MACHINE MEMORY REQUIREMENTS - LOWER BOUNDS

by: Marvin C. Paul

January, 1972

DCS Technical Report #15

Department of Computer Science
Livingston College
Rutgers University
Machine Memory Requirements - Lower Bounds

I

The purpose of this paper is to consider how memory requirements grow with the length of the input sequence recognized by a machine. In particular, we develop lower bounds on such growth.

Such bounds have been developed for a particular form of Turing machine,[1]. In this form of machine, the input is permanently stored on an input tape which may be read in either direction, but the memory used for storing the input is not included in the bound. We wish to consider a bound on all the memory used in a machine, considering the input as coming from outside the machine. The machine may or may not store individual input symbols. We define a machine in terms of states only, without specifying the nature of the form of memory in which these states are recorded. A machine, as defined here, may have an infinite number of states.

A machine specification M will consist of: a set of input symbols, I; a set of states, S (generally infinite); a subset R of S of 'read' states such that if M is in a member of R, M will read the next input; a 'read-next-state' function which for each state in R and each input in I, specifies a member of S as next state; a 'non-read-next-state' function which for each state in S but not in R specifies a member of S as next state; a member of S called the starting state; a subset of S of terminal states.

An input sequence i takes M from state a to state b if starting in a and applying the next state functions to the states and inputs of i in the expected way (similar to the way it is done with finite state machines) M will, after the last input of i is read, be in state b. A sequence is recognized by M if it will take M from starting state s to a terminal state of M, in a way defined analogously to that for finite state machines.

As when considering finite state machines, we can use state diagrams in representing aspects of these generally-infinite-state machines. We develop this state diagram or graph form of
representation now in greater detail in preparation for the use we intend to make of it in developing the memory bound. In fact, the memory bound will be shown to be equivalent to a bound on the number of branches in certain graphs.

Consider an input sequence, \(i\), of length \(k\) recognized by a machine \(M\) having \(l\) input symbols in such a way that \(M\) passes through \(n\) states in the process. The way in which this occurs can be represented by an appropriate state diagram, i.e., by a directed graph \(U\) having:

1. \(n\) nodes, one designated the starting node \((s)\), one the terminal node \((t)\)

2. at least \(k\) branches, with \(k\) of the branches labelled with input symbols (those branches from 'read' states) and the remaining branches unlabelled (those branches from 'non-read states').

3. Each node has either a single unlabelled branch (a non-read-state) or no more than \(l\) labelled branches (read state) emerging from it.

Furthermore, \(U\) must have a path \(p\) from \(s\) to \(t\) traversing each and every branch in \(U\) once, such that writing the labels on the labelled branches in the order they occur in the path \(p\) yields the input sequence \(i\).

Such a directed graph with a single 'complete' path running from the starting to the terminating node and covering every branch in the graph is a 'unicursal' [2] graph.

As described above, the state configuration through which a machine \(M\) with \(l\) input symbols passes in recognizing an input sequence of length \(k\) can be characterized by a unicursal graph.
Similarly, for any unicursal graph with \( n \) nodes and with \( k \) of its branches labelled and with no more than \( k \) branches from each node, we can construct a machine \( M \) which will accept an input sequence \( i \) of length \( k \) in no more than \( n \) input symbols and pass through \( n \) states in the order indicated by the graph.

In describing the bound we are seeking, we will use the following definitions.

If in recognizing a sequence \( i \), machine \( M \) passes through \( n \) states, we will say \( M \) recognized \( i \) in \( n \) states.

If \( M \) recognizes \( i \) in \( n \) states and \( i \) is the shortest sequence \( M \) recognizes in \( n \) states, then \( i \) is a standard sequence for \( n \) states in machine \( M \).

If for each number of input symbols \( \ell \), we consider all machines \( M \) to determine the smallest number of states used by any of them in recognizing a standard sequence of length \( k \), we would have obtained the bound we seek. It is in general a function of \( \ell \) and \( k \).

Our objective then is to find the function which is represented in figure 1.

---

**Figure 1**

- **n** = number of states of a machine
- **k** = length of standard sequences
- **Allowed Region**
- **Disallowed Region**
- Minimum # of states required by any machine in recognizing a standard sequence of given length
In figure 1, the boundary curve is shown to be monotonic non-decreasing. That it will actually be so, is shown by the following simple argument. If a standard sequence \( i \) of length \( k \) requires \( n \) states in some machine \( M \) then certainly there is a standard sequence \( i' \) of length \( k-1 \) in some machine \( M' \) which requires no more than \( n \) states. The sequence \( i' \) can always be constructed from \( i \) by removing the first input symbol of \( i \). If from the state diagram representing the recognition of \( i \) in \( M \), we remove the branch from the starting state labelled with the first symbol of \( i \) and let the state to which this branch goes be the starting state of \( M' \), we have a state diagram which can be used to represent the recognition of \( i' \) by \( M' \). \( i' \) thus recognized will be a standard sequence and will certainly not require more than \( n \) states. Now consider the inverse of the non-decreasing function given in figure 1. This may be obtained by a rotation of 90° counterclockwise and viewing the result from the back of the page. Figure 2 results.

![Diagram](attachment:Figure_2.png)

- boundary curve = max length standard sequence recognizable using \( n \) states

\( k = \text{length of standard sequence} \)

\( n = \text{number of states of a machine} \)

**Figure 2**
In figure 2 it is natural to view the boundary curve as representing the minimum length standard sequence recognizable in \( n \) states. Because its inverse was monotonic non-decreasing this curve is itself monotonic increasing. If we can obtain this 'minimum' boundary curve, then its inverse will give us the boundary curve of figure 1 (at least at every value of \( n \)) - which is our basic objective.

To find the boundary curve of figure 2 we phrase the problem in graph terms using the graph characterization we have developed for the state diagram for a sequence accepted by a machine:

1. To each sequence accepted by a machine with \( n \) states, there corresponds an \( n \)-node unicursal graph.

2. If a sequence \( i \) is a standard sequence, then the corresponding unicursal graph of \( n \) nodes, must be such that removal of any branch can not leave a unicursal graph on the \( n \) nodes; for then a shorter sequence than \( i \) would also require \( n \) nodes and \( i \) would not be standard. A unicursal graph with the above property is a minimal-unicursal graph. (For any \( n \) node minimal unicursal graph \( U \), a machine can always be constructed which recognizes a standard input sequence of length equal to the number of branches in \( U \).)

The curve we seek is characterized in graph terms as follows:

- find the number of branches \( K \) (=the complete path length) for the minimal unicursal graph with the maximum number of branches over all such graphs with \( n \) nodes and no more than 1 branches from each node.

As is shown in section II of this paper that that number is:

\[
K = \begin{cases} 
\left\lfloor \left( \frac{n+1}{2} \right)^2 - 1 \right\rfloor & \text{for } n = 1 \text{ to } 2^k-1 \\
\lfloor (n-k+1) \rfloor - 1 & \text{for } n > 2^k-1
\end{cases}
\]
In machine terms, $K$ may be interpreted as the maximum length standard sequence of figure 2; $n$ the number of states used in recognizing this standard sequence and $l$ the number of input symbols allowed.

The formula for figure 1, which is the inverse of that for figure 2, is given by:

$$n = \begin{cases} \frac{2\sqrt{K+1} - 1}{2} & \text{for } n = 1 \text{ to } 2l-1 \\ \left\lfloor \frac{1+K}{2} + (l-1 + \frac{1}{2}) \right\rfloor & \text{for } n > 2l-1 \end{cases}$$

with $\left\lfloor x \right\rfloor$ read as the integer $y$ such that $y \leq x < y+1$. It is plotted on figure 3.

It may be thought at first that a finite state machine would have a curve which would be represented by a horizontal line infinite to the right on axes of figure 3, and therefore that a finite state machine violates the bound given there. This is not the case because a finite state machine will not have an infinite number of standard input sequences, i.e., there is a bound on the longest possible standard sequence as a function of the number of states for finite state machines. In fact an alternate interpretation for the curve of figure 3 (or rather its inverse) is that it gives the maximum length standard sequence possible for a finite state machine of $n$ states.

For each number of states one can construct a machine which will meet the bounds of figure 3. We may further consider whether there are any infinite state machines which meet the bounds of figure 3 for all numbers of states $n$. There is such a machine with two input symbols. The infinite state machine in figure 4 meets the bound at every node.
For each \( n \) there is a standard sequence which uses the \( 2n-3 \) leftmost branches in recognizing \( 1^n \) \( \varepsilon \) \( 0^n \).

This machine will accept any sequence of 1's and 0's which has one more 0 than 1 and in which each prefix has no more 0's than 1's.

Figure 4
II

Here we develop a theorem of graph theory which has been shown in section I to be equivalent to the development of a memory bound for the machine described there. After some preliminary definitions and conventions, we will state and prove the theorem.

We will only consider directed finite graphs. 'Graph' is to be read as 'directed finite graph'. Subsequently,

A graph $G$ consists of a finite set of nodes and branches. Each branch in $G$ is associated with two nodes in $G$; going 'out' from one node and 'in' to the other node.

If $a$ is a node of $G$, then $\text{in}(a)$ is the number of branches into $a$ in $G$, and $\text{out}(a)$ is the number out of $a$ in $G$.

If $\text{out}(a) \leq 1$ for every node $a$ in $G$ then we say the fanout of $G$ is $1$.

A path $w$ in a graph $G$ is a sequence of nodes of $G$; $w = a_1a_2 \ldots a_m$ such that there is a branch in $G$ from $a_i$ to $a_{i+1}$ for each $1 \leq 1$ to $m-1$.

A graph $G$ is connected if for every pair of nodes $a, b$ in $G$ there is a path between $a$ and $b$, i.e., starting with $a$ and ending with $b$.

A complete path in $G$ is a path $w = a_1 \ldots a_m$ in $G$ such that each branch in $G$ goes from $a_i$ to $a_{i+1}$ for some $1 \leq 1$.

A graph $G$ is unicursal iff it has a complete path.

A graph $G$ is minimal unicursal if it is unicursal and removal of any set of branches leaves it no longer unicursal.

We will need the following well known [?] characterization of unicursal graphs.
Theorem R1: A graph G is unicursal iff it is connected and either

I: its nodes can be partitioned into a set S consisting of
   one node s, a set T consisting of one node t, and a set
   R consisting of all other nodes in G such that:
   (1) out(s) - in(s) = 1
   (2) in(t) - out(t) = 1
   (3) in(r) - out(r) = 1

or II: for every node a in G in(a) - out(a) = 0.

Convention: Small letters are nodes; small greek letters are paths.

Theorem: A minimal unicursal graph of \( n \geq 3 \) nodes and ranout \( \ell \geq 2 \)
has no more than \( \text{MIN} \left( \frac{(n+1)^2}{2} - 1, \ell(n-\ell+1) - 1 \right) \)
branches and there exists a minimal unicursal graph with
these numbers of branches.

First we develop some simple necessary conditions that the
graph be minimal unicursal.

Lemma 1: Consider a complete path \( w \) through any minimal unicursal
graph \( U \). Let's represent a node and let \( \alpha, \beta, \gamma \) each
represent sequences of nodes. If the path \( \omega \) can be represented
as \( \alpha_s \beta \gamma \) then \( \beta \) must contain a node which is not
contained in \( \alpha \) nor in \( \gamma \).

Clearly if \( \alpha \beta \gamma \) is a complete node path through \( U \) then
\( \omega = \alpha \beta \gamma \) is also a path through \( U \) and if \( \beta \) does not contain a
node which is absent from \( \alpha \) and \( \gamma \) then \( \omega \) would be a complete
path. Since \( \omega \) would be smaller than \( \omega \), \( U \) could be constructed
from \( U \) by removing from \( U \) all branches in \( \omega \) and not in \( \omega' \) (of
which there is at least one). Contrary to the hypothesis, \( U \)
could not have been a minimal unicursal graph of n nodes.

Lemma 2: Again consider a complete path \( w \) through the minimal unicursal graph \( U \). If the path \( w \) can be represented as \( \alpha s \beta s \gamma \) then there is a node in \( \beta \) which appears only once in \( w \).

By lemma 1 there is a node, say \( b_1 \), in \( \beta \) which appears neither in \( \alpha s \) nor in \( s \gamma \). If \( b_1 \) appears once in \( \beta \), it is the node we seek since it will appear in \( w \) only once. If it appears more than once in \( \beta \), then \( \delta = \alpha s \beta_1 b_1 \beta_2 b_1 \beta_3 s \gamma \) and again by lemma 1 we see that \( \beta_2 \) - a proper substring of \( \beta \) - a node must appear which is neither in \( \alpha s \beta_1 b_1 \) nor in \( b_1 \beta_3 s \gamma \). By repeating the consideration for \( \beta_2 \) in an analogous manner to that for \( \beta \) we can again find a smaller substring in which a node which appears nowhere else in \( \delta \) must appear. This process can be imbedded in an inductive argument to establish the lemma.

**Corollary:** If a node \( s \) appears \( n \) times in the complete path \( w \) of a minimal unicursal graph \( U \), then there are at least \( n-1 \) nodes \( b_1, \ldots, b_{n-1} \) which appear exactly once in \( w \), and clearly \( \text{in}(b_1) = \text{out}(b_1) = 1 \) for each of these nodes.

If \( s \) appears \( n \) times then \( w = \beta_0 s \beta_1 s \beta_2 \ldots s \beta_{n-1} \) and by Lemma 2, \( \beta_1, \beta_2, \ldots, \beta_{n-1} \) must each contain a node which appears nowhere else in \( w \).
Lemma 3: The terminal node, t, of a minimal unicursal graph U has the property \( \text{out}(t) = 0, \text{in}(t) = 1 \).

Proof: If \( \text{out}(t) > 0 \), then in the complete path \( w \) of U, t must appear followed by some node, say a, as well as in the terminal position, i.e., \( w = \alpha \ldots t \ldots a \ldots \). But in that case the terminal appearance of t would be eliminated, i.e., a branch from c to t removed and the resulting graph U would still be unicursal which contradicts the hypothesis that U was minimal. In summary, a minimal unicursal graph has nodes which may be partitioned four ways:

-12-

Partition I:

(b by theorem R1)

\( \begin{cases} \text{1) } P \text{ such that } p \in P \text{ implies } \text{in}(p) = \text{out}(p) = 1 \\ \text{2) } M \text{ such that } m \in M \text{ implies } \text{in}(m) = \text{out}(m) > 1 \\ \text{3) } \text{A node } s \text{ for which } \text{in}(s) - \text{out}(s) = 1 \\ \text{4) } \text{A node } t \text{ for which } \text{in}(t) = 0, \text{out}(t) = 1 \\ \text{5) } \text{The number of members of } P \text{ is at least as great as the maximum value of } \text{out}(x) - 1, \text{ } x \text{ being a node in } U. \end{cases} \)

Construction of Standard Form

Focusing on the partition of the nodes of a minimal unicursal graph U which is given above (partition I) define an ordering on the nodes in U.

If \( x, y \) are nodes in U, \( x \succ y \) iff \( \text{in}(x) \geq \text{in}(y) \). Note now the information given in Partition I (5); that letting \( n_1 \) be the node in U such that \( n_1 \succ x \) for all \( x \) in U; \( P = \text{number of nodes in } P \)

\( \geq \text{out}(n_1) - 1 \)
Define \( P_1 \) = a subset of \( P \) having \( j = \text{out}(m_1) - 1 \) members

\[ \{ p_1, \ldots, p_j \} \]

Define \( M_1 = M \cup P \setminus F_1 \cup \{ s \} \) having \( m \) members

\[ \{ m_1, \ldots, m_m \} \text{ with } m_1 > m_1 + 1 \]

Now construct a new graph \( U' \) having its nodes partitioned into:

\[ P'_i = \{ p_1', p_2', \ldots, p_j' \} \text{ in 1-1 correspondence with } P_1 \text{ corresponding to } p_i' \]

\[ M'_i = \{ m_1', \ldots, m_m' \} \text{ in 1-1 correspondence with } m_i \text{ corresponding to } m_i' \text{ corresponding to } t \text{ of } U \]

We now have a 1-1 correspondence between the nodes of \( U' \) and \( U \).

The branches of \( U' \) are given as follows:

(1) For each \( i = 1 \) to \( m-1 \); there are \( \text{in}(m_i) \) branches from \( m_i' \) to \( m_i'+1 \)

(2) For each \( i = 1 \) to \( k = \text{in}(m_1) \); there is one branch from \( p_i' \) to \( m_i' \)

(3) For each \( i = 1 \) to \( m \); there are \( \text{out}(m_i) - \text{in}(m_i) = x_1 \) branches from \( m_i' \) to members of \( P' \cup \{ t' \} \). These provide all the inputs for nodes of \( P' \) and the terminal node \( t' \).

We show that (3) is possible, i.e \( \sum_1^m x_1 = j+1 \)

\[ \text{out}(m_1') = \text{in}(m_{i+1}) + x_1 = \text{out}(m_i) \]

\[ \sum_1^m \text{out}(m_1') = \sum_1^m \text{in}(m_{i+1}) + \sum_1^m x_1 = \sum_1^m \text{out}(m_i) \]

\[ \sum_1^m x_1 = \sum_1^m \text{out}(m) - \sum_1^m \text{in}(m_{i+1}) \]

\[ \sum_1^m x_1 = \text{in}(m_1) = \sum_1^m \text{out}(m_j) - \text{in}(m_j) \]

\[ \sum_1^m x_1 = \text{in}(m_1) = 1 \] because each node in \( M_1 \) has an equal number of input and output branches except the starting node for which \( \text{out}(s) - \text{in}(s) = 1 \).
\[ \sum_{j=1}^{m} x_j = \text{in}(m_j) + 1 = j + 1 \]

Note that in this construction of \( U' \), \( m_1 \) is the starting node since \( \text{out}(m_1) - \text{in}(m_1) = 1 \) by construction.

In general then the graph \( U' \) has the form given in figure 1.

Its nodes clearly have a 1-1 correspondence with the nodes of \( U \). Corresponding nodes having the same number of out branches and (except for the new start and pre-terminal node) the same number of in branches. \( U' \) like \( U \) is obviously unicursal. \( U' \) is also minimal. This can be seen if we consider the result of removing a set of branches from figure 1 in such a way that the resulting graph is connected and unicursal and still has \( s \) for a starting and \( t \) for a terminal node.

![Diagram](image)

**Figure 1**
Let \( S \) be the set of nodes in \( U \) on which at least one of the removed branches are incident. Let \( R \) be the set of removed branches. It follows that if we consider the graph \( Z \) formed from the nodes of \( S \) and branches of \( R \), it must be such that in \( Z \) \( \text{in}(s) = \text{out}(s) \) for all \( s \in S \). In other words an equal number of in and out branches must be removed from each node in \( U \) from which branches are removed. This is necessary so that the resulting graph will still meet the conditions of Theorem 1 with \( s \) and \( t \) remaining the starting and terminating nodes respectively of the resulting graph. Because the 'removed' graph \( Z \) must meet the above conditions, it follows that \( Z \) must consist of a number of loops taken from \( U \). But noting figure 1 we see that every loop in \( U \) involves a node in \( P \) and thus removal of any such loop will disconnect \( U \).

**Max-Min-Unicursal Graph**

Of the minimal unicursal graphs with \( n \) nodes and with \( \text{out}(x) \leq 1 \) for all its nodes \( x \), we wish to find one which has a maximum number of branches. We wish also to establish this number of branches.

Refer to the standard form illustrated in figure 1. Assume that \( U \) is minimal unicursal in that form with \( n \) nodes; \( \text{out}(x) \leq 1 \) for all nodes in \( x \) in \( U \) (i.e. fanout = 1) \( M \) has \( m \) members, \( m_1 \) is the starting node, \( m_t \) is the node connected directly to the terminal node \( t \). \( P \) has \( j \) members.
Therefore:

\[ \text{out}(P) = j \]
\[ \text{out}(m_1) = j + 1 \quad 1 = 1 \text{ to } t \]
\[ \text{out}(m_1) = j \quad 1 = t+1 \text{ to } m = n-j-1 \]

The total number of branches is \( k \).

\[ k = t(j+1) + (n-j-1-t+1)j \]
\[ k = t + n-j(j) \]

For any value of \( j \), \( k \) will have its maximum value if \( t = m = (n-j)-1 \).

Note that this implies that there is a branch from \( m_m \) to \( t \).

\[ k_{\text{max}} = (n-j)-1 + n-j(j) \]
\[ = (n-j)(j+1) - 1 \]

Now maximizing on \( j \) which in any case cannot be greater than \( \ell-1 \):

\[ \frac{dk_{\text{max}}}{dj} = 0 = n-2j-1 \]

\[ j_{\text{max}} = \text{MIN}(\lfloor \frac{n-1}{2} \rfloor, \ell-1) \]

Let the maximum values of \( k_{\text{max}} = K \)

\[ K = \lfloor \left( \frac{n+1}{2} \right)^2 - 1 \rfloor \quad \text{for } n = 1 \text{ to } 2\ell-1 \]
\[ = \ell(n-\ell+1) \quad \text{for } n > 2\ell-1 \]

---

1, 2 \( \lfloor x \rfloor = x \) if \( x \) is an integer otherwise = next lowest integer
References:

[1] Hopcroft and Ullman; Formal Languages and their Relation to Automata; Addison-Wesley Publishing Company; 1969