Acta Informatica manuscript No.
(will be inserted by the editor)

Element Distinctness on One-Tape Turing Machines: A Complete Solution

Amir M. Ben-Amram[1], **Omer Berkman**[1], **Holger Petersen**[2*]

[1] The Academic College of Tel-Aviv-Yaffo
4 Antokolski St.
64044 Tel-Aviv, Israel
[2] University of Stuttgart
Breitwiesenstr. 20–22
D-70565 Stuttgart, Germany

The date of receipt and acceptance will be inserted by the editor

Key words. Complexity – Element Distinctness Problem – Turing machines

Abstract. We give a complete characterization of the complexity of the element distinctness problem for n elements of $m \geq \log n$ bits each on deterministic and nondeterministic one-tape Turing machines. We present an algorithm running in time $O(n^2 m(m + 2 - \log n))$ for deterministic machines and nondeterministic solutions that are of time complexity $O(nm(n + \log m))$. For elements of logarithmic size $m = O(\log n)$, on nondeterministic machines, these results close the gap between the known lower bound $\Omega(n^2 \log n)$ and the previous upper bound $O(n^2 (\log n)^{3/2} (\log \log n)^{1/2})$. Additional lower bounds are given to show that the upper bounds are optimal for all other possible relations between m and n. The upper bounds employ hashing techniques, while the lower bounds make use of the communication complexity of set disjointness.

1 Preliminaries

The computational models investigated in this paper are deterministic and nondeterministic one-tape Turing machines. Formal definitions of these devices can be found in text books, e.g., in Chapter 7

* Supported by "Deutsche Akademie der Naturforscher Leopoldina", grant number BMBF-LPD 9901/8-1 of "Bundesministerium für Bildung und Forschung".
E-mail: petersen@informatik.uni-stuttgart.de, fax: +49-711-7816-310

of [7]. An important property with respect to time complexity is that these machines are not equipped with a separate input tape. They receive their input written on the single work-tape.

The problem the machines will have to solve is the *element distinctness problem* (EDP). A list of n binary strings of equal length m, called *elements* (often treated as binary numbers) is given. The task of the Turing machine is to accept if and only if each element occurs at most once. In order to be able to consider varying relations between the number of elements and the length of their encodings, we assume that the ends of elements are marked in order to indicate the borders between them. However, the results also apply to variants of the definition where the borders are not marked but instead a fixed relation between m and n allows the machine to perform the marking fast enough. For typical relations this can be done in $O(s \log s)$ time, for input of total length s, which is low enough not to affect our considerations. Such a typical relation is $m = k \log n$ for a fixed k; this is the case that received most of the attention in previous papers [10, 15,12]. Let us briefly outline how to do the marking within $O(s \log s)$ time, when m is polynomial time computable from the binary encoding of s. The machine first counts the number of input symbols in binary by shifting a counter along the input. Using a separate track this is possible within $O(s \log s)$ steps without destroying the input. Then it computes m in time polynomial in $O(\log s)$, and hence in $O(s \log s)$ time. Finally it shifts the encoding of m and a counter along the input, marking portions of m bits each.

In the case of marked input, we assume that the marking of element borders is valid, i.e., the elements are of equal length (this can also be verified in $O(s \log s)$ time).

The set of positive instances of this problem with n elements of m bits each is denoted by $\mathrm{EDP}(n, m)$. Note that the set is empty if $m < \log n$, and contains every input with $n = 1$, so for the rest of the paper we assume $m \geq \log n$ and $n \geq 2$.

By log we mean the base 2 logarithm. For simplicity we assume that n is a power of two and hence $\log n$ is an integer. This situation can be reached by padding the input with an appropriate number of additional elements, disjoint from the original input. For nondeterministic algorithms this can be done simply by guessing the set of additional elements. For deterministic algorithms the tape alphabet is extended in such a way that before each element an additional 0 can be inserted, increasing the element size by one bit. New elements are then generated with 1 as their most significant bit.

We use the O notation for bivariate functions and write, e.g., $f(n,m) = O(g(n,m))$, with the meaning

$$\exists n_0, m_0, c > 0\, \forall n > n_0, m > m_0 : f(n,m) \leq cg(n,m)\,.$$

2 Previous Work

The complexity of EDP on one-tape Turing machines was first investigated by López-Ortiz [10]. Part of the motivation of this work is derived from the close relationship between EDP and sorting, since EDP can be solved by sorting the input and then comparing neighboring elements. Under reasonable assumptions this leads to a solution of EDP which is as efficient as sorting (up to a constant factor). For any particular model of computation a challenging question remains: Can EDP be solved more efficiently, or is there a lower bound that proves EDP to be as hard as sorting?

López-Ortiz proved a lower bound $\Omega(n^2 \log n)$ on the time required to accept $\mathrm{EDP}(n, k \log n)$ for $k \geq 2$ by a nondeterministic Turing machine, a result also mentioned in [9, Exercise 6.10.1]. The problem can be solved deterministically in $O(n^2 \log^2 n)$ time, and López-Ortiz conjectured that the lower bound could be improved to match the upper bound. Szepietowski [15] extended the lower bound to shorter numbers of size $\log n + c$ (for constant c). In addition, Szepietowski gave a deterministic solution of time complexity $O(n^2 \log n)$ for this size. The conjecture of López-Ortiz turned out to be correct for deterministic machines (see Section 3). On the other hand, in [12] a nondeterministic solution of time complexity $O(n^2 (\log n)^{3/2} (\log\log n)^{1/2})$ was presented, disproving the conjecture with respect to nondeterministic machines, and leaving a gap between the known upper and lower bounds in this setting. In Section 4 we prove that in fact the above lower bound is tight.

It is a natural generalization to consider elements of a size larger than logarithmic in n. This has already been done by López-Ortiz in Theorem 14 of [10] stating the lower bound $\Omega(\max\{n^2 m, nm^2\})$. For deterministic machines the technique presented in Section 3 yields stronger lower bounds showing EDP to be of the same complexity as sorting, while for nondeterministic machines it turns out that the above lower bound is invalid (Section 4).

There are numerous investigations of the EDP on other computational models than the one-tape Turing machines considered here. A recent example is the sub-linear upper bound by Buhrman at al. for the comparison complexity model with quantum computing [1].

3 Deterministic Algorithms

Theorem 1. *EDP(n, m) requires time $\Theta(n^2 m(m+2-\log n))$ on deterministic one-tape Turing machines.*

The proof of the theorem extends over the next two subsections.

3.1 Lower Bounds

The deterministic two-party communication complexity $D(f)$ of a two-place function f is the minimum height of a protocol-tree for the computation of $f(x, y)$ by two players Alice and Bob, where Alice receives x and Bob receives y, see [8] for more details.

The lower bound we prove in this section relies on the communication complexity of the set disjointness problem over a universe U, where $|U| = u$ for some $u \geq 2$. For two sets $S_1, S_2 \subset U$ with $|S_1| = |S_2| = s \leq u/2$ we define $\mathrm{DISJ}_s^u(S_1, S_2) = 1$ if $S_1 \cap S_2 = \emptyset$ and $\mathrm{DISJ}_s^u(S_1, S_2) = 0$ otherwise. The following lower bound is proved in [8, Example 2.12].

Fact 1

$$D(\mathrm{DISJ}_s^u) \geq \log \binom{u}{s}$$

We now apply the technique of Dietzfelbinger [3, Section 7] for deriving a lower bound on the running time of deterministic Turing machines solving the EDP. The technique involves the communication complexity of a function on an array. This is a generalization of the two-party case, where two or more processors are linked in a linear array, each one communicating with its immediate neighbors, and with the two processors at the ends taking the roles of Alice and Bob. Dietzfelbinger [3] has shown that the deterministic communication complexity of a function on an array is proportional to its two-party complexity times the number of links.

Lemma 1. *It requires $\Omega(mn \log \binom{2^m - n/2}{n/4})$ time to decide EDP(n, m) for a deterministic one-tape Turing machine.*

Proof. We reduce the problem of computing $\mathrm{DISJ}_{n/4}^{2^m - n/2}$ by a deterministic protocol on a linear array of $m(n/2) + 2$ processors to the EDP.

Of the possible 2^m numbers of length m we reserve $n/2$ distinct ones and concatenate their encodings to form a string of length $m(n/2)$ which we call a "desert." Any two $(n/4)$-element subsets S_1

and S_2 of the remaining $2^m - n/2$ elements can be encoded and placed to the left and right of the desert, forming an instance of $\mathrm{EDP}(n,m)$. Since by construction the elements in the desert are distinct and do not occur in the two sets, the membership in EDP depends entirely on the disjointness of S_1 and S_2.

Now, a deterministic Turing machine solving the problem in time $T(n,m)$ gives rise to a communication protocol of similar complexity, by letting the array simulate the computation of the Turing machine (note that passing the Turing machine's head from a cell to its neighbor amounts to transmitting a constant number of bits). Thus, the lower bound of Fact 1 and Dietzfelbinger's result provide the lower bound in the lemma. $\square$

The lower bound part of Theorem 1 is obtained from the following simplification:

$$\begin{aligned} mn \log \binom{2^m - n/2}{n/4} &\geq nm \log \left(\frac{n2^{m-\log n} - n/2}{n/4} \right)^{n/4} \\ &\geq nm(n/4) \log \left(2^{m+2-\log n} - 2 \right) \\ &= \Omega(n^2 m(m + 2 - \log n)) \,. \end{aligned}$$

3.2 Upper Bounds

For the upper bounds we first distinguish two cases, according to whether $m \geq 2 \log n$ or $m < 2 \log n$. In the former case the bound of Lemma 1 becomes $\Omega(n^2 m^2)$. A matching upper bound is given by a simple algorithm that shifts each element along the tape comparing it to all other elements.

For elements of length $m < 2 \log n$ let us write $m = \log n - 2 + g$ with $2 \leq g < \log n + 2$. The upper bound in this case is obtained as follows. The Turing machine keeps a counter of $\log n$ bits in a separate track underneath each input number. The machine initializes all counters to zero and then performs passes of incrementing all the counters. Thus there are n passes of total time $O(n^2 m)$. Whenever a counter matches the $\log n$ bit prefix of the element above it, the remaining $g - 2$ bits of the element are carried along the tape and compared to each other element with the same prefix. This takes $O(nmg)$ time, resulting in the $O(n^2 mg) = O(n^2 m(m + 2 - \log n))$ upper bound. For larger m the bound achieved by the second algorithm is $O(n^2 m^2)$, which is why the Turing machine does not have to distinguish the cases. Instead it can use the latter algorithm for the full range of possible values of m.

Note that the $O(n^2 \log n)$ upper bound of [15, Theorem 4] for elements of length $\log n + c$ (with c constant) and proved in a different way follows as a special case of the above solution.

A less direct approach of reducing EDP to sorting (sort and then compare adjacent elements), together with Wiedermann's efficient sorting algorithm [16], gives the same upper bounds.

4 Nondeterministic Algorithms

Theorem 2. *Accepting EDP(n,m) requires time $\Theta(nm(n + \log m))$ on nondeterministic one-tape Turing machines.*

Once again we break the proof of the theorem into subsections.

4.1 Upper Bounds

In the algorithms of this section we will consider a slightly generalized problem EDP(n, m, ℓ). This is the problem of deciding distinctness of n elements having $m \geq \log n$ bits each, where each element appears right-justified in a field of $\ell \geq m$ tape cells. Note that EDP$(n,m) =$ EDP(n,m,m).

The solutions make use of the hashing technique developed by Fredman, Komlós, and Szemerédi. Below we cite some results from their paper [5]. Let $U = \{1, \ldots, u\}$, $W \subseteq U$ with $|W| = r$, $s \geq r$, and assume that $p = u+1$ is prime. Denote by $a \bmod b$ the unique integer x, $1 \leq x \leq b$ such that $x \equiv a \pmod{b}$. For $k \in U$ and $1 \leq j \leq s$ let

$$B(s, W, k, j) = |\{x \mid x \in W \text{ and } (kx \bmod p) \bmod s = j\}|.$$

Corollary 1 (of [5]) *There exists $k \in U$ such that*

$$\sum_{j=1}^{r} B(r, W, k, j)^2 < 3r.$$

Corollary 2 (of [5]) *There exists a $k' \in U$, such that the mapping*

$$x \mapsto (k'x \bmod p) \bmod r^2$$

is one-to-one when restricted to W.

Corollary 4 (of [5]) *The mapping*

$$x \mapsto (k'x \bmod p) \bmod 2r^2$$

is one-to-one when restricted to W for at least one-half of the values k' in U.

In order to apply the hashing techniques, a one-tape Turing machine has to perform a variety of arithmetic operations. Bulnes has given a detailed analysis of addition and multiplication on the deterministic variant of this machine model [2]. His results show that the average case complexity of these two operations for operands of t bits each is $\Omega(t^2)$, while there is an algorithm achieving time complexity $O(t^2)$ in the worst case (here we assume that the operands are written on consecutive tape cells). For division and modulo-operation the usual "black board methods" will also result in $O(t^2)$ steps.

We first present a solution of the traditional version of the problem, where the length of the elements is logarithmic.

Lemma 2. *EDP*(n, m, ℓ) *for* $m = O(\log n)$ *can be accepted by an* $O(n^2\ell)$ *time bounded nondeterministic one-tape Turing machine.*

Proof. Fix a $c \geq 1$ such that $m \leq c \log n$. We describe the procedure carried out by a nondeterministic Turing machine M accepting the EDP from Lemma 2. First we assume in our description that the input encodes a positive instance of EDP, i.e., the elements are all distinct, and show that acceptance can be reached within the time bound. Later we will argue that carrying out the same steps for a negative instance does not lead to acceptance.

Since the hashing technique introduced above handles positive numbers, we assume in the following that a binary string of $c \log n$ zeros represents the number n^c. Let S denote the set of n numbers in the input.

First M guesses a number p with $n^c < p \leq 2n^c$, which can be represented in $1 + c \log n$ bits. The intention is that p should be prime; such a prime number exists by Bertrand's Postulate [6, Theorem 418]. It is not necessary to check whether p really is a prime. The corollaries from [5] will be applied with $U = \{1, \ldots, p-1\}$.

Next M guesses a $k < p$. Corollary 1 guarantees that k can be guessed so that $\sum_{j=1}^{n} B(n, S, k, j)^2 < 3n$. Then M carries the binary encodings of p and k on a separate track along the tape and computes $h(x) = (kx \bmod p) \bmod n$ for every input number x. It also leaves a copy of p next to each element. Shifting the numbers and carrying out the local computations can be done in time $O(\ell n \log n) = O(n^2\ell)$.

The set of numbers $S_j = \{x \mid x \in S \text{ and } h(x) = j\}$ will be called *bucket* j. Let $s_j = |S_j|$ and call $k' \in U$ *good for bucket* j if $x \mapsto (k'x \bmod p) \bmod 2s_j^2$ is one-to-one when restricted to S_j. Suppose a collection C of buckets is given, then by Corollary 4 in at least one-half of the pairs $(S_j, k') \in C \times U$ the value k' is good for bucket S_j. By averaging, at least one k' is good for one-half of the buckets.

Starting with n buckets and repeatedly selecting a value good for one-half of the remaining buckets, a set of at most $1 + \log n$ elements of U can be selected which includes a good k' for each bucket. We let M guess these $1 + \log n$ values $k'_1, \ldots, k'_{1+\log n}$ in the form of a table of space $O(\log^2 n)$. Then M fills a track below the input region of the tape with adjacent copies of this table. The number of copies is $O(n\ell / \log^2 n)$ and each copying operation can be carried out in $O(\log^4 n)$ steps, which leads to an $O(n\ell \log^2 n) = O(n^2\ell)$ bound for this stage of the computation.

Next M starts to cycle through the n buckets by counting within $\log n$ bits on a track underneath the hash values. Stepping up all counters by one is possible by one scan of the tape in $O(n\ell)$ steps. Suppose that M has in this way selected the elements of bucket j; the selected elements are those where $h(x) = j$. The machine computes the number s_j of elements in S_j and places a copy of s_j (in binary) under each such element. Next M guesses an index i_j such that k'_{i_j} is good for bucket j and shifts this value in unary notation along the tape. Whenever it passes a selected element, it leaves a copy of i_j at this element. Then M replaces each copy of i_j by the value k'_{i_j} which is retrieved from the copy of the table closest to the element. Next M hashes each element x belonging to bucket S_j using the function $h_j(x) = (k'_{i_j} x \bmod p) \bmod 2s_j^2$ that (with proper choices) is one-to-one on S_j. Finally M verifies that all elements of bucket j are distinct by shifting an array of $2s_j^2$ flags along the tape. Initially all flags are off. For each flag that, during the shifting of the array, passes the leftmost bit position of an element x of S_j, a counter with initial value $h_j(x)$ at this element is decremented. The flag that passes the element when this value first reaches zero is clearly flag number $h_j(x)$. If it is already set, M rejects. Otherwise it sets this flag. If the entire array of flags has been shifted beyond the last element without rejection, M accepts.

Running time: moving the indices i_j is possible in $O((n\ell)\sum_{j=1}^{n} i_j)$ steps. Now, the way we selected the values k'_i ensures that $i_j = 1$ for at least one-half of the j's, $i_j = 2$ for at least one-half of the rest and so on. Therefore,

$$\sum_{j=1}^{n} i_j \leq \sum_{i=1}^{1+\log n} i \frac{n}{2^i} < 2n \tag{1}$$

so the total time for this part is $O(n^2\ell)$. Copying the value k'_{i_j} from the nearest table is possible in $O(\log^3 n)$ steps, accounting for time $O(n \log^3 n) = O(n^2\ell)$. All elements are rehashed in $O(n \log^2 n + n\ell) = O(n^2\ell)$ time, since the arithmetic operations on numbers of $O(\log n)$

bits can be carried out in $O(\log^2 n)$ steps as indicated above. Computing the sizes of the buckets as well as shifting the flag array is possible in $O((n\ell)\sum_{j=1}^{n} s_j^2) = O(n^2\ell)$ steps (using the guarantee from Corollary 1).

This completes the description of the nondeterministic procedure, and the analysis of its time bound for positive instances of the EDP and correct nondeterministic choices. Acceptance is guaranteed essentially because correct choices yield perfect hash functions, and the machine is able to verify that there are no collisions within any bucket. It remains to argue that negative instances are not accepted. This is true because two identical elements will always be hashed into the same bucket S_j. The same hash function will be applied to them in the rehashing stage and the collision will be detected using the array of flags. Note also that the time analysis depended on making the right guesses, which is good enough for the framework where only the shortest accepting computation counts. However, M can be strengthened to obey the time bound on every input. Indeed, to achieve this it is only necessary for the machine to verify that two upper bounds we have computed under the assumption of a good guess are observed. To verify (1), the machine sums that values of i_j in a designated position on the tape, or rather subtracts these values from a counter initialized first to n. If the counter reaches zero, the machine aborts. The second inequality is $\sum_{j=1}^{n} s_j^2 < 3n$, which can be verified in much the same way. □

The idea of choosing good multipliers in the previous proof has been used before by Schmidt and Siegel [14] for compressing constant-time hash functions.

Combining the $\Omega(n^2 \log n)$ lower bound from [15, Theorem 2] and the upper bound of Lemma 2 for $\ell = m$ we obtain a precise characterization of the complexity of EDP on nondeterministic machines for $m = O(\log n)$:

Lemma 3. *EDP$(n, k\log n)$ (fixed $k \geq 1$) requires time $\Theta(n^2 \log n)$ on nondeterministic one-tape Turing machines.*

Next we extend the range of element sizes for which our EDP solution can be applied by adding another stage of hashing.

Lemma 4. *EDP(n, m, ℓ) for $m \geq 2\log n$ can be reduced to the problem EDP$(n, 2\log n, \ell)$ by a nondeterministic one-tape Turing machine in $O(nm\ell)$ steps.*

Proof. The Turing machine guesses a prime $p \leq 2^{m+1}$ and $k' < p$ which according to Corollary 2 allows for perfect hashing into the range n^2. Then in time $O(nm\ell)$ it shifts the encoding of p and k' along the tape and computes the hash value for every element. The EDP has now been reduced to $\mathrm{EDP}(n, 2\log n, \ell)$.

Note that, like in the proof of Lemma 2, making a "wrong" guess will never lead to acceptance. Thus the machine does not have to check that the hashing is perfect. □

Lemma 5. *EDP(n,m) for $m = O(n)$ can be accepted by an $O(n^2m)$ time bounded nondeterministic one-tape Turing machine.*

Proof. We apply Lemma 4, spending $O(nm^2) = O(n^2m)$ steps to reduce the problem into $\mathrm{EDP}(n, 2\log n, m)$. We now apply Lemma 2 leading to a solution in $O(n^2m)$ steps. □

We finally apply a well known technique in order to derive upper bounds for even larger elements:

Lemma 6. *EDP(n,m) for $m \geq n$ can be reduced to the problem EDP$(n, \lceil c\log m\rceil, m)$, where $c > 2$ is a fixed constant, by a nondeterministic one-tape Turing machine in $O(nm\log m)$ steps.*

Proof. First we claim that for an appropriate c, there exists a prime $p \leq m^c$ such that no two elements are congruent to each other modulo p, provided that the input elements are indeed all distinct. Such a prime (and of course remainders modulo this prime) can be represented in $\lceil c\log m\rceil$ bits. To justify the claim, we note that for a given pair of elements, less than $\log m$ primes can give the same remainder for both (otherwise the elements are equal, by the Chinese Remainder Theorem). Call such primes *misleading for the pair.* By the Prime Number Theorem, there is a $c > 2$ such that there are at least $m^2\log m$ primes below m^c. Omitting from this set of $m^2\log m$ numbers all misleading primes for each of the $n(n-1)/2 < m^2$ pairs of elements leaves at least one prime p to satisfy the claim. The Turing machine M carrying out the reduction guesses p and shifts a binary encoding of p along the tape computing remainders of the elements. This is possible in $O(nm\log m)$ steps. Note that it is not necessary to check that p is suitable, since a collision introduced by the choice of p will eventually lead to rejection. □

Lemma 7. *EDP(n,m) for $m \geq n$ can be accepted by an $O(nm(n + \log m))$ time bounded nondeterministic one-tape Turing machine.*

Proof. First we apply Lemma 6 to obtain $\mathrm{EDP}(n, \lceil c\log m\rceil, m)$ in $O(nm\log m)$ time. Lemma 4 reduces this to $\mathrm{EDP}(n, 2\log n, m)$ in $O(nm\log m)$ steps. Finally, this instance is solved in $O(n^2m)$ time according to Lemma 2. □

The upper-bound part of Theorem 2 can now be proved by designing a NTM that applies Lemma 5 if $m \leq n$ or Lemma 7 if $m > n$. This covers all the cases and provides the desired time bound.

4.2 Lower Bounds

In order to show that our nondeterministic solutions are optimal we will now derive matching lower bounds.

As in the case of deterministic Turing machines we will derive the lower bounds from the communication complexity of the set disjointness problem. We use the following notation:

$D(f)$ The deterministic communication complexity of f, as in Section 3.1.
$N^1(f)$ The nondeterministic communication complexity of f: here Alice and Bob can make nondeterministic choices, but an accepting communication exists if and only if $f(x,y) = 1$.
$N^0(f)$ is the same as $N^1(\overline{f})$ (the complement of f).
$N_k^1(f)$ The nondeterministic communication complexity of f on a linear array of $k+1$ processors. Neighbors are connected by a link and all bits exchanged over the k links are taken into account.

The first result required to establish the connection between the EDP and communication complexity is Theorem 2.11 of [8]:

For every function $f : X \times Y \to \{0,1\}$,

$$D(f) = O(N^0(f) \cdot N^1(f)).$$

Fact 1 (Section 3.1) gives a lower bound on the deterministic communication complexity of set disjointness, while the nondeterministic complexity $N^0(\mathrm{DISJ}_s^u)$ of verifying that two subsets $\{1,\ldots,u\}$ of cardinality s *do intersect* satisfies $N^0(\mathrm{DISJ}_s^u) \leq 1 + \log u$ (Alice transmits the index of one of her elements and Bobs acknowledges the membership in his set). Combining these bounds we obtain:

Fact 2

$$N^1(\mathrm{DISJ}_s^u) = \Omega\left(\frac{\log \binom{u}{s}}{\log u}\right).$$

In a similar way as in Exercise 12.9 of [8] and forming the "desert" like in the deterministic case of Lemma 1 we obtain the lower bound

$$\Omega(nm\left(\frac{\log \binom{2^m - n/2}{n/4}}{m} - \log mn\right))$$

on the time complexity of EDP. Notice that while this bound is strong for a large range of element sizes, it becomes useless for functions like $m = 2^n$ due to the additive term $-\log mn$. This technique has been refined by Dietzfelbinger, and (rephrased in our notation) he showed as Theorem 4.9 of [3]:

For every function $f : X \times Y \to \{0,1\}$ and $k \geq 2$,

$$N_k^1(f) \geq \lceil k/2 \rceil \cdot (N^1(f) - 2).$$

Using this stronger result we obtain:

Fact 3 *It requires*

$$\Omega(n \log \binom{2^m - n/2}{n/4})$$

time to accept EDP(n,m) by a nondeterministic one-tape Turing machine.

For very large m this lower bound still does not match our upper bound $O(nm \log m)$, e.g., $m = 2^{2^n}$. We show a lower bound for this range by a problem-specific pumping argument. This involves the concept of a *crossing sequence*, which is the sequence of states a Turing machine enters when crossing the boundary between two neighboring tape cells. Given that initially M's head reads the leftmost symbol of the input and considering only crossing sequences properly within the input area, the direction of the first crossing of each cell boundary is fixed and hence the sequence of states fully describes the history of the crossings.

Lemma 8. *Accepting EDP(n,m) requires time $\Omega(nm \log m)$ on a nondeterministic one-tape Turing machine.*

Proof. Let EDP(n,m) be accepted by a Turing machine M with q states and denote the size of M's input by $s = nm$. For the sake of contradiction assume that M accepts the EDP in

$$t(s) < \frac{s}{2}\left(\frac{\log s - 1}{\log q} - 1\right)$$

steps. We obtain

$$\left(\frac{2t(s)}{s} + 1\right) \log q < \log s - 1$$

and

$$q^{\left(\frac{2t(s)}{s}+1\right)} < \frac{s}{2}.$$

Observe that at least $s/2$ crossing sequences of M properly within the input area have length at most $k = 2t(s)/s$. The number of different crossing sequences of such length is bounded above by

$$\sum_{i=0}^{k} q^i \leq q^{k+1}.$$

Since $q^{k+1} < s/2$, one of these crossing sequences must be repeated. Thus we can split the input into non-empty portions uvw such that the crossing sequences between u, v and between v, w are identical. Notice that this implies that M does not accept while its head is within v. We are now able to obtain an accepting computation on any input of the form uv^jw for $j \geq 0$ by copying segments of M's computation along with the repetitions of v. Recall that only inputs are accepted that consist of n elements of equal length whose end points are marked. Replacing uvw with uv^2w will create an invalid input, unless v includes at least one mark and repeating it creates an element of the correct length between the rightmost mark of the first copy and the leftmost mark of the second. If this is the case, consider uv^3w: this is now a valid input but contains repeated elements. In any case, we conclude that M accepts incorrectly. Since this is a contradiction to our assumption about M, we conclude that M needs at least

$$\frac{s}{2}\left(\frac{\log s - 1}{\log 2q} - 1\right) = \Omega(nm \log m)$$

steps to accept the EDP. $\square$

The lower-bound part of Theorem 2 now follows by combining three results, depending on the relationship of m to n. If $\log m > n$, Lemma 8 provides the lower bound. If $2 \log n < m \leq 2^n$, we use Fact 3, noting (as in Section 3.1) that

$$n \log \binom{2^m - n/2}{n/4} = \Omega(n^2(m + 2 - \log n)) = \Omega(n^2 m)\,.$$

Finally if $m \leq 2 \log n$, the lower bound $\Omega(n^2 \log n)$ is provided by Szepietowski [15].

5 Discussion

We presented algorithms solving the EDP in asymptotically optimal time on deterministic and nondeterministic one-tape Turing machines. The lower bounds only rely on the flow of information on these computational models and thus the upper bounds show that within their inherent limitations, one-tape Turing machines are quite efficient when solving variants of the EDP.

We remark that the bound of Theorem 2 corresponds to the product of input length and the size of perfect hash functions for linear table sizes [11]. Indeed, the size of perfect hash functions can be directly related to communication complexity: the upper bound $N^1(\mathrm{DISJ}^u_s) = O(s + \log\log u)$ (in our notation $s = n$ and $u = 2^m$) of [13, Theorem 2.8] and [8, Example 2.12] can be obtained by having Alice transmit a perfect hash function for the elements along with a bit vector encoding the hash values of her subset. This however ignores efficiency considerations concerning the hashing algorithm, which become important when the approach of using hash functions is applied to concrete computational models.

References

1. H. Buhrman, C. Dürr, M. Heiligman, P. Høyer, F. Magniez, M. Santha, and R. de Wolf. Quantum algorithms for element distinctness. In *Proceedings of the 16th Annual Conference on Computational Complexity, DePaul University, Chicago IL*, 2001. To appear.
2. J. Bulnes. On the speed of addition and multiplication on one-tape, off-line Turing machines. *Information and Control*, 20:415–431, 1972.
3. M. Dietzfelbinger. The linear-array problem in communication complexity resolved. Technical Report TR96-063, ECCC: Electronic Colloquium on Computational Complexity, Reports Series, 1996. Available via: `http://www.eccc.uni-trier.de/eccc/`, see also [4].
4. M. Dietzfelbinger. The linear-array problem in communication complexity resolved. In *Proceedings of the 29th ACM Symposium on Theory of Computing (STOC), El Paso, Texas*, pages 373–382, 1997.
5. M. L. Fredman, J. Komlós, and E. Szemerédi. Storing a sparse table with $O(1)$ worst case access time. *Journal of the Association for Computing Machinery*, 31:538–544, 1984.
6. G. H. Hardy and E. M. Wright. *An Introduction to the Theory of Numbers.* Clarendon Press, Oxford, 5th edition, 1979.
7. J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, Reading, Mass., 1979.
8. E. Kushilevitz and N. Nisan. *Communication Complexity.* Cambridge University Press, 1997.
9. M. Li and P. Vitányi. *An Introduction to Kolmogorov Complexity and its Applications.* Graduate Texts in Computer Science. Springer, second edition, 1997.
10. A. López-Ortiz. New lower bounds for element distinctness on a one-tape Turing machine. *Information Processing Letters*, 51:311–314, 1994.
11. K. Mehlhorn. On the program size of perfect and universal hash functions. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science (FOCS), Chicago, IL*, pages 170–175, 1982.

12. H. Petersen. Bounds for the element distinctness problem on one-tape Turing machines. 2001. To appear in Information Processing Letters.
13. A. A. Razborov. Applications of matrix methods to the theory of lower bounds in computational complexity. *Combinatorica*, 10:81–93, 1990.
14. J. P. Schmidt and A. Siegel. The spatial complexity of oblivious k-probe hash functions. *SIAM Journal on Computing*, 19:775–786, 1990.
15. A. Szepietowski. The element distinctness problem on one-tape Turing machines. *Information Processing Letters*, 59:203–206, 1996.
16. J. Wiedermann. Optimal algorithms for sorting on single-tape Turing machines. In J. van Leeuwen, editor, *Algorithms, Software, Architecture, Proceedings of the IFIP 12th World Computer Congress, Madrid, Spain*, volume I, pages 306–314. Elsevier Science Publishers, 1992.