Available at www.ComputerScienceWeb.com
POWERED BY SCIENCE @ DIRECT®

Information Processing Letters 86 (2003) 23–26

Information Processing Letters

www.elsevier.com/locate/ipl

Optimal search for rationals

Stephen Kwek [a,1], Kurt Mehlhorn [b,*]

[a] *University of Texas at San Antonio, Department of Computer Science, San Antonio, TX 78249, USA*
[b] *Max-Planck-Institute fur Informatics, Im Stadtwald, Geb. 44, Saarbrücken 66123, Germany*

Received 20 July 2001; received in revised form 1 October 2002

Communicated by P.M.B. Vitányi

Abstract

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

Keywords: Combinatorial algorithms; Binary search; Rational numbers; Algorithms

1. Introduction

Students in all introductory programming courses are taught how to determine an unknown integer x, $x \in \{1, \ldots, M\}$, by posing queries of the form "Is $x \leqslant y$?" 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 $\{1, \ldots, M\}$ and $\{0, \ldots, k\}$, 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

* Corresponding author.
E-mail addresses: kwek@cs.utsa.edu (S. Kwek), mehlhorn@mpi-sb.mpg.de (K. Mehlhorn).

[1] This work is partially supported by NSF Grant CCR-0208935.

0020-0190/02/$ – see front matter
doi:10.1016/S0020-0190(02)00455-6

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 $10\lceil\log_2 M\rceil + \mathrm{O}(1)$.

We present an algorithm that requires only $2\log_2 M + \mathrm{O}(1)$ 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 $\mathrm{O}(\log M)$ 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.

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 $\log_2\lfloor x\rfloor + \mathrm{O}(1)$ queries (see Lemma 2). We then use Lemma 5 to determine the fractional part $\frac{a}{b}$ by asking at most $2\log_2 M - 2\log_2\lfloor x\rfloor + \mathrm{O}(1)$ queries. We obtain:

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

2.1. 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 $x \leqslant 2^k$ (exponential search) and then use binary search to locate x in the interval $[2^{k-1}, 2^k]$. The number of comparisons required is

$$2k + \mathrm{O}(1) = 2\log_2\lfloor x\rfloor + \mathrm{O}(1).$$

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

2.2. Searching for the fractional part

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

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

Proof. Suppose $x = \alpha/b$ where $\alpha = \lfloor x\rfloor b + a$. Then $\alpha/b \geqslant \lfloor x\rfloor$ and thus, $b \leqslant \alpha/\lfloor x\rfloor \leqslant M/\lfloor x\rfloor$. □

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 μ. This can be done by asking $\lceil\log_2(2\mathcal{M}^2)\rceil = 2\log M - 2\log\lfloor x\rfloor + \mathrm{O}(1)$ 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 $\frac{a}{b} = \frac{c}{d}$.*

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

$$\left|\frac{a}{b} - \frac{c}{d}\right| = \left|\frac{ac - bd}{bd}\right| \geqslant \frac{1}{\mathcal{M}^2}. \qquad \square$$

binarySearch: x
  % determine the integer part
  use exponential and binary search to determine $\lfloor x\rfloor$
  % determine the fractional part.
  $\mathcal{M} \leftarrow \lfloor M/\lfloor x\rfloor\rfloor$
  use binary search to determine an interval $[\frac{\mu}{2\mathcal{M}^2}, \frac{\mu+1}{2\mathcal{M}^2}]$
  $a, b \leftarrow$ *findFraction*$(\mu, 2\mathcal{M}^2, \mu + 1, 2\mathcal{M}^2)$ (see Fig. 2)
  return $\lfloor x\rfloor + a/b$

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

$$
\begin{array}{ll}
\textit{findFraction}(\alpha, \beta, \gamma, \delta)\colon a, b & \\
\quad \textbf{if } \lfloor \frac{\alpha}{\beta} \rfloor = \lfloor \frac{\gamma}{\delta} \rfloor \textbf{ and } \frac{\alpha}{\beta} \notin \mathbf{Z} & \text{(Case 1)} \\
\qquad b, a' \leftarrow \textit{findFraction}(\delta, \gamma \bmod \delta, \beta, \alpha \bmod \beta) & \\
\qquad a = \lfloor \frac{\alpha}{\beta} \rfloor b + a' & \text{(Eq. (1))} \\
\qquad \textbf{return } a, b & \\
\quad \textbf{else} & \text{(Case 2)} \\
\qquad \textbf{return } a = \lceil \frac{\alpha}{\beta} \rceil, b = 1 &
\end{array}
$$

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

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 $\mathrm{O}(\log_2(M))$ and $\Theta(1)$ space without asking any further query. The algorithm is essentially continued fraction expansions; see [8, Section 6.1].

2.3. *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 $a_{\min}(I)/b_{\min}(I)$ in $I = [\frac{\alpha}{\beta}, \frac{\gamma}{\delta}]$ such that for all $\frac{a}{b} \in I$, $a_{\min}(I) \leqslant a$, $b_{\min}(I) \leqslant b$. Further, we can determine this fraction in time* $\mathrm{O}(\log_2(\max(\alpha, \beta, \gamma, \delta)))$.

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

Case 1: Assume first that I contains an integer, say it contains the integers $z_1 < \cdots < z_k$. We claim that $\forall \frac{a}{b} \in I,\ a \geqslant z_1$. This is clearly true if $z_1 = 1$ or $b = 1$ or $\frac{a}{b} \geqslant z_1$. 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 $a \geqslant z_1$. Hence we have $a_{\min}(I) = z_1$ and $b_{\min}(I) = 1$.

Case 2: Assume next that I contains no integer. Let $\frac{a}{b}$ be an arbitrary fraction in I. In this case, we have

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

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

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

That is, $\frac{b}{a'} \in I'$ where $I' = [\frac{\delta}{\gamma'}, \frac{\beta}{\alpha'}]$. Notice that if there exists $\hat{b}, \hat{a}' \in I'$ such that for all $\frac{b}{a'} \in I',\ b \geqslant \hat{b}$ and $a' \geqslant \hat{a}'$, then substituting $\hat{b}$ for b and $\hat{a}'$ for a' in Eq. (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 $\gamma' \leqslant \gamma$ and $\alpha' \leqslant \alpha$. Suppose $\gamma' \nless \gamma$ and $\alpha' \nless \alpha$. That is, $\gamma' = \gamma$ and $\alpha' = \alpha$, then by repeating the above argument, we have

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

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 $[\frac{1}{1}, \frac{1}{1}]$.

Fig. 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

$$\mathrm{O}\big(\mathcal{F}^{-1}\big(\max(x, y)\big)\big) = \mathrm{O}\big(\log_2\big(\max(x, y)\big)\big),$$

where $\mathcal{F}^{-1}(\alpha)$ is the largest k such that α is less than the kth Fibonacci number [1]. Thus, we have the desired time complexity. The space bound follows since each level of the recursions requires constant space. □

References

[1] T.H. Cormen, C.E. Leiserson, R.L. Rivest, Introduction to Algorithms, 6th edn., MIT Press, Cambridge, MA, 1992.

[2] P. Goldberg, 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, Morgan Kaufmann, Los Altos, CA, 2000, pp. 225–235.

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

[4] R.M. Karp, A generalization of binary search, in: Lecture Notes in Comput. Sci., Vol. 709, Springer, Berlin, 1993, pp. 27–33.

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

[6] C. Papadimitriou, Efficient search for rationals, Inform. Process. Lett. 8 (1979) 1–4.

[7] S. Reiss, Rational search, Inform. Process. Lett. 8 (1979) 89–90.

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

[9] E. Zemel, On search over rationals, Oper. Res. Lett. 1 (1981) 34–38.