# Optimal Search for Rationals

Stephen Kwek* Kurt Mehlhorn†

July 4, 2001

**Abstract**

We present a $\Theta(\log_2 M)$-time algorithm that determines an unknown rational number $x$ in $\mathcal{Q}_M = \left\{ \frac{p}{q} : p, q \in \{1, ..., M\} \right\}$ by asking at most $2\log_2 M + O(1)$ queries of the form "Is $x \leq y$?".

**keywords:** combinatorial algorithms, binary search, rational numbers

## 1 Introduction

Students in all introductory programming courses are taught how to determine an unknown integer $x$, $x \in \{1, .., M\}$, by posing queries of the form "Is $x \leq y$?" using the fundamental binary search algorithm. The number of queries needed is at most $\lceil \log_2 M \rceil$, the maximum number of bits needed to represent $x$, which meets the decision theoretic lower bound [1, 5]. The binary search technique was further generalized by Hassin and Megiddo [3] and Karp [4]. They developed algorithms to determine a non-decreasing integer-valued function whose range and domain are in the sets $\{1, .., M\}$ and $\{0, .., k\}$ respectively. The original binary search algorithm can be viewed as having $k = 1$.

However, it is clear that if $x$ is an irrational number then we cannot determine $x$ in any finite number of queries. The natural question to ask next is how many queries are needed to determine a positive rational number $x$ where the denominator and numerator are integers bounded by $M$. Note that floating point representation can be interpreted as a rational number. Some applications for searching rationals can be found in [2, 8, 9].

An immediate solution to this problem is to list all the $\Theta(M^2)$ possible rational numbers in an array, sort them and perform a binary search on the sorted array. The maximum number of queries needed in this solution matches the decision theoretic lower bound of $2\log_2 M$. However, the algorithm has a preprocessing phase that requires $\Theta(M^2)$ time and space. Efficient $\Theta(\log_2 M)$-time algorithms had been proposed by Reiss [7] and Papadimitriou [6]. The algorithm of Reiss is a simple binary search that makes at most $\lceil \log_2 2M^3 \rceil$ queries but it does not determine $x$ exactly. Instead, it outputs a rational approximation of $x$ where the error is bounded by $1/(2M^2)$. Papadimitriou's algorithm determines $x$ exactly but the number of queries needed may be as high as $10\lceil \log_2 M \rceil + O(1)$.

We present an algorithm that requires only $2\log_2 M + O(1)$ queries; this matches the information theoretic lower bound. Our algorithm can be viewed as a refinement of Reiss binary search algorithm (which does not identify $x$ exactly). Our algorithm requires no preprocessing and runs in $O(\log M)$ time and space. We assume that rationals in $\mathcal{Q}_M$ can be stored in constant space and that arithmetic operations (addition, subtraction, multiplication, division) on them take constant time.

*University of Texas at San Antonio, Department of Computer Science, Email: kwek@jazz.cs.utsa.edu

†Max-Planck-Institute for Informatics, Email: mehlhorn@mpi-sb.mpg.de

```
binarySearch : x
    % determine the integer part
    use exponential and binary search to determine ⌊x⌋
    % determine the fractional part.
    𝓜 ← ⌊M/⌊x⌋⌋
    use binary search to determine an interval [μ/(2𝓜²), (μ+1)/(2𝓜²)]
    a, b ← findFraction(μ, 2𝓜², μ + 1, 2𝓜²) (see Figure 2)
    return ⌊x⌋ + a/b
```

Figure 1: A simple algorithm that performs binary search on the rational line.

## 2 Our Algorithm

Note that $x$ can be expressed as $\lfloor x \rfloor + \frac{a}{b}$ where $a$ and $b$ are relatively prime and $a < b$. Our algorithm uses exponential and binary search to determine $\lfloor x \rfloor$ using $\log_2 \lfloor x \rfloor + O(1)$ queries (see Lemma 2). We then use Lemma 5 to determine the fractional part $\frac{a}{b}$ by asking at most $2\log_2 M - 2\log_2 \lfloor x \rfloor + O(1)$ queries. We obtain.

**Theorem 1** *Let $x$ be an arbitrary number in $\mathcal{Q}_M = \left\{ \frac{p}{q} : p, q \in \{1, ..., M\} \right\}$. Suppose we are given an oracle that takes an input $y$ and answers query of the form "Is $x \leq y$?". Then we can identify the number $x$ in $\Theta(\log M)$ time and space by making at most $2\log_2 M + O(1)$ queries to the oracle.*

### Searching for the Integer Part

We use exponential and binary search. We first compare $x$ with $2^k$ for $k = 0, 1, 2, \ldots$ until $x \leq 2^k$ (exponential search) and then use binary search to locate $x$ in the interval $[2^{k-1}, 2^k]$. The number of comparisons required is $2k + O(1) = 2\log_2 \lfloor x \rfloor + O(1)$.

**Lemma 2** *The integer part $\lfloor x \rfloor$ can be determined using at most $2\log_2 \lfloor x \rfloor + O(1)$ queries in time $O(\log_2 M)$.*

### Searching for the Fractional Part

To determine $a/b$ efficiently, we need the following lemma that bounds $a$ and $b$ further.

**Lemma 3** $0 \leq a < b \leq \mathcal{M} = \lfloor M/\lfloor x \rfloor \rfloor$. *That is $a/b \in \mathcal{Q}_{\mathcal{M}}$.*

**Proof:** Suppose $x = \alpha/b$ where $\alpha = \lfloor x \rfloor b + a$. Then $\alpha/b \geq \lfloor x \rfloor$ and thus, $b \leq \alpha/\lfloor x \rfloor \leq M/\lfloor x \rfloor$. □

To exactly determine the fractional part, we perform a binary search on the unit interval $[\lfloor x \rfloor, \lfloor x \rfloor + 1]$ so that we know $\frac{a}{b} \in [\frac{\mu}{2\mathcal{M}^2}, \frac{\mu+1}{2\mathcal{M}^2}]$ for some $\mu$. This can be done by asking $\lceil \log_2(2\mathcal{M}^2) \rceil = 2\log M - 2\log \lfloor x \rfloor + O(1)$ queries. The following lemma states that the number in $\mathcal{Q}_{\mathcal{M}}$ that lies in $[\frac{\mu}{2\mathcal{M}^2}, \frac{\mu+1}{2\mathcal{M}^2}]$ is unique.

**Lemma 4** *[7] Suppose $\frac{a}{b}, \frac{c}{d} \in \mathcal{Q}_{\mathcal{M}}$ and $\frac{a}{b}, \frac{c}{d} \in [\frac{\mu}{2\mathcal{M}^2}, \frac{\mu+1}{2\mathcal{M}^2}]$. Then $\frac{a}{b} = \frac{c}{d}$.*

```
findFraction(α, β, γ, δ) : a, b
    if ⌊α/β⌋ = ⌊γ/δ⌋ and α/β ∉ Z                          (Case 1)
        b, a' ← findFraction(δ, γ mod δ, β, α mod β)
        a = ⌊α/β⌋ b + a'                                  (Equation 1)
        return a, b
    else                                                   (Case 2)
        return a = ⌈α/β⌉, b = 1
```

Figure 2: An algorithm for finding a fraction $\frac{a_{min}(I)}{b_{min}(I)} \in I = \left[\frac{\alpha}{\beta}, \frac{\gamma}{\delta}\right]$ such that for all $\frac{a}{b} \in I$, $a \geq a_{min}(I), b \geq b_{min}(I)$.

**Proof:** We enclose the simple proof for the sake of completeness. Assume $\frac{a}{b} \neq \frac{c}{d}$. Then

$$|\frac{a}{b} - \frac{c}{d}| = |\frac{ac - bd}{bd}| \geq \frac{1}{\mathcal{M}^2}$$

$\square$

At this point, we have determined $x$ in the sense that there is only a single choice for $x$. We have not made it explicit yet, i.e., we still need to compute $a$ and $b$. We show that this can be done in time $O(\log_2(M))$ and $\Theta(1)$ space without asking any further query. The algorithm is essentially continued fraction expansions; see [8, section 6.1].

## Making the Fractional Part Explicit

Suppose we know that the desired $\frac{a}{b}$ is in $I = [\frac{\mu}{2\mathcal{M}^2}, \frac{\mu+1}{2\mathcal{M}^2}]$ as in Lemma 4. Further, without loss of generality, we can assume $a$ and $b$ to be relatively prime. Since $\frac{a}{b}$ is the only fraction in $\mathcal{Q}_M \cap I$, all fractions in $I$ not equal to $\frac{a}{b}$ must have denominator greater than $b$. Thus, it suffices to find the fraction that has the smallest denominator in $I$.

**Lemma 5** *Given an interval $I = [\frac{\alpha}{\beta}, \frac{\gamma}{\delta}]$, there exists a fraction $\frac{a_{min}(I)}{b_{min}(I)}$ in $I = \left[\frac{\alpha}{\beta}, \frac{\gamma}{\delta}\right]$ such that for all $\frac{a}{b} \in I$, $a_{min}(I) \leq a, b_{min}(I) \leq b$. Further, we can determine this fraction in time $O(\log_2(\max(\alpha, \beta, \gamma, \delta)))$.*

**Proof:** We prove the existence of $a_{min}(I)$ and $b_{min}(I)$ by constructing it using a recursive algorithm. Our algorithm (see Figure 2) has the same flavor as Euclid's algorithm for finding the greatest common divisor of two integers [1]. We distinguish two cases.

Assume first that $I$ contains an integer, say it contains the integers $z_1 < ... < z_k$. We claim that $\forall \frac{a}{b} \in I, a \geq z_1$. This is clearly true if $z_1 = 1$ or $b = 1$ or $\frac{a}{b} \geq z_1$. Thus, suppose $z_1 - 1 < \frac{a}{b} < z_1$ and $b \neq 1$ and $z_1 \neq 1$. Then $a > b(z_1 - 1)$ which implies $a \geq z_1$. Hence we have $a_{min}(I) = z_1$ and $b_{min}(I) = 1$.

Assume next that $I$ contains no integer. Let $\frac{a}{b}$ be an arbitrary fraction in $I$. In this case, we have

$$\frac{\alpha}{\beta} \leq \frac{a}{b} \leq \frac{\gamma}{\delta} \quad \text{and} \quad \left\lfloor \frac{\alpha}{\beta} \right\rfloor = \left\lfloor \frac{a}{b} \right\rfloor = \left\lfloor \frac{\gamma}{\delta} \right\rfloor.$$

We can express $a$ as

$$a = \left\lfloor \frac{a}{b} \right\rfloor b + a' = \left\lfloor \frac{\alpha}{\beta} \right\rfloor b + a' \tag{1}$$

where $a' = a \bmod b$. Let $\alpha' = \alpha \bmod \beta$ and $\gamma' = \gamma \bmod \delta$. Then, we also have

$$\frac{\alpha'}{\beta} \leq \frac{a'}{b} \leq \frac{\gamma'}{\delta} \quad \text{and hence} \quad \frac{\delta}{\gamma'} \leq \frac{b}{a'} \leq \frac{\beta}{\alpha'}.$$

That is, $\frac{b}{a'} \in I'$ where $I' = \left[\frac{\delta}{\gamma'}, \frac{\beta}{\alpha'}\right]$. Notice that if there exists $\hat{b}, \hat{a'} \in I'$ such that for all $\frac{b}{a'} \in I', b \geq \hat{b}$ and $a' \geq \hat{a'}$, then substituting $\hat{b}$ for $b$ and $\hat{a'}$ for $a'$ in Equation (1) gives us the smallest $a$ among all feasible $b$ and $a'$ such that $\frac{b}{a'} \in I$. That is, to prove the existence of $a_{min}(I)$ and $b_{min}(I)$, it suffices to prove the existence of $a_{min}(I')$ and $b_{min}(I')$. Similarly, to determine $a_{min}(I)/b_{min}(I)$, it is sufficient to solve the problem with the interval $I'$.

If $I'$ contains an integer, then the problem instance is reduced to Case 1. Thus, suppose $I'$ does not contain an integer. Notice that $\gamma' \leq \gamma$ and $\alpha' \leq \alpha$. Suppose $\gamma' \not< \gamma$ and $\alpha' \not< \alpha$. That is, $\gamma' = \gamma$ and $\alpha' = \alpha$, then by repeating the above argument, we have

$$\frac{\alpha'}{\beta'} \leq \frac{a'}{b'} \leq \frac{\gamma'}{\delta'}$$

where $b' = b \bmod a'$, $\beta' = \beta \bmod \alpha'$ and $\delta' = \delta \bmod \gamma'$. That is, the problem is reduced to finding $a_{min}(I'')$ and $b_{min}(I'')$ where $I'' = [\frac{\alpha'}{\beta'}, \frac{\gamma'}{\delta'}]$. Further, as $\gamma = \gamma' = \gamma \bmod \delta$, we have $\gamma' < \delta$ which implies $\delta'(= \delta \bmod \gamma') < \delta$. Similarly, $\gamma' < \gamma$.

In other words, by reducing the problem instance (*i.e.* an interval) in this manner, we are sure that at least one number in $\{\alpha, \beta, \gamma, \delta\}$ is replaced with a smaller number and none of them is replaced with a larger number. Eventually the problem instance must contain an integer and the algorithm terminates (see Case 1). In the worst case, we stop when the interval being considered is $[\frac{1}{1}, \frac{1}{1}]$.

Figure 2 summarizes the algorithm for determining $a_{min}(I)$ and $b_{min}(I)$. The method we used to reduce the denominator and numerator of the endpoints of $I$ is essentially the same as Euclid's algorithm for finding the greatest common divisor of two numbers $x$ and $y$. The time complexity for Euclid's algorithm is $O(\mathcal{F}^{-1}(\max(x, y))) = O(\log_2(\max(x, y)))$ where $\mathcal{F}^{-1}(\alpha)$ is the largest $k$ such that $\alpha$ is less than the $k$th Fibonacci number [1]. Thus, we have the desired time complexity. The space bound follows since each level of the recursions requires constant space. $\square$

# References


[1] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to algorithms*. MIT Press and McGraw-Hill Book Company, 6th edition, 1992.

[2] P. Goldberg and S. Kwek. The precision of query points as a resource for learning convex polytopes with membership queries. In *Proc. 13th Annual Conference on Computational Learning Theory*, pages 225–235. Morgan Kaufmann, 2000.

[3] R. Hassin and N. Megiddo. An optimal algorithm for finding all the jumps of a monotone step-function. *Journal of Algorithms*, 6:265–274, 1985.

[4] R. M. Karp. A generalization of binary search. *Lecture Notes in Computer Science*, 709:27–33, 1993.

[5] D. E. Knuth. *The Art of Computer Programming III: Sorting and Searching.* Addison–Wesley, Reading, Massachusetts, 1973.

[6] C. Papadimitriou. Efficient search for rationals. *Information Processing Letters*, 8:1–4, 1979.

[7] S. Reiss. Rational search. *Information Processing Letters*, 8:89–90, 1979.

[8] A. Schrijver. *Theory of Linear and Integer Programming.* Wiley, 1986.

[9] E. Zemel. On search over rationals. *Operations Research Letters*, 1:34–38, 1981.