更改*它。当然，当输入改变时，新的输入必须与之前的回答一致。

在本文中，我们将限制对手的能力。我们假设输入最多可以改变 k 次。我们注意到早期查询的回答必须得到满足，我们说这些是*固定*。我们将两次输入更改之间的步骤集称为*轮次*。

我们将从对手的角度来看这个游戏。如果我们对对手的最佳情况感兴趣（即，提问者面临的最坏情况），那么根据上述论点，如果他可以任意改变输入，则需要 $D(P,n)$ 次查询来完成游戏；如果他完全不能改变输入，则需要 $D_0(P,n)$ 次查询来完成游戏。

因此我们对 Adversary 的最坏情况感兴趣。他可以选择输入，然后最多更改 k 次，他能将游戏延长多久？我们将这个问题表示为 $P(k)$，在这种情况下所需的查询次数（输入大小为 n）表示为 $D_k(P,n)$。我们假设 Questioner 总是提出最好的可能的查询。因此；我们可以假定 Questioner 实际上一直知道当前输入是什么，他的目标不是找到 f 的值，而是证明该值就是那个值（即找到一个证书）。特别地，如果 $k=0$，那么在选择 x 后，对手将不再参与游戏，提问者知道 x。这确实是非确定性版本，因此我们不会用 $D_0(P,n)$ 来表示两个不同的事物。

另一方面，如果 $k \geq D(P,n)-1$，则对手可以在前 $D(P,n)-1$ 步中随意更改输入，并且提问者的最佳策略会在对手进一步更改输入的机会之前结束。因此，如果我们有 $D_k(P,n) = D(P,n)$ 当 $k \geq D(P,n)-1$ 时。

上述观察表明，对于任意的 k，$D_k(P,n)$ 位于问题 P 的非确定性和确定性复杂度之间。我们继续进行另外两个简单的观察。

命题 1.1. *(i)* $D_k(P,n) \leq D(P,n)$.

（二） $D_k(P,n) \leq \min\{\sum_{i=1}^{l} D_{j_i}(P,n) : l, i_1, \ldots, i_l \text{ are such that } \sum_{i=1}^{l}(j_i+1) > k\}$。

证明. $D(P,n)$ 是定义上的上界。为了证明 (ii)，提问者运行最优算法 $P(j_1)$。如果问题没有解决，对抗者必须改变输入超过 j_1 次。然后提问者运行最优算法 $P(j_2)$，如此继续直到 $P(j_l)$。如果问题从未被解决，在每个部分中对抗者都必须至少改变输入 j_i+1 次，总共改变了超过 k 次，这是矛盾的。 □

命题 1.2. $D_k(P,n) \geq \min\{k+1, D(P,n)\}$.

证明. 令对手的策略 A 是一个在可以任意改变输入的情况下迫使至少 $D(P,n)$ 个问题的策略。首先假设 $k+1 \geq D(P,n)$。那么，对手可以在前 $D(P,n)-1$ 步自由地改变输入，因此他可以遵循策略 A。如果 $k+1 < D(P,n)$，则对手采用相同的策略。只有在他不能再改变输入时才会失败，因此不会早于第 $k+1$ 次查询之后。 □

动机

我们有多种不同的方式来看待这里提出的新问题，它们导致了不同的动机和不同的术语。

我们的主要动机是通过研究中间地带的情况来更深入地理解确定性与非确定性查询复杂度之间的差异。

如果我们把问题视为一个游戏，我们可以考虑其非确定性版本，在这种情况下，对手必须选择一个输入。如果之后我们再看其确定性版本，可以认为对手改变输入的策略是作弊。例如，在著名的《战舰》游戏中，一名玩家可以在对方未察觉的情况下重新排列他的船只。这是有用的（增加了击沉所有船只所需的猜测次数从 $D_0(P,n)$ 到 $D(P,n)$），但显然这是一种作弊行为。在此设定下，我们研究了作弊的限制。请注意，在此设定中提问者也会通过窥探来作弊。

从某种意义上说，我们也在研究我们的算法有多稳健。当我们查看最优算法时，不仅仅是 $D_k(P,n)$ 的值，我们可以看到如果输入改变，我们需要对它们进行多少修改。

让我们提到，我们在检查实际算法时遇到问题并非不可想象。我们对输入有强烈的假设并不罕见。然而，一个查询可能会有一个出乎意料的答案。这导致了一个范式转变，一种完全重新评估输入的方式，从而引出了一个新的、同样强有力的假设。假设最多有 k 个范式转变是非常不自然的，但假设没有太多的转变是合理的。

最后，我们可以从不同的角度看待我们的问题。我们面对的不是一个固定的输入，而是一个动态且不断变化的系统。当我们提出一个查询时，我们得到的是输入的一部分而不是现有的（尽管对我们来说是未知的）信息，我们修复输入的那一部分。对于尚未固定的部分，我们确实有一些了解，但这些了解并不可靠。尽管如此，我们可能仍能假设我们的知识很少出错。

论文结构

在接下来的章节中，我们研究了不同问题中的 $D_k(P,n)$ 和 P。我们在第 2 节讨论搜索理论和群测试，在第 3 节讨论排序，在第 4 节讨论在一个未知顺序的集合中寻找最大值和最小值，在第 5 节讨论图是否连通的性质。我们以结论性备注结束本文，在这部分描述了几种变体。

2 搜索理论和群检测

在搜索理论中，$f = id$ 是恒等函数，即，我们需要从一个基本集合（可能的输入集）大小为 n 中识别出未知元素 x。通常，x 被称为导致整个系统故障的缺陷的元素，我们的目标是定位缺陷部分。允许的查询包括所有关于输入的是/否问题。更准确地说，我们可以将每个查询与答案为“是”的元素集对应起来，因此我们询问基本集合的子集。特别地，提问者可以询问集合 $\{x\}$，显示 $D_0(P,n) = 1$。

另一方面，众所周知，$D(P,n)=\lceil \log n \rceil$。因此命题 1.1和命题 1.2（带 $j_1=\cdots=j_{k+1}=0$）暗示了 $D_k(P,n)=\min\{k+1,\lceil \log n \rceil\}$。

一个自然的推广是考虑存在多个缺陷品的情况。这个版本通常称为群检测。关于这个领域的更多信息，请参见 [1]。在这种情况下，输入是一组元素。通常我们假设恰好有 d 或至多 d 个缺陷品，目标是识别出所有这些缺陷品。让我们用 P_d 和 $P_{\leq d}$ 表示这些问题。众所周知，我们有 $\Omega(d\log(n/d))=D(P_d,n)\leq D(P_{\leq d},n)=O(d\log n)$（例如，请参见 [1]）。

首先观察到，通过询问缺陷集的补集可以得到 $D_0(P_d,n)=1$，但 $D_0(P_{\leq d},n)=d$。实际上，只要询问包含所有缺陷元素 a 的集合 $\{a\}$ 即可。另一方面，对于每一个缺陷元素 a，我们需要询问一个只包含 a 作为缺陷元素的集合，否则 a 可能是非缺陷的。对于一般的 k，我们使用命题 1.1和命题 1.2得到 $D_k(P_d,n)=\min\{k+1,D(P_d,n)\}$ 和 $j_1=\cdots=j_{k+1}=0$。

然而，对于 $P_{\leq d}$ 情况要复杂得多。同样的方式，使用命题 1.1和 1.2，我们得到 $D_k(P_{\leq d},n)$ 的下界 $\min\{k+1,D(P_{\leq d},n)\}$ 和上界 $\min\{D(P_{\leq d},n)\}$。如果 n 很小，这种差异就会消失，因为 $D(P_{\leq d})$ 在两个界限中都给出了最小值。如果 n 很大，我们也可以确定 $D_k(P_{\leq d},n)$ 的确切值。

命题 2.1. 令 $n>(d-1)2^k$。则 $D_k(P_{\leq d},n)=k+d$。

证明. 对于上界，注意到提问者总是可以询问在该点处有缺陷的单元素集合。在每个点处，这样的查询最多得到 d 个 “是” 和最多 k 个 “否” 的答案，因此算法在最多 $k+d$ 次查询后结束。

对于下界，考虑 Adversary 的以下简单算法。对于第一个查询 A，他仅在 $|A|\geq n/2$ 时回答 YES。此外，在这种情况下，他还提供了所有有缺陷元素都在 A 中的额外信息。因此，在得到答案后，我们只知道一个至少包含所有有缺陷元素且大小为 $n/2$ 的集合。对于剩余元素数量而言，他以同样的方式回答前 k 个查询。之后，存在一个至少包含所有有缺陷元素且大小为 $n/2^k>d-1$ 的集合。即使 Adversary 用尽了所有的输入变化可能，Questioner 也需要至少再进行 d 次查询才能识别出所有有缺陷的元素。

□

3 排序

在排序的情况下，输入是一组不同的数字 n，记为 $a_1,\ldots,a_n$，一个查询对应两个数字 a_i 和 a_j，当且仅当 $a_i<a_j$ 时答案是肯定的。目标是对元素进行排序，即找到它们递增的顺序。可以将其视为第 2节研究的搜索理论问题的一个特殊版本，因为我们是在确定许多可能的排列中

的一种。然而，我们只能询问一些特殊的而不是所有可能的 YES/NO 问题。关于更多的排序信息，请参见例如 [2]。

令 S 表示排序问题，则众所周知 $D(S,n)=\Theta(n\log n)$。另一方面，$D_0(S,n)=n-1$。实际上，可以比较最大元素和第二大元素，然后是比较第二大元素和第三大元素，依此类推。另一方面，查询的成对作为边形成一个图，并且如果该图是不连通的，则我们无法知道它们的顶点之间的关系。

该问题的一个重要特殊性质是，那些 $n-1$ 条边始终需要用于解决问题，因为如果一对在排序顺序中相邻的数字没有被作为查询提出，我们就无法证明哪一个更大。其他 $\binom{n-1}{2}$ 个可能的查询最终都是无用的。因此，对手的目标是通过变更确保这些相邻数字中的成对数字不太多地出现在已经提出的查询中。

命题 3.1. $D_1(S,n)=\lceil 3n/2\rceil-2$.

证明. 让我们从对手的策略开始。他回答前 $\lceil n/2\rceil-1$ 次查询不变。然后他改变输入，使得所有被询问过的成对顶点在这新的顺序中都不相邻，因此需要 $n-1$ 次进一步的查询。

我们需要证明这些元素可以按照上述描述的方式重新排序。查询的边形成一个图，每个连通分量上都有一个偏序。对手将该偏序扩展为每个连通分量上的任意全序。设 $a_1<\cdots<a_k$ 是最大的一个分量，并考虑那些元素少于 k 的分量集合。选择其中一个元素数量最多的分量 $\ell\le k$。对手将这些分量上的顺序扩展到一个全序 $b_1<\cdots<b_\ell$，并令 $a_i<b_i<a_{i+1}$ 对每个 $i\le\ell$ 成立。对于每个其他具有排序 $c_1<\cdots<c_m$ 的组件，我们有 $m+\ell\ge k$。对手让 $a_{k-i}<c_i<a_{k-i+1}$ 为 $i\le m$。最后，他将上述给出的排序扩展到一个总排序，并将其作为新的输入。

对于 a_i 和 a_j 与 $i<j$，在新的顺序中它们之间要么有 b_i 要么有 c_{k-i}。从另一个组件中的每个顶点对，它们之间有一个 a_i。这表明对于前 $\lceil n/2\rceil-1$ 次查询中的任意一次，这两个元素在新输入中都不相邻，因此需要进一步的 $n-1$ 次查询。

让我们继续提问者的策略。他试图遵循上述非确定性问题版本中的策略：在查询 i 中，他比较第 i 和第 $i+1$ 小的元素，直到对手执行更改。假设它发生在第 i 个回答之后，那么我们有 $i+1$ 个元素 $b_1<\cdots<b_{i+1}$ 的总排序。变更后，对于每个其他元素我们都知道它在排序中的最终位置，特别是我们知道 b_j 和 b_{j+1} 之间有多少个元素。如果 b_j 和 b_{j+1} 之间有 p 个元素，那么 $p+1$ 次查询足以证明它们的最终位置是我们所怀疑的位置，如果是 $p=0$，那么 0 次查询就足够了。如果有 p 个元素小于 b_1（或大于 b_{i+1}），则 p 次查询就足够了。

这表明我们可以用每个元素一次查询加上一些额外的查询来找到剩余 $n-i-1$ 个元素

的位置：对于每一个 $j < i$，我们需要一个额外的查询来确定最终排序中位于 b_j 和 b_{j+1} 之间的那些元素，除非在 b_j 和 b_{j+1} 之间没有其他元素。一方面，最多需要 $i-1$ 个额外查询（因此总共 $n-2$ 个），因为在 $i-1$ 个区间中我们可能需要额外的查询。另一方面，最多需要 $n-i-1$ 个额外查询，因为我们至少需要在 b_j 和 b_{j+1} 之间有一个元素来进行额外查询。因此 $D_1(S,n) \le \min\{i+n-2, i+2(n-i-1)\}$，这很容易得出结论。 □

让我们简要讨论一下 $D_k(S,n)$。提问者的一个合理算法是上述算法的直接推广。他查看当前输入 $a_1 < \cdots < a_n$ 并询问 a_i, a_{i+1} 这个查询尚未被问过的最小 i。

针对这一策略，对手可以采取以下行动。他在第 ℓ_1 次查询后进行更改，并将尚未被查询的元素在早期查询中出现的元素之间以平衡的方式放置。然后，在第 ℓ_i 次查询之后应用第 i 次更改，最后一次更改发生在对手可以改变输入的最后一个点上，使得之前查询的所有对在新输入中都不相邻，因此后续还需要 n 次进一步的查询。考虑算法过程中的任意一点。设 $b_1 < \cdots < b_p$ 为当前轮询问者所询问的元素，$c_1 < \ldots, c_q$ 为之前询问过的元素。

然后固定的是（带有小误差的）$b_1 \cdots < b_p < c_1 \cdots < c_q$ 排序，使得当前排序中的 b_i 是 p 个最小元素，并且 c_i 在它们之上，以平衡的方式划分。最后一个变化必须在 $p+q$ 大约是 $n/2$ 时发生，因此额外的查询是在之前的轮次中获得的。然而，由于前面提到的误差，对这一点进行精确分析会相当复杂。确实，在给定的变化中，早期的查询给出了顺序 $a_1 < \cdots < a_r$，而当我们后来询问包含当前排序中位于 a_i 和 a_{i+1} 之间的元素时，最大的变成了 b_p，而 $c_1 = a_{i+1}$ 则不然。然而，我们并不一定知道关系 $b_p < c_1$ 是否成立。这会产生一个小的误差，在各轮中可能会累积。

4 寻找最大值和最小值

一个比前一节所考虑的任务更简单的任务是，在具有相同输入和查询的情况下，只寻找某些特殊元素。找到最大的元素需要 $n-1$ 次查询，无论是确定性方式还是非确定性方式都是如此。确实，其他每个元素都必须在答案中较小，因此需要 $n-1$ 次查询，并且可以通过不询问任何不可能成为最大值的元素来轻松实现，因为这些元素已被证明早前小于另一个元素。这表明，无论有多少变化，都需要 $n-1$ 次查询。

显然，同样的方法也适用于找到最小元素。令 L 表示同时寻找最大和最小元素的问题。Pohl[4] 证明了 $D(L,n) = \lceil 3n/2 \rceil - 2$。很明显，$D_0(L,n) = n-1$。下界来源于这样的事实，即这显然不比只找最大元素更容易，而上界则通过形式为 a_i, a_{i+1} 的查询来展示，其中 a_i 是第 i

大的元素。注意这些查询是大小为 $n-1$ 的唯一证书。确实，除了 a_1 外每个元素都应该是较小的，并且除了 a_n 外每个元素都应该在一个查询中较大。这些都是 $2n-2$ 个固定位置，因此如果我们只有 $n-1$ 次查询，则没有一个元素可以在多于一次的查询中更大（或更小）。显然，a_2 只比 a_1 小，因此查询 a_1, a_2 必须存在。然后不能询问 a_3, a_1，所以 a_3, a_2 是唯一一个查询，其中 a_3 较小，因此必须进行询问，依此类推。

定理 4.1. $D_k(L,n) = \min\{n+k-1, \lceil 3n/2 \rceil - 2\}$.

证明. 首先我们证明 $D_k(L,n) \leq n+k-1$。注意到这一点，结合 $D(L,n) = \lceil 3n/2 \rceil - 2$ 证明了 $D_k(L,n) \leq \min\{n+k-1, \lceil 3n/2 \rceil - 2\}$。我们对 k 使用归纳法，基本情况 $k=0$ 如上所述。在第一轮中，提问者询问了 a_i, a_{i+1} 作为 i 次查询，其中 a_i 是第 i 大的元素。假设在第 j 次查询后发生了变化。那么我们知道 $a_2, \ldots, a_j$ 不能是最大的也不能是最小的元素。因此我们需要在其他元素中找到最大的和最小的元素，所以 $D_{k-1}(L, n-j+1) \leq n-j+k-1$ 进一步的查询就足够了。总共最多有 $n+k-1$ 次查询，完成了上界的证明。

我们继续讨论对抗者的策略。如果一个元素在查询中既较小又较大，我们就说该元素是输出的。在算法的第一部分，只要可能，对抗者会以这样的方式回答问题，使得没有任何元素被移除。在这种情况下，他也会以下面的方式改变输入。假设 A 是已被移除的元素集合，B 是未被移除且在查询中较小的元素集合，C 是未被移除且在查询中较大的元素集合（并且存在一个由 $n-|A|-|B|-|C|$ 个元素组成的集合 D，这些元素尚未出现在任何查询中）。然后在新的输入中，B 的元素是最小的（任意排序），接着是 A 的元素，然后是 D 的元素（任意排序），最后是 C 的元素（任意排序）。注意任何这样的排序都与迄今为止的答案一致，除了 A 内部的排序，Adversary 选择一个与其答案一致的任意排序。

如果在这一部分中，对手必须回答得让一个元素出局（因为查询了 B 中的两个元素或 C 中的两个元素），那么他会在不改变输入的情况下作出回答。这部分在 $\min\{k, \lceil n/2 \rceil\}$ 个答案后结束，此时没有元素出局（因此最多经过 k 次变化）。在第二部分中，他不再改变输入。

注意，对手在第一次回答中并没有改变输入，并且没有一个元素在第一次回答后被移出。因此第一部分至少有 $1 + \min\{k, \lceil n/2 \rceil\} + |A|$ 次查询。在第二部分，B 中除了一个元素外的所有元素在一个查询中必须更大，而 C 中除了一个元素外的所有元素在一个查询中必须更小。由于当前输入中的 B 的元素比 C 的元素要小，所有这些都需要不同的查询。因此我们至少有 $1 + \min\{k, \lceil n/2 \rceil\} + |A| + |B| - 1 + |C| - 1 = \min\{n+k-1, \lceil 3n/2 \rceil - 2\}$ 次查询。 □

5 连通图

输入是一个未知图 G，该图有 n 个顶点，允许的查询对应于顶点对 u, v，答案告知 uv 是否为 G 的一条边。令 C 表示判定问题，即判断 G 是否连通。

众所周知（参见例如 [3]），$D(C,n) = \binom{n}{2}$，即连通性是一个规避的图属性。换句话说，提问者必须询问所有配对并完全识别输入的图 G 以确定其是否连通。展示这一点的一个简单策略是对手采取以下方式。除非一个查询会使该图不连通，否则他会对每个查询回答“否”，即他只改变输入中的那一条边。这显示了 $D_{n-2}(C,n) = \binom{n}{2}$，因为在这一策略中总是有 $n-1$ 个“是”的答案，并且最后一个“是”在最后的查询后到达（因此对手可以对那个查询回答“否”而不是“是”，从而避免最后一次改变）。

另一方面，$D_0(C,n) = \lfloor n^2/4 \rfloor$。事实上，对手可以选择一个由大小为 $\lfloor n/2 \rfloor$ 和 $\lceil n/2 \rceil$ 的两个组件组成的图 G。然后 G 是不连通的，但要证明这一点，必须询问这两个组件之间的所有可能边。

令 $T(n,k)$ 表示 Turán 图，即 k 部完全平衡图有 n 个顶点，并且 $t(n,k)$ 表示其边的数量。这里平衡意味着每一部分的大小为 $\lfloor n/k \rfloor$ 或 $\lceil n/k \rceil$。请注意，该图在所有 k 部图中具有最多的边。

定理 5.1. $t(n,k+2) \le D_k(C,n) \le t(n,k+2) + n - 1$.

证明. 让我们从对手的策略开始。他选择补图 G 作为原始输入，该补图对应于 Turán 图 $T(n,k+2)$。设 $A_1, \ldots, A_{k+2}$ 是 Turán 图中的部分；它们是 G 中的团。他对同一 A_i 内的顶点的回答总是相连；实际上他免费提供了这些信息，在接下来的内容中我们把这些边视为已知。对手会在 NO 回答会使图不连通（并结束算法）时更改输入。在这种情况下，他会把当前的非边更改为边。这样最终图将不会是连通图，但最后一次更改使图只有两个连通分量。假设 Turán 图中的边 uv 尚未被询问。然后在算法结束时，u 和 v 不可能在同一组件中。确实，在那种情况下，某个时刻有一个查询 $u'v'$ 将两个组件连接起来，其中一个包含 u，另一个包含 v。但是那时对该查询的回答将是“否”，因为这不会使图不连通。因此，u 和 v 在不同的组件中，但这样图可能是连通的，这是一个矛盾。

对于上限，我们对 k 应用归纳法，基数步骤 $k = 0$ 在定理陈述之前已描述。请注意，如果 $k \ge n-2$，上限大于 $\binom{n}{2}$，因此该命题是显然的。因此从现在开始，我们假设 $k \le n-3$。

让我们描述提问者的策略。首先，他询问包含每个组件中的生成树的生成森林的边。每当当前图还没有他已经询问过的这样的生成森林时，他就选择最少数量的边来形成一个与我们知道在图中的一些边构成的生成森林。每当当前图已经有这样的生成森林时，他就选取最小割并

询问其边。

声明 5.2. 找到一个生成森林（该图 G' 是此时的输入图）之后，如果有 i 次更改剩余，则最多需要 $t(n, i+2)$ 次查询。

声明的证明. 首先观察到，G' 的每个连通分支都包含一棵生成树。如果最多有 $i+2$ 个 G' 的连通分支，则（因为询问两端点在同一分支的边是没有意义的）我们所需询问的数量不会超过补图中 G' 的边数，这个数量至多为 $t(n, i+2)$。

如果 G' 中有超过 $i+2$ 个组件，设 $a_1, \ldots, a_{i+1}$ 是我们试图从剩余顶点中分离出的组件大小，并设 $a_{i+2} = n - \sum_{j=1}^{i+1} a_j$。然后我们在找到生成树后提出的查询次数最多为 $\prod_{1 \le j < l \le i+2} a_j a_l$，这是具有大小为 a_j 的部分的完全 $(i+2)$ 部图中的边的数量，因此至多为 $t(n, i+2)$。

□

让我们回到定理的证明。如果在寻找生成森林的过程中没有发生变化，则上述声明完成了证明。

如果在对应最初的生成树森林的查询过程中发生了变化，那么我们继续进行直到找到一个生成树森林，通过询问当前输入图中存在的边，但这些边不在已知边的连通分量内。注意，在最多 $n-1$ 次肯定回答之后，我们在当前图中就有了一个生成树森林。如果我们在此期间得到了 $k-i$ 次否定回答，这意味着有 $k-i$ 次变化，因此根据断言 5.2，后续最多需要 $t(n, i+2)$ 次查询。这说明总共有 $n-1+k-i+t(n, i+2)$ 次查询。注意，每当 $i < n$ 时，$t(n, i) < t(n, i+1)$ 成立。这表明每当 $k \le n-2$ 时，$k - i + t(n, i+2) \le t(n, k+2)$ 成立，从而完成证明。 □

6 结论与展望

我们定义了一个介于确定性和非确定性查询复杂度之间的量。我们在几个示例中进行了研究，但还有无数同样有趣的问题有待探讨。此外，即使对于我们考虑的大多数问题，我们也无法完全确定 $D_k(P, n)$。

在这里我们描述一些我们研究过的模型的潜在变体。

在图的情况中，或者更广泛地说，如果输入是可能查询的不相交并集（例如，在任何类型的布尔函数的情况下），可以将变化限制为所询问的查询。请注意，实际上这是 Adversary 在定理 5.1中改变输入图的方式。对于这个模型，我们可以考虑以下动机：提问者有一些过时的信息，例如一张地图，并且自从发布以来有些东西已经改变了。然而，可以合理假设变化的数

量是有限的。此外，这些变化是独立的，即，当我们了解到我们在某个地方的信息不正确时，我们没有理由假设任何特定的其他改变（将此与本文研究模型的一个动机中的更大范式转变进行比较）。

我们研究中的变化在某种意义上与其它研究中的谎言相反。在那里，对手有一个不同的选择：他可以最多 l 次给出任意的错误答案。这除了他能够随心所欲地改变输入次数外。仍然，将这些能力结合起来并研究一个可以改变输入 k 次并且撒谎 l 次的对手将是有趣的。在这种情况下，我们不会假设提问者知道一切。

还有其他的谎言模型，当固定比例的回答可以是虚假的，或者每个回答以概率 p 独立地是虚假的。类似地，我们可以修改我们的模型，并允许变化与查询数量成正比，或在每个点上定义可能输入图的分布，并随机选择一个。

我们也可以以更复杂的方式结合变化和谎言。例如，对手可以说他之前的答案是一个谎言，因为他当时更改了输入。

资金支持研究得到国家研究、发展和创新办公室——NKFIH 的资助，资助编号为 SNN 129364、KH 130371 和 K 116769，并由匈牙利科学院的 János Bolyai 研究奖学金支持。

参考文献

[1] D.Z. Du, F.K. Hwang, Combinatorial Group Testing and Its Applications, World Scientific, first ed., (1994)

[2] D. E. Knuth, The Art of Computer Programming, Volume 3: Sorting and Searching, Addison-Wesley, Reading, MA, 1973

[3] L. Lovász, N. Young, 1991. Lecture notes on evasiveness of graph properties. Tech. Rep. TR-317-91. Computer Science Department, Princeton Univ., Princeton, N.J.

[4] I. Pohl, A sorting problem and its complexity, Communications of the ACM 15 (1972), 462–464.