Noisy (Binary) Searching: Simple, Fast and Correct

Dariusz Dereniowski
Faculty of Electronics, Telecommunications and Informatics,
Gdańsk University of Technology, Poland

Aleksander Łukasiewicz
Institute of Computer Science, University of Wrocław, Poland
Computer Science Institute of Charles University, Prague, Czech Republic

Przemysław Uznański
Institute of Computer Science, University of Wrocław, Poland

Abstract

This work considers the problem of the noisy binary search in a sorted array. The noise is modeled by a parameter p that dictates that a comparison can be incorrect with probability p, independently of other queries. We state two types of upper bounds on the number of queries: the worst-case and expected query complexity scenarios. The bounds improve the ones known to date, i.e., our algorithms require fewer queries. Additionally, they have simpler statements, and work for the full range of parameters. All query complexities for the expected query scenarios are tight up to lower order terms. For the problem where the target prior is uniform over all possible inputs, we provide an algorithm with expected complexity upperbounded by $(\log_2 n + \log_2 \delta^{-1} + 3)/I(p)$, where n is the domain size, $0 \leq p < 1/2$ is the noise ratio, and $\delta > 0$ is the failure probability, and $I(p)$ is the information gain function. As a side-effect, we close some correctness issues regarding previous work. Also, en route, we obtain new and improved query complexities for the search generalized to arbitrary graphs. This paper continues and improves the lines of research of Burnashev–Zigangirov [Prob. Per. Informatsii, 1974], Ben-Or and Hassidim [FOCS 2008], Gu and Xu [STOC 2023], and Emamjomeh-Zadeh et al. [STOC 2016], Dereniowski et al. [SOSA@SODA 2019].

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases Graph Algorithms, Noisy Binary Search, Query Complexity, Reliability

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.29

Related Version *Full Version*: `https://arxiv.org/abs/2107.05753`

Funding *Dariusz Dereniowski*: Partially supported by National Science Centre (Poland) grant number 2018/31/B/ST6/00820.
Aleksander Łukasiewicz: Partially supported by the ERC-CZ project LL2406 of the Ministry of Education of Czech Republic.
Przemysław Uznański: Partially supported by National Science Centre (Poland) grant number 2018/31/B/ST6/00820.

1 Introduction

1.1 Problem statement

An adaptive search problem for a general *search domain* $\mathcal{S}$ and an arbitrary *adversary* can be formulated as follows. The goal is to design an adaptive algorithm, also referred to as a *strategy*, that finds a *target* initially unknown to the algorithm. Adaptivity means that the subsequent actions of the algorithm depend on the answers already received. The process is divided into *steps*: in each step the algorithm performs a *query* and receives an *answer*. Each query-reply pair provides new information to the algorithm: it learns that some part of the search space $S \subseteq \mathcal{S}$ does not contain the target while its complement does. From both

theoretical and practical viewpoints, it is of interest to develop error-resilient algorithms for such a search process. This can be modeled, for example, by the presence of a probabilistic noise: each reply can be erroneous with some fixed probability $0 < p < \frac{1}{2}$, independently. The performance of a strategy is measured by the number of performed queries.

In this work, we focus on searching with probabilistic noise in two particular types of search domains. The first is a sorted array (which, in the absence of noise, would lead to the classical binary search problem). Formally, for a linear order $v_1 < \cdots < v_n$ with an unknown position of the target $v^* = v_j$, each query selects an element v_i, and the algorithm learns from the reply whether $v^* < v_i$ or $v^* \geq v_i$.

The second search domain we consider is a simple, undirected graph. More precisely, for an input graph G and an unknown target vertex v^*, each query selects some vertex v. The answer either states that v is the target or provides a neighbor u of v, that lies on a shortest path from v to v^*.

Searching through a graph can be viewed as a certain generalization of the former setting, as searching a linear order resembles searching a graph that is a path. However, it is important to note that the two models are not directly comparable, as in a graph search on a path there are three possible replies to a query, whereas in a search through an array, the answers are binary.

1.2 Overview of our results

To complete the search process, we need to learn roughly $\log_2 n$ bits of information (the identifier of the target). We can extract approximately $I(p) = 1 - H(p)$ bits of information from each reply, where $H(p) = -p \log_2 p - (1-p) \log_2(1-p)$ is the binary entropy function. Therefore we expect the optimal algorithm to use around $\frac{\log_2 n}{I(p)}$ queries. In an idealized scenario where there always exists a query that perfectly bisects the search space, regardless of the answer, one could achieve this bound. However, perfect bisection is typically not possible due to the discrete nature of the search space, which causes algorithms to lose some lower-order terms.

Our results are summarized in Table 1. We use the following naming convention: if we are interested in the worst-case analysis of the query complexity, we refer to it as the *worst case* setting throughout the paper. Conversely, if we analyze the expected number of queries made by an algorithm, we call this the *expected query complexity* setting. We note that in each setting the process is randomized due to answers of the adversary. Additionally, some of our algorithms use random bits as well.

The binary search algorithms referenced in the theorems below are detailed in Section 3: Algorithms 16 and 13 correspond to Theorems 1 and 3, respectively. Interestingly, both algorithms are essentially the same and differ only by the stopping condition.

▶ **Theorem 1.** *For any noise parameter $0 < p < \frac{1}{2}$ and a confidence threshold $0 < \delta < 1$, there exists a binary search algorithm that after the **expected** number of*

$$\frac{1}{I(p)} \left(\log_2 n + \log_2 \delta^{-1} + 3 \right)$$

*queries finds the target in any linear order correctly with probability at least $1 - \delta$, given that **the target position is chosen uniformly at random**.*

Using a previously known reduction from adversarial target placement to the uniformly random choice of a target ([3], see Lemma 10), we automatically obtain the following result for the adversarial version of the problem.

▶ **Corollary 2.** *For any noise parameter $0 < p < \frac{1}{2}$ and a confidence threshold $0 < \delta < 1$, there exists a binary search algorithm that after the **expected** number of*

$$\frac{1}{I(p)}\left(\log_2 n + \mathcal{O}(\log \delta^{-1})\right)$$

queries finds the target in any linear order correctly with probability at least $1 - \delta$.

▶ **Theorem 3.** *For any noise parameter $0 < p < \frac{1}{2}$ and a confidence threshold $0 < \delta < 1$, there exists a binary search algorithm for any linear order that after*

$$\frac{1}{I(p)}\left(\log_2 n + \mathcal{O}(\sqrt{\log n \log \delta^{-1}}) + \mathcal{O}(\log \delta^{-1})\right)$$

queries returns the target correctly with probability at least $1 - \delta$.

For graph searching we obtain two analogous results (detailed in Section 4): Algorithms 23 and 25 correspond to Theorems 5 and 4, respectively.

▶ **Theorem 4.** *For an arbitrary connected graph G, a noise parameter $0 < p < \frac{1}{2}$ and a confidence threshold $0 < \delta < 1$, there exists an graph searching algorithm that after the **expected** number of at most*

$$\frac{1}{I(p)}\left(\log_2 n + \mathcal{O}(\log \log n) + \mathcal{O}(\log \delta^{-1})\right)$$

queries returns the target correctly with probability at least $1 - \delta$.

▶ **Theorem 5.** *For an arbitrary connected graph G, a noise parameter $0 < p < \frac{1}{2}$ and a confidence threshold $0 < \delta < 1$, there exists an graph searching algorithm that after*

$$\frac{1}{I(p)}\left(\log_2 n + \mathcal{O}(\sqrt{\log n \log \delta^{-1}}) + \mathcal{O}(\log \delta^{-1})\right)$$

queries returns the target correctly with probability at least $1 - \delta$.

Table 1 The summary of the results.

Setting	*Binary search*	*Graph search*
Worst case query complexity:	$\frac{1}{I(p)}\left(\log_2 n + \mathcal{O}(\sqrt{\log n \log \delta^{-1}}) + \mathcal{O}(\log \delta^{-1})\right)$ (Thm. 3)	(Thm. 5)
Expected query complexity:	$\frac{1}{I(p)}(\log_2 n + \mathcal{O}(\log \delta^{-1}))$ (Cor. 2)	$\frac{1}{I(p)}(\log_2 n + \mathcal{O}(\log \log n) + \mathcal{O}(\log \delta^{-1}))$ (Thm. 4)

1.3 Comparison with previous works

1.3.1 Upper bounds for noisy binary search

Table 2 provides an overview of the known algorithms for noisy binary search that have complexity close to the optimal $\frac{\log_2 n}{I(p)}$. We provide a detailed comparison with our results below.

- Burnashev and Zigangirov [7] studied this problem from an information-theoretic perspective as early as 1974[1], in a setting where the location of the target element is chosen uniformly at random. We highlight that their query complexity is worse than ours by an additive term of $\log_2 \frac{1-p}{p}$, which tends to infinity when $p \to 0$. This behavior is rather unnatural, since when $p = 0$, the noise disappears, and the problem reduces to standard binary search.
- Feige et al. [15], in their seminal work, considered several problems in the noisy setting and, in particular, developed an asymptotically optimal algorithm for noisy binary search (with an adversarially placed target). However, their method intrinsically incurs a non-optimal constant in front of $\frac{\log_2 n}{I(p)}$.
- Later, Ben-Or and Hassidim [3], likely unaware of [7], developed algorithms with an expected query complexity of $\frac{1}{I(p)}(\mathcal{O}(\log n) + \mathcal{O}(\log\log n) + \mathcal{O}(\log \delta^{-1}))$. However, we claim that their proofs contain two *serious issues*.
  Firstly, in the proof of Lemma 2.6 in [3], they consider all the queries made by the algorithm throughout its execution and sort them by their positions. Then, the number of '$<$' answers in a fixed interval of positions is claimed to follow a binomial distribution. Notice, however, that while the answers to the particular queries are independent random variables, the *positions* of the queries depend on the answers to the previous queries, and the act of forgetting the order introduces correlation. To further illustrate this point, we note the rightmost query in their algorithm is guaranteed to have a '$<$' answer, because a '$\geq$' answer would have changed the weights maintained by the algorithm, causing the next query to be asked further to the right, which contradicts the assumption that this query is the rightmost.
  Secondly, the final expected number of steps is bounded by the ratio of total information needed to identify the target and the expected information gain per step (without any additional comments). However, the expected value does not work in this manner directly – to make this approach effective, one needs to employ additional probabilistic tools. Our paper uses Wald's identity for this purpose, see [19] for an example of the usage of martingales and the Optional Stopping Theorem. Moreover, the choice of a particular probabilistic theorem and the way this tool is handled may incur additional lower-order terms, making it unclear what the final complexity would be.

1.3.2 Reductions in complexity for expected length setting

Ben-Or and Hassidim in their work [3] showed a general technique that can transform any of the aforementioned algorithms (regardless if they are in the worst case or expected complexity setting) into an algorithm with the expected query complexity that is better by roughly a multiplicative factor of $(1 - \delta)$ at the cost of additive lower order term of order $\frac{\mathcal{O}(\log\log n)}{I(p)}$. Very recently Gu and Xu [19] showed how to improve that reduction in order to obtain a better constant in front of $\log \delta^{-1}$. They plug in our algorithm for noisy binary search (Corollary 2) as a black-box in order to get the $(1 + o(1))((1 - \delta)(\frac{\log_2 n}{I(p)} + \frac{\mathcal{O}(\log\log n)}{I(p)}) + \frac{\log_2 1/\delta}{(1-2p)\log\frac{1-p}{p}})$ expected query complexity.

[1] Curiously, it appears that until recently, this work has been largely unknown to the algorithmic community, despite the fact that the paper in question has over 100 citations. There is no mention of [7] in the well-known survey by Pelc [27], nor in the subsequent works that we reference. We suspect that the main reason for this oversight is that, until very recently, the work was only available in Russian. For the English translation of the algorithm and the proof, see [34].

Table 2 Upper bounds for noisy binary search.

Setting	*Query complexity*	*References and Notes*
Expected, uniform prior	$\frac{1}{I(p)}(\log_2 n + \log_2 \delta^{-1} + \log_2 \frac{1-p}{p})$	Burnashev and Zigangirov [7]
	$\frac{1}{I(p)}(\log_2 n + \mathcal{O}(\log\log n) + \mathcal{O}(\log \delta^{-1}))$	Ben-Or and Hassidim [3] (Correctness issues, see Sec.1.3.1).
	$\frac{1}{I(p)}\left(\log_2 n + \log_2 \delta^{-1} + 3\right)$	This work, Thm. 1.
Expected, adversarial target	$\frac{1}{I(p)}(\mathcal{O}(\log n) + \mathcal{O}(\log \delta^{-1}))$	Feige et al. [15]
	$\frac{1}{I(p)}(\log_2 n + \mathcal{O}(\log\log n) + \mathcal{O}(\log \delta^{-1}))$	Ben-Or and Hassidim [3] (Correctness issues, see Sec.1.3.1).
	$\frac{1}{I(p)}(\log_2 n + \mathcal{O}(\log_2 \delta^{-1}))$	This work, Cor. 2.
Worst case	$\frac{1}{I(p)}(\log_2 n + \mathcal{O}(\sqrt{\log n \log \delta^{-1}}) + \mathcal{O}(\log \delta^{-1}))$	This work, Thm. 3.

1.3.3 Upper bounds for noisy graph search

Known algorithms for noisy graph search are summarized in Table 3. The problem for arbitrary graphs was first considered by Emamjomeh-Zadeh et al.[14]. Later on Dereniowski et al. [11] simplified that algorithm and obtained an improved dependence on $\log \delta^{-1}$. We make another progress in that direction: we further simplify the algorithms and the analysis while simultaneously improving the dependence on $\log \delta^{-1}$ even further.

Table 3 Upper bounds for noisy graph search.

Setting	*Query complexity*	*References and Notes*
Expected	$\frac{1}{I(p)}(\log_2 n + \mathcal{O}(\log\log n) + \mathcal{O}(\log \delta^{-1}))$	This work, Thm. 4.
Worst case	$\frac{1}{I(p)}(\log_2 n + o(\log n) + \mathcal{O}(\log^2 \delta^{-1}))$	Emamjomeh-Zadeh et al., 2016 [14]
	$\frac{1}{I(p)}(\log_2 n + \mathcal{O}(\sqrt{\log n \log \delta^{-1}} \cdot \log \frac{\log n}{\log \delta^{-1}}) + \mathcal{O}(\log \delta^{-1}))$	Dereniowski et al., 2019 [11]
	$\frac{1}{I(p)}(\log_2 n + \mathcal{O}(\sqrt{\log n \log \delta^{-1}}) + \mathcal{O}(\log \delta^{-1}))$	This work, Thm. 5.

1.3.4 Lower bounds

For the expected complexity of noisy binary search, Ben-Or and Hassidim [3] established the first lower bound of $(1-\delta)\frac{\log_2 n - 10}{I(p)}$. Recently Gu and Xu [19] improved the lower bound to $(1-o(1))((1-\delta)\frac{\log_2 n}{I(p)} + \frac{\log_2 1/\delta}{(1-2p)\log\frac{1-p}{p}})$. However it works only for constant noise parameter p. They leave the question of improving the lower bound for an arbitrary p as an open problem.

Very recently, Gretta and Price [18] obtained a lower bound for the worst case setting of a more general problem (known as *Noisy Binary Search with Monotonic Probabilities*, which was introduced for the first time by [22]). For the worst case noisy binary search, their work implies a lower bound of the natural $\frac{\log_2 n}{I(p)}$. We note that we are not aware of any lower bounds for the lower order terms dependent on n in any of the considered settings.

1.4 Overview of the techniques

The core building block of our algorithms is Multiplicative Weights Update technique (MWU). This method has been employed in the past for noisy binary search and related problems [3, 5, 11, 22, 28]. The general outline of the method is as follows: we maintain weights that denote the "likelihood" of particular elements being the target and multiplicatively update them according to the answers to subsequent queries. After a certain number of steps, the algorithm returns the element with the highest weight, as it is the element we deem most likely to be the target.

The typical problem with this approach is that, at some point, we may encounter a situation where there is no good element to query, meaning that no query divides the search space close to the bisection. This usually occurs when a particular element becomes heavy, and subsequently querying this element yields less and less information. Previous works tried to different ways to resolve this issue, e.g. by ensuring that a good approximation of the target has been found and calling the algorithm recursively [3], introducing a phase with majority voting [14], etc.

We take a different approach by using a specifically tailored measure of progress of our algorithms, which differs from those used in previous works. For binary search, we define this measure as the total weight minus the weight of the target element. In the context of graph searching, the measure is slightly different – we use the total weight minus the weight of the heaviest vertex. This contrasts with the approach taken in prior studies, such as in [11], where the total weight itself was utilized. It is important to note that, in the case of graph search, the identity of the heaviest vertex may change throughout the search process, and at times, it may not even be the target vertex. However, our analysis guarantees the target will become the heaviest vertex by the end of the search, within the desired probability threshold.

This subtle change proves to be powerful and plays a vital role in all our proofs. We believe this is the key idea that enabled us to overcome the obstacles that the authors of the previous works might have faced.

Furthermore, in the case of binary search, when selecting which vertex to query, we employ a technique similar to that of [7]. Specifically, whenever we identify two elements that are closest to bisecting the search space, we randomly choose one of them with appropriate probability. Our analysis demonstrates that this effectively simulates the ideal subdivision of the search space and ensures the desired progress of our algorithm.

1.5 Other related work

There are many variants of the interactive query games, depending on the structure of queries and the way erroneous replies occur. The study of such games was initiated by Rényi [29] and Ulam [31]. A substantial amount of literature deals with a fixed number of errors for arbitrary membership queries or comparison queries; here we refer the reader to surveys [10, 27]. Among the most successful tools for tackling binary search with errors, is the idea of a volume [4, 28], which exploits the combinatorial structure of a possible distribution of errors. A natural approach of analyzing decision trees has been also successfully applied,

see e.g. [15]. See [5, 14] for examples of partitioning strategies into stages, where in each stage the majority of elements is eliminated and only few "problematic" ones remain. For a different reformulation (and asymptotically optimal results) of the noisy search see [22].

Although the adversarial and noisy models are most widely studied, some other ones are also considered. As an example, we mention the (linearly) bounded error model in which it is guaranteed that the number of errors is a r-fraction, $r < \frac{1}{2}$, of (any initial prefix of) the number of queries, see e.g. [1, 5, 12]. Interestingly, it might be the case that different models are so strongly related that a good bound for one of them provides also tight bounds for the other ones, see e.g. [11]. We refer the reader to distributional search where an arbitrary target distribution is known to the algorithm a priori [8, 9, 30]. A closely related theory of coding schemes for noisy communication is out of scope of this paper and we only point to some recent works [6, 16, 17, 20, 25].

The first steps towards generalizing binary search to graph-theoretic setting are works on searching in partially ordered data [2, 23, 24]. Specifically for the node search that we consider in this work, the first results are due to Onak and Parys for trees [26], where an optimal linear-time algorithm for error-less case was given.

2 Preliminaries

Whenever we refer to a *search space*, we mean either an (undirected and unweighted) graph or a linear order. Consequently, by an *element* of a search space, we refer to a vertex or an integer, respectively. In the following, let n denote the size of the search space, i.e., either the number of vertices in a graph or the number of integers in a linear order.

Throughout the search process, the strategies will maintain the weights $\omega(v)$ for the elements v of a search space V. For any $0 \leq c \leq 1$, v is *c-heavy* if $\omega(v)/\omega(V) \geq c$, where for any subset $U \subseteq V$ we write $\omega(U) = \sum_{u \in U} \omega(u)$. $\frac{1}{2}$-heavy elements play a special role and we call them *heavy* for brevity. The weight of an element v at the end of step t is denoted by $\omega_t(v)$, with $\omega_0(v)$ being the initial value. The initial values are set uniformly by putting $\omega_0(v) = 1$ for each v in our algorithms.

Noisy binary search definition and model specifics

In the *noisy binary search* problem, we operate on a linear order $v_1 < \cdots < v_n$. We are given an element v^* and the values $p \in [0, \frac{1}{2}), \delta \in (0, 1)$ as input. We are promised that there exists some $i \in [n]$ such that $v^* = v_i$. We call v^* the *target* element. We can learn about the search space by asking if $v^* < v_j$ for any $j \in [n]$, and receiving an answer that is correct with probability $1 - p$, independently for each query. The goal is to design an algorithm that finds the $i \in [n]$ such that $v^* = v_i$, and returns this index correctly with probability at least $1 - \delta$. We strive to minimize the number of queries performed in the process.

We adopt the following naming convention for query results. When we ask if $v^* \stackrel{?}{<} v_i$ and receive an affirmative answer (i.e., v^* is less than v_i), we call it a *yes-answer*. If the reply indicates v^* is greater than or equal to v_i (i.e., a negative answer), we call it a *no-answer*. An element v_j of a search space is considered *compatible* with the reply to a query $v^* \stackrel{?}{<} v_i$ if and only if:

- For a yes-answer (indicating $v^* < v_i$), $j < i$.
- For a no-answer (indicating $v^* \geq v_i$), $j \geq i$.

Noisy graph searching definition and model specifics

In the *noisy graph searching* problem we are given an unweighted, undirected, simple graph G and the values $p \in [0, \frac{1}{2}), \delta \in (0, 1)$. We know that one vertex v^* of G is marked as the *target*, but we don't know which one is it.

We can query the vertices of G, upon querying a vertex q we get one of two possible answers:

- $v^* = q$, i.e. the queried vertex is the target. We call it a *yes-answer*.
- $v^* \neq q$, but some neighbor u of q lies on a shortest path from q to v^*. We call it a *no-answer*. If there are multiple such neighbors (and hence shortest paths), then we can get an arbitrary one as an answer.

In fact, we assume for simplicity that each reply is given as a single vertex u. If $u = q$, then we interpret it as a yes-answer. If $u \neq q$, then u is a neighbor of q that lies on a shortest path from q to v^*. Again, we are interested in the noisy setting, therefore every reply is correct independently with probability $1 - p$. Observe that if the answer is *incorrect* then it can come in different flavors:

- if $q = v^*$, then an incorrect answer is any neighbor u of q,
- if $q \neq v^*$, then an incorrect answer may be either q or any neighbor of q that does not lie on a shortest path from q to v^*.

Clearly, in both cases there might be several possible vertices that constitute an incorrect answer. Here we assume the strongest possible model where every time the choice among possible incorrect replies is made adversarially and independently for each query. The goal is, similarly as in noisy binary search, to design an algorithm that finds a target correctly with probability at least $1 - \delta$ and minimizes the number of queries. In fact, in this work we operate in a slightly weaker model of replies (as compared to [11, 13, 14, 26]) in which an algorithm receives less information in some cases. This is done in somewhat artificial way for purely technical reasons, i.e., to simplify several arguments during analysis. The only change to the model we have just described happens when we query a vertex that is heavy at the moment and a *no-answer* has been received. More specifically, if a *heavy* q is queried and a *no-answer* is given, the algorithm reads this reply as: the *target is not* q (ignoring the direction the target might be). Observe that this only makes our algorithms stronger, since they operate in a weaker replies model and any algorithmic guarantees for the above model carry over to the generic noisy graph search model.

Similarly to the case of noisy binary search, we say that a vertex v is *compatible* with the reply to the query q if and only if:

- $v = q$ in case of a yes-answer.
- The neighbor u given as a no-answer lies on a shortest path from q to v and q was not heavy.
- $v \neq q$ in case of a no-answer when q was heavy.

Common mathematical tools and definitions

We adopt the notation from [11] and denote $\varepsilon = \frac{1}{2} - p$ and $\Gamma = \frac{1-p}{p}$. These quantities appear frequently throughout the proofs, and this notation helps to make the presentation more concise.

The *information function*, denoted by $I(p)$, appears in all our running times. It is defined as follows $I(p) = 1 - H(p) = 1 + p \log_2 p + (1-p) \log_2 (1-p)$. In the analysis of our algorithms we frequently use the following quantitative fact about $I(p)$ and $\log_2 \Gamma$.

▶ **Proposition 6.** *We have* $I(p) = \Omega(\varepsilon^2)$ *and* $(\log_2 \Gamma)^2 p(1-p) = \mathcal{O}(\varepsilon^2)$.

Proof. We first observe that by Taylor's series expansion (which can be derived using elementary calculus, see e.g. [32]):

$$I(p) = \frac{1}{2\ln 2} \sum_{n=1}^{\infty} \frac{(2\varepsilon)^{2n}}{n(2n-1)} \geq \frac{4\varepsilon^2}{2\ln 2}.$$

To show the second bound we compute

$$p(1-p)(\log_2 \Gamma) = (1/2-\varepsilon)(1/2+\varepsilon)\left(\log_2 \frac{1+2\varepsilon}{1-2\varepsilon}\right)^2$$

$$= (1-4\varepsilon^2)\left(\tanh^{-1} 2\varepsilon\right)^2 \frac{1}{(\ln 2)^2} \leq \frac{4}{(\ln 2)^2}\varepsilon^2$$

where the last step follows by observing that under the substitution $\varepsilon = 1/2 \cdot \tanh \gamma$ it reduces to $\gamma^2 \leq \sinh^2 \gamma$ and that inequality follows immediately from the Taylor expansion of $\sinh^2 \gamma$. ◀

Let $(X_m)_{m\in\mathbb{N}}$ be a sequence of i.i.d random variables. We say that a random variable T is a *stopping time* (with respect to $(X_m)_{m\in\mathbb{N}}$) if $\mathbb{1}_{\{T\leq m\}}$ is a function of $X_1, X_2, \ldots, X_m$ for every m.

We will use the following version of the Wald's identity.

▶ **Proposition 7** (Wald's Identity [33])**.** *Let* $(X_m)_{m\in\mathbb{N}}$ *be i.i.d with finite mean, and* T *be a stopping time with* $\mathbb{E}[T] < \infty$. *Then* $\mathbb{E}[X_1 + \cdots + X_T] = \mathbb{E}[X_1]\mathbb{E}[T]$.

Let us also recall a basic version of a Hoeffding bound, which we use in our calculations of query complexities in the worst case setting.

Multiplicative Weights Update Method

The core building block of our strategies for both binary and graph search is a standard Multiplicative Weights Update (MWU) technique. Below we formally define the version of MWU that we use in our algorithms (Algorithm 8).

▶ **Algorithm 8.** (MWU updates.)

In a step $t+1$, *for each element* v *of the search space do:*
if v *is compatible with the answer, then* $\omega_{t+1}(v) \leftarrow \omega_t(v) \cdot 2(1-p)$,
if v *is not compatible with the answer, then* $\omega_{t+1}(v) \leftarrow \omega_t(v) \cdot 2p$.

Directly from the statement of our MWU method we obtain the following bound on the weight of the target. This bound applies to both binary and graph search, as the analysis is based solely on the number of erroneous replies and the fact that the target is always compatible with a correct answer.

▶ **Lemma 9.** *If* v^* *is the target, then after* τ *queries, with probability at least* $1-\delta$ *it holds*

$$\omega_\tau(v^*) \geq \Gamma^{-\sqrt{2p(1-p)\tau \ln \delta^{-1}}} 2^{I(p)\tau}.$$

Proof. After τ queries with at most ℓ erroneous replies, the weight of the target satisfies:

$$\omega_\tau(v^*) \geq (2p)^\ell (2(1-p))^{\tau-\ell} = \Gamma^{p\tau-\ell} 2^{I(p)\tau}.$$

Denote $a = \sqrt{2p(1-p)\tau \ln \delta^{-1}}$. Then by Chernoff-Hoeffding bound [21], with probability at most δ there is $\ell - p\tau \geq a$. Thus, after τ queries, with probability at least $1-\delta$ the weight of the target satisfies $\omega_\tau(v^*) \geq \Gamma^{-a} 2^{I(p)\tau}$. ◀

Uniform prior for binary search

One can assume that the distribution of the target element in noisy binary search is *a priori* uniform by using a shifting trick described by Ben-Or and Hassidim [3]. We formally state it as a lemma below.

▶ **Lemma 10** (c.f. [3]). *Assume that the target element in noisy binary search problem was chosen adversarially. One can reduce that problem to the setting where the target element is chosen uniformly at random using* $\mathcal{O}(\frac{\log \delta^{-1}}{I(p)})$ *queries.*

3 Binary Search Algorithm

Each query performs the MWU updates using Algorithm 8. The element to be queried is selected as follows: let k be such that $\sum_{i=1}^{k-1} \omega(v_i) \le \omega(V)/2$ and $\sum_{i=1}^{k} \omega(v_i) \ge \omega(V)/2$. Since the queries v_k and v_{k+1} are the closest possible to equi-division of the total weight, the algorithm chooses one of those with appropriate probability (cf. Algorithm 11.)

▶ **Algorithm 11.** (Query selection procedure.)

In step τ*: let* k *be such that* $\sum_{i=1}^{k-1} \omega_\tau(v_i) \le \frac{\omega_\tau(V)}{2} \le \sum_{i=1}^{k} \omega_\tau(v_i)$.
Then, query v_k *with probability* $\frac{1}{2\omega_\tau(v_k)}(\sum_{i=1}^{k} \omega_\tau(v_i) - \sum_{i=k+1}^{n} \omega_\tau(v_i))$*, and otherwise query* v_{k+1}.

In order to turn Algorithm 11 into a particular strategy, we will provide a stopping condition for each model. We start by determining the expected weight preservation during the search.

▶ **Lemma 12.** $\mathbb{E}[\omega_{\tau+1}(V \setminus \{v^*\}) \mid \omega_\tau(V \setminus \{v^*\})] \le \omega_\tau(V \setminus \{v^*\})$.

Proof. We consider three cases, and show that this bound holds in each of those independently. Denote $A = \sum_{i=1}^{k-1} \omega_\tau(v_i)$, $B = \omega_\tau(v_k)$ and $C = \sum_{i=k+1}^{n} \omega_\tau(v_i)$. Denote the probability of querying v_k as $\alpha = \frac{A+B-C}{2B}$, and the probability of querying v_{k+1} as $\beta = \frac{C+B-A}{2B}$.

Case 1: $v^* = v_k$.

$$\begin{aligned}\mathbb{E}[\omega_{\tau+1}(V\setminus\{v^*\}) \mid \omega_\tau(V \setminus \{v^*\})] &= \alpha[2p^2C + 2(1-p)^2C + 2p(1-p)A + 2p(1-p)A] \\ &\quad +\beta[2p^2A + 2(1-p)^2A + 2p(1-p)C + 2p(1-p)C] \\ &= \alpha[(1+4\varepsilon^2)C + (1-4\varepsilon^2)A] + \beta[(1+4\varepsilon^2)A + (1-4\varepsilon^2)C].\end{aligned}$$

Using the definition of α and β, we obtain

$$\begin{aligned}\mathbb{E}[\omega_{\tau+1}(V \setminus \{v^*\}) \mid \omega_\tau(V \setminus \{v^*\})] &= -4\varepsilon^2\frac{(A-C)^2}{B} + (A+C) \\ &= -4\varepsilon^2\frac{(A-C)^2}{B} + \omega_\tau(V \setminus \{v^*\}).\end{aligned}$$

Case 2: $v^* < v_k$. In this case,

$$\begin{aligned}\mathbb{E}[\omega_{\tau+1}(V \setminus \{v^*\}) \mid \omega_\tau(V \setminus \{v^*\})] &= (2p^2 + 2(1-p)^2)(A - \omega_\tau(v^*)) \\ &\quad +[\alpha 4p(1-p) + \beta(2p^2 + 2(1-p)^2)]B + 4p(1-p)C.\end{aligned}$$

Denote $p_1 = p^2 + (1-p)^2$ and $p_2 = 2p(1-p)$. Observe $p_1 + p_2 = 1$ and $p_1 \ge \frac{1}{2} \ge p_2$. Then,

$$\begin{aligned}\mathbb{E}[\omega_{\tau+1}(V \setminus \{v^*\}) \mid \omega_\tau(V \setminus \{v^*\})] &= 2p_1A + (C+B-A)p_1 + (A+B-C)p_2 + 2p_2C - 2p_1\omega_\tau(v^*) \\ &= A + B + C - 2p_1\omega_\tau(v^*) \\ &\le \omega_\tau(V \setminus \{v^*\}).\end{aligned}$$

Case 3: $v^* > v_{k+1}$ is symmetric to case 2. ◀

3.1 Proof of Theorem 1 (The expected strategy length)

▶ **Algorithm 13.** (The expected strategy length for binary search.)

Initialization: $\omega_0(v_i) \leftarrow 1$ *for each* $v_i \in V$.
Execute Algorithm 11 until in some step τ *it holds* $\frac{\omega_\tau(v_j)}{\omega_\tau(V)} \geq 1-\delta$ *for some* v_j.
Return v_j.

To show correctness of Algorithm 13, we need to observe that in the case of binary search, our MWU updates are, in fact Bayesian updates, that is, the normalized weights follow a posterior distribution conditioned on the replies seen so far. We state this as a lemma below.

▶ **Lemma 14.** *After any step* τ *of Algorithm 13 we have for every* $v_i \in V$

$$\Pr[v^* = v_i \mid \tau \text{ observed replies}] = \frac{\omega_\tau(v_i)}{\omega(V)}.$$

Proof. The proof is by induction on τ. The base case is trivial, because we assign the weights uniformly in the initialization step. The inductive step follows immediately from our definition of MWU updates (Algorithm 8) and the Bayes' rule. ◀

The correctness of Algorithm 13 follows directly from Lemma 14 and the stopping condition. To prove Theorem 1 it remains to analyze the expected length of the strategy.

▶ **Lemma 15.** *Algorithm 13 terminates after the expected number of at most* $\frac{1}{I(p)}(\log_2 n + \log_2 \frac{1}{\delta} + 3)$ *steps.*

Proof. We measure the progress at any given step by a random variable $\zeta_t = \log_2 \omega_t(v^*)$. If the answer in step $t+1$ is erroneous, then $\zeta_{t+1} = \zeta_t + 1 + \log_2 p$ and otherwise $\zeta_{t+1} = \zeta_t + 1 + \log_2(1-p)$.

For the sake of bounding the number of steps of the algorithm, we consider it running indefinitely. Let Q be the smallest integer such that $\frac{\omega_Q(v^*)}{\omega_Q(V\setminus\{v^*\})} \geq \frac{1-\delta}{\delta}$, that is v^* is $(1-\delta)$-heavy in round Q. Obviously Q upper bounds the strategy length. By the definition, Q is a stopping time.

First, let us show that $\mathbb{E}[Q]$ is finite. To this end, let $\xi_t = \log_2 \frac{\omega_t(v^*)}{\omega_t(V\setminus\{v^*\})}$ and $X_t = \xi_t - \xi_{t-1}$ for $t \geq 1$. Observe that X_t's are i.i.d and $\mathbb{E}[X_t] = p(-\log_2 \Gamma) + (1-p)\log_2 \Gamma = (1-2p)\log_2 \Gamma > 0$. Therefore, using Lemma 29 for sequence X_t, with $\ell = -\log\Gamma$, $r = \log\Gamma$ and $T = \log_2 \frac{1-\delta}{\delta} - \log_2 \frac{\omega_0(v^*)}{\omega_0(V\setminus\{v^*\})}$, we indeed obtain $\mathbb{E}[Q] < \infty$.

Let Q_i for any positive integer i be smallest value such that $\omega_{Q_i}(v^*) \geq \frac{n}{\delta}\cdot 2^i$ (for completeness of notation, we define $Q_0 = 0$). Consider an event $\{Q > Q_i\}$. It means that in round Q_i the target v^* is not yet $(1-\delta)$-heavy. Hence, $\omega_{Q_i}(V\setminus\{v^*\}) > \frac{\delta}{1-\delta}\omega_{Q_i}(v^*) \geq \delta\omega_{Q_i}(v^*) \geq n\cdot 2^i$. But we know from Lemma 12 that $\mathbb{E}[\omega_{Q_i}(V\setminus\{v^*\})] \leq \mathbb{E}[\omega_0(V\setminus\{v^*\})] \leq n$. Using Markov's inequality we conclude that $\Pr[Q > Q_i] \leq \Pr[\omega_{Q_i}(V\setminus\{v^*\}) > n\cdot 2^i] \leq 2^{-i}$.

Additionally, since $\omega_{Q_i-1}(v^*) < \frac{n}{\delta}\cdot 2^i$, there is $\omega_{Q_i}(v^*) < 2\frac{n}{\delta}\cdot 2^i$. We can then bound

$$\begin{aligned}
\mathbb{E}[\zeta_Q] &< \sum_{i=1}^{\infty} \Pr[Q_{i-1} < Q \leq Q_i] \log_2(2\cdot\frac{n}{\delta}2^i) \\
&= \log_2(2\cdot\frac{n}{\delta}) + \sum_{i=1}^{\infty} \Pr[Q_{i-1} < Q \leq Q_i]\cdot i \\
&= \log_2(2\frac{n}{\delta}) + \sum_{i=1}^{\infty} \Pr[Q > Q_{i-1}]
\end{aligned}$$

$$\begin{aligned} &\leq 1 + \log_2 n + \log_2 \frac{1}{\delta} + \sum_{i=0}^{\infty} 2^{-i} \\ &\leq \log_2 n + \log_2 \frac{1}{\delta} + 3. \end{aligned}$$

Let $Y_t = \zeta_t - \zeta_{t-1}$. Obviously, Y_i's are independent. We have already established that Q is a stopping time and that $\mathbb{E}[Q] < \infty$. This means we can employ the Wald's identity (Proposition 7) to obtain (using $\zeta_0 = 0$) $\mathbb{E}[\zeta_Q] = \mathbb{E}[Y_1 + \cdots + Y_Q] = \mathbb{E}[Q]I(p)$. Therefore,

$$\log_2 n + \log_2 \frac{1}{\delta} + 3 \geq \mathbb{E}[\zeta_Q] = \mathbb{E}[Q]I(p). \qquad \blacktriangleleft$$

3.2 Proof of Theorem 3 (Worst-case strategy length)

Take Q to be the smallest positive integer for which

$$I(p)Q > \log_2 n + \log_2 \frac{2}{\delta} + \sqrt{2p(1-p)Q \ln \frac{2}{\delta}} \log_2 \Gamma. \tag{1}$$

The Q gives our strategy length (see Algorithm 16). To prove Theorem 3 we bound the strategy length and the failure probability (see Lemma 17 below). The algorithm essentially remains the same except for the stop condition (cf. Algorithm 16).

▶ **Algorithm 16.** (Worst-case strategy length for binary search.)

Initialization: $\omega_0(v_i) \leftarrow 1$ *for each element* v_i.
Execute Algorithm 11 for exactly Q *steps with* Q *as in* (1).
Return the heaviest element.

▶ **Lemma 17.** *For any* $0 < \delta < 1$, *Algorithm 16 finds the target correctly with probability at least* $1 - \delta$ *in* $\frac{1}{I(p)} \left(\log_2 n + \mathcal{O}(\log \delta^{-1}) + \mathcal{O}(\sqrt{\log n \log \delta^{-1}}) \right)$ *steps.*

Proof. Firstly, observe that solving (1) for Q using Lemma 28 with parameters $a = I(p)$, $b = \log_2 n + \log_2 \frac{2}{\delta}$ and $c = 2p(1-p)(\log_2 \Gamma)^2 \ln \frac{2}{\delta}$ yields

$$Q = \frac{1}{I(p)} \left(\log_2 n + \log_2 \frac{2}{\delta} + \mathcal{O}\left(\ln \frac{2}{\delta} + \sqrt{\log_2 n + \log_2 \frac{2}{\delta}} \sqrt{\ln \frac{2}{\delta}} \right) \right)$$

where we have used, by Proposition 6, that $\frac{p(1-p)(\log_2 \Gamma)^2}{I(p)} = \mathcal{O}(1)$. The above equation can be simplified to

$$Q = \frac{1}{I(p)} \left(\log_2 n + \mathcal{O}(\log \delta^{-1}) + \mathcal{O}(\sqrt{\log \delta^{-1} \log n}) \right).$$

It remains to prove correctness. By Lemma 9, with probability at least $1 - \delta/2$ we have

$$\omega_Q(v^*) \geq \Gamma^{-\sqrt{2p(1-p)Q \ln 2/\delta}} 2^{I(p)Q}. \tag{2}$$

From Lemma 12 we also get $\mathbb{E}[\omega_Q(V \setminus \{v^*\})] \leq \mathbb{E}[\omega_0(V \setminus \{v^*\})] \leq n$. Therefore, by Markov's inequality with probability $1 - \delta/2$ we have

$$\omega_Q(V \setminus \{v^*\}) \leq \frac{2n}{\delta}. \tag{3}$$

It remains to observe that our definition of Q (Equation (1)) is equivalent to

$$\Gamma^{-\sqrt{2p(1-p)Q\ln 2/\delta}}2^{I(p)Q} > \frac{2n}{\delta}.$$

Thus, by the union bound applied to (2) and (3), with probability at least $1-\delta$ we get $\omega_Q(v^*) > \omega_Q(V \setminus \{v^*\})$. Then, v^* is the heaviest element and Algorithm 16 returns it. ◀

4 Graph Searching Algorithm

Denoting by $d(u,v)$ the graph distance between u and v, i.e., the length of the shortest path between these vertices, a *median* of the graph is a vertex

$$q = \arg\min_{v\in V} \sum_{u\in V} d(u,v)\cdot\omega(u).$$

For a query v and a reply u, let $C(v,u) = \{x \in V | u \text{ lies on some shortest path from } v \text{ to } x\}$ for $v \neq u$, and $C(v,v) = \{v\}$. We note a fundamental bisection property of a median:

▶ **Lemma 18** (cf. [14] Lemma 4). *If q is a median, then $\max_{u\in N(q)} \omega(C(q,u)) \leq \omega(V)/2$.*

Proof. Denote for brevity $\Phi(x) = \sum_{v\in V} d(x,v)\cdot\omega(v)$ for any $x \in V$. Suppose towards the contradiction that $C(q,u) > \omega(V)/2$ for some $u \in N(q)$. Observe that $\Phi(u) \leq \Phi(q) - \omega(C(q,u)) + \omega(V \setminus C(q,u))$ since by moving from q to u we get closer to all vertices in $C(q,u)$. But $\Phi(q) - \omega(C(q,u)) + \omega(V \setminus C(q,u)) = \Phi(q) + \omega(V) - 2\omega(C(q,u)) < \Phi(q)$ by our assumption, hence $\Phi(u) < \Phi(v)$, which yields a contradiction. ◀

We now analyze how the weights behave when in each step a median is queried and the MWU updates are made. This analysis is common for both graph searching algorithms given later. Essentially we prove that, in an amortized way, the total weight (with the heaviest vertex excluded) remains the same in each step. In absence of heavy vertices we use Lemma 19. Lemmas 20 and 21 refer to an interval of queries to the same heavy vertex x. If the interval ends (cf. Lemma 20), then the desired weight drop can be claimed at its end. For this, informally speaking, the crucial fact is that x received many no-answers during this interval. If a strategy is at a step that is within such interval, then Lemma 21 is used to bound the total weight with the weight of x excluded. Hence, at any point of the strategy the weight behaves appropriately, as summarized in Lemma 22.

▶ **Lemma 19** (see also [11, 14]). *If in a step t there is no heavy vertex, then $\omega_{t+1}(V) \leq \omega_t(V)$.*

Proof. Let q be a query and u an answer in step t. Note that if there is no heavy vertex, then $C(q,u)$ is the set of vertices compatible with the reply. If $q \neq u$ then $\omega_t(C(q,u)) \leq \omega_t(V)/2$ by Lemma 18 and in case $q = u$ we have $C(q,u) = \{q\}$ and thus the same bound holds. Then in both cases, $\omega_{t+1}(V) = 2(1-p)\cdot\omega_t(C(q,u)) + 2p\cdot\omega_t(V \setminus C(q,u)) \leq \omega_t(V)$. ◀

▶ **Lemma 20** (see also [11]). *Consider an interval $I = \{\tau, \tau+1, \ldots, \tau+k-1\}$ of k queries such that some x is heavy in each query in I and is not heavy after the last query in the sequence. Then $\omega_{\tau+k}(V) \leq \omega_\tau(V)$.*

Proof. First note that in each query in the interval I the queried vertex is x. Consider any two queries i and j in I such that they receive different replies. The contribution of these two queries is that together they scale down each weight multiplicatively by $2p\cdot 2(1-p) \leq 1$. Also, for a single no-answer in a query $i \in I$ we get $\omega_{i+1}(V) = 2p\omega_i(x) + 2(1-p)\omega_i(V \setminus \{x\}) \leq \omega_i(V)$

because $\omega_i(x) \geq \omega_i(V \setminus \{x\})$ for the heavy vertex x. By assumption, the number of no-answers is at least the number of yes-answers in I. Thus, the overall weight drop is as claimed in the lemma. ◀

▶ **Lemma 21.** *Consider an interval $I = \{\tau, \tau+1, \ldots, \tau+k-1\}$ of k queries such that some x is heavy in each query in I, and x remains heavy after the last query in I. Then*

$$\omega_{\tau+k}(V \setminus \{x\}) \leq \omega_\tau(V).$$

Proof. Recall that in each query in the interval I, the queried vertex is x. Assume that there were a yes-answers in I and b no-answers, with $a+b=k$. If $a \geq b$, then $\omega_{\tau+k}(V \setminus \{x\}) = (2p)^a(2(1-p))^b\omega_\tau(V \setminus \{x\}) \leq \omega_\tau(V \setminus \{x\}) \leq \omega_\tau(V)$. If $a < b$, then we bound as follows: $\omega_{\tau+k}(V \setminus \{x\}) \leq \omega_{\tau+k}(x) = (2p)^b(2(1-p))^a\omega_\tau(x) \leq \omega_\tau(x) \leq \omega_\tau(V)$. ◀

The bound in the next lemma immediately follows from Lemmas 19, 20 and 21. We say that an element v is *heaviest* if $\omega(v) \geq \omega(u)$ for each $u \in V$. For each step i, we denote by x_i a heaviest vertex at this step, breaking ties arbitrarily.

▶ **Lemma 22.** $\omega_\tau(V \setminus \{x_\tau\}) \leq \omega_0(V) = n$.

Proof. We consider the first τ queries and observe that they can be partitioned into a disjoint union of maximal intervals in which either there is a heavy vertex present (in the whole interval) or there is no heavy vertex (in the whole interval). We apply Lemma 19 for intervals with no heavy vertex and Lemmas 20, 21 otherwise (note that Lemma 21 can be applied only to the last interval. The latter happens only when there exists a heavy vertex after we perform all τ queries). ◀

4.1 Proof of Theorem 5 (Worst-case strategy length)

In this section we prove Theorem 5. Take Q to be the smallest positive integer for which

$$I(p)Q \geq \log_2 n + \sqrt{2p(1-p)Q \ln \delta^{-1}} \log_2 \Gamma. \tag{4}$$

The Q gives our strategy length (see Algorithm 23). To prove Theorem 5 we bound the strategy length and the failure probability (see Lemma 24 below).

▶ **Algorithm 23.** (Worst-case strategy length for graph search.)

Initialization: $\omega_0(v) = 1$ *for each* $v \in V$.
In each step: query the median and perform the MWU updates (Algorithm 8).
Stop condition: do exactly Q queries with Q defined by (4) *and return the heaviest vertex.*

▶ **Lemma 24.** *For any $0 < \delta < 1$, Algorithm 23 finds the target correctly with probability at least $1-\delta$ in $\frac{1}{I(p)}\left(\log_2 n + \mathcal{O}(\log \delta^{-1}) + \mathcal{O}(\sqrt{\log n \log \delta^{-1}})\right)$ steps.*

Proof. The proof is very similar to that of Lemma 17 (worst-case noisy binary search). It is actually simpler, thanks to the fact that Lemma 22 gives even stronger bound than Lemma 12.

Using Lemma 28 we solve (4) for Q. We bound the result further with Proposition 6:

$$Q = \frac{\log_2 n + \mathcal{O}(\sqrt{\log n \log \delta^{-1}}) + \mathcal{O}(\log \delta^{-1})}{I(p)}. \tag{5}$$

By Lemma 9 and the definition of Q in (4) it holds with probability $1-\delta$

$$\log_2 \omega_Q(v^*) \geq -\sqrt{2p(1-p)Q\ln\delta^{-1}}\log_2\Gamma + I(p)Q \geq \log_2 n \geq \log_2 \omega_Q(V\setminus\{x_Q\}),$$

where the last inequality is due to Lemma 22. Since the weights are non-negative at all times, the only way for this to happen is to have $v^* = x_Q$, that is the target being found correctly. ◀

4.2 Proof of Theorem 4 (The expected strategy length)

The solution for this case (given in Algorithm 25) paraphrases Algorithm 11 except for the proper adjustement of the confidence threshold.

▶ **Algorithm 25.** (The expected strategy length for graph search.)

Let $\delta' = c(\delta^2\cdot(\log n + \log 1/\delta)^{-2})$ *for small enough constant* $c>0$.
Initialization: $\omega_0(v)=1$ *for each* $v\in V$.
In each step: query the median and perform the MWU updates (Algorithm 8).
Stop condition: if for any v *in some step* τ *it holds* $\frac{\omega_\tau(v)}{\omega_\tau(V)}\geq 1-\delta'$, *then return* v.

▶ **Lemma 26.** *Algorithm 25 stops after the expected number of at most* $\frac{1}{I(p)}(\log_2 n+\log_2\frac{1}{\delta'}+1)$ *steps.*

Proof. We argue that within the promised expected number of steps, target v^* reaches the threshold weight. This clearly upperbounds the runtime. We measure the progress at any given moment by a random variable $\zeta_t = \log_2\omega_t(v^*)$. Observe that if the reply is erroneous in a step $t+1$, then $\zeta_{t+1}=\zeta_t+1+\log_2 p$, and if it is correct, then $\zeta_{t+1}=\zeta_t+1+\log_2(1-p)$.

For the sake of bounding the number of steps of the algorithm, we assume it is simulated indefinitely. Let Q be the smallest integer such that $\zeta_Q \geq \log_2 n + \log_2\frac{1-\delta'}{\delta'}$.

By Lemma 22 we have that $\zeta_Q = \log_2\omega_Q(v^*) \leq \log_2\frac{\omega_Q(v^*)}{\omega(V\setminus\{x_Q\})/n}$, thus $\frac{\omega_Q(v^*)}{\omega(V\setminus\{x_Q\})}\geq\frac{1-\delta'}{\delta'}>1$ since w.l.o.g. $\delta'<1/2$. But if for any t there is $\frac{\omega_t(v^*)}{\omega_t(V\setminus\{x_t\})}>1$, then $x_t=v^*$, since $\omega_t(v^*)>\omega_t(V\setminus\{x_t\})$ implies $v^*\notin V\setminus\{x_t\}$. Thus we deduce that $x_Q=v^*$. Additionally, from $\frac{\omega_Q(v^*)}{\omega(V\setminus\{v^*\})}\geq\frac{1-\delta'}{\delta'}$ we get that v^* is $(1-\delta')$-heavy, hence Q bounds the strategy length.

From $\zeta_{t+1}\in\{\zeta_t+1+\log_2 p, \zeta_t+1+\log_2(1-p)\}$ and the minimality of Q we deduce $\zeta_Q\leq\log_2 n+\log_2\frac{1-\delta'}{\delta'}+1+\log_2(1-p)\leq\log_2 n+\log_2\frac{1}{\delta'}+1$. In particular

$$\mathbb{E}[\zeta_Q]\leq\log_2 n+\log_2\frac{1}{\delta'}+1. \tag{6}$$

Let $X_t=\zeta_t-\zeta_{t-1}$ and observe that $\mathbb{E}[X_t]=p(1+\log_2 p)+(1-p)(1+\log_2(1-p))=I(p)>0$. Also, X_i's are independent and Q is a stopping time. Finally, we have $\mathbb{E}[Q]<\infty$ from Lemma 29 [2]. Therefore, we meet all conditions of the Wald's identity (Proposition 7) and we get (since $\zeta_0=0$) $\mathbb{E}[\zeta_Q]=\mathbb{E}[X_1+\cdots+X_Q]=\mathbb{E}[Q]I(p)$. Thus, by (6) we have $1+\log_2\frac{1}{\delta'}+\log_2 n\geq\mathbb{E}[\zeta_Q]=\mathbb{E}[Q]I(p)$, from which the claim follows. ◀

▶ **Lemma 27.** *Algorithm 25 finds the target correctly with probability at least* $1-\delta$.

[2] By plugging in $\ell=1+\log_2 p$, $r=1+\log_2(1-p)$ and $T=\log_2\frac{1-\delta'}{\delta'}+\log_2 n$.

Proof. We first show correctness. Denote by $A \leq \frac{\log \frac{1-\delta'}{\delta'}}{\log \frac{1-p}{p}} + 1$ the number of yes-answers required to go from a vertex being 1/2-heavy to being $(1-\delta')$-heavy. For now assume that $A \geq 2$, we will deal with the other case later. For a non-target vertex u to be declared by the algorithm as the target, it has to observe a suffix of the strategy being a random walk on a 1-dimensional discrete grid $[0, \ldots, A]$ and transition probabilities p for $i \to i+1$ and $1-p$ for $i \to i-1$. We consider a random walk starting at position $A/2$ and ending when reaching either 0 or A and call it a *subphase* (w.l.o.g. assume that A is even). Any execution of the algorithm can be partitioned into maximal in terms of containment, disjoint subphases. Each subphase starts when one particular heavy vertex v receives $A/2$ more yes-answers than no-answers within the interval in which v is heavy. Then, a subphase ends when either the algorithm declares v to be the target or v stops being heavy. By the standard analysis of the gamblers ruin problem, each subphase (where the heavy vertex is not the target) has failure probability $\delta'' = \frac{1}{1+\left(\frac{1-p}{p}\right)^{A/2}} \leq \frac{1}{1+\sqrt{\frac{1-\delta'}{\delta'}}} = O(\sqrt{\delta'})$. Let us denote by a random variable D the number of subphases in the execution of the algorithm. Let F_i be the length of i-th subphase. By the standard analysis of the gamblers ruin problem,

$$\mathbb{E}[F_i] = \frac{A/2}{1-2p} - \frac{A}{1-2p}\frac{1}{1+(\frac{1-p}{p})^{A/2}} \geq \frac{A/2}{1-2p}\left(1 - \frac{2}{1+\sqrt{\frac{1-\delta'}{\delta'}}}\right) = \Omega\left(\frac{1}{\varepsilon^2}\right),$$

where the asymptotic holds since w.l.o.g. $\delta' < 1/3$, and also since if $\varepsilon < 1/3$, then $A = \Omega(1/\varepsilon)$, and otherwise $A \geq 2 = \Omega(1/\varepsilon)$. Let $F = F_1 + \cdots + F_D$ be the total length of all subphases. Observe that D is a stopping time, hence we have $\mathbb{E}[F] = \mathbb{E}[D] \cdot \Omega(\frac{1}{\varepsilon^2})$ by Proposition 7. By Lemma 26, $\mathbb{E}[Q] = \mathcal{O}(\varepsilon^{-2}(\log n + \log \delta'^{-1}))$ holds for the strategy length Q. Since $F \leq Q$, $\mathbb{E}[D] = \mathcal{O}(\log n + \log 1/\delta') = \mathcal{O}(\log n + \log 1/\delta)$.

By application of the union bound, the error probability for the whole procedure is bounded by $\delta''\mathbb{E}[D] \leq \delta$ for appropriately chosen constant in the definition of δ'.

We now deal with case of $A \leq 1$. This requires $p < \delta'$, and $\varepsilon > 1/3$ (since if $\varepsilon < 1/3$, appropriate choice of constant in δ' enforces $A \geq 2$) and so the expected strategy length is $\mathbb{E}[Q] = O(\log n + \log 1/\delta)$. By the union bound, algorithm receives a single erroneous response with probability at most $p\mathbb{E}[Q] \leq \delta'\mathbb{E}[Q] = O(\delta^2/(\log n + \log 1/\delta)) \leq \delta$. ◀

References

1 Javed A. Aslam and Aditi Dhagat. Searching in the presence of linearly bounded errors (extended abstract). In *STOC*, pages 486–493, 1991. doi:10.1145/103418.103469.

2 Yosi Ben-Asher, Eitan Farchi, and Ilan Newman. Optimal search in trees. *SIAM J. Comput.*, 28(6):2090–2102, 1999. doi:10.1137/S009753979731858X.

3 Michael Ben-Or and Avinatan Hassidim. The bayesian learner is optimal for noisy binary search (and pretty good for quantum as well). In *FOCS*, pages 221–230, 2008. doi:10.1109/FOCS.2008.58.

4 Elvyn R. Berlekamp. *Block Coding For The Binary Symmetric Channel With Noiseless, Delayless Feedback*, pages 61–88. Wiley & Sons, New York, 1968.

5 Ryan S. Borgstrom and S. Rao Kosaraju. Comparison-based search in the presence of errors. In *STOC*, pages 130–136, 1993. doi:10.1145/167088.167129.

6 Mark Braverman, Klim Efremenko, Ran Gelles, and Bernhard Haeupler. Constant-rate coding for multiparty interactive communication is impossible. In *STOC*, pages 999–1010, 2016. doi:10.1145/2897518.2897563.

7 Marat Valievich Burnashev and Kamil'Shamil'evich Zigangirov. An interval estimation problem for controlled observations. *Problemy Peredachi Informatsii*, 10(3):51–61, 1974.

8 Yuval Dagan, Yuval Filmus, Ariel Gabizon, and Shay Moran. Twenty (simple) questions. In *STOC*, pages 9–21, 2017. doi:10.1145/3055399.3055422.

9 Yuval Dagan, Yuval Filmus, Daniel Kane, and Shay Moran. The entropy of lies: Playing twenty questions with a liar. In James R. Lee, editor, *12th Innovations in Theoretical Computer Science Conference, ITCS 2021*, volume 185 of *LIPIcs*, pages 1:1–1:16, 2021. doi:10.4230/LIPICS.ITCS.2021.1.

10 Christian Deppe. *Coding with Feedback and Searching with Lies*, pages 27–70. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. doi:10.1007/978-3-540-32777-6_2.

11 Dariusz Dereniowski, Stefan Tiegel, Przemysław Uznański, and Daniel Wolleb-Graf. A framework for searching in graphs in the presence of errors. In *SOSA@SODA*, pages 4:1–4:17, 2019. doi:10.4230/OASIcs.SOSA.2019.4.

12 Aditi Dhagat, Péter Gács, and Peter Winkler. On playing "twenty questions" with a liar. In *SODA*, pages 16–22, 1992. URL: http://dl.acm.org/citation.cfm?id=139404.139409.

13 Ehsan Emamjomeh-Zadeh and David Kempe. A general framework for robust interactive learning. In *NIPS*, pages 7085–7094, 2017.

14 Ehsan Emamjomeh-Zadeh, David Kempe, and Vikrant Singhal. Deterministic and probabilistic binary search in graphs. In *STOC*, pages 519–532, 2016. doi:10.1145/2897518.2897656.

15 Uriel Feige, Prabhakar Raghavan, David Peleg, and Eli Upfal. Computing with noisy information. *SIAM J. Comput.*, 23(5):1001–1018, 1994. doi:10.1137/S0097539791195877.

16 Ran Gelles, Bernhard Haeupler, Gillat Kol, Noga Ron-Zewi, and Avi Wigderson. Towards optimal deterministic coding for interactive communication. In *SODA*, pages 1922–1936, 2016. doi:10.1137/1.9781611974331.ch135.

17 Ran Gelles, Ankur Moitra, and Amit Sahai. Efficient coding for interactive communication. *IEEE Trans. Information Theory*, 60(3):1899–1913, 2014. doi:10.1109/TIT.2013.2294186.

18 Lucas Gretta and Eric Price. Sharp Noisy Binary Search with Monotonic Probabilities. In *ICALP 2024*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.ICALP.2024.75.

19 Yuzhou Gu and Yinzhan Xu. Optimal bounds for noisy sorting. In *STOC*, pages 1502–1515, 2023. doi:10.1145/3564246.3585131.

20 Bernhard Haeupler. Interactive channel capacity revisited. In *FOCS*, pages 226–235, 2014. doi:10.1109/FOCS.2014.32.

21 Wassily Hoeffding. Probability Inequalities for Sums of Bounded Random Variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963. doi:10.2307/2282952.

22 Richard M. Karp and Robert Kleinberg. Noisy binary search and its applications. In *SODA*, pages 881–890, 2007. URL: http://dl.acm.org/citation.cfm?id=1283383.1283478.

23 Eduardo Sany Laber, Ruy Luiz Milidiú, and Artur Alves Pessoa. On binary searching with nonuniform costs. *SIAM J. Comput.*, 31(4):1022–1047, 2002. doi:10.1137/S0097539700381991.

24 Tak Wah Lam and Fung Ling Yue. Optimal edge ranking of trees in linear time. *Algorithmica*, 30(1):12–33, 2001. doi:10.1007/s004530010076.

25 Debbie Leung, Ashwin Nayak, Ala Shayeghi, Dave Touchette, Penghui Yao, and Nengkun Yu. Capacity approaching coding for low noise interactive quantum communication. In *STOC*, pages 339–352, 2018. doi:10.1145/3188745.3188908.

26 Krzysztof Onak and Pawel Parys. Generalization of binary search: Searching in trees and forest-like partial orders. In *FOCS*, pages 379–388, 2006. doi:10.1109/FOCS.2006.32.

27 Andrzej Pelc. Searching games with errors—fifty years of coping with liars. *Theoretical Computer Science*, 270(1):71–109, 2002. doi:10.1016/S0304-3975(01)00303-6.

28 Ronald L. Rivest, Albert R. Meyer, Daniel J. Kleitman, Karl Winklmann, and Joel Spencer. Coping with errors in binary search procedures. *J. Comput. Syst. Sci.*, 20(3):396–404, 1980. doi:10.1016/0022-0000(80)90014-8.

29 Alfréd Rényi. On a problem of information theory. *MTA Mat. Kut. Int. Kozl.*, 6B:505–516, 1961.

30 Claude Elwood Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 1948. doi:10.1002/J.1538-7305.1948.TB01338.X.

31 Stanislaw M. Ulam. *Adventures of a Mathematician*. Scribner, New York, 1976.

32 Claude Leibovici (https://math.stackexchange.com/users/82404/claude-leibovici). The taylor expansion of the binary entropy. Mathematics Stack Exchange. URL: https://math.stackexchange.com/q/4502235 (version: 2022-07-29).

33 A. Wald. Sequential Tests of Statistical Hypotheses. *The Annals of Mathematical Statistics*, 16(2):117–186, 1945. URL: https://www.jstor.org/stable/2235829.

34 Ziao Wang, Nadim Ghaddar, and Lele Wang. Noisy sorting capacity. In *IEEE International Symposium on Information Theory, ISIT 2022*, pages 2541–2546. IEEE, 2022. doi:10.1109/ISIT50566.2022.9834370.

A Delegated Proofs

▶ **Lemma 28.** *The solution to* $ax = b + \sqrt{cx}$ *is of the form* $x = \frac{1}{a}\left(b + \mathcal{O}\left(\frac{c}{a} + \sqrt{b}\cdot\sqrt{\frac{c}{a}}\right)\right)$.

Proof. Solving the quadratic equation $a^2x^2 + b^2 - 2abx - cx = 0$, we get:

$$\Delta = (2ab + c)^2 - 4a^2b^2 = c(c + 4ab)$$

$$x = \frac{2ab + c + \sqrt{c^2 + 4abc}}{2a^2} = \frac{b}{a} + \mathcal{O}\left(\frac{c + \sqrt{abc}}{a^2}\right)$$

◀

▶ **Lemma 29.** *Let* $(X_i)_{i\in\mathbb{N}_+}$ *be a sequence of i.i.d random variables with* $\Pr(X_i = \ell) = p$ *and* $\Pr(X_i = r) = (1-p)$ *for some* $\ell, r \in \mathbb{R}$ *and* $0 < p < \frac{1}{2}$. *Let us fix* $T > 0$ *and let* $Q = \inf\{m : \sum_{i=1}^{m} X_i \geq T\}$. *If* $\mathbb{E}[X_i] > 0$, *then* $\mathbb{E}[Q] < \infty$.

Proof. Let $\zeta_m = \sum_{i=1}^{m} X_i$. If $Q > m$, then in particular $\zeta_m \leq T$, hence

$$\Pr(Q > m) \leq \Pr(\zeta_m \leq T) \tag{7}$$

for any $m \in \mathbb{N}_+$.

Let $\mu = \mathbb{E}[X_i]$. Obviously, $\mathbb{E}[\zeta_m] = m\mathbb{E}[X_i] = m\mu$. Now, let us define $N = \lceil \frac{T}{\mu} \rceil$. For any $m > N$ we have

$$\zeta_m \leq T \iff \zeta_m - m\mu \leq -(m\mu - T) \tag{8}$$

and $m\mu - T > 0$. Using Hoeffding bound [21] we get

$$\Pr(\zeta_m - m\mu \leq -(m\mu - T)) \leq \exp\{\frac{-2(m\mu - T)^2}{m(\ell + r)^2}\} \leq \exp\{\frac{-2m\mu^2}{(\ell + r)^2} + \frac{4\mu T}{(\ell + r)^2}\} = C\beta^m \tag{9}$$

with $C = e^{\frac{4\mu T}{(\ell+r)^2}}$ and $\beta = e^{\frac{-2\mu^2}{(\ell+r)^2}}$. Observe that $0 < \beta < 1$.

Putting together equations (7), (8) and (9) we get

$$\begin{aligned}\mathbb{E}[Q] &= \sum_{m=0}^{\infty} \Pr(Q > m) \\ &= \sum_{m\leq N} \Pr(Q > m) + \sum_{m>N} \Pr(Q > m) \leq \sum_{m\leq N} \Pr(Q > m) + C\sum_{m>N} \beta^m < \infty.\end{aligned}$$

◀