Theoretical Computer Science 36 (1985) 231-237
North-Holland

IMPROVED LOWER BOUNDS FOR THE CYCLE DETECTION PROBLEM

Eric ALLENDER*
School of Information and Computer Science, Georgia Institute of Technology, Atlanta, GA 30332, U.S.A.

Maria M. KLAWE
Computer Science Department, IBM Research Laboratory, San Jose, CA 95193, U.S.A.

Communicated by M.S. Paterson
Received September 1983
Revised June 1984

Abstract. Lower bounds for the 'cycle detection problem' were recently investigated by Fich (1981, 1983). She showed that Floyd's algorithm was optimal among those algorithms which have $M=2$ memory locations and which make a finite number of 'jumps'. A lower bound for the case where $M>2$ was also presented, but the question of whether having more than two memory locations could actually yield a better algorithm was left open. In this report, we show that it cannot.

A lower bound was also presented by Fich (1981, 1983) for algorithms which have two memory locations and which make a finite number of 'back advances'. We show here that the same lower bound holds even if the restriction on back advances is dropped.

1. Introduction

Let $\mathbb{N}$ be the set of nonnegative integers; let D be any set, let f be an arbitrary function from D to D, and let $x \in D$ be given. The cycle detection problem is to find $i, j \in \mathbb{N}$, $i<j$, such that $f^i(x)=f^j(x)$, if such a pair exists.

We assume that no inferences can be made about the behavior of f, except that when given an element of D as input, it produces an element of D as output. We assume that any algorithm to solve the cycle detection problem is able to store representations of elements of D in memory locations, that it can compare two memory locations for equality, and for any two memory locations x and y, it can perform $x \leftarrow f(x)$ and $y \leftarrow x$. We restrict our attention to algorithms with a finite number M of memory locations.

Note that if D is infinite, the sequence $x, f(x), f(f(x)), f^3(x), \ldots$ may be cycle-free. In this case, a cycle-detection algorithm will not terminate. (It is clearly impossible to determine that a sequence does not have a cycle.)

* Research has been supported by the National Science Foundation under Grant No. MCS 81-03608.

Let l and t denote the *least* integers such that $l < t$ and $f^l(x) = f^t(x)$. Then note that

(1) we do *not* require that a cycle detection algorithm find l and t,

(2) $f^{l+m}(x) = f^{t+m}(x)$ for all $m \in \mathbb{N}$,

(3) $f^l(x) = f^{t+k(t-l)}(x)$ for all $k \in \mathbb{N}$.

Results about the cycle detection problem often seem easier to present when the problem is cast in a different setting. Consider the following situation. You are given a number line for $\mathbb{N}$, and you have M markers, all of which are initially on zero. (We follow convention and assume that zero is the *leftmost* number on the number line.) Someone has selected two integers l and t (you do not know what l and t are), and your object is to move your markers into an (l, t) *stopping configuration*, which is defined to be any configuration in which you have markers on the numbers i and j, where $l \leqslant i < j$ and $t - l$ divides $j - i$.

You are allowed to make the following moves:

(1) Move a marker forward one position.

(2) Pick up a marker and put it down on top of another marker.

Moves of type 1 are called *advances*. Advances from the 'front' (i.e., rightmost) pile of markers are called *front advances*. All other advances are *back advances*. Moves of type 2 are called *jumps*.

Continuing to follow Fich [3, 4], we equate the running time t' of an algorithm with the number of function evaluations performed by the algorithm. That is, transferring values from one memory location to another, and testing values for equality, are 'free'. (Equivalently, we charge for all advances, but all jumps are free.) As in [3, 4], we restrict our attention to those algorithms whose behavior depends only on l and t. Thus we denote the running time by $t'(l, t)$. We will often write t' to mean $t'(l, t)$.

Clearly, $t'(l, t) \geqslant t$. It is thus reasonable to measure the complexity of an algorithm by

$$\sup\{t'(l, t)/t : l, t \in \mathbb{N}\}.$$

Fich analyzed the cycle detection problem using this measure of complexity. Her results are summarized in Table 1.

The first result in this paper deals with cycle detection algorithms which make only a finite number of jumps. Among all the cycle detection algorithms which have appeared in the literature, only Floyd's algorithm (one of the earliest, see [5, p. 7] and [2, 6, 7]) falls into this class. All of the improvements on Floyd's algorithm presented in [1–4, 6, 7] involve making use of jumps. Fich showed that that had to be the case if one was restricted to using $M = 2$ memory locations. We show that it is true in general. That is, Floyd's algorithm is optimal among all algorithms making a finite number of jumps—having more than two memory locations does not help.

Another problem left open by Fich is the question of whether or not back advances can be used to yield a better algorithm, if jumps are allowed. Although we are

Table 1

Class of algorithms	Lower bound	Upper bound	
Arbitrary, M memory locations	$1+1/(M-1)$	$1+2/(M-1)$	(using no back advances)
Finite number of jumps, M memory locations	$1+\sqrt{2}$	3	(using no jumps, $M=2$)
Finite number of jumps, 2 memory locations	3	3	(using no jumps)
Finite number of back advances, 2 memory locations	$\frac{1}{2}(3+\sqrt{5})$	$\frac{1}{2}(3+\sqrt{5})$	(using no back advances)

unable to answer this question in general, we are able to show in our second theorem that back advances do not help in the case where only two memory locations are used. Thus the algorithms presented in [3, 4] for solving the cycle detection problem using only two memory locations are optimal, using this measure of complexity.

2. Improved lower bounds

Theorem 1. *For any cycle detection algorithm which uses at most M locations and which performs at most a fixed finite number of jumps,*

$$\sup\{t'(l,t)/t\colon l, t\in\mathbb{N}\}\geqslant 3.$$

Proof. Assume we are given an algorithm A with M markers. It has been proved in [3] that the theorem is true if $M=2$. Thus we will assume below that $M>2$.

Let $0\leqslant a_1(i)\leqslant a_2(i)\leqslant\cdots\leqslant a_M(i)$ denote the positions of the M markers immediately after step i in the algorithm. (A 'step' is either a jump or an advance.) Let the last jump be made at time τ and let ϕ be the number of advances performed during the first τ steps. Then we will let

$$\alpha=a_1(\tau)+a_2(\tau)+\cdots+a_M(\tau)-\phi.$$

Note that $a_1(\tau)+\cdots+a_M(\tau)$ is the number of advances needed to achieve the configuration at time τ if no jumps are used. Thus α represents the number of advances 'saved' by performing jumps.

We will also need the constant γ. If time τ is defined as above, then

$$\gamma=\max\{a_M(j):j\leqslant\tau\}.$$

Constant γ is the rightmost position visited by any marker prior to the last jump. In particular, no marker may ever move from the right of γ to the left of γ, since the only way to move to the left is to make a jump.

Case I: There exists some integer k such that, for all times i, $a_M(i)-a_{M-1}(i)<k$.

Let $l \in \mathbb{N}$ and let $t = l + k$. Assume that an (l, t) stopping configuration occurs at time i. Since $a_M(i) - a_{M-1}(i) < t - l$ (and thus the two rightmost markers are not far enough apart to detect the cycle), we must have $a_{M-2}(i) \geqslant l = t - k$. Thus

$$\begin{aligned} t'(l, t) &\geqslant a_M(i) + a_{M-1}(i) + a_{M-2}(i) - \alpha \\ &\geqslant 3(a_{M-2}(i)) - \alpha \\ &\geqslant 3(t-k) - \alpha, \end{aligned}$$

and hence

$$\begin{aligned} \sup t'/t &= \sup\{t'(l, t)/t \colon l, t \in \mathbb{N}\} \\ &\geqslant \sup\{3 - (3k + \alpha)/(l + k) \colon l \in \mathbb{N}\} \\ &= 3. \end{aligned}$$

Case II: The set $\{a_M(i) - a_{M-1}(i) \colon i \in \mathbb{N}\}$ is infinite.

In this case, if $i_n = \min\{i \colon a_M(i) - a_{M-1}(i) \geqslant n\}$, then i_n is defined for all $n \in \mathbb{N}$. Now let $X = \{i_n \colon n > \gamma \text{ and } a_{M-1}(i_n) > \gamma\}$. Since the set $\{a_{M-1}(i) \colon i \in \mathbb{N}\}$ must be infinite in order to detect cycles when l is arbitrarily large, and since $a_{M-1}(i) > \gamma$ implies that the algorithm has finished performing jumps and hence that $a_{M-1}(i + 1) \geqslant a_{M-1}(i)$, it follows easily that

(1) X is infinite,

(2) $\{a_{M-1}(i) \colon i \in X\}$ is infinite,

(3) $\{a_M(i) - a_{M-1}(i) \colon i \in Y\}$ is infinite, where Y is any infinite subset of X, and

(4) if $i \in X$, then, for all $j < i$, $a_M(j) - a_{M-1}(j) < a_M(i) - a_{M-1}(i)$.

Case II.1: For infinitely many $i \in X$, $a_{M-1}(i) > \frac{1}{2}(a_{M-2}(i) + a_M(i))$ (see Fig. 1).

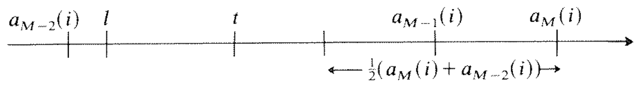

Fig. 1. Case II.1.

Let $i \in X$, where $a_{M-2}(i) + a_M(i) < 2a_{M-1}(i)$. Let $l = a_{M-2}(i) + 1$ and $t = l + a_M(i) - a_{M-1}(i) + 1 = a_{M-2}(i) + a_M(i) - a_{M-1}(i) + 2 < a_{M-1}(i) + 2$.

Note that since $t > a_M(i) - a_{M-1}(i) > \gamma$, no (l, t) stopping configuration can be reached until after the last jump has been made. Moreover, since only locations $a_M(i)$ and $a_{M-1}(i)$ are to the right of l, and (by point (4) above) the front two markers have never been far enough apart to detect a cycle of length $t - l$, no (l, t) stopping configuration has been reached by time i. Now

$$\begin{aligned} t'(l, t) &\geqslant a_M(i) + a_{M-1}(i) + a_{M-2}(i) - \alpha \\ &= a_M(i) + a_{M-1}(i) + (t - 2 + a_{M-1}(i) - a_M(i)) - \alpha \\ &= 2(a_{M-1}(i)) + (t-2) - \alpha \\ &> 3(t-2) - \alpha \end{aligned}$$

and hence

$$\frac{t'}{t} > 3 - \frac{6+\alpha}{t} > 3 - \frac{6+\alpha}{a_M(i) - a_{M-1}(i)}.$$

Now it follows by point (3) above that

$$\sup t'/t \geqslant 3.$$

Case II.2: For all large $i \in X$, $\frac{1}{2}(a_M(i) + a_{M-2}(i)) \geqslant a_{M-1}(i)$ (see Fig. 2).

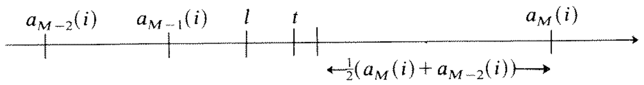

Fig. 2. Case II.2.

Let $i \in X$, where $a_M(i) + a_{M-2}(i) \geqslant 2a_{M-1}(i)$. Let $l = a_{M-1}(i) + 1$ and $t = l + 1$. Since there is only one marker to the right of $l > \gamma$, the algorithm cannot have reached an (l, t) stopping configuration yet. Thus we have that

$$\begin{aligned} t'(l, t) &\geqslant a_M(i) + a_{M-1}(i) + a_{M-2}(i) - \alpha \\ &\geqslant 3(a_{M-1}(i)) - \alpha \\ &= 3(t-2) - \alpha \end{aligned}$$

and hence

$$\sup \frac{t'}{t} \geqslant \sup\left\{3 - \frac{6+\alpha}{a_{M-1}(i) + 2} : i \in X\right\} = 3. \qquad \square$$

Theorem 2. *For any cycle detection algorithm which uses only two memory locations,*

$$\sup\{t'(l, t)/t : l, t \in \mathbb{N}\} \geqslant \tfrac{1}{2}(3 + \sqrt{5}).$$

Proof. Assume we are given an algorithm A with two markers. As in Theorem 1, let $a_1(i) \leqslant a_2(i)$ denote the positions of the two markers at step i in the algorithm. Since $\frac{1}{2}(3+\sqrt{5}) < 3$, we can assume by Theorem 1 that A makes an infinite number of jumps. For each $n \in \mathbb{N}$, let i_n be a step in the algorithm which occurs between the nth and the $(n+1)$st jump, such that $a_2(i_n) - a_1(i_n)$ is maximal and i_n is as small as possible. That is, if the nth jump occurs at time j, and the $(n+1)$st jump occurs at time k, then

$$i_n = \min\{i : j \leqslant i \leqslant k, \text{ and if } j \leqslant h \leqslant k, \text{ then } a_2(h) - a_1(h) \leqslant a_2(i) - a_1(i)\}.$$

As an aid to clarity, let $x_n = a_1(i_n)$ and $y_n = a_2(i_n)$. In the following, we will make use of the set X defined as

$$X = \{n\colon y_n \geqslant (\tfrac{1}{2}(3+\sqrt{5}))x_n\}.$$

Case I: The set X is infinite.

Let $n \in X$. Let $l = x_n + 1$ and $t = x_n + 2$. Note that at time i_n the leftmost marker has never been to the right of x_n, since it is impossible for the leftmost marker to move to the left. Thus an (l, t) stopping configuration is not reached before time $i_n + 1$, and

$$t'(l, t)/t > y_n/t \geqslant \tfrac{1}{2}(3+\sqrt{5})(t-2)/t.$$

Now as x_n and hence t can be chosen arbitrarily large it is clear that $\sup(t'/t) \geqslant \frac{1}{2}(3+\sqrt{5})$.

Case II: The set X is finite.

In this case, there is some K such that $y_n < \frac{1}{2}x_n(3+\sqrt{5})$ for all $n \geqslant K$. Let $Y = \{n \geqslant K\colon a_2(h) - a_1(h) < a_2(i_n) - a_1(i_n)$ for all $h < i_n\}$. Note that if $n \in Y$, then i_n is the first time that the distance between the markers has been as great as $y_n - x_n$. Since the set $\{a_2(i) - a_1(i)\colon i \in \mathbb{N}\}$ is infinite, it follows that Y is infinite.

Let $n \in Y$, so $y_n < \frac{1}{2}x_n(3+\sqrt{5})$. Let $l = 0$ and $t = y_n - x_n + 1$. Since the distance between the markers is never more than $y_n - x_n$ between jumps n and $n+1$, there is no chance of reaching an (l, t) stopping configuration until after the $(n+1)$st jump. At least t moves will be necessary after the $(n+1)$st jump, in order to get the markers far enough apart to detect the cycle. Thus,

$$\frac{t'}{t} \geqslant \frac{t+y_n}{t} = 1 + \frac{1}{t/y_n} = 1 + \frac{1}{1-(x_n-1)/y_n}$$

$$> 1 + \frac{1}{1-[2(x_n-1)]/[x_n(3+\sqrt{5})]} = 1 + \frac{1}{1-2/(3+\sqrt{5})+2/[x_n(3+\sqrt{5})]}.$$

Now it follows that

$$\sup \frac{t'}{t} \geqslant \sup\left\{1 + \frac{1}{1-2/(3+\sqrt{5})+2/[x_n(3+\sqrt{5})]}\colon n \in Y\right\}$$

$$= 1 + \frac{1}{1-2/(3+\sqrt{5})} = \tfrac{1}{2}(3+\sqrt{5}). \qquad \square$$

3. Conclusions and open problems

Our knowledge about the cycle-detection problem using this complexity measure is summed up in Table 2.

Table 2

Class of algorithms	Lower bound	Upper bound	
Arbitrary, M memory locations	$1+1/(M-1)$	$1+2/(M-1)$	(using no back advances)
Arbitrary, 2 memory locations	$\frac{1}{2}(3+\sqrt{5})$	$\frac{1}{2}(3+\sqrt{5})$	(using no back advances)
Finite number of jumps, M memory locations	3	3	(using no jumps, $M=2$)

Obvious open problems remaining are the following:

(1) Can back advances be used to obtain a better algorithm if $M>2$ memory locations are used?

(2) Can the gap between the upper and lower bounds in the general case be closed?

Acknowledgment

The first author would like to thank Kim King for bringing this problem to his attention and for offering many helpful comments, which contributed to making this presentation clearer. Thanks are also due to Faith Fich, for putting the authors in touch with each other.

References

[1] M. Beeler, R. Gosper and R.W. Schroeppel, Hakmem, M.I.T. Artificial Intelligence Lab. Memo No. **239** (item 132) (1972) 64.

[2] R.P. Brent, An improved Monte Carlo factorization algorithm, *BIT* **20** (1980) 176–184.

[3] F.E. Fich, Lower bounds for the cycle detection problem, in: *Proc. 13th Annual ACM Symposium on Theory of Computing*, Milwaukee, WI (1981) 96–105.

[4] F.E. Fich, Lower bounds for the cycle detection problem, *Journal of Computer and System Sciences* **26** (1983) 392–409.

[5] D.E. Knuth, *Seminumerical Algorithms, The Art of Computer Programming, Vol. 2* (Addison-Wesley, Reading, MA, 1969).

[6] R. Sedgewick and T.G. Szymanski, The complexity of finding periods, in: *Proc. 11th Annual ACM Symposium on Theory of Computing*, Atlanta, GA (1979) 74–80.

[7] R. Sedgewick, T.G. Szymanski and A.C. Yao, The complexity of finding cycles in periodic functions, *SIAM Journal on Computing* **11** (1982) 376–390.