# Randomized and quantum query complexities of finding a king in a tournament

**Nikhil S. Mande**
University of Liverpool, UK

**Manaswi Paraashar**
University of Copenhagen, Denmark

**Nitin Saurabh**
Indian Institute of Technology Hyderabad, India

## Abstract

A tournament is a complete directed graph. It is well known that every tournament contains at least one vertex $v$ such that every other vertex is reachable from $v$ by a path of length at most 2. All such vertices $v$ are called *kings* of the underlying tournament. Despite active recent research in the area, the best-known upper and lower bounds on the deterministic query complexity (with query access to directions of edges) of finding a king in a tournament on $n$ vertices are from over 20 years ago, and the bounds do not match: the best-known lower bound is $\Omega(n^{4/3})$ and the best-known upper bound is $O(n^{3/2})$ [Shen, Sheng, Wu, SICOMP'03]. Our contribution is to show *tight* bounds (up to logarithmic factors) of $\widetilde{\Theta}(n)$ and $\widetilde{\Theta}(\sqrt{n})$ in the *randomized* and *quantum* query models, respectively. We also study the randomized and quantum query complexities of finding a maximum out-degree vertex in a tournament.

**2012 ACM Subject Classification** Theory of computation → Quantum complexity theory; Theory of computation → Oracles and decision trees

**Keywords and phrases** Query complexity, quantum computing, randomized query complexity, tournament solutions, search problems

**Digital Object Identifier** 10.4230/LIPIcs.CVIT.2016.23

## 1 Introduction

A tournament is a complete directed graph. Many important properties of tournaments were studied by Landau [18] in the context of modelling dominance relations among a flock of chickens. Relevant to our paper is the notion of a *king* in a tournament. This notion was defined by Maurer [19], also with the goal of identifying a reasonable measure of dominance to identify a 'most dominant' chicken in a flock. Soon after Maurer's article, Reid [22] showed existence of tournaments in which all vertices are kings. Tournaments also arise naturally in social choice theory where directions of edges depict preferences. A large amount of work has been devoted to defining a notion of a 'winner' in a tournament, and determining the complexity of finding such winners. For instance, Dey [11] studied the complexity of certain tournament solutions with motivations from social choice theory. The monograph by Moon [20] sparked a line of research on tournaments and their structural properties.

A natural computational model to study the complexity of computing specific properties of a tournament, or more generally, a graph, is that of *query complexity*. In this setting an algorithm may query presence/directions of edges in an unknown input graph. The goal is to minimize the number of such queries made in the worst case. There is a rich literature on query complexity of graph problems, starting over 50 years ago [24, 23, 28, 15, 9, 12, 10, 11]. The famous Aanderaa-Karp-Rosenberg conjecture [24] or evasiveness conjecture posits that the query complexity of any non-trivial monotone graph property on $n$-vertex graphs has maximal deterministic query complexity, i.e., $\binom{n}{2}$. While the deterministic and randomized


42nd Conference on Very Important Topics (CVIT 2016).
Editors: John Q. Open and Joan R. Access; Article No. 23; pp. 23:1–23:19
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

variants of this conjecture are wide open, the quantum version was recently resolved in the positive [1] using Huang's breakthrough resolution of the sensitivity conjecture [16].

Our work deals with the query complexity of certain graph problems. In the next section, we describe the main graph problem of interest to us, and prior work on it.

## 1.1 Related work

It is well known that every tournament has at least one vertex $v$ such that every other vertex is reachable from $v$ via a path of length at most 2 (see Lemma 4 for a proof). Such a vertex $v$ is called a *king* in the underlying tournament. Formally, one may define the following relation that captures this definition.

▶ **Definition 1** (Kings in a tournament). *Let $T$ be a complete directed graph on $n$ vertices. Identify the orientation of the tournament with a string $T$ in $\{0,1\}^{\binom{n}{2}}$, one variable ($\{i,j\}$ with $i \neq j \in [n]$) per edge (between vertex $i$ and vertex $j$) defining its direction. Define the relation $\mathsf{KING}_n \subseteq \{0,1\}^{\binom{n}{2}} \times [n]$ by*

$$(T, v) \in \mathsf{KING}_n \text{ if } \forall u \in [n], \text{ either } v \to u \text{ or } \exists w \textit{ such that } v \to w \to u.$$

*Here the directions of the edges $v \to u$ and $v \to w \to u$ are as in $T$.*

A natural question arises: what is the query complexity of finding a king in an $n$-vertex tournament? The study of this was initiated by Shen, Sheng and Wu [25]. They showed an algorithm with query complexity $O(n^{3/2})$ and also showed a non-matching lower bound of $\Omega(n^{4/3})$. For the upper bound, they crucially used the fact that a king in an in-neighbourhood of an arbitrary subset of vertices is also a king in the original tournament (see Lemma 5). An outline of their upper bound is as follows: first arbitrarily choose a sub-tournament of a fixed size and find the maximum-out-degree vertex in it by querying all edges in this sub-tournament. Remove this vertex along with its out-neighbours, and proceed iteratively. When the number of remaining vertices is small enough, find a king using brute force (query all the edges in the remaining sub-tournament). Simple manipulation of parameters gives an upper bound of $O(n^{3/2})$. For the lower bound they design an adversary who answers an algorithm's queries using a fixed strategy, and show that every algorithm must make $\Omega(n^{4/3})$ queries in the worst case. Ajtai et al. [3] independently showed the same bounds, in a different context. Despite active recent research in the area (see the next paragraph), these bounds from over 20 years ago remain state-of-the-art. It can be shown that a vertex with maximum out-degree is a king (see Lemma 4 and its proof). However, finding a vertex with maximum out-degree is known to be hard: it has deterministic query complexity $\Omega(n^2)$ [4].

Biswas et al. [6] recently showed that the adversary used by [3, 25] to show an $\Omega(n^{4/3})$ lower bound cannot be used to prove a stronger lower bound. They additionally showed a query complexity upper bound of $O(n^{4/3})$ on finding a vertex from which at least half of all vertices are reachable by paths of length at most 2. They also considered variants of kings, and the complexity of finding such vertices. In a more recent work, Lachish, Reidl and Trehan [17] showed an $O(n^{4/3})$-query algorithm to find a vertex from which at least $(\frac{1}{2} + \frac{2}{17})$ of the vertices are reachable by paths of length at most 2.

## 1.2 Our contributions

While the question of pinning down the deterministic query complexity of finding a king has been open and unimproved since the work of Shen, Sheng and Wu [25], the corresponding question in the randomized and quantum query models does not seem to have been studied in the literature. Our contribution is to give tight bounds in these models. We refer the reader to Section 2 for a formal description of these models. Let $\mathsf{R}(\cdot)$ and $\mathsf{Q}(\cdot)$ denote bounded-error randomized and quantum query complexity, respectively. Our main theorems are as follows.

▶ **Theorem 2.** *For all positive integers $n$,*

$$\mathsf{R}(\mathsf{KING}_n) = O(n \log\log n), \qquad \mathsf{R}(\mathsf{KING}_n) = \Omega(n),$$
$$\mathsf{Q}(\mathsf{KING}_n) = O(\sqrt{n}\ \mathrm{polylog}(n)), \qquad \mathsf{Q}(\mathsf{KING}_n) = \Omega(\sqrt{n}).$$

We mentioned earlier that a vertex of maximum out-degree in a tournament is a king, and finding a vertex of maximum out-degree is known to have deterministic query complexity $\Omega(n^2)$. We show that even the randomized query complexity is $\Omega(n^2)$, and we also show bounds for the quantum query complexity of this task. Define the relation $\mathsf{MOD}_n \subseteq \{0,1\}^{\binom{n}{2}} \times [n]$ to consist of the elements $(T, v)$ where $v$ is a maximum out-degree vertex in the $n$-vertex tournament described by $T$.

▶ **Theorem 3.** *For all positive integers $n$,*

$$\mathsf{R}(\mathsf{MOD}_n) = \Theta(n^2), \qquad \mathsf{Q}(\mathsf{MOD}_n) = O(n^{3/2}), \qquad \mathsf{Q}(\mathsf{MOD}_n) = \Omega(n).$$

We suspect that $\mathsf{Q}(\mathsf{MOD}_n) = \Theta(n^{3/2})$, but we leave open the problem of closing the gap between the upper and lower bounds in the quantum setting.

**Sketch of randomized upper bound for finding a king** As mentioned in Section 1.1, the upper bound of Shen, Sheng and Wu crucially uses the fact that a king in the in-neighbourhood of an arbitrary vertex is also a king in the original tournament (Lemma 5). A simple counting argument shows that a *uniformly random* vertex in an $n$-vertex tournament has out-degree $\Omega(n)$ with high probability. This suggests a natural randomized iterative algorithm: in each step sample a few vertices and query all edges incident on them, until a vertex with large out-degree in the current sub-tournament is seen. We then remove this vertex along with all its out-neighbours from the tournament, and iterate. Since a random vertex has out-degree that is linear in the number of vertices with high probability, this process results in a small sub-tournament (with at most $\sqrt{n}$ vertices) after $O(\log n)$ iterations. At this point we can afford to query the entire remaining sub-tournament to find a king in it, and it can be shown by applying Lemma 5 iteratively that this vertex is also a king the original tournament.

**Sketch of quantum upper bound for finding a king** Our quantum algorithm follows the same structure as our randomized one, but we run into some issues during a naive simulation. The following are the issues, along with how we handle them:

- When trying to sample a vertex with high out-degree, we cannot afford to query all edges incident on a vertex to compute its out-degree since our algorithm needs to have query complexity essentially $O(\sqrt{n})$. To circumvent the need of querying all edges incident on a vertex to compute its in-degree, we use the subroutine of *approximate counting* [8] that returns an approximation of the in-degree but offers a quadratic speedup. It may seem like one could use a classical algorithm for approximate counting here, but such a classical sampling-based algorithm would require $\widetilde{\Omega}(n)$ queries if the number of in-neighbours is small, say $\mathrm{polylog}(n)$ (see the fourth bullet as to why such a case may arise).
- A second issue that arises is when we need to sample a vertex from the current sub-tournament. It is no longer clear how to do this in the quantum setting since we do not

explicitly know the vertices remaining. However, we keep track of the set of vertices $W$ whose out-neighbours have effectively been removed in previous iterations. The number of iterations of the algorithm, and hence the size of $W$, is bounded by $O(\log n)$. We then use key properties of Grover's search algorithm: first set up a uniform superposition over all vertices. Next we 'mark' the vertices in the current sub-tournament (call such vertices 'good') using $O(\log n)$ queries: for a given vertex we only need to check if it is an out-neighbour of any of the vertices in $W$. Making these queries in superposition allows us to mark the good vertices in $O(\log n)$ queries. We then apply the Grover iterate a suitable number (at most $O(\sqrt{n})$) of times. At this point we make a measurement in the computational basis: by the correctness of Grover's algorithm, the probability of seeing a good vertex (i.e., a vertex in the current sub-tournament) is large. The structure of Grover's algorithm implies that conditioned on not seeing a bad vertex, each good vertex is seen with equal probability. This effectively simulates sampling a uniformly random vertex from the current sub-tournament.

- Having sampled a vertex $v$ from the in-neigbourhood of $W$, the randomized algorithm next computes the out-degree of $v$ in the in-neighbourhood of $W$. We cannot afford to do this exactly since we do not explicitly know the in-neighbourhood of $W$, and moreover it may be very large. We are able to get around this using similar ideas to that in the previous bullet.
- Finally, it is no longer clear how to do the final brute-force step in the last remaining sub-tournament since we do not explicitly know the remaining $\sqrt{n}$ vertices. To handle this, we first modify the randomized algorithm so as to only have $O(\text{polylog}(n))$ vertices remaining in this 'brute-force' step, while still having only $O(\log n)$ iterations overall. Thus, the query complexity so far is still $\widetilde{O}(\sqrt{n})$. We can then use Grover's search repeatedly (or an improvement thereof, Theorem 11) to find all the remaining vertices with high probability in $\widetilde{O}(\sqrt{n})$ queries. At this point we can find a king in the remaining sub-tournament using $O(\text{polylog}(n))$ queries. By the same argument as in the randomized case, this vertex is also a king in the original tournament.

**Sketch of lower bounds for finding a king** To show our lower bounds, we restrict our attention to a special class of tournaments, described below. Fix an arbitrary tournament $T$ on $n$ vertices where vertex $n$ is a source. This immediately implies that vertex $n$ is the unique king. For each $i \in [n-1]$, define the tournament $T_i$ to be $T$ with edges incident on vertex $i$ flipped so as to make vertex $i$ the source. Note that these sets of edges are disjoint for every $i \neq j$. If we assign one variable to each such set and promise that at most one of them has value 1 (i.e., has edges in the opposite direction from those in $T$), an algorithm that finds a king in these tournaments (which are unique by construction) also solves the Search problem on $n-1$ input variables. Our lower bounds of $\Omega(n)$ and $\Omega(\sqrt{n})$ on the randomized and quantum query complexities, respectively, then immediately follow from corresponding well-known lower bounds on the complexity of the Search problem.

**Sketch of bounds for finding a maximum out-degree vertex** In the randomized setting, we use Yao's minimax principle (Lemma 21). By this principle, it suffices to exhibit a hard distribution on input tournaments such that any deterministic algorithm with small query complexity must make large error when inputs are drawn from this distribution. We now describe the distribution: fix an $n$-vertex regular tournament, say $T$ with $n$ odd and each vertex having out-degree exactly $(n-1)/2$ (such a tournament is easy to construct iteratively, for example) and flip a uniformly random edge of $T$. This causes a unique vertex of the new tournament to be a maximum out-degree vertex. Intuitively, finding this vertex is as hard as finding the edge that has been flipped, and it is well known that searching for a marked element among $k$ elements has randomized query complexity $\Omega(k)$. We formalize this argument in Theorem 20. Our quantum lower bound uses similar ideas, and involves a reduction from the Search problem on an $\binom{n}{2}$-bit string, which has quantum query complexity $\Omega(n)$ [7]. For the quantum upper bound, we use a maximum finding routine over the degrees of the vertices. Each degree can be computed using $n-1$ queries, and the maximum can be found in $O(\sqrt{n})$ queries [13], giving us an $O(n^{3/2})$ upper bound.

## 2 Preliminaries

All logarithms in this paper are base 2. We use the notation polylog$(n)$ to denote a quantity that is $\log^c n$ for a constant $c > 0$ (independent of $n$). For a positive integer $n$, we use the notation $[n]$ to denote the set $\{1, 2, \ldots, n\}$. For an event $X$, let $\mathbb{I}[X]$ denote the indicator of $X$, i.e., $\mathbb{I}[X] = 1$ if $X$ occurs, and $\mathbb{I}[X] = 0$ if $X$ does not occur.

### 2.1 Tournaments

A tournament $T$ on a vertex set $V$ is a complete graph such that each edge is directed. Throughout this paper, unless mentioned otherwise, we consider tournaments $T$ on $n$ vertices and denote the vertex set by $V = [n]$. Such a tournament has $\binom{n}{2}$ directed edges. We identify an $n$-vertex tournament with a binary string in $\{0,1\}^{\binom{n}{2}}$: an element of $[n]$ corresponds to the label of a vertex, and there is one variable ($\{i, j\}$ with $i \neq j \in [n]$) per edge (between vertex $i$ and vertex $j$) that defines its direction. For a tournament $T$ and vertex $v \in V$, let $N^-(v)$ denote the set of in-neighbours of $v$, i.e., $N^-(v) = \{u \in [n] \setminus \{v\} \,|u \to v \text{ is an edge in } T\}$ and let $N^+(v)$ denote the set of out-neighbours of $v$ (i.e., $\{u \in [n] \setminus \{v\} \,|v \to u \text{ is an edge}\}$). Also, let $d^+(v) = |N^+(v)|$ and $d^-(v) = |N^-(v)|$ denote the out-degree and in-degree of $v$, respectively. Since $T$ is a tournament, $d^+(v) + d^-(v) = (n-1)$ for all $v \in V$. For $S \subseteq V$, let $T[S]$ be the tournament induced on the vertices in $S$. For a subset $W \subseteq V$, define $W^- = \{v \in V \mid v \to w \text{ is an edge for all } w \in W\}$. If $W = \emptyset$ then define $W^- = V$. A vertex $v \in V$ is a *king* if every vertex in $V \setminus \{v\}$ is reachable from $v$ by a path of length at most 2. This is formally captured in Definition 1 and repeated below for convenience. Define the relation $\mathsf{KING}_n \subseteq \{0,1\}^{\binom{n}{2}} \times [n]$ by $(G, v) \in \mathsf{KING}_n$ if $\forall u \in [n] \setminus \{v\}$, either $v \to u$ or $\exists w : v \to w \to u$. Here the directions of the edges $v \to u$ and $v \to w \to u$ are as in the tournament $G$. A well-known fact about tournaments is that every tournament has a king. We give a proof for completeness.

▶ **Lemma 4** (Folklore). *Let $T \in \{0,1\}^{\binom{n}{2}}$ be a tournament. Then there exists a vertex $v \in [n]$ such that $(T, v) \in \mathsf{KING}_n$.*

**Proof.** Consider a vertex $v$ of maximum out-degree. We show that such a vertex is a king. Consider the partition of $V$ into three disjoint sets: $\{v\}$, $N^+(v)$ and $N^-(v)$. Clearly, every vertex in $N^+(v)$ is at a distance at 1 from $v$. Towards a contradiction, assume that there is a vertex $w$ in $N^-(v)$ such that there is no path of length 2 of the form $v \to u \to w$, for some $u \in N^+(v)$. Thus every vertex in $N^+(v)$ is an out-neighbour of $w$. Since $v$ is also an out-neighbour of $w$, the out-degree of $w$ is greater than that of $v$, which is a contradiction. ◀

The above lemmas shows that any vertex with maximum out-degree in a tournament is a king in that tournament. However, as discussed in Section 1.1, finding a vertex of maximum out-degree is known to be hard. We need the following result due to [19].

▶ **Lemma 5** ([19]). *Let $T \in \{0,1\}^{\binom{n}{2}}$ be a tournament and $v \in [n]$. If a vertex $u$ in $N^-(v)$ is a king in $T[N^-(v)]$, then $u$ is a king in $T$.*

The proof of the above lemma is easy: If $u$ is a king of the tournament $T[N^-(v)]$, then every vertex in $N^-(v)$ is at a distance at most 2 from $u$. Also, since $u$ is an in-neighbour of $v$, every vertex in $N^+(v)$ is at a distance 2 from $u$. We also need the following lemma from [19].

▶ **Lemma 6** ([19]). *Let $T \in \{0,1\}^{\binom{n}{2}}$ be a tournament. $\sum_{i=1}^n d^+(i) = \sum_{i=1}^n d^-(i) = \binom{n}{2}$.*

We also need the following observation on the structure of a tournament (see e.g., [4]).

▶ **Lemma 7.** *Let $T \in \{0,1\}^{\binom{n}{2}}$ be a tournament and $k \geq 0$. Then, the number of vertices $v$ such that $d^+(v) \leq k$ is at most $2k+1$.*

## 2.2 Query complexity

A deterministic decision tree $T$ on $m$ variables is a binary tree where the internal nodes are labeled by variables and leaves are labeled with elements of a set $\mathcal{R}$. Each internal node has a left child, corresponding to an edge labeled 0, and a right child corresponding to an edge labeled 1. On an input $x \in \{0,1\}^m$, $T$'s computation traverses a path from root to leaf as follows. At an internal node, the variable associated with that node is *queried*: if the value obtained is 0, the computation moves to the left child, otherwise it moves to the right child. The output of $T$ on input $x$, denoted by $T(x)$, is the label of leaf node reached. We say that a decision tree $T$ computes the relation $f \subseteq \{0,1\}^m \times \mathcal{R}$ if $(x, T(x)) \in \mathcal{R}$ for all $x \in \{0,1\}^m$. The deterministic query complexity of $f$, is $\mathsf{D}(f) := \min_{T: T \text{ computes } f} \text{depth}(T)$. A randomized decision tree $\mathcal{A}$ is a distribution $\mathcal{D}_\mathcal{A}$ over deterministic decision trees. On input $x \in \{0,1\}^m$, the computation of $\mathcal{A}$ proceeds by first sampling a deterministic decision tree $T$ according to $\mathcal{D}_\mathcal{A}$, and outputting the label of the leaf reached by $T$ on $x$. We say $\mathcal{A}$ computes $f$ with bounded error if for every input $x$, $\Pr[(x, \mathcal{A}(x)) \in \mathcal{R}] \geq 2/3$. The randomized query complexity of $f \subseteq \{0,1\}^m \times \mathcal{R}$ is defined as follows. $\mathcal{R}(f) = \min_{\substack{\mathcal{A} \text{ computing } f \\ \text{with error } \leq 1/3}} \max_{T: \mathcal{D}_\mathcal{A}(T) > 0} \text{depth}(T)$.

## 2.3 Preliminaries for quantum query complexity

We refer the reader to [21, 26] for basics of quantum computing. A quantum query algorithm $\mathcal{A}$ computing a relation $f \subseteq \{0,1\}^m \times \mathcal{R}$ begins in an input-independent initial state $|\psi_0\rangle$, applies a sequence of unitaries $U_0, O_x, U_1, O_x, \cdots, U_T$, and performs a measurement. Here, the unitaries $U_0, U_1, \ldots, U_T$ are independent of the input. The unitary operation $O_x$ represents the 'query' operation, and maps $|i\rangle|b\rangle$ to $|i\rangle|b \oplus x_i\rangle$ for all $i \in [m]$ and $|0\rangle$ to $|0\rangle$. We say that $\mathcal{A}$ is a bounded-error algorithm computing $f$ if for all $x \in \{0,1\}^m$, the probability of outputting $b \in \mathcal{R}$ such that $(x, b) \in f$ is at least $2/3$. The bounded-error quantum query complexity of $f$, denoted by $\mathsf{Q}(f)$, is the least number of queries required for a quantum query algorithm to compute $f$ with error at most $1/3$.

We also need some basic notions from Grover's search algorithm [14], a fundamental quantum algorithm, referring the reader to [26, Chapter 7] for more details. In the search problem, a quantum algorithm is given quantum query access to a string $x \in \{0,1\}^n$. It is convenient to work with the 'phase-query' unitary $O_{x,\pm}$ which satisfies $O_{x,\pm}|i\rangle = (-1)^{x_i}|i\rangle$. The goal is to find an $i \in [n]$ such that $x_i = 1$ with probability at least $2/3$ if such an $i$ exists, otherwise return that there is no such element. An $i$ which satisfies $x_i = 1$ is also called a *marked element* and thus the goal is to find a marked element with high probability, if such an element exists.

Let $t := |\{i \in [n] : x_i = 1\}|$. Grover's algorithm starts with the uniform superposition $|U\rangle = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} |i\rangle$, and proceeds by applying Grover's iterate (which is an application of $O_{x,\pm}$ followed by a reflection about $|U\rangle$) several times. After $k$ applications of Grover's iterate the resulting state is

$$\sin((2k+1)\theta) \sum_{i:x_i=1} |i\rangle + \cos((2k+1)\theta) \sum_{i:x_i=0} |i\rangle, \tag{1}$$

where $\theta = \arcsin(\sqrt{t/n})$. It is known that Grover's algorithm finds a marked element in $x$ (if it exists) with $O(\sqrt{n})$ applications of the query oracle $O_{x,\pm}$, and probability at least $2/3$. Standard error reduction yields the following theorem.

▶ **Theorem 8.** *Given query access to $x \in \{0,1\}^n$, there is a quantum algorithm that decides whether the Hamming weight of $x$ is 0 or returns an $i \in [n]$ such that $x_i = 1$, with error at most $\delta$. The query complexity of this algorithm is $O(\sqrt{n \cdot \log(1/\delta)})$.*

Grover's algorithm is known to be asymptotically optimal.

▶ **Theorem 9** ([7]). *A quantum algorithm that solves the Search problem with error $2/5$ on $n$-bit inputs must have query complexity $\Omega(\sqrt{n})$, even when the inputs are promised to have Hamming weight either 0 or 1.*

The following theorem, due to Dürr and Høyer [13], is a generalization of Grover's search algorithm, to find the maximum number in an input list.

▶ **Theorem 10** ([13]). *Let $T$ be an unsorted table of $n$ items. There exists a quantum query algorithm of cost $O(\sqrt{n})$ that has query access to $T$ and returns the maximum element of $T$ with probability at least $2/3$.*

We require the following theorem, essentially due to Boyer et al. [7].[1]

▶ **Theorem 11** ([7]). *Given query access to $x \in \{0,1\}^n$ with $|x| \geq k$, there is a quantum algorithm that outputs, with query complexity $O(\sqrt{(n/k)} \log(1/\delta))$ and error probability at most $\delta$, an index $i \in [n]$ with $x_i = 1$.*

We obtain the following immediate corollary by repeating the algorithm in Theorem 11 $k$ times and updating the 'marked' elements after each application.

▶ **Corollary 12.** *Given an input parameter $k$ and query access to $x \in \{0,1\}^n$, there is a quantum algorithm that does the following with query complexity $O(\sqrt{nk} \log\log(n))$ and error probability at most $1/\mathrm{polylog}(n)$:*

- *If $|x| \geq k$, it returns $k$ distinct indices $i_1, \ldots, i_k \in [n]$ such that $x_{i_j} = 1$ for $j \in [k]$.*
- *If $|x| < k$, it outputs all indices $i$ with $x_i = 1$, along with the information that $|x| < k$.*

Our quantum algorithm also uses quantum approximate counting as a sub-routine. Here, an algorithm is given query access to a string $x \in \{0,1\}^n$. The indices $i \in [n]$ such that $x_i = 1$ are again called 'marked'. For an input parameter $\varepsilon$ the goal of the algorithm is to output a multiplicative $(1 \pm \varepsilon)$-approximation of the number of marked indices of $x$. An optimal quantum algorithm for approximate counting was first given by Brassard et al. [8]. We use a version due to Aaronson and Rall [2].

[1] Their bound is for bounded-error algorithms and does not have polylogarithmic factors in the query complexity. Standard error reduction gives us Theorem 11.

▶ **Theorem 13** ([2]). *There exists a quantum algorithm that, given $\varepsilon > 0$ and query access to a string $x \in \{0,1\}^n$, outputs an estimate $\widetilde{K}$ of $K = |\{i : x_i = 1\}|$ such that $K(1-\varepsilon) \leq \widetilde{K} \leq K(1+\varepsilon)$ with probability at least $(1-\delta)$. The query complexity of this algorithm is $O(\sqrt{n/K} \cdot 1/\varepsilon \cdot \log(1/\delta))$.*

## 3 Randomized algorithm

Throughout this section and the next, unless mentioned otherwise, a tournament $T$ is assumed to be in $\{0,1\}^{\binom{n}{2}}$, and its vertex set is denoted by $V = [n]$. Query algorithms are assumed to have classical/quantum query access to the edge directions of $T$, that is, the individual bits of the corresponding $\binom{n}{2}$-bit string.

In this section we give a randomized algorithm for finding a king in a tournament $T \in \{0,1\}^{\binom{n}{2}}$ with query complexity $O(n \log\log n)$ and success probability at least $2/3$. First, we make the following simple observation, which shows that a randomly chosen vertex from $V = [n]$ has a large number of out-neighbours with high probability.

▶ **Lemma 14** (Out-degree of a random vertex is large). *For all positive integers $n$, a tournament $T \in \{0,1\}^{\binom{n}{2}}$ and a vertex $v \in V$ chosen uniformly at random, $d^+(v) \geq \lfloor (n-1)/5 \rfloor$ with probability at least $3/5$.*

**Proof.** From Lemma 7, $|\{v \in V \mid d^+(v) < \lfloor (n-1)/5 \rfloor\}| \leq 2((n-1)/5 - 1) + 1 = (2n-7)/5 < 2n/5$. Thus, the fraction of vertices with out-degree at least $\lfloor (n-1)/5 \rfloor$ is at least $3/5$. ◀

Lemma 14 suggests a natural randomized query algorithm, given in Algorithm 1. We show in Theorem 15 that the algorithm makes $O(n \log\log n)$ queries to $T$ in the worst case, and returns a king with probability at least $2/3$.

**Algorithm 1** Randomized Query Algorithm

**Input:** Query access to edge directions of a tournament $T \in \{0,1\}^{\binom{n}{2}}$ where $V = [n]$.
**while** $|V| \geq \sqrt{n}$ **do**
  $t \leftarrow |V|$, $k \leftarrow \lceil \log\log n \rceil$
  $v_1, \ldots, v_k \leftarrow$vertices chosen uniformly at random from $V$
  $w \leftarrow \arg\max_{u \in \{v_1,\ldots,v_k\}} d^+(u)$ ▷ querying all edges incident on $\{v_1,\ldots,v_k\}$ in $T[V]$ and breaking ties arbitrarily
  **if** $d^+(w) = t-1$ **then**
    Return $w$
  **else if** $d^+(w) < \lfloor (t-1)/5 \rfloor$ **then**
    Return a random vertex $v \in V$
  **else** ▷ $\lfloor (t-1)/5 \rfloor \leq d^+(w) < t-1$ here
    $V \leftarrow N^-(w)$ ▷ This is the in-neighbourhood of $w$ in the set $V$, and not in the whole vertex set $[n]$.
    **continue**
  **end if**
**end while**
$w \leftarrow$ a king in $T[V]$ ▷ query all edges in the sub-tournament $T[V]$
Output $w$

▶ **Theorem 15.** *Let $n > 0$ be a positive integer. Then,* $\mathsf{R}(\mathsf{KING}_n) = O(n \log\log n)$.

**Proof.** Consider Algorithm 1. We first analyze the query cost of the algorithm. For the correctness, we define 'bad events', argue correctness of the algorithm conditioned on no bad event occurring, and then upper bound the probability of a bad event happening.

**Query complexity** In order to upper bound the query complexity, first note that each iteration of the **while** loop (Line 2) uses $k \cdot |V| \leq |V| \log\log n$ queries in the worst case. Furthermore, the **while** loop goes into the next iteration (Line 12) if and only if $|V| > \sqrt{n}$ (Line 2) and a vertex $w$ of out-degree at least $\lfloor (t-1)/5 \rfloor$ has been found in Line 5 (see comment on Line 10). This means that the size of the vertex set reduces by a factor of at least $4/5$ in the next iteration of the **while** loop. In particular, this means in the $i$'th iteration of the **while** loop, we have $|V| \leq (4/5)^i \cdot n$, and thus there are $O(\log n)$ iterations of the **while** loop in the worst case. Finally, Line 15 accounts for at most $O(n)$ queries since $|V| < \sqrt{n}$ here. The worst-case query complexity is thus upper bounded by

$$n + \sum_{i=0}^{O(\log n)} \left(\frac{4}{5}\right)^i \cdot n \cdot O(\log\log n) = O(n \log\log n).$$

**Bad event, and correctness assuming no bad event** The event of Line 9 occurring during the run (i.e., Line 8 being triggered in any iteration) is defined to be the bad event. Conditioned on the bad event not occurring, the algorithm either terminates on Line 7 or Line 16. Clearly when the algorithm terminates on Line 7 or Line 16, the output vertex is a king in the sub-tournament being considered at the moment. If the **while** loop has not even completed once, the current sub-tournament is the same as the original tournament, and we are done. If the **while** loop has completed at least once, the sub-tournament being considered at the moment is the sub-tournament of a tournament $T'$ (which itself may be a sub-tournament of $T$) induced by the in-neighbourhood of a specific vertex. Applying Lemma 5, we conclude that the king in the current sub-tournament is also a king in $T'$, and also the whole tournament by applying Lemma 5 repeatedly now. Hence conditioned on the bad event not occurring, the algorithm indeed outputs a correct answer.

**Probability of bad event** From Lemma 14, the probability that Line 8 is run in an iteration is at most $(2/5)^k \leq 1/\log^{\log 2.5}|V| \leq 1/\log^{1.3} n$. By a union bound, the probability that Line 8 gets executed in any of the $O(\log n)$ iterations is at most $O(\log n)/\log^{1.3}(n) = o(1)$. ◀

## 4 Quantum algorithm

For $W \subseteq [n]$ and $v \in V$, we can decide whether $v$ is an out-neighbour of any $w \in W$ by making $|W|$ queries, by checking $x_{wv}$ for all $w \in W$. Similarly, $|W|$ queries are sufficient to decide whether $v$ is an in-neighbour of some vertex $w \in W$. This simple classical algorithm can easily be simulated in the quantum setting, which gives us the following observation.

▶ **Observation 16.** *For a tournament $T \in \{0,1\}^{\binom{n}{2}}$ and a known subset of the vertices $W \subseteq V$, there exists a unitary transformation that maps the basis state $|v\rangle$ to $(-1)^{\mathbb{I}[v \in W^-]}|v\rangle$ using $|W|$ queries to $T$. In other words, there is a unitary transformation that has query cost $|W|$ and 'marks' vertices in $W^-$.*

Before proving the main theorem of this section, we give two lemmas (proven in the appendix). The algorithm in these lemmas will be used in the proof of the main theorem.

▶ **Lemma 17.** *Let $T \in \{0,1\}^{\binom{n}{2}}$ be a tournament, $W \subseteq V$ and $t = \Theta(\log\log n)$ be an integer. There exists a quantum algorithm* In-Sample$(T, W, t)$, *Algorithm 2, that with error probability at most $1/(\text{polylog}(n))$, returns a set of uniformly distributed and independent samples from $W^-$ of size $t$. The query complexity of this algorithm is $O(|W| \cdot \sqrt{n} \cdot \text{polyloglog}(n))$.*

**Algorithm 2** The In-Sample$(T, W, t)$ algorithm for sampling many uniformly independent samples from a subset of vertices

1: **Input:** Query access to the adjacency matrix of a tournament $T \in \{0,1\}^{\binom{n}{2}}$ where $V = [n]$, $W \subseteq V$ such that $|W^-| \geq \log^{100} n$, and $t \in \mathbb{N}$ such that $t = \Theta(\log\log n)$.
2: $N \leftarrow 10^4 n$
3: $|\phi\rangle \leftarrow \sum_{i=1}^{N} \frac{1}{\sqrt{N}} |i\rangle$ ▷ $|\phi\rangle$ is used as the starting state in Line 8 and Line 14 with vertices in $W^- \subseteq [n]$ marked (by first checking if $j \in [N]$ satisfies $j \leq n$, and marking such a $j$ using $|W|$ queries).
4: **if** $W = \emptyset$ **then**
5: $\quad S \leftarrow t$ samples from uniform superposition over $V$
6: $\quad$ Return $S$
7: **else**
8: $\quad \widetilde{w} \leftarrow$ estimate of $|W^-|$ from Theorem 13 with $\varepsilon = 1/100, \delta = 1/\text{polylog}(N) = 1/\text{polylog}(n)$.
9: $\quad w' \leftarrow \lfloor \widetilde{w}/2 \rfloor$
10: $\quad \widetilde{k} \leftarrow \left\lfloor \left( \frac{\pi}{400 \arcsin \sqrt{w'/N}} + \frac{1}{2} \right) \right\rfloor$
11: $\quad R \leftarrow \emptyset$
12: $\quad \mathsf{count} \leftarrow 0$
13: $\quad$ **while** $\mathsf{count} < O(t \text{ polyloglog}(n))$ **do**
14: $\quad\quad |\psi_i\rangle \leftarrow$ state obtained by applying Grover's iterate $\widetilde{k}$ times on $|\phi\rangle$, with vertices in $W^-$ being the marked elements
15: $\quad\quad v_i \leftarrow$ measurement outcome of $|\psi_i\rangle$ in computational basis
16: $\quad\quad$ **if** $v_i \in W^-$ **then** ▷ query edges between $v_i$ and $W$
17: $\quad\quad\quad R \leftarrow R \cup \{v_i\}$
18: $\quad\quad$ **end if**
19: $\quad\quad \mathsf{count} \leftarrow \mathsf{count} + 1$
20: $\quad\quad$ **if** $|R| = t$ **then** ▷ If we have collected enough samples
21: $\quad\quad\quad$ Return $R$ ▷ This is a set of uniformly distributed and independent samples from $W^-$ of size $t$ (See Lemma 17)
22: $\quad\quad$ **end if**
23: $\quad$ **end while**
24: **end if**
25: Return $[t]$ ▷ The algo makes error in this case.

▶ **Lemma 18.** *Let $T \in \{0,1\}^{\binom{n}{2}}$ be a tournament, $W$ be a subset of $V$ satisfying $|W^-| \geq \log^{100} n$ and $u$ be a vertex in $V$. There exists a quantum algorithm* Decide-High-Out-Degree$(T, W, u)$, *Algorithm 3, that returns with error probability at most $1/(\text{polylog}(n))$,* True *if the out-degree of $u$ in $W^-$ is at least $|W^-|/5$ and* False *if the out-degree of $u$ in $W^-$ is at most $|W^-|/10$. The query complexity of this algorithm is $O(|W| \cdot \sqrt{n} \text{ polylog}(n))$.*

**Algorithm 3** The Decide-High-Out-Degree$(T, W, u)$ subroutine

**Input:** Query access to the edge directions of a tournament $T \in \{0,1\}^{\binom{n}{2}}$ where $V = [n]$, $W \subseteq V$ such that $|W^-| \geq \log^{100} n$, and $u \in V$.
$\widetilde{w}_1 \leftarrow$ estimate of $|W^-|$ using Theorem 13 with $\varepsilon = 1/100, \delta = 1/\text{polylog}(n)$.
▷ Since the algorithm is given $W$ is input, it can decide whether $v \in W^-$ by making $|W|$ queries.
$\widetilde{w}_2 \leftarrow$ estimate of $|N^+(u) \cap W^-|$ using Theorem 13 with $\varepsilon = 1/100, \delta = 1/\text{polylog}(n)$.
▷ Note that we do not have query access to the presence/absence of a vertex $v$ in $N^+(u) \cap W^-$. However such a query can be implemented with $1 + |W|$ queries: check if $v \to u$ is an edge, and check if $v \to w$ is an edge for any $w \in W$.
**if** $\widetilde{w}_2/\widetilde{w}_1 \geq 99/505$ **then**
    return True
**else**
    return False
**end if**

We now show our main result of this section.

▶ **Theorem 19.** *Let $n > 0$ be a positive integer. Then* $\mathsf{Q}(\mathsf{KING}_n) = O(\sqrt{n}\ \text{polylog}(n))$.

**Proof.** Consider Algorithm 4. We first analyze the query cost of the algorithm. For the correctness, we define 'bad events', argue correctness of the algorithm conditioned on no bad event occurring, and then upper bound the probability of a bad event happening.

**Query complexity** First we upper bound $|W|$ at the end of the run of the algorithm. The **while** loop in Line 3 runs for at most $O(\log n)$ iterations. The algorithm starts with $W$ initialized to $\emptyset$ and is updated only in Line 14 where one new element is added to $W$. Thus we have $|W| = O(\log n)$.

Consider Line 5. Since $|W| = O(\log n)$ and $k = \log^{100} n$, by Corollary 12 the number of queries in this step is upper bounded by $O(|W|\sqrt{n}\ \text{polylog}(n)) = O(\sqrt{n}\ \text{polylog}(n))$, and thus the overall cost of queries executed in this line over at most $O(\log n)$ iterations is also $O(\sqrt{n}\ \text{polylog}(n))$.

In Line 9, the In-Sample algorithm (Algorithm 2) is called at most $O(\log n)$ times with $t = \Theta(\log\log n)$ and $|W| = O(\log n)$. Thus by Lemma 17, the cost of this step is upper bounded by $O(|W|\sqrt{n}\ \text{polylog}(n)) = O(\sqrt{n}\ \text{polylog}(n))$.

Now consider the **for** loop in Line 11. This loop is executed at most $O(\log n)$ times and each iteration of this loop invokes the algorithm Decide-High-Out-Degree, with $|W| = O(\log n)$, at most $|S|$ many times. Since $|S| = O(\text{polylog}(n))$ (see Lemma 17) the query cost in this loop is upper bounded by $O(|W| \cdot |S| \cdot \sqrt{n}\ \text{polylog}(n)) = O(\sqrt{n}\ \text{polylog}(n))$ in the worst case.

The only remaining step in Line 23. In this case, since $|U| \leq \log^{100} n$ throughout the algorithm, at most $O(\text{polylog}(n))$ queries are made.

**Bad event, and correctness assuming no bad event** If any of the following events happen, we say that a bad event has happened for Algorithm 4:

- **(I)** The algorithm in Corollary 12 which is used in Line 5 gives an incorrect answer.
- **(II)** The algorithm In-Sample (Algorithm 2) in Line 9 fails to return a set of $\Omega(t) = \Omega(\log\log n)$ uniformly distributed and independent samples from $W^-$.
- **(III)** The set $S$ obtained from In-Sample in Line 9 does not contain a vertex of out-degree at least $|W^-|/5$ in $W^-$.
- **(IV)** The algorithm Decide-High-Out-Degree (Algorithm 3) in Line 13 returns False.

We prove the correctness of the algorithm assuming that these bad events do not happen. Consider the $j$'th iteration of the **while** loop in Line 3, for $j \geq 1$, and let $W^{(j)}$ denote the set $W$ in this iteration. $W^{(j)}$ is updated only in Line 14 by a $v$ which satisfies $v \in (W^{(j)})^-$. This is because each vertex of the set $S$ belongs to $W^-$ (see Line 16 of Algorithm 2). In the next iteration of the **while** loop, $(W^{(j+1)})^-$ is defined as $(W^{(j)})^- \cap N^-(v)$. Thus by applying Lemma 5 iteratively, $(W^{(j+1)})^-$ contains a king in the tournament $T[(W^{(j)})^-]$, and hence a king in $T$.

Assuming that the bad events do not happen, we now argue that in $O(\log n)$ iterations the size of $W^-$ becomes smaller than $\log^{100} n$. In this case $U = W^-$ because of the property of Corollary 12 used in Line 5, and the algorithm correctly returns the king in Line 23 by a similar argument as in the previous paragraph by iteratively applying Lemma 5. The analysis is similar to that of proof of Theorem 15. Since Decide-High-Out-Degree (Algorithm 3) in Line 13 does not return False, the out-degree of $v$ in $(W^{(j)})^-$ must be at least $|(W^{(j)})^-|/10$.

Thus $|(W^{(j+1)})^-| \leq (9/10) \cdot |(W^{(j)})^-|$, and after $O(\log n)$ iterations the size of $W^-$ is smaller than $\log n < \log^{100} n$.

**Probability of bad event** The probability of events I, II, IV are each upper bounded by $O(1/\text{polylog}(n))$ by Corollary 12, Lemma 17 and Lemma 18, respectively. The probability of event III conditioned on II not happening is upper bounded by $(2/5)^{\Theta(\log\log(n))} = O(1/\text{polylog}(n))$, thus the probability of event III is upper bounded by $O(1/\text{polylog}(n))$. The number of times that the events I, II, III can happen is at most $O(\log n)$, and IV can happen is at most $O(\text{polylog}(n))$, a union bound implies the probability of a bad event happening is upper bounded by $O(1/\text{polylog}(n))$. ◀

# 5 Lower bounds

We show our lower bounds in this section. We first show our lower bounds for the query complexity of finding a vertex of maximum out-degree, and then our lower bounds for finding a king in a tournament.

## 5.1 Maximum out-degree

We show in this subsection that the randomized query complexity of finding a vertex of maximum out-degree in an $n$-vertex tournament is $\Omega(n^2)$. This task is formally defined as the relation $\mathsf{MOD}_n \subseteq \{0,1\}^{\binom{n}{2}} \times [n]$: $(G, v) \in \mathsf{MOD}_n$ if $d^+(v) \geq d^+(w)\ \forall w \neq v \in [n]$. Here the out-degrees of $v, w$ are according to the tournament $G$.

▶ **Theorem 20.** *For sufficiently large positive integers $n$, $\mathsf{R}(\mathsf{MOD}_n) \geq n^2/100$.*

We use Yao's minimax principle [27], stated below in a form convenient for us.

▶ **Lemma 21** (Yao's minimax principle)**.** *For a relation $f \subseteq \{0,1\}^m \times \mathcal{R}$, we have $\mathsf{R}(f) \geq k$ if and only if there exists a distribution $\mu : \{0,1\}^m \to [0,1]$ such that $\mathsf{D}_\mu(f) \geq k$. Here, $\mathsf{D}_\mu(f)$*

**Algorithm 4** Quantum Algorithm

1: **Input:** Query access to the edge directions of a tournament $T \in \{0,1\}^{\binom{n}{2}}$ with $V = [n]$
2: $W \leftarrow \emptyset$, $t \leftarrow \Theta(\log\log n)$, and $\mathsf{COUNT} \leftarrow O(\log n)$ ▷ Recall that $\emptyset^- := V$
3: **while** $\mathsf{COUNT} > 0$ **do**
4: $\quad$ $\mathsf{COUNT} \leftarrow \mathsf{COUNT} - 1$
5: $\quad$ $U \leftarrow$ the output of the algorithm in Corollary 12 with the string in $\{0,1\}^{[n]}$ as input where indices corresponding to vertices in $W^-$ are equal to 1 (marked), and $k = \log^{100} n$ ▷ query access to this string can be done using $|W|$ edge queries to $T$
6: $\quad$ **if** $|U| < \log^{100} n$ **then**
7: $\quad\quad$ **break** ▷ Go to Line 23
8: $\quad$ **else**
9: $\quad\quad$ $S \leftarrow$ In-Sample$(T, W, t)$ ▷ We reach here if $|W^-| \geq \log^{100} n$ (Line 7 gets executed otherwise)
10: $\quad\quad$ $S' \leftarrow S$
11: $\quad\quad$ **for** $v \in S$ **do**
12: $\quad\quad\quad$ $S' \leftarrow S' \setminus \{v\}$
13: $\quad\quad\quad$ **if** Decide-High-Out-Degree$(T, W, v) ==$ True **then** ▷ Decide-High-Out-Degree can be applied since $|W^-| \geq \log^{100} n$
14: $\quad\quad\quad\quad$ $W \leftarrow W \cup \{v\}$
15: $\quad\quad\quad\quad$ **break** ▷ Go to Line 3
16: $\quad\quad\quad$ **end if**
17: $\quad\quad\quad$ **if** $S' == \emptyset$ **then**
18: $\quad\quad\quad\quad$ Return a random vertex $v \in V$
19: $\quad\quad\quad$ **end if**
20: $\quad\quad$ **end for**
21: $\quad$ **end if**
22: **end while**
23: Return a king in $U$ ▷ query all edges in $T[U]$

*is the minimum depth of a deterministic decision tree that computes $f$ to error at most $1/3$ when inputs are drawn from the distribution $\mu$.*

**Proof of Theorem 20.** Assume without loss of generality that $n$ is odd. We construct a hard distribution $\mu$ on $n$-vertex tournaments. We show that any deterministic query algorithm of cost less than $n^2/100$ must make error at least $1/3$ on inputs drawn from $\mu$, and this would prove the theorem by Yao's principle (Lemma 21). Let $G$ be a fixed $n$-vertex regular tournament where every vertex has out-degree exactly $(n-1)/2$ (such a tournament is easy to construct, by induction, for example). The distribution $\mu$ is defined by taking $G$ and flipping the direction of a uniformly random edge. Note that all resultant tournaments have a unique vertex with maximum out-degree.

Consider a deterministic query algorithm (decision tree) that queries less than $n^2/100$ edges. Consider the leaf $L$ of this tree for which answers of all queries on its path are consistent with directions of edges in $G$. Say the label of this leaf is vertex $i$. Consider the set $S$ of all unqueried edges on the path to $L$ that are not incident on vertex $i$. We have $|S| \geq \binom{n}{2} - \frac{n^2}{100} - (n-1)$. For each $e \in S$, the graph $G_e$ defined by flipping the direction of

$e$ in $G$ reaches the leaf $L$. Moreover, the unique maximum out-degree vertex of $G_e$ is not vertex $i$ since $e$ is not incident on $i$ by the definition of $S$. This implies that the tree outputs the wrong answer on $G_e$. By the definition of $\mu$, we have $\mu(G_e) = 1/\binom{n}{2}$ for all $e \in S$. Thus, the mass of inputs under $\mu$ on which the decision tree makes an error is at least

$$\sum_{e \in S} \mu(G_e) \geq \frac{\binom{n}{2} - \frac{n^2}{100} - n + 1}{\binom{n}{2}} > \frac{97}{100} > \frac{1}{3},$$

where the second-to-last inequality holds for sufficiently large $n$. Lemma 21 yields the theorem. ◀

We now give our quantum bounds for $\mathsf{MOD}_n$.

▶ **Theorem 22.** *For all positive integers $n$,* $\mathsf{Q}(\mathsf{MOD}_n) = O(n^{3/2}), \mathsf{Q}(\mathsf{MOD}_n) = \Omega(n)$.

**Proof.** For the upper bound we apply the maximum finding subroutine in Theorem 10 to the degree sequence of the input tournament. Finding the degree of a vertex (and hence a query of the maximum-finding algorithm) can be done with $n-1$ edge queries. Thus, this algorithm has cost $O(\sqrt{n} \cdot (n-1)) = O(n^{3/2})$.

For the lower bound, we give a reduction from the Search problem on an $\binom{n}{2}$-bit string. As in the proof of the randomized lower bound, assume $n$ is odd and let $G$ be a fixed $n$-vertex regular tournament where every vertex has out-degree exactly $(n-1)/2$. Towards a contradiction, suppose we have an algorithm $\mathcal{A}$ that finds a maximum out-degree vertex in an $n$-vertex graph with query complexity $o(n)$ and probability at least $2/3$. We use $\mathcal{A}$ to solve the Search problem on $\binom{n}{2}$-bit strings with the promise that the input has Hamming weight at most 1. On input $x \in \{0,1\}^{\binom{n}{2}}$ with $|x| \leq 1$, do the following:

1. Run the algorithm $\mathcal{A}$ on the tournament $G \oplus x$. Here $G \oplus x$ denotes the bitwise XOR of $G$ and $x$. Suppose the output is $v \in [n]$.
2. Run a (99/100)-error Search algorithm with query cost $O(\sqrt{n})$ on the $n-1$ indices of $x$ that are indexed by pairs with one element as $v$ (that is, indexed by the edges adjacent to $v$ in the corresponding tournament).
3. Output the index returned by the search algorithm.

The cost of this algorithm is clearly $o(n) + O(\sqrt{n})$. For the correctness, first note that when $|x| = 1$ and $G$ is such that all out-degrees are equal, the tournament $G \oplus x$ has exactly one maximum out-degree vertex. Thus, by the correctness of $\mathcal{A}$, it outputs this vertex with probability at least $2/3$. Observe that the edge flipped in $G \oplus x$ from $G$ is adjacent to this vertex. In the event that the first step outputs the correct vertex, the edge that has been flipped in $G \oplus x$ from $G$ (i.e., the index $\{i, j\}$ with $x_{\{i,j\}} = 1$) is caught in the second step with probability at least $99/100$. Thus, this gives an algorithm solving the Search problem on $\binom{n}{2}$-bit strings with the promise that the input has Hamming weight at most 1, with success probability at least $(2/3) \cdot (99/100) > 3/5$. The query cost of this algorithm is $o(n)$ from the first step, by our assumption, and $O(\sqrt{n})$ from the second step. Thus the total cost is $o(n)$, which is a contradiction in view of Theorem 9. ◀

We leave open the question of closing the gap in Theorem 22.

## 5.2 Finding a king

We show an $\Omega(n)$ lower bound for the randomized query complexity of finding a king in a tournament, and an $\Omega(\sqrt{n})$ quantum query lower bound. To show these lower bounds, we restrict our attention on input tournaments of a particular structured form that have the property that there is only one king (which is a source in the tournament). We then show a lower bound on the randomized and quantum query complexities of finding a king in these promised inputs, by a reduction from the Search problem on $n-1$ variables with the promise that the input has Hamming weight either 0 or 1, for which we know an $\Omega(n)$ lower bound in the randomized setting and an $\Omega(\sqrt{n})$ lower bound in the quantum setting. Our reductions use a simple modification of block sensitivity.

We require the following relation.

▶ **Definition 23.** *Let $n$ be a positive integer. Define the relation $\mathsf{USEARCH}_n \subseteq \{0,1\}^n \times \{\emptyset\} \cup [n]$ as $(0^n, \emptyset) \in \mathsf{USEARCH}_n$ and $(x,i) \in \mathsf{USEARCH}_n$* when $x = e_i$.

▷ Claim 24. Let $n$ be a positive integer. Then,

$$\mathsf{R}(\mathsf{KING}_n) \geq \mathsf{R}(\mathsf{USEARCH}_{n-1}), \qquad \mathsf{Q}(\mathsf{KING}_n) \geq \mathsf{Q}(\mathsf{USEARCH}_{n-1}).$$

**Proof.** Consider an arbitrary input $x \in \{0,1\}^{\binom{n}{2}}$ such that the vertex $n$ is the source. For each $j \in [n-1]$, let $V_j \subseteq \left[\binom{n}{2}\right]$ be the set of edges incident on vertex $j$ that need to be flipped in the input $x$ to make vertex $j$ the source. We first make the following two observations:

$$V_j \cap V_k = \emptyset \quad \forall j \neq k \in [n-1], \qquad \bigcup_{i=1}^{n-1} V_j = \left[\binom{n}{2}\right]. \tag{2}$$

The first observation follows by considering an edge from vertex $\ell$ to vertex $m$. This edge only appears in $V_m$. Clearly every edge belongs to exactly one $V_j$, proving the second observation.

Using these two observations, the input set $\{0,1\}^{\binom{n}{2}}$ can also be expressed as $\{0,1\}^{V_1} \times \{0,1\}^{V_2} \times \cdots \times \{0,1\}^{V_{n-1}}$. For the remaining part of this proof we treat inputs to be of the latter form. In fact, we only restrict our attention to the case where each coordinate in a 'block' has the same value.

For a string $y \in \{0,1\}^{n-1}$, define the tournament $x_y = \bigotimes_{i=1}^{n-1} y_i^{V_i}$. Thus we have the following tournaments when $|y| \leq 1$:

$$x_{e_j} = \begin{cases} 0^{V_1} \times \cdots \times 0^{V_{j-1}} \times 1^{V_j} \times 0^{V_{j+1}} \times \cdots \times 0^{V_{n-1}} & y = e_{j-1} \\ 0^{V_1} \times \cdots \times 0^{V_{n-1}} & y = 0^{n-1}. \end{cases}$$

In other words, $x_{e_j}$ equals the tournament $x$ with variables in $V_j$ flipped, and $x_{0^{n-1}} = x$.

Note that vertex $j$ is the source (and thus the unique king) in the tournament $x_{e_j}$. Thus, finding a king in the set of tournaments $\{x_{e_j} : j \in [n-1]\}$ is the same as finding a source in these tournaments. Thus, a query algorithm finding a king in the restricted input set $x_y : |y| \leq 1$ yields a query algorithm for $\mathsf{USEARCH}_{n-1}$ on input $y$, which proves the claim. ◀

From the well-known lower bounds of $\mathsf{Q}(\mathsf{USEARCH}_{n-1}) = \Omega(\sqrt{n})$ [5] and $\mathsf{R}(\mathsf{USEARCH}_{n-1}) = \Omega(n)$, we obtain our main theorem of this section.

▶ **Theorem 25.** *Let $n$ be a positive integer and $\mathsf{KING}_n \subseteq \{0,1\}^{\binom{n}{2}} \times [n]$. Then,*

$$\mathsf{R}(\mathsf{KING}_n) = \Omega(n), \qquad \mathsf{Q}(\mathsf{KING}_n) = \Omega(\sqrt{n}).$$

## References


1 Scott Aaronson, Shalev Ben-David, Robin Kothari, Shravas Rao, and Avishay Tal. Degree vs. approximate degree and quantum implications of huang's sensitivity theorem. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1330–1342. ACM, 2021. URL: https://doi.org/10.1145/3406325.3451047.

2 Scott Aaronson and Patrick Rall. Quantum approximate counting, simplified. In *Symposium on simplicity in algorithms*, pages 24–32, 2020.

3 Miklós Ajtai, Vitaly Feldman, Avinatan Hassidim, and Jelani Nelson. Sorting and selection with imprecise comparisons. *ACM Transactions on Algorithms (TALG)*, 12(2):1–19, 2015.

4 Ramachandran Balasubramanian, Venkatesh Raman, and G Srinivasaragavan. Finding scores in tournaments. *Journal of Algorithms*, 24(2):380–394, 1997.

5 Charles H. Bennett, Ethan Bernstein, Gilles Brassard, and Umesh V. Vazirani. Strengths and weaknesses of quantum computing. *SIAM J. Comput.*, 26(5):1510–1523, 1997. doi:10.1137/S0097539796300933.

6 Arindam Biswas, Varunkumar Jayapaul, Venkatesh Raman, and Srinivasa Rao Satti. Finding kings in tournaments. *Discret. Appl. Math.*, 322:240–252, 2022. doi:10.1016/j.dam.2022.08.014.

7 Michel Boyer, Gilles Brassard, Peter Høyer, and Alain Tapp. Tight bounds on quantum searching. *Fortschritte der Physik: Progress of Physics*, 46(4-5):493–505, 1998.

8 Gilles Brassard, Peter Hoyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation. *Contemporary Mathematics*, 305:53–74, 2002.

9 Amit Chakrabarti and Subhash Khot. Improved lower bounds on the randomized complexity of graph properties. In *Automata, Languages and Programming: 28th International Colloquium, ICALP 2001 Crete, Greece, July 8–12, 2001 Proceedings 28*, pages 285–296. Springer, 2001.

10 Andrew M Childs and Robin Kothari. Quantum query complexity of minor-closed graph properties. *SIAM Journal on Computing*, 41(6):1426–1450, 2012.

11 Palash Dey. Query complexity of tournament solutions. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 2992–2998. AAAI Press, 2017. URL: http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14180.

12 Christoph Dürr, Mark Heiligman, Peter Høyer, and Mehdi Mhalla. Quantum query complexity of some graph problems. *SIAM Journal on Computing*, 35(6):1310–1328, 2006.

13 Christoph Dürr and Peter Høyer. A quantum algorithm for finding the minimum. *arXiv preprint quant-ph/9607014*, 1996.

14 Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing (STOC)*, pages 212–219. ACM, 1996. doi:10.1145/237814.237866.

15 Péter Hajnal. An $\Omega(n^{4/3})$ lower bound on the randomized complexity of graph properties. *Combinatorica*, 11:131–143, 1991.

16 Hao Huang. Induced subgraphs of hypercubes and a proof of the sensitivity conjecture. *Annals of Mathematics*, 190(3):949–955, 2019.

17 Oded Lachish, Felix Reidl, and Chhaya Trehan. When you come at the king you best not miss. In *42nd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2022*, volume 250 of *LIPIcs*, pages 25:1–25:12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.FSTTCS.2022.25.

18 HG Landau. On dominance relations and the structure of animal societies: Iii the condition for a score structure. *The bulletin of mathematical biophysics*, 15:143–148, 1953.

19 Stephen B Maurer. The king chicken theorems. *Mathematics Magazine*, 53(2):67–80, 1980.

20 John W Moon. *Topics on tournaments in graph theory.* Courier Dover Publications, 2015.

21 Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information (10th Anniversary edition)*. Cambridge University Press, 2016. URL: https://www.cambridge.org/de/academic/subjects/physics/quantum-physics-quantum-information-and-quantum-computation/quantum-computation-and-quantum-information-10th-anniversary-edition?format=HB.

22 K.B. Reid. Every vertex a king. *Discrete Mathematics*, 38(1):93–98, 1982. doi:10.1016/0012-365X(82)90173-X.

**23** Ronald L Rivest and Jean Vuillemin. On recognizing graph properties from adjacency matrices. *Theoretical Computer Science*, 3(3):371–384, 1976.

**24** Arnold L Rosenberg. On the time required to recognize properties of graphs: A problem. *ACM SIGACT News*, 5(4):15–16, 1973.

**25** Jian Shen, Li Sheng, and Jie Wu. Searching for sorted sequences of kings in tournaments. *SIAM J. Comput.*, 32(5):1201–1209, 2003. doi:10.1137/S0097539702410053.

**26** Ronald de Wolf. Quantum computing: Lecture notes, 2019. arXiv:1907.09415, version 5. URL: http://arxiv.org/abs/1907.09415, arXiv:1907.09415.

**27** Andrew Chi-Chih Yao. Probabilistic computations: Toward a unified measure of complexity. In *18th Annual Symposium on Foundations of Computer Science (SFCS 1977)*, pages 222–227. IEEE Computer Society, 1977.

**28** Andrew Chi-Chih Yao. Lower bounds to randomized algorithms for graph properties. In *28th Annual Symposium on Foundations of Computer Science (sfcs 1987)*, pages 393–400. IEEE, 1987.


# A Proofs of Lemmas from Section 4

In this section we prove Lemma 17 and Lemma 18.

**Proof of Lemma 17.** Consider Algorithm 2. We first analyze the query cost of the algorithm. For the correctness, we define 'bad events', argue correctness of the algorithm conditioned on no bad event occurring, and then upper bound the probability of a bad event happening.

## Query complexity

We upper bound the worst-case query complexity of the algorithm. Line 8 of the algorithm case costs $O(|W| \cdot \sqrt{N} \cdot \text{polylog}(N))$ from Theorem 13. The **while** loop from Line 13 runs for $O(t \cdot \text{polyloglog}(N)) = O(\text{polyloglog}(N))$ times, and each Grover's iterate in each of these iterations makes $O(|W| \cdot \sqrt{N})$ queries in Line 14. Also, Line 16 uses $|W|$ many queries. Thus, the overall query cost of the algorithm is upper bounded by $O(|W| \cdot \sqrt{N} \cdot \text{polyloglog}(N))$. Since $N = 10^4 n$, we have an upper bound of $O(|W| \cdot \sqrt{n} \cdot \text{polyloglog}(n))$.

## Bad event, and correctness assuming no bad event

If the estimate in Line 8 is incorrect or if the algorithm has reached Line 25 is not in $W^-$ then we say that a bad event has occurred for Algorithm 2. We assume that these events have no happened. Thus the estimate in Line 8 is correct then $\widetilde{w}$ satisfies

$$|W^-|(1 - 1/100) \leq \widetilde{w} \leq |W^-|(1 + 1/100).$$

Define $w' = \lfloor \widetilde{w}/2 \rfloor$, thus $w'$ satisfies the following equations.

$$\begin{aligned} |W^-|/4 \leq w' \leq |W^-|, \\ 1/2 \cdot \sqrt{|W^-|/N} \leq \sqrt{w'/N} \leq \sqrt{|W^-|/N}. \end{aligned} \quad (3)$$

Let $x = |W^-|/N$. Since $|W^-| \geq 0$ and $|W^-| \leq n$, we have

$$0 \leq x \leq 1/10^4.$$

Let $C = 1/10^4$. For $x \in [0, \sqrt{C}]$ and $A \geq 1$ (whose value is to be fixed later), define

$$g(x) = A \arcsin x/2 - \arcsin x.$$

The derivative of $g$ is given by

$$g'(x) = \frac{A/2}{\sqrt{1-x^2/4}} - \frac{1}{\sqrt{1-x^2}}$$
$$\geq \frac{A}{\sqrt{4-x^2}} - \frac{1}{\sqrt{1-C}}$$
$$\geq A/2 - \frac{1}{\sqrt{1-C}}.$$

Thus for $A = 3\sqrt{1-C}$ the above derivative is positive for all $x \in [0, \sqrt{C}]$. Since $g(0) = 0$, we have, for $A \arcsin x/2 \geq \arcsin x$.

From monotonicity of arcsin in $[0,1]$ and Equation (3) we have

$$\arcsin(1/2 \cdot \sqrt{|W^-|/N}) \leq \arcsin(\sqrt{w'/N}) \leq \arcsin(\sqrt{|W^-|/N})$$
$$1/A \cdot \arcsin(\sqrt{|W^-|/N}) \leq \arcsin(\sqrt{w'/N}) \leq \arcsin(\sqrt{|W^-|/N}). \quad (4)$$

In Line 10 we choose $\widetilde{k}$ to be $\left\lfloor \left( \frac{\pi}{400 \arcsin \sqrt{w'/N}} + \frac{1}{2} \right) \right\rfloor$. From Equation (4) we have

$$\left( \frac{\pi}{400 \arcsin \sqrt{|W^-|/N}} + \frac{1}{2} \right) \leq \left( \frac{\pi}{400 \arcsin \sqrt{w'/N}} + \frac{1}{2} \right) \leq (A+1) \cdot \left( \frac{\pi}{400 \arcsin \sqrt{|W^-|/N}} + \frac{1}{2} \right). \quad (5)$$

which implies

$$\left( \frac{\pi}{400 \arcsin \sqrt{|W^-|/N}} - \frac{1}{2} \right) \leq \left\lfloor \left( \frac{\pi}{400 \arcsin \sqrt{w'/N}} + \frac{1}{2} \right) \right\rfloor \leq (A+1) \cdot \left( \frac{\pi}{400 \arcsin \sqrt{|W^-|/N}} + \frac{1}{2} \right). \quad (6)$$

From Equation (1), if we apply Grover's iterate $k$ times then the resulting state in Line 14 is of the following form:

$$\beta \sum_{v \in W^-} |v\rangle + \sqrt{(1-\beta^2)} \sum_{v \in W^+} |v\rangle, \quad (7)$$

where $\beta = \sin((2k+1) \cdot \arcsin \sqrt{|W^-|/N})$. From Equation (6) we have

$$\frac{\pi}{200} \leq (2\widetilde{k}+1) \cdot \arcsin \sqrt{|W^-|/N} \leq (A+1)\frac{\pi}{200} + (A+2) \arcsin(\sqrt{|W^-|/N}) < \pi/2,$$

where the last inequality follows due to the choice of $A$ ($A \leq 3$) and since $\sqrt{|W^-|/N} \leq 1/100$. Thus after $\widetilde{k}$ iterations, $\beta^2 = \sin^2((2\widetilde{k}+1) \cdot \arcsin \sqrt{|W^-|/N})$ is a constant smaller than $\pi/2$.

Since we have assumed that the bad event in Line 25 has not occurred, this means that $t$ sample obtained is in $W^-$. From Equation 7 each vertex in $W^-$ has an equal probability of being sampled. Clearly, for different iterations of the **while** loop in Line 13 the samples are independent. Also, in this case the algorithm returns in Line 21 after $t$ iterations and hence $\Omega(t)$ uniformly distributed and independent samples from $W^-$ are returned.

## Probability of bad event

The probability of the bad event happening in Line 8 by Theorem 13 is $O(1/\text{polylog}(n))$. To upper bound the probability of the algorithm reaching Line 25, observe that with probability $\beta^2 = \Omega(1)$ (see Equation (7)) a vertex sampled in Line 15 is in the set $W^-$. Thus the probability that after $O(t \text{ polyloglog}(n))$, less than $t$ vertices are seen in $W^-$ is upper bounded by $O(1/\text{polylog}(n))$ by a Chernoff bound. ◀

**Proof of Lemma 18.** Consider Algorithm 3. We first analyze the query cost of the algorithm. For the correctness, we define a 'bad event', argue correctness of the algorithm conditioned on the bad event not occurring, and then upper bound the probability of the bad event happening.

## Query complexity

The only queries used are in Line 2 and Line 3 of the algorithm. The query cost of these steps are upper bounded by $O(|W| \cdot \sqrt{n} \cdot \mathrm{polylog}(n))$ by Theorem 13.

## Bad event, and correctness assuming no bad event

The only bad event for Algorithm 3 are that either the estimates Line 2 or Line 3 is incorrect. Let us assume that the bad event has not happened. Then

$$(1 - 1/100)|W^-| \leq \widetilde{w}_1 \leq (1 + 1/100)|W^-|,$$

and

$$(1 - 1/100)|N^+(u) \cap W^-| \leq \widetilde{w}_2 \leq (1 + 1/100)|N^+(u) \cap W^-|.$$

We have

$$\frac{99}{101} \cdot \frac{|N^+(u) \cap W^-|}{|W^-|} \leq \frac{\widetilde{w}_2}{\widetilde{w}_1} \leq \frac{101}{99} \cdot \frac{|N^+(u) \cap W^-|}{|W^-|}.$$

Thus if $|N^+(u) \cap W^-|/|W^-| \geq 1/5$ then $\widetilde{w}_2/\widetilde{w}_1 \geq 99/505$ and if $|N^+(u) \cap W^-|/|W^-| \leq 1/10$ then $\widetilde{w}_2/\widetilde{w}_1 \leq 101/990$.

## Probability of bad event

By Theorem 13 and a union bound, the probability of the bad event is upper bounded by $O(1/\mathrm{polylog}(n))$. ◀