Available online at www.sciencedirect.com

ScienceDirect

Theoretical Computer Science 370 (2007) 240–253

Theoretical Computer Science

www.elsevier.com/locate/tcs

On searching a table consistent with division poset[☆]

Yongxi Cheng[a,*], Xi Chen[a], Yiqun Lisa Yin[b]

[a] *Department of Computer Science, Tsinghua University, Beijing 100084, China*
[b] *Independent Security Consultant, Greenwich CT, USA*

Received 5 April 2006; received in revised form 18 October 2006; accepted 31 October 2006

Communicated by E. Pergola

Abstract

Suppose $P_n = \{1, 2, \ldots, n\}$ is a partially ordered set with the partial order defined by divisibility, that is, for any two elements $i, j \in P_n$ satisfying i divides j, we have $i \leq_{P_n} j$. A table $A_n = \{a_i | i = 1, 2, \ldots, n\}$ of real numbers is said to be *consistent* with P_n, provided that for any two elements $i, j \in \{1, 2, \ldots, n\}$ satisfying i divides j, $a_i \leq a_j$. Given a real number x, we want to determine whether $x \in A_n$, by comparing x with as few entries of A_n as possible. In this paper, we investigate the complexity $\tau(n)$, measured by the number of comparisons, of the above search problem. We present a $\frac{55n}{72} + O(\ln^2 n)$ search algorithm for A_n and prove a lower bound $(\frac{3}{4} + \frac{17}{2160})n + O(1)$ on $\tau(n)$ using an adversary argument.
© 2006 Elsevier B.V. All rights reserved.

Keywords: Search algorithm; Complexity; Partially ordered set; Divisibility

1. Introduction

Suppose $P = \{1, 2, \ldots, n\}$ is a partially ordered set (*poset*), we say a table $A = \{a_i | i = 1, 2, \ldots, n\}$ of n real numbers *consistent* with P, provided that for any $i, j \in P$ satisfying $i \leq_P j$, $a_i \leq a_j$. Given a table A of real numbers which are consistent with a known poset P, and given a real number x, we would like to determine whether $x \in A$, by making a series of comparisons between x and certain elements $a_i \in A$. The problem is considered in a model using pairwise comparisons of the form $x : a_i$ ($a_i \in A$) as the basic operations. These comparisons have ternary outcomes $x < a_i$, $x = a_i$, or $x > a_i$. Our aim is to make as few comparisons as possible. The *complexity* of the problem is defined to be the minimum(over all search algorithms for P) of the maximum number of comparisons required in the worst-case.

In this article, we consider the above search problem for the case where the partial order is defined by divisibility. Let $P_n = \{1, 2, \ldots, n\}$ be a poset with a partial order such that, for any two elements $i, j \in P_n$ satisfying i divides

[☆] This work was supported in part by National Natural Science Foundation of China Grant 60553001 and National Basic Research Program of China Grant 2007CB807900, 2007CB807901.

[*] Corresponding author. Tel.: +86 10 62797304.
E-mail addresses: cyx@mails.tsinghua.edu.cn (Y. Cheng), xichen00@mails.tsinghua.edu.cn (X. Chen), yiqun@alum.mit.edu (Y.L. Yin).

0304-3975/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2006.10.027

j, $i \leq_{P_n} j$. Thus, we can say a table $A_n = \{a_i | i = 1, 2, \ldots, n\}$ of real numbers is consistent with P_n if for any two elements $i, j \in \{1, 2, \ldots, n\}$ satisfying i divides j, $a_i \leq a_j$. Denote by $\tau(n)$ the complexity of the problem of searching for a given real number x in A_n. We will investigate both the upper bounds (i.e., search algorithms for A_n) and the lower bounds on $\tau(n)$, and our main result is the following:

Theorem 1.1. *For $n \geq 1$, $c_1 n + O(1) \leq \tau(n) \leq c_2 n + O(\ln^2 n)$, where c_1 and c_2 are constants, and $c_1 = \frac{3}{4} + \frac{17}{2160} \approx 0.758$, $c_2 = \frac{55}{72} \approx 0.764$.*

Throughout the paper, we denote by $\lceil x \rceil$ (the ceiling of x) the least integer that is not less than x, and denote by $\lfloor x \rfloor$ (the floor of x) the largest integer that is not greater than x.

2. Related work

In [5,6], Linial and Saks have studied the above class of search problems for the general finite partially ordered set P. In [5] some general bounds on the complexity are provided; more precise results are also presented for the case that P is a product of chains and that P is a rooted forest. In [6] it was proved that, for general finite partially ordered sets, the information theoretically bound for the complexity is tight up to a multiplicative constant.

There are different perspectives on the problem of searching for posets. In [2], the authors have studied the tradeoff between the preprocessing time and the subsequent search time in a partial order. Let $P(n)$ be the worst-case cost of a preprocessing algorithm, which builds some partial orders, and let $S(n)$ be the maximum number of comparisons required to answer a membership query. They proved that $P(n)+n \log_2 S(n) \geq (1+o(1))n \log_2 n$ for any comparison-based algorithm.

A different notion of searching a poset was studied in [1,3]. They considered searching for a given element x in a poset P, instead of searching for a given real number in a table of real numbers consistent with P. In this case, for each comparison $x : p_i$ ($p_i \in P$), there are two possible outcomes: ‘yes’ indicates that x is ‘below’ p_i (less than or equal to p_i); ‘no’ indicates that x is not below p_i. The aim is to find the optimal search strategy. In spite of the similarity in the definitions, this turns out to be quite a different model. In [1], the authors have given a polynomial time algorithm for posets that have tree structures. In [3], the authors proved that the problem is NP-hard in general, and they also gave a $(1 + o(1))$-approximation algorithm under the *random graph* model and a 6.34-approximation algorithm under the *uniform* model; both of these run in polynomial time.

3. Easy bounds on $\tau(n)$

First, we give an easy lower bound $\frac{3n}{4}$ on $\tau(n)$ and a simple asymptotical cn algorithm searching A_n, where $c \approx 0.81$ is a constant.

3.1. Lower bounds $(3/4)n$

It is easy to see that the following simple *response strategy* for the adversary can guarantee that at least $\frac{3n}{4}$ comparisons of the form $x : a_i$ ($a_i \in A_n$) are required to determine whether $x \in A_n$, for any search algorithm.

Response Strategy RS_1*: when the algorithm asks about $x : a_i$, $i = 1, 2, \ldots, \lfloor \frac{n}{2} \rfloor$, answer $x > a_i$; when the algorithm asks about $x : a_i$, $i = \lfloor \frac{n}{2} \rfloor + 1, \ldots, n$, answer $x < a_i$.*

Focus on the subset $A_n^* = \{a_i | \frac{n}{4} < i \leq n\}$ of A_n; if the adversary answers queries in the above manner, the algorithm needs the comparison of x with each element $a_i \in A_n^*$ to determine whether $x = a_i$; it thus needs at least $\frac{3n}{4}$ comparisons. In fact, the set $P_n^* = \{i \in P_n | \frac{n}{4} < i \leq n\}$ is a *section* (see [5]) of P_n , and there is no ordered chain that has a length more than two in P_n^*.

Remark 3.1. The following is a further exposition of the above argument for the lower bounds $(3/4)n$. For any algorithm searching A_n, we answer the queries $x : a_i$ according to RS_1, for any step S to which the algorithm had been implemented, if there exists $a_k \in A_n^*$ such that the algorithm doen not compare x with a_k, consider the following two assignments to A_n:

Assignment 1: $a_i = x - 1$ for $1 \leq i \leq \frac{n}{2}$; $a_i = x + 1$ for $\frac{n}{2} < i \leq n$.

Assignment 2: The same as Assignment 1, except that $a_k = x$ (notice that $\frac{n}{4} < k \leq n$).

It is easy to see that for both the above assignments, A_n is consistent with P_n, and when the assigned array A_n is searched by the same algorithm to step S, the answers will be the same as in RS_1. Notice that $x \notin A_n$ in Assignment 1, and $x \in A_n$ in Assignment 2; however, the algorithm could not distinguish the above assignments if they were only implemented to step S. It follows that no search algorithm could determine whether $x \in A_n$ if it hasn't compared all the elements in A_n^* with x.

3.2. A simple search algorithm

We will give an asymptotical cn algorithm searching A_n based on a binary search, where $c = \sum_{t=0}^{\infty} \frac{1}{2^{2^t}} \approx 0.81$ is a constant. Define $B_i = \{a_j \in A_n | j = (2i-1) \times 2^k\ (\le n),\ k = 0, 1, \ldots\}$, $i = 1, 2, \ldots, \lceil n/2 \rceil$. Then $B_1, B_2, \ldots, B_{\lceil n/2 \rceil}$ is a partition of A_n, and each B_i is in linear ordering. The algorithm runs as follows:

Algorithm 1 (*Searching Table A_n*). Binary search $B_1, B_2, \ldots, B_{\lceil n/2 \rceil}$ one by one.

Now, we analyze the number of comparisons required by Algorithm 1. For each B_i, the binary search needs at most $\lceil \log_2(|B_i| + 1) \rceil$ comparisons, $i = 1, 2, \ldots, \lceil n/2 \rceil$. Since $1 \le |B_i| \le \lfloor \log_2 n \rfloor + 1$, and the number of the sets B_i that have exactly k elements is less than $(\frac{n}{2^{k+1}} + 1)$, it follows that the total number of comparisons required by Algorithm 1 is at most:

$$\begin{aligned} s_1(n) &= \sum_{k=1}^{\lfloor \log_2 n \rfloor + 1} \left(\frac{n}{2^{k+1}} + 1 \right) \lceil \log_2(k+1) \rceil \\ &< \sum_{k=1}^{\infty} \frac{n}{2^{k+1}} \lceil \log_2(k+1) \rceil + \sum_{k=1}^{\lfloor \log_2 n \rfloor + 1} \lceil \log_2(k+1) \rceil \\ &= n \cdot \sum_{t=0}^{\infty} \frac{1}{2^{2^t}} + O(\ln n \cdot \ln \ln n). \end{aligned}$$

4. Upper bounds $\frac{55n}{72} + O(\ln^2 n)$ on $\tau(n)$

In this section, we present a $\frac{55n}{72} + O(\ln^2 n)$ search algorithm for A_n, by partitioning A_n into 2-dimensional *layers* and then searching them one by one.

Let $I_n = \{i | 1 \le i \le n,\ i$ does not have factor 2 or 3$\}$. For each $i \in I_n$, extend a_i to a subset of A_n, L_i, such that $L_i = \{a_{i \times 2^k \times 3^s} \in A_n | k, s = 0, 1, 2, \ldots\}$. For instance, for $n = 15$, we have subsets $L_1, L_5, L_7, L_{11}, L_{13}$, and

$$L_1 = \begin{bmatrix} a_1 & a_2 & a_4 & a_8 \\ a_3 & a_6 & a_{12} & \\ a_9 & & & \end{bmatrix} \quad L_5 = \begin{bmatrix} a_5 & a_{10} \\ a_{15} & \end{bmatrix} \quad L_7 = \begin{bmatrix} a_7 & a_{14} \end{bmatrix} \quad L_{11} = \begin{bmatrix} a_{11} \end{bmatrix} \quad L_{13} = \begin{bmatrix} a_{13} \end{bmatrix}.$$

It is easy to see that all these subsets L_i (which sometimes will be referred to as *layers*), $i \in I_n$, form a partition of A_n.

An algorithm searching for a real number x in an $m \times n$ $(m, n \ge 1)$ *monotone matrix* (a matrix with entries increasing along each row and each column) was described in [5], which repeats comparing x with the element e at the top right corner of the current matrix, and either the first row or the rightmost column of the current matrix will be eliminated depending on whether $x > e$ or $x < e$; thus the algorithm requires at most $m+n-1$ comparisons (see [4,5] for the lower bounds on the number of comparisons required for this problem when $m = n$). Based on this algorithm and the fact that a layer is a "triangular" portion of a monotone matrix, we can apply the above "$m+n-1$" algorithm searching for a layer. Furthermore, for some of the layers we can do slightly better by exploiting the properties of the layers.

Lemma 4.1. *If a layer L with $|L| \notin \{1, 2, 3, 5\}$, then one can search x in L using at most $m + n - 2$ comparisons, where m and n are the numbers of rows and columns of L, respectively.*

Proof. See Appendix A. ■

By Lemma 4.1 we can obtain an improved search algorithm for A_n.

Algorithm 2 (*Searching Table* A_n). Search all layers L_i one by one. If $|L_i| \in \{1, 2, 3, 5\}$, search L_i using the "$m + n - 1$" algorithm in [5]; otherwise, search L_i using the "$m + n - 2$" algorithm in Lemma 4.1.

Next, we analyze the number of comparisons required by Algorithm 2. Define $r(L_i)$ and $c(L_i)$ to be the numbers of rows and columns of L_i respectively. The total number of comparisons required by Algorithm 2 is at most

$$\begin{aligned} s_2(n) &= \sum_{i\in I_n,\ |L_i|\in\{1,2,3,5\}} (r(L_i) + c(L_i) - 1) + \sum_{i\in I_n,\ |L_i|\notin\{1,2,3,5\}} (r(L_i) + c(L_i) - 2) \\ &= \sum_{i\in I_n} r(L_i) + \sum_{i\in I_n} c(L_i) - 2\sum_{i\in I_n} 1 + \sum_{i\in I_n,\ |L_i|\in\{1,2,3,5\}} 1. \end{aligned}$$

Clearly, $r(L_i) \le 1 + \log_3 n$ for any $i \in I_n$. The number of layers L_i with $r(L_i) = p$ is less than $(\frac{2n}{3^{p+1}} + 1)$, since $3^{p-1}i \le n$, $3^p i > n$ and i has no factor 2 or 3. Similarly, $c(L_i) \le 1 + \log_2 n$ for any $i \in I_n$, and the number of layers L_i with $c(L_i) = q$ is less than $(\frac{n}{3\times 2^q} + 1)$. It follows that:

$$\sum_{i\in I_n} r(L_i) < \sum_{p=1}^{1+\log_3 n} \left(\frac{2n}{3^{p+1}} + 1\right) p < \sum_{p=1}^{\infty} \frac{2np}{3^{p+1}} + \sum_{p=1}^{1+\log_3 n} p = \frac{n}{2} + O(\ln^2 n),$$

$$\sum_{i\in I_n} c(L_i) < \sum_{q=1}^{1+\log_2 n} \left(\frac{n}{3\times 2^q} + 1\right) q < \sum_{q=1}^{\infty} \frac{nq}{3\times 2^q} + \sum_{q=1}^{1+\log_2 n} q = \frac{2n}{3} + O(\ln^2 n).$$

In addition, $\sum_{i\in I_n} 1 = |I_n| = \frac{n}{3} + O(1)$, and

$$\sum_{i\in I_n,\ |L_i|\in\{1,2,3,5\}} 1 = \sum_{i\in I_n,\ \frac{n}{2}<i\le n} 1 + \sum_{i\in I_n,\ \frac{n}{3}<i\le\frac{n}{2}} 1 + \sum_{i\in I_n,\ \frac{n}{4}<i\le\frac{n}{3}} 1 + \sum_{i\in I_n,\ \frac{n}{8}<i\le\frac{n}{6}} 1 = \frac{19n}{72} + O(1).$$

Therefore, $s_2(n) \le \frac{55n}{72} + O(\ln^2 n)$.

5. Lower bounds $(\frac{3}{4} + \frac{1}{432})n + O(1)$ on $\tau(n)$

In this section, we prove a lower bound $(\frac{3}{4} + \frac{1}{432})n + O(1)$ on $\tau(n)$, by using an adversary argument. Thus, the previous easy lower bound $\frac{3n}{4}$ is not the best possible.

5.1. The main idea in constructing lower bounds

Recall the response strategy RS_1 for the adversary given in Section 3, which guarantees that at least $\frac{3n}{4}$ comparisons are needed to determine whether $x \in A_n$ for any search algorithm. We can view RS_1 in the following way:

In Algorithm 2, A_n is partitioned into layers L_i. For each row R in each layer, if R has only one element, we pick this element and say that it forms a *unit*; if R has at least two elements, we pick the last two elements as a unit. We will call a unit consisting of one or two elements a *1-unit* or *2-unit*, respectively. In total. $\frac{3n}{4}$ elements are picked, which form exactly the subset $A_n^* = \{a_i \mid \frac{n}{4} < i \le n\}$ of A_n. We can now restate the response strategy RS_1 as follows:
The response strategy for the elements in a 1-unit: for $\frac{n}{2} < i \le n$, i is odd, when the algorithm asks about $x : a_i$, answer $x < a_i$.

$$\begin{bmatrix} \cdots & \cdots \\ \{\ulcorner a_i\ \} & \end{bmatrix}$$

The response strategy for the elements in a 2-unit: for $\frac{n}{4} < i \le \frac{n}{2}$, when the algorithm asks about $x : a_i$, answer $x > a_i$; when the algorithm asks about $x : a_{2i}$, answer $x < a_{2i}$.

$$\left[\begin{array}{llll} \cdots & \cdots & \cdots & \cdots \\ \cdots & \{\, a_i \lrcorner\,, & \ulcorner a_{2i} \,\} & \\ \cdots & \cdots & & \end{array}\right]$$

The response strategy for the elements not in any unit: for $1 \le i \le \frac{n}{4}$, when the algorithm asks about $x : a_i$, answer $x > a_i$.

In general, if an algorithm compares x with a_i and gets the result $x < a_i$, then all the elements a_k with k divisible by i are known to be larger than x, and thus could be eliminated from consideration, we say that these elements are *cut* by a_i. Similarly, if the algorithm gets the result $x > a_i$, then all the elements a_k with k that divide i are known to be smaller than x and could be eliminated, we also say that these elements are cut by a_i.

It is easy to see that under the response strategy RS_1, any element that belongs to a particular unit could not be cut by any other element. Therefore, if the strategy RS_1 is adopted by the adversary, for any search algorithm in order to determine whether $x \in A_n$, the $\frac{3n}{4}$ comparisons of x with all the elements in all the units are necessary, thus obtaining the lower bound $\frac{3n}{4}$.

Hereafter, for a response strategy, the notation '$a_i \lrcorner$' means that when queried by any algorithm with the comparison $x : a_i$, answer $x > a_i$, and we say that a_i *cuts to the left up*; '$\ulcorner a_i$' means that when queried by the comparison $x : a_i$, answer $x < a_i$, and we say that a_i *cuts to the right bottom.*

A natural thought for constructing better lower bounds could be: If more elements from each row are picked, can we prove that more than $\frac{3n}{4}$ comparisons are required? Picking three elements from a row makes no difference, since all the elements in a row are in linear order, and two comparisons are sufficient to search three ordered elements. Therefore, at least four elements should be chosen from some rows, to guarantee that at least three comparisons are required to search for x in them.

We will pick elements in the following way. If there are less than four elements in a row, we pick all the elements as a unit; if there are at least four elements in a row, we pick the last four elements as a unit. However, in general, we could no longer guarantee, as we do under the strategy RS_1, that each of the elements picked cannot be cut by other elements. Actually, we cannot even guarantee that any element picked would not be cut by the elements in the other units. An element that cannot be cut by any element outside its unit, under some response strategy, can guarantee the number of comparisons required by any search algorithm. We call this kind of element *essential* (notice that an essential element may be cut by other elements in the same unit). Next, we will present a more effective response strategy, in which there are sufficient essential elements to guarantee that more than $\frac{3n}{4}$ comparisons are required, for any search algorithm.

5.2. Units and special units

Let us start with some definitions. As described above, there is exactly one *unit* in each row of each layer. If in a row there are less than four elements, all these elements form a unit; if in a row there are at least four elements, the last four elements form a unit. A unit consisting of one, two, three, or four elements is called *1-unit*, *2-unit*, *3-unit*, or *4-unit* respectively. E.g., in the below layer (1), we have a 1-unit $\{a_{9i}\}$, a 3-unit $\{a_{3i}, a_{6i}, a_{12i}\}$, and a 4-unit $\{a_{2i}, a_{4i}, a_{8i}, a_{16i}\}$.

Next, we introduce an important subcollection of the units defined above, called *special units*, which is the key to the proof of new lower bounds.

Definition 5.1. *Special units:* A unit u is called a *special unit* if u is the 4-unit in a layer L_i with $|L_i| = 9$ (i.e., a layer L_i with $i \in S_n$, where $S_n = \{i \mid \frac{n}{18} < i \le \frac{n}{16}, i$ is not divisible by 2 or 3$\}$). We denote by 4-*unit*$_s$ a special unit.

Since the form of a layer L_i is determined by its number of elements, $|L_i|$, a layer containing a special unit must have the following form (the marks '$\lrcorner$' and '$\ulcorner$' indicating the cut directions of the elements will be explained later in the new response strategy, RS_2). Each such layer contains exactly one special unit, $\{a_{2i}, a_{4i}, a_{8i}, a_{16i}\}$, in its first row.

$$\begin{bmatrix} a_i\ , & \{\, a_{2i}\lrcorner\ , & \ulcorner a_{4i}\lrcorner\ , & \ulcorner a_{8i}\lrcorner\ , & \ulcorner a_{16i}\,\} \\ a_{3i}\lrcorner\ , & \ulcorner a_{6i}\ , & \ulcorner a_{12i} & & \\ \ulcorner a_{9i} & & & & \end{bmatrix} \tag{1}$$

A unit that is not a special unit is called a *general unit*, thus all 1-units, 2-units, and 3-units are general units, and subcollections of 4-units are general units.

Special units can help establish better lower bounds as we will prove later that, under the response strategy RS_2 described below, all elements in special units are *essential*; thus each special unit guarantees at least three necessary comparisons for any search algorithm, which is one more than the two necessary comparisons guaranteed by the 2-unit in its row under RS_1. Consequently, the lower bound $\frac{3n}{4}$ can be improved by the number of special units, $|S_n|$.

5.3. New response strategy

Now, we are ready to describe the new response strategy for the adversary, and show that it can guarantee that at least $(\frac{3}{4} + \frac{1}{432})n + O(1)$ comparisons are required for any search algorithm to determine whether $x \in A_n$.

Response Strategy RS_2***:***

For the elements not in a unit: for $1 \le i \le \frac{n}{16}$, when the algorithm asks about $x : a_i$, answer $x > a_i$.

For the elements in a 1-unit: for $\frac{n}{2} < i \le n$, i is odd, when the algorithm asks about $x : a_i$, answer $x < a_i$.

$$\begin{bmatrix} \cdots & \cdots \\ \{\,\ulcorner a_i\,\} & \end{bmatrix} \tag{2}$$

For the elements in a 2-unit: for $\frac{n}{4} < i \le \frac{n}{2}$, i is odd, when the algorithm asks about $x : a_i$, answer $x > a_i$; when the algorithm asks about $x : a_{2i}$, answer $x < a_{2i}$.

$$\begin{bmatrix} \cdots & \cdots & \cdots \\ \{\, a_i\lrcorner\ , & \ulcorner a_{2i}\,\} & \\ \cdots & & \end{bmatrix} \tag{3}$$

The 3-units are partitioned into two classes according to the number of elements of the next row below them, which are denoted by 3-*unit*$_1$'s and 3-*unit*$_2$'s respectively.

For the elements in a 3-unit$_1$ *(the 1st class of 3-units whose next row contains one element):* for $\frac{n}{6} < i \le \frac{n}{4}$, i is odd, when the algorithm asks about $x : a_i$, answer $x > a_i$; when the algorithm asks about $x : a_{2i}$ or $x : a_{4i}$, answer $x < a_{2i}$ or $x < a_{4i}$ respectively.

$$\begin{bmatrix} \cdots & \cdots & \cdots & \cdots \\ \{\, a_i\lrcorner\ , & \ulcorner a_{2i}\ , & \ulcorner a_{4i}\,\} & \\ \ulcorner a_{3i} & & & \end{bmatrix} \tag{4}$$

For the elements in a 3-unit$_2$ *(the 2nd class of 3-units whose next row contains two elements):* for $\frac{n}{8} < i \le \frac{n}{6}$, i is odd, when the algorithm asks about $x : a_i$ or $x : a_{2i}$, answer $x > a_i$ or $x > a_{2i}$ respectively; when the algorithm asks about $x : a_{4i}$, answer $x < a_{4i}$.

$$\begin{bmatrix} \cdots & \cdots & \cdots & \cdots \\ \{\, a_i\lrcorner\ , & a_{2i}\lrcorner\ , & \ulcorner a_{4i}\,\} & \\ a_{3i}\lrcorner\ , & \ulcorner a_{6i} & & \end{bmatrix} \tag{5}$$

There are two classes of 4-units, general 4-units and special 4-units, which are denoted by 4-*unit*$_g$'s and 4-*unit*$_s$'s respectively.

For the elements in a 4-unit$_g$ (general 4-unit): for $\frac{n}{16} < i \leq \frac{n}{8}$, $i \notin 2S_n$ (where $2S_n = \{2j | j \in S_n\}$, and see Definition 5.1 for S_n), when the algorithm asks about $x : a_i$ or $x : a_{2i}$, answer $x > a_i$ or $x > a_{2i}$ respectively; when the algorithm asks about $x : a_{4i}$ or $x : a_{8i}$, answer $x < a_{4i}$ or $x < a_{8i}$ respectively.

$$\begin{bmatrix} \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \{ a_i \lrcorner, & a_{2i} \lrcorner, & \ulcorner a_{4i}, & \ulcorner a_{8i} \} & \\ \cdots & \cdots & \cdots & \cdots & & \end{bmatrix} \tag{6}$$

For the elements in a 4-unit$_s$ (special 4-unit): for $i \in S_n$, $\{a_{2i}, a_{4i}, a_{8i}, a_{16i}\}$ is a special unit, the response strategy for the elements in a special unit is adaptive, depending on the order of comparisons with x made by the search algorithm, to guarantee that at least three comparisons are needed to determine whether $x \in \{a_{2i}, a_{4i}, a_{8i}, a_{16i}\}$.

If the algorithm first asks about $x : a_{2i}$, answer $x > a_{2i}$. Then a_{2i} will be eliminated, and the remaining three elements will follow the strategy $\{a_{4i}\lrcorner, \ulcorner a_{8i}, \ulcorner a_{16i}\}$ for possible subsequent comparisons with x.

If the algorithm first asks about $x : a_{4i}$, answer $x > a_{4i}$. Then a_{2i} and a_{4i} are known to be smaller than x and will be eliminated, and the remaining two elements will follow the strategy $\{a_{8i}\lrcorner, \ulcorner a_{16i}\}$.

If the algorithm first asks about $x : a_{8i}$, answer $x < a_{8i}$. Then a_{8i} and a_{16i} will be eliminated, and the remaining two elements will follow the strategy $\{a_{2i}\lrcorner, \ulcorner a_{4i}\}$.

If the algorithm first asks about $x : a_{16i}$, answer $x < a_{16i}$. Then a_{16i} will be eliminated, and the remaining three elements will follow the strategy $\{a_{2i}\lrcorner, a_{4i}\lrcorner, \ulcorner a_{8i}\}$.
We denote the response strategy of the elements in a special unit by $\{a_{2i}\lrcorner, \ulcorner a_{4i}\lrcorner, \ulcorner a_{8i}\lrcorner, \ulcorner a_{16i}\}$, see (1).

5.4. *Lower bounds* $(\frac{3}{4} + \frac{1}{432})n + O(1)$ *on* $\tau(n)$

Recall that we called an element $a_i \in A_n$ *essential* under the strategy RS$_2$, if a_i belongs to some unit u (i.e., $i > \frac{n}{16}$), and any element not in u cannot cut a_i under RS$_2$.
Define set $E_n = \{$*all elements of 1-units*$\} \cup \{$*all elements of 2-units*$\} \cup \{$*all the first and second elements of 3-unit*$_1$*'s*$\} \cup \{$*all the second and third elements of 3-unit*$_2$*'s*$\} \cup \{$*all the second and third elements of 4-unit*$_g$*'s*$\} \cup \{$*all elements of 4-unit*$_s$*'s*$\}$. We can prove the following lemma.

Lemma 5.1. *Under the response strategy* RS$_2$*, all the elements of* E_n *are essential.*

Proof of Lemma 5.1. See Appendix B. ■

By Lemma 5.1 and the response strategy RS$_2$, each essential element in the general units needs one comparison with x to determine whether it equals x. For each special unit u, at least three comparisons between x and its elements are needed to determine whether $x \in u$. By comparing the 1-units and 2-units picked for RS$_1$ in Section 5.1, we can see that under the strategy RS$_2$, each row containing a general unit contributes the same number of necessary comparisons as it contributes under RS$_1$. While for each row containing a special unit, it contributes one more necessary comparison than it does under RS$_1$. Therefore, the lower bound $\frac{3n}{4}$ could be improved by the number of special units, $|S_n|$, which is $|\{i : \frac{n}{18} < i \leq \frac{n}{16}, i \text{ is not divisible by 2 or 3}\}| = (\frac{n}{16} - \frac{n}{18}) \times \frac{1}{3} + O(1) = \frac{n}{432} + O(1)$. Thus, we get a new lower bound $(\frac{3}{4} + \frac{1}{432})n + O(1)$.

Remark 5.2. Similarly to Remark 3.1 in Section 3.1, we can give a further exposition of the above lower bound arguments. That is, for any search algorithm that doesn't make sufficiently many comparisons, we can give two explicit assignments to A_n (one contains x and the other doesn't) such that the algorithm cannot distinguish them; thus it cannot determine whether $x \in A_n$. Since the ideas are the same, we append the detailed discussion in Appendix C.

6. Improved lower bounds $(\frac{3}{4} + \frac{17}{2160})n + O(1)$ on $\tau(n)$

By recognizing more special units, we can extend the above method to obtain a better lower bound $(\frac{3}{4} + \frac{17}{2160})n + O(1)$ on $\tau(n)$. The units are defined in the same way as in Section 5.2, and we will introduce three classes of special units.

Definition 6.1. *The 1st class of special units (4-unit$_{s,1}$):* A unit u is of the 1st class of special units if u is the 4-unit in a layer L_i with $|L_i| = 9$ and i is not divisible by 5.

The definition of 4-*unit*$_{s,1}$ is just Definition 5.1 of 4-*unit*$_s$ with the extra restriction that i is not divisible by 5. A layer L_i containing a 4-*unit*$_{s,1}$ must have the form (1), each such layer contains exactly one special unit $\{a_{2i}, a_{4i}, a_{8i}, a_{16i}\}$ in its first row. The subscripts $j = 2i$ of the first elements of all 4-*unit*$_{s,1}$'s form a set $S_{n,1} = \{j(=2i) | \frac{n}{9} < j \le \frac{n}{8}$, j is divisible by 2, but is not divisible by 3, 4 or 5$\}$.

Definition 6.2. *The 2nd class of special units (4-unit$_{s,2}$):* A 4-unit u is of the 2nd class of special units if it has the following properties:

1. u is the 4-unit in the *first* row of a layer L_j with j not divisible by 5.
2. The first row of L_j has at least six elements and has two more elements than the second row of L_j.

The layers containing a 4-*unit*$_{s,2}$ must have the following form (7), where the symbol '•' indicates that there must exist an element at the position. Each such layer contains exactly one special unit, $\{a_i, a_{2i}, a_{4i}, a_{8i}\}$, in its first row. The subscripts i of the first elements of all 4-*unit*$_{s,2}$'s form a set $S_{n,2} = \{i | \frac{n}{12} < i \le \frac{n}{8}$, i is divisible by 4, but is not divisible by 3 or 5$\}$.

$$\begin{bmatrix} \cdots & \bullet & \bullet & \{a_i\lrcorner & \ulcorner a_{2i}\lrcorner & \ulcorner a_{4i}\lrcorner & \ulcorner a_{8i}\} \\ \cdots & \bullet\lrcorner & \bullet\lrcorner & \ulcorner a_{3i} & \ulcorner a_{6i} & & \\ \cdots & \cdots & \cdots & \cdots & & & \end{bmatrix} \tag{7}$$

Definition 6.3. *The 3rd class of special units (4-unit$_{s,3}$):* A 4-unit $u = \{a_i, a_{2i}, a_{4i}, a_{8i}\}$ is of the 3rd class of special units if it is in a layer L_j with j not divisible by 5, and has the following properties:

1. The subscript of its first element, i, is divisible by 36 (i.e., in L_j, there are at least two rows above u, and at least two columns before the first element a_i of u).
2. Denote by R the row containing u in L_j. The next row above R has two more elements than R, and the next row below R has two less elements than R.

A 4-*unit*$_{s,3}$ in a layer must have the following form (8) (see Lemma D.1 in Appendix D), here we only list the subscripts.

$$\begin{bmatrix} \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \bullet & \bullet & \bullet & \bullet & \\ \cdots & \cdots & \cdots & \cdots & \cdots & \frac{4i}{3}\lrcorner & \frac{8i}{3}\lrcorner & \ulcorner\frac{16i}{3} & \ulcorner\frac{32i}{3} & & \\ \cdots & \cdots & \cdots & \{i\lrcorner & \ulcorner 2i\lrcorner & \ulcorner 4i\lrcorner & \ulcorner 8i\} & & & & \\ \cdots & \bullet\lrcorner & \bullet\lrcorner & \ulcorner 3i & \ulcorner 6i & & & & & & \\ \cdots & \bullet & \bullet & \ulcorner 9i & & & & & & & \\ \cdots & \cdots & & & & & & & & & \end{bmatrix} \tag{8}$$

The subscripts i of the first elements of all 4-*unit*$_{s,3}$'s form a set $S_{n,3} = \{i \mid \frac{n}{12} < i \leq \frac{3n}{32}$, i is divisible by 4 and 9, but is not divisible by 5$\}$.

Based on the above newly defined special units, we have the following response strategy:

Response Strategy RS_2^****:*** In RS_2^*, the response strategy for the elements in general units, special units and not in any units are the same as their strategy in RS_2 respectively, with only a different partition of 4-units into general units and special units.

Similarly, define set E_n^*=\{*all elements of 1-units*\} ∪ \{*all elements of 2-units*\} ∪ \{*all the first and second elements of 3-unit*$_1$*'s*\} ∪ \{*all the second and third elements of 3-unit*$_2$*'s*\} ∪ \{*all the second and third elements of 4-unit*$_g$*'s*\} ∪ \{*all elements of special units 4-unit*$_{s,1}$*'s, 4-unit*$_{s,2}$*'s and 4-unit*$_{s,3}$*'s*\}. We can prove the following lemma:

Lemma 6.1. *Under* RS_2^**, all elements of* E_n^* *are essential.*

Proof of Lemma 6.1. See Appendix D. ■

Using similar arguments as in Section 5.4, the lower bound $\frac{3n}{4}$ could be improved by the number of the newly defined special units, $|S_{n,1}| + |S_{n,2}| + |S_{n,3}|$, which is

$$\left(\frac{n}{8} - \frac{n}{9}\right) \times \frac{1}{4} \times \frac{2}{3} \times \frac{4}{5} + O(1) + \left(\frac{n}{8} - \frac{n}{12}\right) \times \frac{1}{4} \times \frac{2}{3} \times \frac{4}{5} + O(1) + \left(\frac{3n}{32} - \frac{n}{12}\right) \times \frac{1}{4} \times \frac{1}{9} \times \frac{4}{5} + O(1)$$
$$= \frac{17n}{2160} + O(1).$$

Remark 6.2. By simply replacing RS_2 and E_n by RS_2^* and E_n^* respectively, the argument in Appendix C can be directly applied here for the new lower bound $(\frac{3}{4} + \frac{17}{2160})n + O(1)$.

7. Concluding remarks and open problems

In this paper, we investigate the complexity, $\tau(n)$, of the problem of searching a table consistent with division posets. Our main result is the following: For $n \geq 1$, $c_1 n + O(1) \leq \tau(n) \leq c_2 n + O(\ln^2 n)$, where c_1 and c_2 are constants, and $c_1 = \frac{3}{4} + \frac{17}{2160} \approx 0.758$, $c_2 = \frac{55}{72} \approx 0.764$. It may be of interest to further close the gap.

Notice that in the model presented in this paper. we only allow comparisons of the form $x : a_i$, i.e., all comparisons must involve x. If we also allow pairwise comparisons among the elements of A_n, then the techniques used in this paper to prove lower bounds will not apply directly. It may be interesting to investigate the complexity of the search problem in this new model.

Acknowledgments

The authors are grateful to Andrew C. Yao for introducing this interesting problem and for his insightful comments, and to Xiaoming Sun and Chen Wang for helpful discussions. Also, the authors would like to thank the referees for their valuable suggestions and comments.

Appendix A. Proof of Lemma 4.1

It is easy to see that the form of L is determined by its cardinality $|L|$. First, we prove two lemmas that will be useful later.

Lemma A.1. *In any layer, the difference of the lengths of any two consecutive rows must be 1 or 2.*

Proof of Lemma A.1. We prove the lemma by contradiction. Otherwise, at least one of the following two situations exists:

$$\begin{bmatrix} \dots & a_i \\ \dots & a_{3i} \end{bmatrix} \qquad \begin{bmatrix} \dots & a_i & a_{2i} & a_{4i} & a_{8i} & \dots \\ \dots & a_{3i} & & & & \end{bmatrix}$$

However, according to the definition of layers, the left one cannot happen because the element a_{2i} should be in the layer, and the right one cannot happen because the element a_{6i} should be in the layer. Thus the lemma holds.

Lemma A.2. *In any layer, there cannot be three consecutive rows with lengths successively increased by one.*

Proof of Lemma A.2. Once again, we proceed by contradiction. Otherwise, the following situation must be the case:

$$\begin{bmatrix} \dots & a_i & a_{2i} & a_{4i} \\ \dots & a_{3i} & a_{6i} & \\ \dots & a_{9i} & & \end{bmatrix}$$

However, this cannot happen since the element a_{8i} should be in the layer. The lemma holds. ■

Now, we are ready to prove Lemma 4.1. By Lemma A.1, we have the following two cases:
Case 1. The first row of L has two more elements than the second row. Since $|L| \geq 4$, L must have the following form

$$\begin{bmatrix} \dots & a_i & a_{2i} & a_{4i} \\ \dots & a_{3i} & & \\ \dots & & & \end{bmatrix}$$

First compare x with a_{2i}.

If $x < a_{2i}$, then a_{2i} and a_{4i} are known to be greater than x, and will be eliminated from L, leaving a portion of an $m \times (n-2)$ monotone matrix, which can be searched using at most $m + (n-2) - 1$ comparisons. In total, at most $m + n - 2$ comparisons are needed.

If $x > a_{2i}$, then all elements in the first row except a_{4i} are known to be smaller than x, and will be eliminated. Then we compare x with a_{4i} and eliminate it, leaving a portion of an $(m-1) \times (n-2)$ monotone matrix, which can be searched using at most $(m-1) + (n-2) - 1$ comparisons. In total, at most $m + n - 2$ comparisons are needed.
Case 2. The first row of L has one more element than the second row. We can assume $|L| \geq 8$, since for $|L| = 4$, 6, or 7 it is easy to verify that the first row of L has two more elements than the second row, which belong to Case 1. When $|L| \geq 8$, by Lemmas A.2 and A.1 the second row of L has two more elements than the third row, thus L must have the following form:

$$\begin{bmatrix} \dots & a_i & a_{2i} & a_{4i} & a_{8i} \\ \dots & a_{3i} & a_{6i} & a_{12i} & \\ \dots & a_{9i} & & & \\ \dots & & & & \end{bmatrix}$$

First, compare x with a_{4i}.

If $x < a_{4i}$, then the two rightmost columns are known to be greater than x and will be eliminated, leaving a portion of an $m \times (n-2)$ monotone matrix, which can be searched using at most $m + (n-2) - 1$ comparisons. In total, at most $m + n - 2$ comparisons are needed.

If $x > a_{4i}$, then all the elements in the first row except a_{8i} are known to be smaller than x, and will be eliminated. Then, compare x with a_{8i} and eliminate it, leaving a portion of an $(m-1) \times (n-1)$ monotone matrix whose first row has two more elements than the second row. Thus, it reduces to the situation of Case 1, which needs at most $(m-1) + (n-1) - 2$ comparisons. In total, at most $m + n - 2$ comparisons are needed.
Therefore, in either case $m + n - 2$ comparisons suffice.

Appendix B. Proof of Lemma 5.1

We prove the lemma by contradiction. Suppose that, under RS$_2$, $a_j \in E_n$ is not essential, i.e., there exists an element $a_i \in A_n$ that cuts a_j, and a_i is not in the unit containing a_j. We have the following two cases:
Case 1. a_i and a_j are in one layer. If a_i does not belong to any unit, then $i \leq n/16$ and a_i always cuts to the left up. If a_i cuts an element a_j, then $2j \leq i$, $32j \leq 16i \leq n$, which implies that $a_{2j}, a_{4j}, a_{8j}, a_{16j}$ all exist in the row containing a_j, and thus a_j cannot be in a unit. Therefore, a_i cannot cut any element in a unit. If a_i belongs to a special unit, 4-*unit*$_s$, notice the form and the response strategy of special units, (1), and it is easy to see that a_i cannot cut any element $a_j \in E_n$ in a different unit in the same layer. If a_i belongs to a general unit, we have the following eight subcases:

1. a_i is the last element of a unit. Then a_i always cuts to the right bottom, $\{\dots, \ulcorner a_i\}$, and $2i > n$. If a_i cuts an element a_j, then $j \geq 2i > n$, which contradicts with $j \leq n$. Thus a_i cannot cut any element.

2. a_i is the second last (first) element of a 2-unit, then it cuts to the left up, $\{a_i\lrcorner, a_{2i}\}$. If a_i cuts $a_j \in E_n$ in the same layer, then a_j must be the first element of a row above a_i. In addition, a_j must be in the next row R above a_i (since by Lemma A.2 the rows above R have at least 5 elements, if a_j is in a unit in those rows, then a_j cannot be the first element of that row). By Lemma A.1, R has 3 or 4 elements. If R has 3 elements, then it is a 3-*unit*$_2$ and its first element does not belong to E_n. If R has 4 elements, then it is a 4-*unit*$_g$ and its first element does not belong to E_n.
3. a_i is the second last (second) element of a 3-*unit*$_1$, then it cuts to the right bottom, $\{a_{i/2}, \ulcorner a_i, a_{2i}\}$. Notice the form of 3-*unit*$_1$, (4), a_i cannot cut any element in a different unit in the same layer.
4. a_i is the second last (second) element of a 3-*unit*$_2$, then it cuts to the left up, $\{a_{i/2}, a_i\lrcorner, a_{2i}\}$. Since the row below a_i has two elements, then by Lemmas A.1 and A.2, the next row above a_i, R, has 5 elements. Thus, if a_i cuts $a_j \in E_n$ in the same layer, a_j must be in R, and is the first element of the 4-unit U of R. Notice the form of the special units, (1), in which the bottom row has one element. Therefore, U is not a special unit and its first element does not belong to E_n.
5. a_i is the second last (third) element of a 4-*unit*$_g$, then it cuts to the right bottom, $\{a_{i/4}, a_{i/2}, \ulcorner a_i, a_{2i}\}$. Denote by U the 4-*unit*$_g$ which a_i is in. Notice the form of the general units, (6); a_j must be the last element of R, where R is the next row below a_i and has one less element than the row containing a_i. If R has at least four elements, then a_j must be the last element of a 4-*unit*$_g$, thus $a_j \notin E_n$. Otherwise, R must have exactly three elements, and then a_j must be the third element of R. Since R has one less element than the next row above it, by Lemmas A.1 and A.2, the next row below R has one element, and it follows that R is a 3-*unit*$_1$; thus its third element $a_j \notin E_n$.
6. a_i is the third last (first) element of a 3-*unit* (3-*unit*$_1$ or 3-*unit*$_2$), then it cuts to the left up, $\{a_i\lrcorner, a_{2i}, a_{4i}\}$. If a_i cuts $a_j \in E_n$ in the same layer, then a_j must be the first element of R, where R is the next row above a_i. In addition, R must contain exactly four elements. Notice the form of the special units, (1); R is not a 4-*unit*$_s$, and thus its first element $a_j \notin E_n$.
7. a_i is the third last (second) element of a 4-*unit*$_g$, then it cuts to the left up, $\{a_{i/2}, a_i\lrcorner, a_{2i}, a_{4i}\}$. In this case, a_j must be in the next row above a_i, R. In addition, R has one more element than the row containing a_i, and a_j is the first element of the 4-unit U in R. Notice the form of the special units, (1); U is not a 4-*unit*$_s$, and thus its first element $a_j \notin E_n$.
8. a_i is the fourth last (first) element of a 4-*unit*$_g$, then it cuts to the left up, $\{a_i\lrcorner, a_{2i}, a_{4i}, a_{8i}\}$. If a_i cuts an element a_j, then $2j \le i$, $16j \le 8i \le n$, which implies that $a_{2j}, a_{4j}, a_{8j}, a_{16j}$ all exist in the row containing a_j, and thus a_j cannot be in a unit. Therefore, a_i cannot cut any element in a unit.

Case 2. a_i and a_j are in different layers. We first prove the following lemma that will be useful later.

Lemma B.1. *For any $a_{j_1}, a_{j_2} \in A_n$ in different layers, if j_1 divides j_2, then the quotient is at least 5.*

Proof of Lemma B.1. Suppose that $a_{j_1} \in L_{i_1}$ with $j_1 = i_1 \times 2^{k_1} \times 3^{s_1}$, $a_{j_2} \in L_{i_2}$ with $j_2 = i_2 \times 2^{k_2} \times 3^{s_2}$, where L_{i_1} and L_{i_2} are different layers (i.e., $i_1 \ne i_2$) and j_1 divides j_2. Since i_1, i_2 have no factor 2 or 3, we have $k_1 \le k_2$ and $s_1 \le s_2$, and i_1 divides i_2 with quotient at least 5. It follows that j_1 divides j_2 with quotient at least 5. ■

There are seven subcases in Case 2.

1. a_i does not belong to any unit. Then $i \le \frac{n}{16}$ and a_i always cuts to the left up, and by using the same argument as at the beginning of Case 1, a_i cannot cut any element in a unit.
2. a_i is the last element of a unit. Then a_i always cuts to the right bottom, $\{\dots, \ulcorner a_i\}$, and $2i > n$. By using the same argument as in subcase 1 of Case 1, a_i cannot cut any element.
3. a_i is the second last element of a unit and cuts to the left up, $\{\dots, a_i\lrcorner, a_{2i}\}$. By Lemma B.1, if a_i cuts $a_j \in E_n$ in a different layer then $5j \le i$, $10j \le 2i \le n$, thus a_{2j}, a_{4j}, a_{8j} all exist in the row containing a_j. If $a_j \in E_n$, a_j can only be the first element of some special unit, it follows that $j/2 \in S_n = \{k \in I_n | \frac{n}{18} < k \le \frac{n}{16}\}$ (see Definition 5.1), thus $9j > n$, which contradicts with $10j \le n$.
4. a_i is the second last element of a unit and cuts to the right bottom, $\{\dots, \ulcorner a_i, a_{2i}\}$. Thus $4i > n$. By Lemma B.1, if a_i cuts $a_j \in E_n$ in a different layer, then $j \ge 5i > n$, which contradicts $j \le n$.
5. a_i is the third last element of a unit and cuts to the left up, $\{\dots, a_i\lrcorner, a_{2i}, a_{4i}\}$. By Lemma B.1, if a_i cuts $a_j \in E_n$ in a different layer then $5j \le i$, and thus $20j \le 4i \le n$. It follows that $a_{2j}, a_{4j}, a_{8j}, a_{16j}$ all exist in the row containing a_j; thus a_j cannot be in a unit, which contradicts $a_j \in E_n$.

6. a_i is the third last element of a unit and cuts to the right bottom. In this case, a_i must be the second element of a special unit, $\{a_{i/2}, \ulcorner a_i, a_{2i}, a_{4i}\}$. It follows that $i/4 \in S_n = \{k \in I_n | \frac{n}{18} < k \leq \frac{n}{16}\}$, thus $i > \frac{2n}{9}$. However, by Lemma B.1, if a_i cuts $a_j \in E_n$ in a different layer, then $j \geq 5i > \frac{9i}{2} > n$, which contradicts $j \leq n$.
7. a_i is the fourth last element of a unit, i.e., the first element of a 4-unit, $\{a_i\lrcorner, a_{2i}, a_{4i}, a_{8i}\}$. Then a_i always cuts to the left up, and by using the same argument as in subcase 8 of Case 1, a_i cannot cut any element in a unit.

Appendix C. Further exposition of the lower bound arguments

For any algorithm searching A_n, we answer the queries $x : a_i$ according to RS_2, for any step S to which the algorithm had been implemented, consider the following assignment to A_n (in which $x \notin A_n$).

Assignment to A_n:

For the elements a_i's not in the special units, under RS_2 they have a *fixed* answer when compared with x. Assign $x - 1$ to a_i if the answer is "$x > a_i$", and assign $x + 1$ to a_i if the answer is "$x < a_i$".

For the special units, there are two cases:

Case 1. If, to step S, the special unit u has at least one element that has been compared with x by the algorithm, then from RS_2 the elements in u will also have *fixed* answers (though the answers may be different depending on which element has been first compared with x). Assign $x - 1$ to $a_i \in u$ if the answer is "$x > a_i$", and assign $x + 1$ to $a_i \in u$ if the answer is "$x < a_i$".

Case 2. If, to step S, the special unit $u = \{a_{2i}\lrcorner, \ulcorner a_{4i}\lrcorner, \ulcorner a_{8i}\lrcorner, \ulcorner a_{16i}\}$ has no element that has been compared with x by the algorithm, we assign $x - 1$ to its first and second elements a_{2i} and a_{4i}, and assign $x + 1$ to its third and fourth elements a_{8i} and a_{16i}.

Lemma C.1. *The above assignment is consistent with P_n, and when the assigned array A_n is searched by the same algorithm to step S, the answers will be the same as in* RS_2.

Proof of Lemma C.1. It is easy to see that, up to step S, the answers will be the same as in RS_2. For their consistency with P_n, we will give a proof by contradiction.

Assume that in the above assignment there exists $a_i = x - 1$ and $a_j = x + 1$ with $j | i$. Notice that the above assignment has the property that if an element is assigned $x - 1$, then it *can* cut to the left up in RS_2; if an element is assigned $x + 1$, then it *can* cut to the right bottom in RS_2. It follows that in RS_2, a_i cuts a_j. Therefore, either a_j is not an essential element, or, a_i and a_j are in the same unit. However, if a_j is not an essential element and is assigned $x + 1$ in the above assignment, so a_j must be the last element of a 3-*unit*$_1$ or a 4-*unit*$_g$, which is contradictory since we will have $i \geq 2j > n$. If a_i and a_j are in the same unit, the only possibility is that they are in a special unit $\{a_{2k}\lrcorner, \ulcorner a_{4k}\lrcorner, \ulcorner a_{8k}\lrcorner, \ulcorner a_{16k}\}$, and $i = 8k$, $j = 4k$, but it is easy to see that this cannot happen in the above assignment. ■

For any step S to which the search algorithm has been implemented, define the set (of *remained* essential elements) $\mathrm{Re}(S) = \{a_i \in E_n |$ up to step S, a_i has neither been compared with x by the algorithm nor been cut by any other elements in the same unit$\}$. If $\mathrm{Re}(S)$ is not empty, we can modify the above assignment by one element in the following way, such that in the modified assignment $x \in A_n$.

Modified Assignment to A_n:

Consider an arbitrary unit u containing an element $a_k \in \mathrm{Re}(S)$, define $u' = \{a_i \in u \cap E_n |$ up to step S, a_i has neither been compared with x by the algorithm nor been cut by any other elements in $u\}$. Notice that u' is not empty since $a_k \in u'$.

If u' is still a full special unit containing four elements, modify the assignment of its second element from $x - 1$ to be x;

Otherwise, all the elements in u' have fixed answers. If there exist elements in u' having fixed answer "$<x$", modify the assignment of the element in u' with the largest subscript having the answer "$<x$" to be x;

Else, there must exist elements in u' having the fixed answer "$>x$", and we modify the assignment of the element in u' with the smallest subscript having the answer "$>x$" to be x.

Lemma C.2. *The above modified assignment is consistent with P_n, and when the re-assigned array A_n is searched by the same algorithm to step S, the answers will be the same as in* RS_2.

Proof of Lemma C.2. Since the modified element has not been queried by the algorithm to step S, the answers will still be the same as in RS_2. For the consistency with P_n, by Lemma C.1 any possible inconsistency between two elements of A_n must involve the modified element, and there are two possible cases of inconsistency:

Case 1. In the above modification, we have a_i modified from $x - 1$ to be x, which results in an inconsistency with a_j. It must be the case that $a_j = x - 1$ (if $a_j = x + 1$, a_i and a_j will form an inconsistency with the original assignment, which is a contradiction) and $i|j$. Notice that $a_j = x - 1$ implies that in RS_2, a_j can cut to the left up. Thus a_j cuts a_i. Since $a_i \in u'$ is essential, a_j must be in the same unit containing a_i. However, this cannot happen in the above modification, since if $a_j = x - 1$ with $j > i$ exists we will not modify a_i.

Case 2. In the above modification, we have a_i modified from $x + 1$ to be x, which results in inconsistency with a_j. Similarly, it must be the case that $a_j = x + 1$ and $j|i$. $a_j = x + 1$ implies that in RS_2, a_j can cut to the right bottom. Thus a_j cuts a_i. Since a_i is essential, again a_j must be in the same unit containing a_i. However, this also cannot happen in the above modification, since if $a_j = x + 1$ with $j < i$ exists. we will not modify a_i. ■

By the above arguments, if $\mathrm{Re}(S)$ is not empty, the algorithm cannot distinguish between the above two assignments; thus it cannot determine whether $x \in A_n$. In other words, if the response strategy RS_2 is adopted by the adversary, for any search algorithm to determine whether $x \in A_n$, the algorithm must be implemented to a step such that $\mathrm{Re}(S)$ is empty.

Suppose $\mathrm{Re}(S)$ is empty. For any $a_i \in E_n$ in a general unit, since a_i cannot be cut by any other elements (including the elements in the same unit), a_i must cost one comparison with x; for any special unit u, by the response strategy of special units in RS_2, if each element of u either has been compared with x or has been cut by other elements in u, at least three comparisons between x and the elements in u are required. Therefore, the lower bound $(\frac{3}{4} + \frac{1}{432})n + O(1)$ is justified.

Appendix D. Proof of Lemma 6.1

We prove the lemma by contradiction. Suppose that, under RS_2^*, $a_j \in E_n^*$ is not essential, i.e., there exists an element $a_i \in A_n$ that cuts a_j, and a_i is not in the unit containing a_j. We have the following two cases:
Case 1. a_i and a_j are in one layer. If a_i does not belong to any unit, then a_i always cuts to the left up, and using the same argument at the beginning of Case 1 in the proof of Lemma 5.1, a_i cannot cut any element in a unit. If a_i belongs to a special unit of the first or the second class, i.e., a 4-$unit_{s,1}$ or 4-$unit_{s,2}$, notice the forms of these special units, (1) and (7); it is easy to see that a_i cannot cut any element $a_j \in E_n^*$ in a different unit in the same layer (for the case where a_i is in a 4-$unit_{s,2}$, notice that the second row of any layer cannot contain a special unit). For the case where a_i belongs to a special unit of the third class, i.e. a 4-$unit_{s,3}$, we first give the two lemmas that will be useful. Similarly as Lemma A.2, we have the following:

Lemma D.1. *In any layer, there cannot be four consecutive rows with lengths successively increased by 2.*

The correctness of Lemma D.1 can be easily seen by noticing that in the third, fourth, fifth and sixth row in (8), $9i$ exists in the layer since $9i < \frac{32i}{3}$.

Lemma D.2. *If R is the next row above or below a 4-$unit_{s,3}$ in the same layer, then R does not contain a special unit.*

Lemma D.2 is true since by the definition of 4-$unit_{s,3}$, R is not the first row of its layer; thus it cannot contain a 4-$unit_{s,1}$ or a 4-$unit_{s,2}$; and by Lemma D.1, R either has one less element than the next row above R or has one more element than the next row below R; thus it cannot contain a 4-$unit_{s,3}$.

By Lemma D.2, if a_i belongs to a 4-$unit_{s,3}$, a_i cannot cut any element $a_j \in E_n^*$ in a different unit in the same layer. If a_i belongs to a general unit, then similarly to in Case 1 in the proof of Lemma 5.1, we have the eight subcases. For all these subcases, we can show that, using the same arguments in Lemma 5.1 correspondingly (sometimes when comes to special units, we only need to replace them with the newly defined special units 4-$unit_{s,1}$, 4-$unit_{s,2}$ and 4-$unit_{s,3}$), a_i cannot cut $a_j \in E_n^*$.
Case 2. a_i and a_j are in different layers. We first give the following two lemmas that will be useful later:

Lemma D.3. *For any $a_{j_1}, a_{j_2} \in A_n$ in different layers, if j_1 divides j_2, and j_2 is not divisible by 5, then the quotient is at least 7.*

Lemma D.4. *If i is the subscript of a first element in a special unit 4-unit$_{s,1}$, 4-unit$_{s,2}$ or 4-unit$_{s,3}$, then $12i > n$.*

Lemma D.3 can be proved in a similar way to Lemma B.1, and Lemma D.4 is true since by the definitions of 4-*unit*$_{s,1}$'s, 4-*unit*$_{s,2}$'s and 4-*unit*$_{s,3}$'s, the row containing a special unit always has two more elements than the next row below it.

Similarly to Case 2 in the proof of Lemma 5.1, we have the following seven subcases. Except subcase 3 and subcase 6, we can apply the same arguments in Lemma 5.1 to all the following subcases correspondingly.

1. a_i does not belong to any unit. By using the same argument given at the beginning of Case 1 in the proof of Lemma 5.1, a_i cannot cut $a_j \in E_n^*$.
2. a_i is the last element of a unit. Then a_i always cuts to the right bottom, $\{\ldots, \ulcorner a_i\}$. By using the same argument as in subcase 1 of Case 1 in the proof of Lemma 5.1, a_i cannot cut $a_j \in E_n^*$.
3. a_i is the second last element of a unit and cuts to the left up, $\{\ldots, a_i\lrcorner, a_{2i}\}$. By Lemma B.1, if a_i cuts $a_j \in E_n^*$ in a different layer, we have $5j \le i$, $10j \le 2i \le n$; thus a_{2j}, a_{4j}, a_{8j} all exist in the row containing a_j. If $a_j \in E_n^*$, a_j can only be the first element of a special unit. We have two possibilities for a_i.
   (a) a_i is the first element of a 2-unit or the second element of a 3-*unit*$_2$. In this case, a_j must be the first element of a 4-*unit*$_{s,1}$, since otherwise in the row containing a_j, there will be at least two elements before a_j, and thus a_i cannot cut a_j. However, by the form of 4-*unit*$_{s,1}$, (1), if a_j is the first element of a 4-*unit*$_{s,1}$, then $9j > n$, and this contradicts $10j \le n$. Thus, this possibility is eliminated.
   (b) a_i is the third element of a special unit. By the definition of the new defined special units, i is not divisible by 5. Thus, by Lemma D.3, $7j \le i$, $14j \le 2i \le n$. However, by Lemma D.4, if a_j is the first element of a special unit, then $12j > n$, which contradicts $14j \le n$.
4. a_i is the second last element of a unit and cuts to the right bottom, $\{\ldots, \ulcorner a_i, a_{2i}\}$. Using the same argument as in subcase 4 of Case 2 in the proof of Lemma 5.1, a_i cannot cut $a_j \in E_n^*$.
5. a_i is the third last element of a unit and cuts to the left up, $\{\ldots, a_i\lrcorner, a_{2i}, a_{4i}\}$. Using the same argument as in subcase 5 of Case 2 in the proof of Lemma 5.1, a_i cannot cut $a_j \in E_n^*$.
6. a_i is the third last element of a unit and cuts to the right bottom. In this case, a_i must be the second element of a special unit, $\{a_{i/2}, \ulcorner a_i, a_{2i}, a_{4i}\}$. By the form of 4-*unit*$_{s,1}$, (1), a_i cannot be in a 4-*unit*$_{s,1}$, since otherwise we have $9 \times i/2 > n$, it follows that $j \ge 5i > n$. Therefore, a_i can only be the second element of a 4-*unit*$_{s,2}$ or 4-*unit*$_{s,3}$; thus, i is divisible by 8. By Lemma D.4, $12 \times i/2 > n$, and thus $6i > n$; since $j \ge 5i$ it must be the case that $j = 5i$. Thus $2j = 10i > 6i > n$, a_j must be the last element of a unit. Since i is divisible by 8, $j = 5i$ is also divisible by 8; thus $a_{j/2}, a_{j/4}, a_{j/8}$ all exist in the row containing a_j, and it follows that a_j is the last element of a 4-unit. Therefore, if $a_j \in E_n^*$, a_j must be the last element of a special unit. By the definitions of the newly defined special units, j is not divisible by 5, which contradicts $j = 5i$.
7. a_i is the fourth last element of a unit, i.e., the first element of a 4-unit, $\{a_i\lrcorner, a_{2i}, a_{4i}, a_{8i}\}$. Using the same argument as in subcase 8 of Case 1 in the proof of Lemma 5.1, a_i cannot cut $a_j \in E_n^*$.

References

[1] Y. Ben-Asher, E. Farchi, I. Newman, Optimal search in trees, SIAM Journal on Computing 28 (6) (1999) 2090–2102.
[2] A. Borodin, L.J. Guibas, N.A. Lynch, A.C. Yao, Efficient searching using partial ordering, Information Processing Letters 12 (2) (1981) 71–75.
[3] R. Carmo, J. Donadelli, Y. Kohayakawa, E. Laber, Searching in random partially ordered sets, Theoretical Computer Science 321 (1) (2004) 41–57.
[4] R.L. Graham, R.M. Karp, Calif., 1968. unpublished.
[5] N. Linial, M. Saks, Searching ordered structures, Journal of Algorithms 6 (1985) 86–103.
[6] N. Linial, M. Saks, Every poset has a central element, Journal of Combinatorial Theory, Series A 40 (1985) 195–210.