Noisy binary search and its applications

Richard M. Karp* Robert Kleinberg†

Abstract

We study a noisy version of the classic binary search problem of inserting an element into its proper place within an ordered sequence by comparing it with elements of the sequence. In the noisy version we can not compare elements directly. Instead we are given a coin corresponding to each element of the sequence, such that as one goes through the ordered sequence the probability of observing heads when tossing the corresponding coin increases. We design online algorithms which adaptively choose a sequence of experiments, each consisting of tossing a single coin, with the goal of identifying the highest-numbered coin in the ordered sequence whose heads probability is less than some specified target value. Possible applications of such algorithms include investment planning, sponsored search advertising, admission control in queueing networks, college admissions, and admitting new members into an organization ranked by ability, such as a tennis ladder.

1 Introduction

1.1 Problem Statements and Main Results An algorithm is given n biased coins labeled with the elements of the set $[n] = \{1, 2, \ldots, n\}$. Let p_i denote the probability of observing heads when tossing coin i. We assume that $p_1 \leq \ldots \leq p_n$, but that the algorithm does not know the values of $p_1, \ldots, p_n$, only their ordering. For convenience we define $p_0 = 0, p_{n+1} = 1$. A target value τ in $[0, 1]$ is specified as part of the input, but the values of $p_1, p_2, \ldots, p_n$ are not revealed. Instead, the algorithm may flip any of the coins at any time and observe the outcome of the coinflip. The algorithm's goal is to find a pair of consecutive coins $i, i+1$ such that the interval $[p_i, p_{i+1}]$ contains (or nearly contains) the specified number τ.

An instance P of size n is specified by the given number τ and the hidden array $(p_1, p_2, \cdots, p_n)$ of heads probabilities. We consider two versions of the problem to be solved by the algorithm. In the first version a positive number ε is given, and the task is to identify a coin i such that the interval $[p_i, p_{i+1}]$ intersects the interval $[\tau - \varepsilon, \tau + \varepsilon]$; such a coin is called a *good coin*. It is required that, for every instance P, the algorithm has probability at least 3/4 of correctly identifying a good coin. In the second version the task is the same, except that ε is not given, but is specified implicitly for each instance P as $\min_i |\tau - p_i|$, which we denote by $\varepsilon(P)$.

For the first version of the problem we give an algorithm whose expected number of coin flips on every instance (P, ε) is $O(\frac{\log n}{\varepsilon^2})$. For the second version we give an algorithm whose expected number of coin flips on every instance P is

$$O\left(\frac{\log n + \log\log(\frac{1}{\varepsilon(P)})}{\varepsilon(P)^2}\right).$$

Both results are optimal, since in both cases we provide information-theoretic proofs that no algorithm can improve on these bounds by more than a constant factor.

1.2 A Naive Algorithm A naive algorithm for the first version of the problem would use binary search to locate a good coin, by maintaining a pair of indices a, b (initialized to $1, n$) and always testing the coin midway between a and b to compare its heads probability p with τ. If p lies outside $(\frac{1}{2} - \varepsilon, \frac{1}{2} + \varepsilon)$, we can test whether $p > \frac{1}{2}$ or $p < \frac{1}{2}$ using $O(1/\varepsilon^2)$ coin flips. This suggests that there is an algorithm with running time $O(\log(n)/\varepsilon^2)$. Unfortunately, designing such an algorithm is not straightforward because a test which uses $O(1/\varepsilon^2)$ coin flips to compare p with $\frac{1}{2}$ has a constant probability of returning the wrong answer, and the algorithm must perform $\log(n)$ such tests. The naive solution to this problem is to design the algorithm so that each of the $\log(n)$ comparisons has $O(1/\log(n))$ error probability, then apply the union bound to say that the algorithm outputs the correct answer with probability 3/4. This requires $O(\log\log(n)/\varepsilon^2)$ coin flips per comparison, leading to a running time of $O(\log(n)\log\log(n)/\varepsilon^2)$. Our optimal result is based on a more subtle algorithm design that eliminates the factor

*Computer Science Division, University of California, Berkeley. Supported in part by NSF grant CCF-0515259. karp@icsi.berkeley.edu

†Computer Science Division, University of California, Berkeley. On leave from the Department of Computer Science, Cornell University. Supported by an NSF Mathematical Sciences Postdoctoral Research Fellowship. rdk@cs.cornell.edu

of $\log\log(n)$.

1.3 Optimal Simulations and Optimal Algorithms Our algorithms are based on optimal procedures for simulating a coin with large bias by successive independent tosses of a coin with a small bias ε. In both procedures one is given a coin with bias p, a parameter $\delta > 0$, and two disjoint sets $A, B \subseteq [0,1]$. The goal is to simulate a coin whose heads probability is less than δ if $p \in A$ and greater than $1-\delta$ if $p \in B$. In the first simulation, we take $A = [0, \frac{1}{2} - \varepsilon), B = (\frac{1}{2} + \varepsilon, 1]$ for some number $\varepsilon > 0$, and the algorithm requires $O(\log(1/\delta)/\varepsilon^2)$ flips of the original coin to produce one flip of the simulated coin. In the second simulation lemma, we take $A = [0, \frac{1}{2}), B = (\frac{1}{2}, 1]$, and the algorithm requires $O(\log(1/\delta)\log\log(1/\varepsilon)/\varepsilon^2)$ flips of the original coin to produce one flip of the simulated coin, where $\varepsilon = |p - \frac{1}{2}|$. The first simulation is essentially a restatement of the Chernoff bounds. The second simulation — along with the result that its running time is optimal within a constant factor — is new to the best of our knowledge, and appears likely to find applications beyond the scope of the present study. We also use a third simulation lemma which improves on naive application of the second simulation lemma when the algorithm must flip several different coins sequentially.

Given these simulations, our task is to design an algorithm that finds a good coin with probability at least $3/4$ when $\varepsilon \geq 1/4$, and requires only $O(\log n)$ coin tosses. We give two algorithms for this task. The first algorithm maintains for each coin i a weight that, roughly speaking, represents our degree of belief in the assertion that $p_i \leq 1/2 \leq p_{i+1}$. The weight of coin i increases or decreases by a multiplicative factor after each coin flip, according to whether the outcome of the coin flip supports the assertion about coin i. We prove that, after $O(\log n)$ coin tosses, the probability is greater than $3/4$ that either most of the coin tosses that have occurred involve good coins, or the coin with highest weight is good; in either case, it is easy to identify a good coin. The second algorithm modifies the naive binary search algorithm by allowing *backtracking*: if the algorithm finds evidence that it has gone down the wrong branch of the tree, it may backtrack to the parent node and retry an earlier comparison.

1.4 Applications The problems we consider are motivated by a noisy version of the the classic binary search problem of inserting an element x into its proper place within an ordered sequence $x_0, x_1, x_2, \cdots, x_n, x_{n+1}$ by comparing x with elements of the sequence. In the noisy version the target is $1/2$, $x > x_i$ iff $p_i < 1/2$, and the probability of error in comparing x with x_i is $\min(p_i, 1-p_i)$. Such a noisy binary search problem might model the process of arranging matches to determine the rank of a new tennis player within the established ranking of a set of n other players; here p_i is the probability that player i wins a match against the new player. This probabilistic model of binary search can be contrasted with adversarial models for binary search and related problems [2, 8, 4, 7, 9], in which an adversary seeking to prolong the search is allowed to give a specified number of incorrect answers. The only previous work we have found on a probabilistic model similar to ours is [6], which, among other models, considers a special case of our model and gives a result that is dominated by the naive algorithm mentioned above.

In other interpretations the coins represent levels of investment, p_i represents the probability of success at the ith level of investment, and the goal is to find the least investment yielding a probability of success greater than or equal to τ. As one example, the levels of investment might represent the possible placements of ad words on a Google Web page, and one seeks the least costly placement that would yield a desired click rate.

In another class of applications the coins correspond to a sequence of decreasingly stringent thresholds for admitting students to a college, packets to an Internet link, etc. Here p_i denotes the fraction of candidates admitted using the ith threshold, and τ is the desired admission rate.

Finally, the coins might correspond to a decreasing sequence of possible prices for copies of some good, in which case the goal would be to find the highest price that will yield a given volume of sales.

Several generalizations and extensions of the questions we consider suggest themselves. It would be natural to consider versions of the problem in which the p_i vary over time in some constrained manner, and the goal is to track, for all t, the index $i(t)$ such that $p_{i(t)} < \tau \leq p_{i(t)+1}$. In the scenario of admitting applicants to a college, we might assume that the desirabilities of the successively arriving applicants are i.i.d. samples from am underlying distribution, that the goal is to admit those applicants whose desirabilities are above a given percentile of the distribution, and that each coin flip corresponds to comparing the applicant's desirability with a particular threshold. In this setting, it is considered a mistake to admit a student whose desirability is below the target percentile or reject one whose desirability is above, and one could investigate the problem of minimizing the number of mistakes incurred while probing for the best threshold. For the problem of pricing a good, one might consider the problem of maximizing revenue rather than achieving a given volume of sales.

This version of the problem is studied in [5].

2 Reduction to the Case $\tau = 1/2$

For any target τ we can construct an experiment which, given a coin with unknown heads probability p, accepts the coin with probability $f(p)$, where $f(p)$ is a strictly increasing linear function such that $f(\tau) = .5$, $f(1+\varepsilon)\tau = 1/2(1+\varepsilon')$ and $f(1-\varepsilon)\tau = 1/2(1-\varepsilon')$, where $\varepsilon' = \frac{\varepsilon}{2\tau}$ if $\tau \geq 1/2$) and $\varepsilon' = \frac{\tau(1/2-\tau)\varepsilon}{1-\tau}$ if $\tau < 1/2$. This experiment assumes the availability of a fair coin, and flips the unknown coin once and the fair coin twice on average. The experiment is based on a well-known construction which simulates a coin with an arbitrary given heads probability s using, on average, two flips of the fair coin. If $\tau > 1/2$ the experiment flips the unknown coin and a simulated coin with success probability $\frac{1}{2\tau}$, and accepts the unknown coin if both outcomes are heads. If $\tau < 1/2$ the experiment flips the unknown coin and a simulated coin with success probability $\frac{1/2-\tau}{1-\tau}$ and accepts the unknown coin if at least one of the two outcomes is heads. In view of this simulation, we may assume without loss of generality that the target is $\frac{1}{2}$, and this assumption is made throughout the rest of the paper.

3 Reduction to the case $\varepsilon = \Theta(1)$

We begin by recalling some exponential tail inequalities from probability theory. Recall that a submartingale is a sequence of random variables $X_0, X_1, X_2, \ldots, X_n$ satisfying

$$\mathbb{E}\left[X_{i+1} \mid X_1, \ldots, X_i\right] \geq X_i,$$

for $1 \leq i < n$. The following inequality for submartingales is a generalization of Azuma's inequality; its proof is the same as the proof of Azuma's inequality, e.g. [1].

LEMMA 3.1. *Suppose $X_0, X_1, X_2, \ldots, X_n$ is a submartingale and $|X_{i+1} - X_i| \leq 1$ for $0 \leq i < n$. Then*

$$\Pr(X_n \leq X_0 - t) \leq \exp\left(\frac{-t^2}{2n}\right).$$

COROLLARY 3.1. *For $q \in [0,1]$ and $\varepsilon > 0$, suppose $y_1, y_2, \ldots, y_n$ are independent Bernoulli random variables, each with expected value at least $q + \varepsilon$. Then*

$$\Pr\left(\sum_{i=1}^{n} y_i \leq qn\right) < e^{-\varepsilon^2 n/2}.$$

Proof. Apply Azuma's inequality to the submartingale $X_i = \sum_{j=1}^{i}(y_j - q - \varepsilon)$. (The result can also be obtained quite easily using Chernoff's bound.) □

LEMMA 3.1. (FIRST SIMULATION LEMMA) *Given numbers $p \in [0,1]$ and $\delta, \varepsilon > 0$, let $n = \lceil 2\log(1/\delta)/\varepsilon^2 \rceil$, let $y_1, y_2, \ldots, y_n$ be independent Bernoulli random variables, each with expected value p, and let E be the event $y_1 + y_2 + \ldots + y_n > n/2$. Then:*

1. $\Pr(E)$ *is an increasing function of p.*
2. $\Pr(E) > 1 - \delta$ *if $p > \frac{1}{2} + \varepsilon$.*
3. $\Pr(E) < \delta$ *if $p < \frac{1}{2} - \varepsilon$.*

Proof. The fact that $\Pr(E)$ is an increasing function of p follows from an easy coupling argument. When $p > \frac{1}{2} + \varepsilon$, the bound $\Pr(E) > 1 - \delta$ follows from Corollary 3.1 and the fact that $\varepsilon^2 n/2 \geq -\log(\delta)$. When $p < \frac{1}{2} - \varepsilon$, the bound $\Pr(E) < \delta$ follows from applying Corollary 3.1 to the random variables $1 - y_1, \ldots, 1 - y_n$. □

LEMMA 3.2. (SECOND SIMULATION LEMMA) *Given a constant $\delta > 0$ and a coin whose heads probability is not equal to $\frac{1}{2}$, there is a deterministic online algorithm which performs coin flips and eventually halts, outputting a 0 or 1. The algorithm has the following properties. If the coin flips are independent with heads probability $p = \frac{1}{2} \pm \varepsilon$ for some $\varepsilon > 0$, then the algorithm's expected running time is $O(\log(1/\delta)\log\log(1/\varepsilon)/\varepsilon^2)$, and the probability of the event $E = \{$the algorithm outputs 1$\}$ satisfies:*

1. $\Pr(E)$ *is an increasing function of p.*
2. $\Pr(E) > 1 - \delta$ *if $p > \frac{1}{2}$.*
3. $\Pr(E) < \delta$ *if $p < \frac{1}{2}$.*

Proof. It suffices to prove the lemma when $\delta = 1/4$; the result for general δ follows by standard probability amplification techniques. We design an algorithm which runs in phases numbered $0, 1, \ldots$. In phase k, the algorithm sets $\gamma = e^{-k}$ and $n = \lceil 16\gamma^{-2}\ln(k+9) \rceil$ and it flips the coin n times. If the number of heads in phase k is between $\left(\frac{1-\gamma}{2}\right)n$ and $\left(\frac{1+\gamma}{2}\right)n$ it continues to phase $k+1$. Otherwise it stops and outputs the outcome which was observed more frequently in phase k.

Note that if the number of observed heads in phase k differs from its expected value by at most $\gamma n/2$, then the algorithm does not output an incorrect answer in phase k. By Azuma's inequality, the probability that the number of observed heads in phase k differs from its expected value by more than $\gamma n/2$ is bounded above by

$$2e^{-\gamma^2 n/8} \leq 2e^{-2\ln(k+9)} = \frac{2}{(k+9)^2}.$$

The algorithm's failure probability is bounded above by $1/4$ because $\sum_{k=0}^{\infty} 2/(k+9)^2 < 1/4$.

Finally, to bound the running time of the algorithm we again use Azuma's inequality. Let $\ell = \lceil \ln(1/\varepsilon) \rceil$, so that $\gamma \leq \varepsilon$ in phase ℓ and afterward. In order for the algorithm to continue running past phase $k \geq \ell$, the number of heads must differ from its expected value by more than $\varepsilon n/2$, an event which has probability bounded above by

$$2e^{-\varepsilon^2 n/8} \leq 2e^{-2\varepsilon^2 \exp(2k) \ln(k+9)} = 2(k+9)^{-2\varepsilon^2 \exp(2k)}.$$

The running time of phases $k \geq \ell$ grows exponentially in k while the probability of reaching phase k decreases faster than exponentially. So the expected running time is dominated by the running time of phase ℓ, which is $O(\varepsilon^{-2} \log\log(1/\varepsilon))$. □

PROPOSITION 3.1. *Let ε_0 be a constant. Suppose there is an ε_0-good algorithm for the coins problem, whose expected running time is bounded above by $T(n)$. Then for every $\varepsilon > 0$ there is an ε-good algorithm for the coins problem whose expected running time is $O(T(n)/\varepsilon^2)$. Moreover, there is a good algorithm for the coins problem whose expected running time, on every instance P, is*

$$O\left(\frac{T(n) \log\log(1/\varepsilon(P))}{\varepsilon(P)^2}\right).$$

Proof. Given coins with heads probabilities $p_1 \leq p_2 \leq \ldots \leq p_n$, let us use coin i to simulate a coin σ_i with some other heads probability q_i by applying the algorithm in Lemma 3.1 with $\delta = \frac{1}{2} - \varepsilon_0$. We require $O(1/\varepsilon^2)$ flips of coin i for each flip of the simulated coin σ_i. These heads probabilities of the simulated coins, $q_1, \ldots, q_n$, satisfy $q_1 \leq q_2 \leq \ldots \leq q_n$, and $[q_i, q_{i+1}]$ is disjoint from $[\delta, 1-\delta]$ if $[p_i, p_{i+1}]$ is disjoint from $[\frac{1}{2} - \varepsilon, \frac{1}{2} + \varepsilon]$. The ε_0-good algorithm performs $T(n)$ simulated coin flips in expectation — this requires us to perform $O(T(n)/\varepsilon^2)$ coin flips in expectation — and with probability 3/4 it outputs a pair $i, i+1$ such that $[q_i, q_{i+1}]$ intersects $[\frac{1}{2} - \varepsilon_0, \frac{1}{2} + \varepsilon_0] = [\delta, 1-\delta]$. As noted above, this implies that $[p_i, p_{i+1}]$ intersects $[\frac{1}{2} - \varepsilon, \frac{1}{2} + \varepsilon]$, as desired.

Similarly, we may use coin i to simulate a coin τ_i with heads probability r_i by applying the algorithm in Lemma 3.2 with $\delta = \frac{1}{2} - \varepsilon_0$. We require $O(\log\log(1/\varepsilon(P))/\varepsilon(P)^2)$ flips of coin i to simulate coin τ_i. The probabilities $r_1, \ldots, r_n$ satisfy $r_1 \leq \ldots \leq r_n$, and $[r_i, r_{i+1}]$ is disjoint from $[\delta, 1-\delta]$ if $[p_i, p_{i+1}]$ does not contain $\frac{1}{2}$. As above, this implies a reduction from good algorithms to ε_0-good algorithms, with a $O(\log\log(1/\varepsilon(P))/\varepsilon(P)^2)$ blow-up in the running time. □

In Sections 4 and 5 we will present algorithms with running time $T(n) = O(\log n)$ which are ε_0-good when $\varepsilon_0 = 1/3$. Hence, by Proposition 3.1 there is an ε-good algorithm with running time $O(\log(n)/\varepsilon^2)$ for every $\varepsilon > 0$, and there is a good algorithm whose running time on an instance P is $O(\log(n) \log\log(1/\varepsilon(P))/\varepsilon(P)^2)$. The running time of the good algorithm can be further improved to

$$O\left(\frac{\log n + \log\log(1/\varepsilon(P))}{\varepsilon(P)^2}\right)$$

using a more efficient simulation lemma which we now present. (In applying the lemma, we define the function A mentioned in the lemma's statement to be the rule that the ε_0-good algorithm uses to choose which coin it flips next.)

LEMMA 3.3. (THIRD SIMULATION LEMMA) *Given a constant $\delta > 0$, a set of coins $\mathcal{C} = \{c_1, c_2, \ldots, c_n\}$ with unknown heads probabilities $p(c_i)$, and a function $A : \{0,1\}^* \to \mathcal{C}$ for choosing the next coin to simulate based on a history of past simulated coin flips, there is a deterministic online algorithm which performs coin flips and outputs a sequence of bits $y_1, y_2, \ldots$ at times $\tau_1 < \tau_2 < \ldots$ such that:*

1. *At each time t, the algorithm flips coin $A(y_1, \ldots, y_s)$, where s is such that $\tau_s < t \leq \tau_{s+1}$.*
2. *If ε is any number satisfying $|p(c_i) - 1/2| \geq \varepsilon > 0$ for all $c_i \in \mathcal{C}$, then for every m the running time of the first m simulated coin flips satisfies*

$$\mathbb{E}(\tau_m) = O([m + \log\log(1/\varepsilon)]/\varepsilon^2).$$

3. *For each $c_i \in \mathcal{C}$, $\Pr(y_s = 1 \mid A(y_1, \ldots, y_{s-1}) = c_i)$ is an increasing function of $p(c_i)$.*
4. *If $p(c_i) > 1/2$ then*

$$\Pr(y_s = 1 \mid A(y_1, \ldots, y_{s-1}) = c_i) > 1 - \delta.$$

5. *If $p(c_i) < 1/2$ then*

$$\Pr(y_s = 1 \mid A(y_1, \ldots, y_{s-1}) = c_i) < \delta.$$

Proof. [Sketch] Intuitively, we need to learn the value of ε in a sequence of phases as in Lemma 3.2, but once we have learned ε we should remember its value and not try to re-learn it each time we simulate a new coin flip. This allows us to pay the cost of $O(\log\log(1/\varepsilon)/\varepsilon^2)$ only once (in expectation) rather than m times.

As above, it suffices to prove the lemma when $\delta = 1/4$. The algorithm is a modification of the one given in Lemma 3.2. It sets a sequence of state variables $\alpha_0, \alpha_1, \ldots$, where α_s may be interpreted as the algorithm's estimate of $\ln(1/\varepsilon)$ when it commences simulating the s-th coin flip. Let $\alpha_0 = \tau_0 = 0$.

Between time τ_s and τ_{s+1}, the algorithm flips coin $c_i = A(y_1, \ldots, y_s)$; this sequence of coin flips is divided into phases numbered $0, 1, \ldots$ In phase k it sets $\gamma = e^{-\alpha_s - k}$ and $n = \lceil 16\gamma^{-2} \ln(k+9) \rceil$ and it performs n flips of coin c_i. If the number of heads in phase k is between $\left(\frac{1-\gamma}{2}\right) n$ and $\left(\frac{1+\gamma}{2}\right) n$ then it continues to phase $k+1$. Otherwise it sets $y_s = 0$ or $y_s = 1$ according to whether tails or heads was observed more frequently among the coin flips in phase k. Finally, it sets τ_{s+1} equal to the current time and it sets $\alpha_{s+1} = \alpha_s + k - 1$.

Every property except property (2) in the statement of the lemma is proved using a straightforward modification of the corresponding argument from Lemma 3.2. The proof of (2) depends on a delicate analysis of the stochastic process $(\alpha_s)_{s \geq 0}$ and is deferred to the full version of the paper. □

4 A multiplicative weights algorithm

In this section we set $\varepsilon_0 = \frac{1}{3}$ and present an ε_0-good algorithm with running time $O(\log n)$.

The algorithm maintains an array $W = (w(0), w(1), \cdots, w(n))$ of positive numbers summing to 1; $w(i)$ can be thought of as a weight associated with the interval $(p_i, p_{i+1}]$. Initially, $w(i) = \frac{1}{n+1}$ for all i. In the analysis of the algorithm, we will write $w_t(i)$ to denote the value of $w(i)$ at the end of step t of the algorithm.

At each step t the algorithm identifies $i(t)$, the largest index i such that $\sum_{j=0}^{i} w(i) < .5$. It then chooses an index $j(t)$ uniformly at random from the 2-element set $\{i(t), i(t)+1\}$, tosses coin $j(t)$ and updates the array W as follows:

Let $q = \sum_{i=0}^{j(t)-1} w(i)$. If coin $j(t)$ comes up "heads" then

1. for $i < j(t)$, $w(i) \leftarrow \frac{3w(i)}{2+q}$;
2. for $i \geq j(t)$,$w(i) \leftarrow \frac{2w(i)}{2+q}$.

If coin $j(t)$ comes up "tails" then

1. for $i \geq j(t)$, $w(i) \leftarrow \frac{3w(i)}{3-q}$;
2. for $i < j(t)$,$w(i) \leftarrow \frac{2w(i)}{3-q}$.

The intuition for this update rule is that $w_t(i)$ is analogous to a Bayesian posterior probability (after t trials) that the interval $(p_i, p_{i+1}]$ contains $1/2$, starting from a uniform prior. The multiplicative update rule given above is analogous to updating this posterior distribution according to Bayes' law.

The algorithm continues for $T(n)$ steps, where $T(n) = \lceil 200 \ln(.51(n+1)) \rceil$. Let $u = \arg\max_i w_{T(n)}(i)$. and let v be the median element of the multiset $\{i(1), i(2), \cdots, i(T(n))\}$. We prove below that, with probability at least $1 - exp(-2)$, at least one of the following statements holds:

1. $p_u < 1/2 \leq p_{u+1}$
2. index v is $\frac{1}{4}$-good.

The final stage of the algorithm is to flip coins v and $v+1$ each $\lceil \ln \ln n \rceil$ times. Let h_v and h_{v+1} be the frequencies of heads for coins v and $v+1$ respectively. If $[h_v, h_{v+1}] \cap [5/24, 19/24] \neq \phi$ then declare that index v is $\frac{1}{3}$-good, else declare that index u is $\frac{1}{3}$-good. We shall prove that the declaration is correct with probability at least $3/4$.

Let i^* be the unique index such that $p_{i^*} < 1/2 \leq p_{i^*+1}$ Let $Z(t) = \ln(w_{t+1}(i^*)) - \ln(w_t(i^*))$.

LEMMA 4.1. *Let $R(t)$ denote the sequence of outcomes of the first $t-1$ coin flips of the algorithm. If $i(t)$ is a $\frac{1}{4}$-good index then $\mathbb{E}(Z(t)|R(t)) \geq \frac{1}{2}\ln(\frac{24}{25})$. If $i(t)$ is not a $\frac{1}{4}$-good index then $\mathbb{E}(Z(t)|R(t)) \geq \frac{1}{4}\ln(\frac{864}{625})$. The first of these constants is greater than $-.0204$ and the second is greater than .0404.*

Proof. Without loss of generality we may assume that $i(t)+1 \leq i^*$, since the case $i(t)+1 > i^*$ is a mirror image of this case and can be analyzed similarly. Let $j(t)$ be the index drawn from $\{i(t), i(t)+1\}$ at step t. Conditional upon this choice $\mathbb{E}[Z(t)] = p_{j(t)} \ln(\frac{2}{2+q}) + (1 - p_{j(t)}) \ln(\frac{3}{3-q})$. For any q this is a monotone nonincreasing function of p_j. Also, when $j(t) = i(t)+1$, $q \geq 1/2$. In all cases $p_{j(t)} \leq 1/2$, and if $i(t)$ is not a $\frac{1}{4}$-good index then $p_{j(t)} \leq 1/4$. Using these facts and elementary calculus the result follows. □

LEMMA 4.2. *Let $T = T(n)$. With probability at least $1 - \exp(-2)$, at least one of the following holds:*
(i) The number of steps in which $i(t)$ is a $\frac{1}{4}$-good index is greater than $T/2$, and the median element of the multiset $\{i(1), i(2), \cdots, i(T)\}$ is a $\frac{1}{4}$-good index.
(ii) $w_T(i^*) \geq .51$.

Proof. We make the following observations.
(1) $w_{T(n)}(i^*) \geq .51$ if and only if

$$\sum_{t=1}^{T(n)} Z(t) \geq \ln(.51(n+1)).$$

(2) Define $X_t = \sum_{k=1}^{t} Z(k) - \mathbb{E}[Z(k)|R(k)]$. Since $\{X_t\}$ is a martingale with differences bounded by $\ln(3/2)$, Azuma's Inequality yields that

$$\Pr[X(T) \leq -2\ln^2(3/2)\sqrt{T}] \leq \exp(-2).$$

(3) Let G denote the number of steps in which $i(t)$ is a $\frac{1}{4}$-good index. If $G > T/2$, then it is immediate that the median element of the multiset $\{i(1), i(2), \cdots, i(T)\}$ is a $\frac{1}{4}$-good index, so we are in case (i).
(4) If $G \leq T/2$, then $\mathbb{E}[Z(t)|R(t)]$ is at least .0404 for at least half of the indices $1, 2, \cdots, T(n)$ and it is at least $-.0204$ for the remaining indices. Thus $\sum_{t=1}^{T} \mathbb{E}\left[Z(t)|R(t)\right] \geq .01T$ and hence

$$\sum_{t=1}^{T} Z(t) \geq .01T + X_T.$$

(5) Let $\widetilde{Z} = .01T - 2\ln^2(3/2)\sqrt{T}$. We have

$$\Pr\left[(G \leq T/2) \wedge \left(\sum_{t=1}^{T(n)} Z(t) < \widetilde{Z}\right)\right] \leq \exp(-2).$$

(6)

$$\Pr\left[(G > T/2) \vee \left(\sum_{t=1}^{T(n)} Z(t) \geq \widetilde{Z}\right)\right] \geq 1 - \exp(-2).$$

(7) For $T = \lceil 200\ln(.51(n+1))\rceil$ and all sufficiently large n, $\widetilde{Z} = .01T - 2\ln^2(3/2)\sqrt{T} \geq \ln(.51(n+1))$.
(8) For $T = \lceil 200\ln(.51(n+1))\rceil$ and all sufficiently large n, $\Pr((G > T/2) \vee (w_T(i^*) \geq .51)) \geq 1 - \exp(-2)$. □

THEOREM 4.1. *With probability at least* 3/4*, the algorithm correctly declares a* $\frac{1}{3}$*-good index.*

Proof. Since $\sum_i w_{T(n)}(i) = 1$, there can be at most one index i such that $w_{T(n)}(i) > 1/2$. By the preceding lemma, with probability at least $1 - \exp(-2)$, at least one of the following two indices is $\frac{1}{4}$-good upon termination of the main loop of the algorithm: v, the median of the multiset of indices $\{i(1), i(2), \cdots, i(T(n))\}$; and u, $\arg\max_i(w_{T(n)}(i))$. Moreover, by application of a Chernoff bound, the probability that $[h_v, h_{v+1}] \cap [5/24, 19/24]$ will be nonempty if $[v, v+1]$ is not $\frac{1}{3}$-good tends to zero as $n \to \infty$. Hence the algorithm correctly declares a good index with probability at least $1 - \exp(-2) - o(1)$. □

5 A binary search algorithm with backtracking

This section describes a second ε_0-good algorithm with running time $O(\log(n))$, when $\varepsilon_0 = \frac{1}{3}$. The algorithm bears a close resemblance to the naive binary search algorithm which was briefly described in Section 1.2, except that the binary search considered here allows *backtracking*: if the algorithm finds evidence that it has gone down the wrong branch of the tree, it may backtrack to the parent node and retry an earlier comparison.

Let $\mathbb{T}$ be an infinite rooted binary tree each of whose nodes is labeled with a pair of indices from {0,1,...,n+1}. The labeling is defined recursively as follows. First, the root vertex is labeled with $(0, n+1)$. Then, for every vertex v with label (a, b), let $m = \lfloor (a+b)/2 \rfloor$ and label the left child $L(v)$ with (a, m) and the right child $R(v)$ with (m, b). For future reference, we will denote the parent of a node v by $P(v)$. (When v is the root, we adopt the convention that $P(v) = v$.)

The algorithm takes a random walk $\nu(0), \nu(1), \ldots$ in $\mathbb{T}$, starting at the root. In round t, when the algorithm is at node $\nu(t)$ whose label is (a, b) and whose children are labeled (a, m) and (m, b), the algorithm operates as follows. First, it flips coin a twice and coin b twice. If both a-flips come up heads or both b-flips come up tails, then it moves to the parent node $\nu(t+1) = P(\nu(t))$. Otherwise, it flips coin m and moves to the left child $\nu(t+1) = L(\nu(t))$ if it flips heads, or to the right child $\nu(t+1) = R(\nu(t))$ if it flips tails. Finally, with probability $1/\ln(n)$, at the end of round t the algorithm performs a special "termination test" to decide if it should halt before progressing to round $t+1$, because either coin a or coin b is ε_0-good. The termination test works as follows. Let $k = \lceil 300\ln(n)\rceil$. The algorithm performs k flips of coin a and k flips of coin b, noting the number of heads h_a, h_b, respectively. If $\frac{1}{4} \leq h_a/k \leq \frac{3}{4}$ then it halts and outputs a. If $\frac{1}{4} \leq h_b/k \leq \frac{3}{4}$ then it halts and outputs b. If $b = a+1$ and $h_a/k < \frac{1}{2} < h_b/k$ then it halts and outputs a. Otherwise it proceeds to round $t+1$.

LEMMA 5.1. *For all* t*, the probability that the algorithm halts and outputs a coin which is not* ε_0*-good at time* t *is less than* $4/n$*.*

Proof. Suppose the algorithm performs a termination test at time t. Conditional on this event, we will prove that the probability of halting at time t and outputting a coin which is not ε_0-good is less than $4/n$. By Azuma's inequality, the probability that $|h_a/k - p_a| > \frac{1}{12}$ is bounded above by $2\exp(-k/288) < 2/n$. Similarly the probability that $|h_b/k - p_b| > \frac{1}{12}$ is bounded above by $2/n$. So assume now that $|h_a/k - p_a| \leq \frac{1}{12}$ and $|h_b/k - p_b| \leq \frac{1}{12}$. If the algorithm decides to halt at time t, then one of the following cases applies:

1. $h_a/k \in [\frac{1}{4}, \frac{3}{4}]$ and the algorithm outputs a. In this case $p_a \in [\frac{1}{6}, \frac{5}{6}]$ and a is ε_0-good.

2. $h_b/k \in [\frac{1}{4}, \frac{3}{4}]$ and the algorithm outputs b. In this case $p_b \in [\frac{1}{6}, \frac{5}{6}]$ and b is ε_0-good.

3. $b = a + 1$ and $\frac{1}{2} \in [h_a/k, h_b/k]$, and the algorithm outputs a. In this case $[\frac{1}{2} - \frac{1}{12}, \frac{1}{2} + \frac{1}{12}]$ intersects $[p_a, p_b]$ and a is ε_0-good.

We have shown that the algorithm outputs a coin which is ε_0-good unless one of $|h_a/k - p_a|$, $|h_b/k - p_b|$ is greater than $1/12$, an event with probability less than $4/n$. □

For the remainder of the analysis, define a node of $\mathbb{T}$ to be *promising* if its label (a, b) satisfies at least one of the following:

1. $p_a \in [\frac{1}{4}, \frac{3}{4}]$.
2. $p_b \in [\frac{1}{4}, \frac{3}{4}]$.
3. $b = a + 1$ and $\frac{1}{2} \in [p_a, p_b]$.

Otherwise we say the node is *bad*. The set of promising nodes in $\mathbb{T}$ will be denoted by W.

LEMMA 5.2. *Conditional on the event $\nu(t) \in W$, the probability that the algorithm halts at time t is $\Omega(1/\ln(n))$.*

Proof. The statement of the lemma is tantamount to saying that if a termination test is performed at time t and $\nu(t)$ is promising, then there is a constant probability that the algorithm terminates. There are three cases (depending on which of the three criteria for a promising node are satisfied by $\nu(t)$) and in each case the lemma follows from the fact that each of the following events has constant probability and the first two are independent of the second two: **(A)** $h_a/k \in [p_a - \frac{1}{4}, p_a]$; **(B)** $h_a/k \in [p_a, p_a + \frac{1}{4}]$; **(C)** $h_b/k \in [p_b - \frac{1}{4}, p_b]$; **(D)** $h_b/k \in [p_b, p_b + \frac{1}{4}]$. □

LEMMA 5.3. *Let $d(u, W)$ denote the distance from a node $u \in \mathbb{T}$ to the closest node of W. Conditional on the event $\nu(t) \notin W$, the probability that $d(\nu(t+1), W) = d(\nu(t), W) - 1$ is at least $9/16$.*

Proof. Let (a, b) be the label of $\nu(t)$. If $\nu(t) \notin W$ then one of the following cases applies.

1. $[p_a, p_b] \subseteq [0, \frac{1}{4}] \cup [\frac{3}{4}, 1]$.
2. $p_a < \frac{1}{4}$ and $p_b > \frac{3}{4}$ and $b - a > 1$.

In case (1), either $p_a > \frac{3}{4}$ or $p_b < \frac{1}{4}$; assume the former without loss of generality. The probability that both of the algorithm's a-flips in round t come up heads is at least $9/16$; if so, then $\nu(t + 1)$ is the parent of $\nu(t)$ and this is closer to W because no descendant of $\nu(t)$ belongs to W. In case (2), let $\mathcal{E}$ denote the event $\nu(t + 1) \neq P(\nu(t))$. This is equivalent to the event that at least one of the a-flips is tails and at least one of the b-flips is heads; these two events are independent and each has probability at least $15/16$. So $\Pr(\mathcal{E}) \geq (15/16)^2$. Let $m = \lfloor (a + b)/2 \rfloor$. If $p_m \in [\frac{1}{4}, \frac{3}{4}]$ then both children of $\nu(t)$ are in W and we are done: with probability at least $(15/16)^2 = 0.87...$, event $\mathcal{E}$ occurs and the distance to W decreases from 1 to 0. If $p_m \notin [\frac{1}{4}, \frac{3}{4}]$ then assume without loss of generality that $p_m < \frac{1}{4}$. This implies that all of W is contained in the right subtree of $\nu(t)$, so to finish we must prove that

$$\Pr(\nu(t + 1) = R(\nu(t))) \geq 9/16.$$

Conditional on $\mathcal{E}$, the algorithm will choose $\nu(t + 1) = R(\nu(t))$ as long as one toss of coin m produces tails. By our assumption that $p_m < \frac{1}{4}$, we see that this event has conditional probability at least $\frac{3}{4}$, so the probability that $\nu(t + 1) = R(\nu(t))$ is at least $(3/4) \Pr(\mathcal{E})$, which is bounded below by $(3/4) \cdot (15/16)^2 > 9/16$. □

PROPOSITION 5.1. *The algorithm is ε_0-good and its expected running time is $O(\log n)$.*

Proof. We will prove that the expected running time is $O(\log n)$. The fact that the algorithm is ε_0-good will follow immediately using Lemma 5.1. Note that each round of the algorithm has constant expected running time, since it requires at most 5 coin flips followed possibly by a termination test whose running time is $O(\log n)$, but the termination test is only executed with probability $O(1/\log n)$. So it suffices to prove that the expected number of rounds is $O(\log n)$. We will do this using a potential function argument.

Let $Y_t = d(\nu(t), W)$, let $Z_t = \#\{s < t \mid \nu(s) \in W\}$, and let

$$\Phi_t = \lceil \log_2(n) \rceil + Z_t - Y_t - \frac{1}{8}(t - Z_t).$$

We have $\Phi_0 = \lceil \log_2(n) \rceil - d(\nu(0), W) \geq 0$, and we claim that $\{\Phi_t\}$ is a submartingale. If $\nu(t) \in W$, then $Z_{t+1} = Z_t + 1$ so $\Phi_{t+1} \geq \Phi_t$. If $\nu(t) \notin W$ then $Z_{t+1} = Z_t$ while Lemma 5.3 implies $\mathbb{E}(Y_{t+1} \mid Y_t, \nu(t)) \leq Y_t - 1/8$, which confirms that $\mathbb{E}(\Phi_{t+1} \mid \Phi_t, \nu(t)) \geq \Phi_t$.

If the algorithm does not terminate before round $t > 160(1 + \log_2(n))$, then either $Z_t > t/10$ or

$$\Phi_t < \log_2(n) + 1 + \frac{9}{8} Z_t - \frac{1}{8} t < -t/160.$$

Lemma 5.2 ensures that $\Pr(Z_t > t/10) < (1 - \delta)^{t/\ln(n)}$ for some constant $\delta > 0$, while Azuma's inequality for submartingales implies that $\Pr(\Phi_t < t/160) < (1 - \delta')^t$ for some other constant $\delta' > 0$. It follows that for $t > 160(1 + \log_2(n))$, the probability of the algorithm running for more than t rounds decays exponentially in $t/\log(n)$. Hence the expected number of rounds is $O(\log n)$. □

6 Information-theoretic lower bounds

In this section we prove lower bounds which establish that the running times of the algorithms in this paper are optimal up to constant factors. We begin by recalling some facts about Kullback-Leibler divergence. See [3] for a more thorough introduction to these techniques.

DEFINITION 6.1. *Let Ω be a finite set with two probability measures p, q. Their* Kullback-Leibler divergence*, or KL-divergence, is the sum*

$$KL(p;q) = \sum_{x \in \Omega} p(x) \ln\left(\frac{p(x)}{q(x)}\right),$$

with the convention that $p(x)\ln(p(x)/q(x))$ is interpreted to be 0 when $p(x) = 0$ and $+\infty$ when $p(x) > 0$ and $q(x) = 0$. If Y is a random variable defined on Ω and taking values in some set Γ, the conditional Kullback-Leibler divergence *of p and q given Y is the sum*

$$KL(p;q \mid Y) = \sum_{x \in \Omega} p(x) \ln\left(\frac{p(x \mid Y = Y(x))}{q(x \mid Y = Y(x))}\right),$$

where terms containing $\log(0)$ or $\log(\infty)$ are handled according to the same convention as above.

The following lemma summarizes some standard facts about KL-divergence; for proofs, see [3].

LEMMA 6.1. *Let p, q be two probability measures on a measure space $(\Omega, \mathcal{F})$ and let Y be a random variable defined on Ω and taking values in some finite set Γ. Define a pair of probability measures p_Y, q_Y on Γ by specifying that $p_Y(y) = p(Y = y), q_Y(y) = q(Y = y)$ for each $y \in \Gamma$. Then*

$$KL(p;q) = KL(p;q \mid Y) + KL(p_Y;q_Y),$$

and $KL(p;q \mid Y)$ is non-negative.

The KL-divergence of two distributions can be thought of as a measure of their statistical distinguishability. We will need three lemmas concerning KL-divergence. The first lemma asserts that a sequence of n experiments can not be very good at distinguishing two possible distributions if none of the individual experiments is good at distinguishing them. The second one shows that if the KL-divergence of p and q is small, then an event which is reasonably likely under distribution p can not be too unlikely under distribution q. The third lemma estimates the KL-divergence of distributions which are very close to the uniform distribution on $\{0, 1\}$.

LEMMA 6.2. *Suppose $\Omega_0, \Omega_1, \ldots, \Omega_n$ is a sequence of finite probability spaces, and suppose we are given two probability measures p_i, q_i on Ω_i $(0 \le i \le n)$ and random variables $Y_i : \Omega_i \to \Omega_{i-1}$ such that $p_{i-1} = (p_i)_{Y_i}$, $q_{i-1} = (q_i)_{Y_i}$ for $i = 1, 2, \ldots, n$. If $p_0 = q_0$ and $KL(p_i; q_i \mid Y_i) < \delta$ for all i, then $KL(p_n; q_n) < \delta n$.*

Proof. The proof is by induction on n, the base case $n = 1$ being trivial. For the induction step, Lemma 6.1 implies

$$\begin{aligned} KL(p_n;q_n) &= K(p_n;q_n \mid Y_n) + KL(p_{n-1};q_{n-1}) \\ &< \delta + KL(p_{n-1};q_{n-1}), \end{aligned}$$

and the right side is less than δn by the induction hypothesis. □

LEMMA 6.3. *Let Ω be a probability space with two probability measures p, q and an event $\mathcal{E}$ such that $p(\mathcal{E}) \ge 1/3$ and $q(\mathcal{E}) < 1/3$. Then*

$$KL(p;q) \ge \frac{1}{3}\ln\left(\frac{1}{3q(\mathcal{E})}\right) - \frac{1}{e}.$$

Proof. Let $\Gamma = \{0, 1\}$ and let Y be the indicator random variable of the event $\mathcal{E}$. Let a, b be the probabilities of event $\mathcal{E}$ under the measures p, q, respectively. Lemma 6.1 implies that

$$\begin{aligned} KL(p;q) &\ge KL(p_Y;q_Y) \\ &= a\ln\left(\frac{a}{b}\right) + (1-a)\ln\left(\frac{1-a}{1-b}\right) \\ &\ge \frac{1}{3}\ln\left(\frac{1}{3b}\right) + (1-a)\ln(1-a) \\ &\ge \frac{1}{3}\ln\left(\frac{1}{3b}\right) - \frac{1}{e}. \end{aligned}$$

□

LEMMA 6.4. *If $0 < \varepsilon < 1/2$ and p, q, r are the distributions on $\{0, 1\}$ defined by*

$$p(1) = \frac{1+\varepsilon}{2} \qquad q(1) = \frac{1}{2} \qquad r(1) = \frac{1-\varepsilon}{2}$$

$$p(0) = \frac{1-\varepsilon}{2} \qquad q(0) = \frac{1}{2} \qquad r(0) = \frac{1+\varepsilon}{2}$$

then $KL(p;q) < 2\varepsilon^2$ and $KL(p;r) < 4\varepsilon^2$.

Proof.

$$\begin{aligned} KL(p;q) &= \frac{1+\varepsilon}{2}\ln(1+\varepsilon) + \frac{1-\varepsilon}{2}\ln(1-\varepsilon) \\ &= \frac{1}{2}\ln(1-\varepsilon^2) + \frac{\varepsilon}{2}\ln\left(1 + \frac{2\varepsilon}{1-\varepsilon}\right) \\ &< \frac{\varepsilon}{2}\left(\frac{2\varepsilon}{1-\varepsilon}\right) < 2\varepsilon^2. \end{aligned}$$

$$\begin{aligned} KL(p;r) &= \frac{1+\varepsilon}{2}\ln\left(\frac{1+\varepsilon}{1-\varepsilon}\right) + \frac{1-\varepsilon}{2}\ln\left(\frac{1-\varepsilon}{1+\varepsilon}\right) \\ &= \frac{1}{2}\ln(1) + \frac{\varepsilon}{2}\ln\left(\left(\frac{1+\varepsilon}{1-\varepsilon}\right)^2\right) \\ &= \varepsilon\ln\left(1+\frac{2\varepsilon}{1-\varepsilon}\right) \\ &< 4\varepsilon^2. \end{aligned}$$

□

THEOREM 6.1. *There does not exist an ε-good algorithm for the n-coins problem whose expected running time on every input is $o(\log(n)/\varepsilon^2)$.*

Proof. Suppose we are given an ε-good algorithm f. Let T be its running time, let $m = 4\mathbb{E}[T]$, and let g be the algorithm which runs f until either f halts (in which case g halts with the same output) or f performs more than m coin flips (in which case g halts and outputs 1 instead of performing the $(m+1)$-th coin flip). By Markov's inequality, $\Pr(T > m) < 1/4$, hence $\Pr(g \neq f) < 1/4$. Recalling that f outputs an ε-good coin with probability at least $3/4$, we find that g outputs an ε-good coin with probability greater than $1/2$.

Define n different input instances $P_1, P_2, \ldots, P_n$ by specifying that in input instance P_i, the heads probability of coin j is $\frac{1}{2}-2\varepsilon$ for $j \leq i$ and $\frac{1}{2}+2\varepsilon$ for $j > i$. These input instances define probability measures q_i^j on $\{0,1\}^j$, for $1 \leq i \leq n$, $0 \leq j \leq m$; the measure q_i^j is defined as the distribution of the outcomes of the first j coin tosses observed when running algorithm g using the coins specified by input instance P_i. Let $\mathcal{E}_i \subseteq \{0,1\}^m$ denote the event that g outputs i. Recalling that g always outputs an ε-good coin with probability greater than $1/2$, and that coin i is the only ε-good coin in instance P_i, we find that $q_i^m(\mathcal{E}_i) > 1/2$ for every i. Now define q_0^j to be the uniform distribution on $\{0,1\}^j$, for $j = 0, 1, \ldots, m$. If we let $\Omega_j = \{0,1\}^j$ for $j = 0, 1, \ldots, m$, and we let $Y_j : \Omega_j \to \Omega_{j-1}$ be the function which omits the last element in a sequence of j coin-toss outcomes, then Lemma 6.4 implies that for all i, $KL(q_i^j; q_0^j \mid Y_j) < 32\varepsilon^2$ and Lemma 6.2 then implies that $KL(q_i^m; q_0^m) < 32\varepsilon^2 m$. However the events $\mathcal{E}_i$ are disjoint, so at least one of them satisfies $q_0^m(\mathcal{E}_i) \leq \frac{1}{n}$. Recalling that $q_i^m(\mathcal{E}_i) > 1/2$, we may apply Lemma 6.3 to conclude that $KL(q_i^m; q_0^m) \geq \frac{1}{3}\ln(n/3) - 1/e$. Hence $32\varepsilon^2 m > \frac{1}{3}\ln(n/3) - 1/e$. Recalling that $m = 4\mathbb{E}[T]$, where T is the running time of the original algorithm f, we see that the algorithm's expected running time is $\Omega(\log(n)/\varepsilon^2)$. □

THEOREM 6.2. *There does not exist a good algorithm for the n-coins problem whose expected running time on every instance P is $o(\log\log(1/\varepsilon(P))/\varepsilon(P)^2)$, even when $n = 1$, i.e. the problem of detecting whether $p < 1/2$ or $p > 1/2$ given a coin with unknown bias p.*

Proof. As in the proof of Theorem 6.1, we define disjoint events, each with a corresponding input instance. Each of these input instances defines a probability distribution on coin-flip outcomes; under this distribution the corresponding event holds with probability greater than $1/2$. Using a KL-divergence argument this implies a lower bound on the algorithm's running time. Unlike the proof of Theorem 6.1, which used a family of n disjoint events (distinguished by the algorithm's output), in this proof we will need an infinite family of disjoint events, distinguished by the algorithm's output as well as its running time.

Define a sequence of small real numbers ε_k and large integers T_k by specifying that $\varepsilon_1 = 1/3$ and using the recursion:

$$\begin{aligned} T_k &= \left\lceil \frac{1}{200\varepsilon_k^2}\ln\ln\left(\frac{1}{\varepsilon_k}\right)\right\rceil \\ \frac{1}{2000\varepsilon_{k+1}^2} &= T_k. \end{aligned}$$

Define input instances $P_{i,k}$ for $1 \leq i \leq n$, $k \geq 1$ by specifying that in instance $P_{i,k}$, the heads probability of coin j is $\frac{1}{2} - 2\varepsilon_k$ for $j \leq i$ and $\frac{1}{2} + 2\varepsilon_k$ for $j > i$. These input instances define probability measures $q_{i,k}^j$ on $\{0,1\}^j$, for $1 \leq i \leq n$, $0 \leq j < \infty$; the measure $q_{i,k}^j$ is defined as the distribution of the outcomes of the first j coin tosses observed when running algorithm f using the coins specified by input instance $P_{i,k}$ (If there is a sequence of $j' < j$ outcomes which causes the algorithm to halt after j' coin tosses, we define the probability measure on $\{0,1\}^j$ by stipulating that all continuations of this sequence of j' outcomes are equally likely.) Let $\mathcal{E}_{i,k} \subseteq \{0,1\}^{T_k}$ denote the event that algorithm f halts and outputs i at a time T_f satisfying $T_{k-1} < T_f \leq T_k$. By abuse of notation, for $j > T_k$ we will also use $\mathcal{E}_{i,k}$ to refer to the event in $\{0,1\}^j$ consisting of all sequences whose first T_k elements constitute an element of $\mathcal{E}_{i,k}$.

As before, we also define a "reference measure" $q_{0,k}^j$ which is the uniform distribution on $\{0,1\}^j$. (Note that this measure doesn't depend on k; the double subscript "$0,k$" is purely a notational convenience.) Combining Lemma 6.2 with Lemma 6.4 we obtain the bound

$$KL(q_{i,k}^j; q_{i',k}^j) < 64\varepsilon_k^2 j$$

for $0 \leq i, i' \leq n$ and $0 \leq j, k < \infty$. This implies, using Lemma 6.3, that if $\mathcal{E}$ is any event in $\{0,1\}^j$ such that

$q^j_{i,k}(\mathcal{E}) \geq 1/2$ then for $0 \leq i' \leq n$ we have

$$q^j_{i',k}(\mathcal{E}) > \frac{1}{3}\exp(-192\varepsilon_k^2 j - 3/e). \tag{6.1}$$

Assume we have a good algorithm f whose running time is $o(\log\log(1/\varepsilon)/\varepsilon^2)$. Increasing the running time of f by a constant factor if necessary, we can assume that for every non-degenerate input instance, the probability that f outputs a good coin is at least 0.9; we summarize this property by saying that f is *very good*. (A good algorithm can always be transformed into a very good algorithm as follows. Run the algorithm seven times. If at least four of the seven runs give the same answer $i \in [n]$, then output i; otherwise output an arbitrary answer.) Since $\varepsilon_k < \varepsilon(P_{i,k})$ and $T_k = \Theta(\log\log(\varepsilon_k)/\varepsilon_k^2)$, we may assume that for all sufficiently large k (say, for $k > k_0$) the probability that f runs past time T_k on instance $P_{i,k}$ is less than 0.01. In the notation introduced above, this is expressed as:

$$q^{T_k}_{i,k}(T_f > T_k) < 0.01.$$

Above, we defined $\mathcal{E}_{i,k}$ to be the event that f halts and outputs i at a time T_f satisfying $T_{k-1} < T_f \leq T_k$. We may also define $\mathcal{E}^-_{i,k}$ to be the event that f halts and outputs i at a time T_f satisfying $T_f \leq T_{k-1}$. The fact that f is a very good algorithm implies that

$$q^{T_k}_{i,k}(\mathcal{E}_{i,k}) + q^{T_k}_{i,k}(\mathcal{E}^-_{i,k}) + q^{T_k}_{i,k}(T_f > T_k) \geq 0.9,$$

and above we stipulated that the third term on the left side is at most 0.01. Note that $\mathcal{E}^-_{i,k}$ depends only on the outcomes of the first T_{k-1} coin tosses, so $q^{T_k}_{i,k}(\mathcal{E}^-_{i,k}) = q^{T_{k-1}}_{i,k}(\mathcal{E}^-_{i,k})$. If this probability is at least $1/3$, then (6.1) implies that for every $i' \neq i$, $q^{T_{k-1}}_{i',k}(\mathcal{E}^-_{i,k}) > 0.1$. (Here we are using the fact that $\varepsilon_k^2 T_{k-1} = 1/2000$ and that $\frac{1}{3}\exp(192/2000 - 3/e) > 0.1$.) This contradicts the fact that on input instance $P_{i',k}$, the probability that f outputs an answer other than i' is less than 0.1. Hence $q^{T_k}_{i,k}(\mathcal{E}^-_{i,k}) < 1/3$, and $q^{T_k}_{i,k}(\mathcal{E}_{i,k}) > 0.9 - 0.01 - 1/3 > 1/2$.

Now that we have established that $\mathcal{E}_{i,k}$ has probability at least $1/2$ when the input instance is $P_{i,k}$, we can use (6.1) once again to prove a lower bound on the probability of $\mathcal{E}_{i,k}$ under the "reference measure" $q^{T_k}_{0,k}$:

$$\begin{aligned} q^{T_k}_{0,k}(\mathcal{E}_{i,k}) &> \frac{1}{3}\exp(-96\varepsilon_k^2 T_k - 3/e) \\ &> \frac{1}{3e^2}\exp\left(-\frac{1}{2}\ln\ln\left(\frac{1}{\varepsilon_k}\right)\right) \\ &= \frac{1}{3e^2\sqrt{\ln(1/\varepsilon_k)}}. \end{aligned}$$

From the recursion defining ε_k we find that

$$\ln(1/\varepsilon_{k+1}) < \ln(1/\varepsilon_k) + \frac{1}{2}\ln\ln\ln(1/\varepsilon_k) + \frac{1}{2}\ln(10),$$

which implies that $\ln(1/\varepsilon_k) = O(k\ln\ln(k))$. Consequently the sum $\sum_{k=k_0}^{\infty}(3e^2\sqrt{\ln(1/\varepsilon_k)})^{-1}$ diverges. If we take K large enough that $\sum_{k=k_0}^{K}(3e^2\sqrt{\ln(1/\varepsilon_k)})^{-1} > 1$, we obtain a contradiction because for any i, the events $\{\mathcal{E}_{i,k} \mid k_0 \leq k \leq K\}$ are disjoint subsets of $\{0,1\}^{T_K}$ whose combined probability, under the measure $q^{T_K}_{0,K}$, exceeds 1. □

Theorem 6.1 demonstrates that the ε-good algorithms presented above are optimal up to a constant factor. The combination of Theorems 6.1 and 6.2 demonstrates that the good algorithms presented above are also optimal up to a constant factor.

References

[1] Noga Alon, Joel H. Spencer, and Paul Erdős. *The Probabilistic Method*. Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley and Sons, Inc., New York, 1992.

[2] J. Aslam and A. Dhagat. Searching in the presence of linearly bounded errors. In *Proc. ACM Symposium on the Theory of Computing (STOC)*, pages 486–493, 1991.

[3] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. John Wiley and Sons, Inc., New York, 1991.

[4] A. Dhagat, P. Gacs, and P. Winkler. On playing "twenty questions" with a liar. In *Proc. ACM Symposium on Discrete Algorithms (SODA)*, pages 16–22, 1992.

[5] Robert Kleinberg and Tom Leighton. The value of knowing a demand curve: Bounds on regret for posted-price auctions. In *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 594–605, 2003.

[6] A. Pelc. Searching with known error probability. *Theoretical Computer Science*, 63:185–202, 1989.

[7] R.L. Rivest, A.R. Meyer, D.J. Kleitman, K. Winklmann, and J. Spencer. Coping with errors in binary search procedures. *Journal of Computer and System Sciences*, 20:396–404, 1980.

[8] R.S. Borgstrom snd S.R. Kosaraju. Comparison-based search in the prsence of errors. In *Proc. ACM Symposium on the Theory of Computing (STOC)*, pages 130–136, 1993.

[9] J. Spencer. Ulam's searching game with a fixed number of lies. *Theoretical Computer Science*, 95:307–321, 1992.