CS1200: Intro. to Algorithms and their Limitations Prof. Anurag Anshu

Sender–Receiver Exercise 3: Reading for Receivers

Harvard SEAS - Spring 2026 *2026-02-23*

The goals of this exercise are:

- to develop your skills at understanding, distilling, and communicating proofs and the conceptual ideas in them
- to practice reasoning about the equivalence of more involved variants of the Word-RAM model via simulations

1 Recap: Word-RAM model

Please review the definition of Word-RAM model, part of which was covered in class.

Definition 1.1. The *Word-RAM program* P is defined like a RAM program except that it has a (static) word length parameter w and a (dynamic) *memory size* S.

- The word length w is *static*, meaning that while it can be set to an arbitrary value in $\mathbb{N}$ prior to running P on an input x, it does not change during the computation.
- The memory size S is *dynamic*, meaning that it can change during the computation, as described below.

These are used as follows:

- **Memory**: the memory is an array of length S, with entries in $\{0, 1, \ldots, 2^w - 1\}$. Reads from and writes to memory locations larger than S have no effect. Initially $S = n$, the length of the input array x. Additional memory can be allocated via a MALLOC command, which increments S by 1.
- **Output**: if the program halts, the output is defined to be

  $$(M[\texttt{output\_ptr}], M[\texttt{output\_ptr} + 1], \ldots, M[\min\{\texttt{output\_ptr} + \texttt{output\_len} - 1, S - 1\}]).$$

  That is, portions of the output outside allocated memory are ignored.
- **Operations**: Addition and multiplication are redefined from RAM Model to return $2^w - 1$ (the max possible value) if the result would be $\geq 2^w$.
- **Crashing**: A Word-RAM program P *crashes* on input x and word length w if any of the following occur:
  1. One of the constants c in the assignment commands ($\texttt{var} = c$) in P is $\geq 2^w$.
  2. $x[i] \geq 2^w$ for some $i \in [n]$
  3. $S > 2^w$. (This can happen because $n > 2^w$, or if $2^w - n + 1$ MALLOC commands are executed.)

We denote the computation of a Word-RAM program on input x with word length w by $P[w](x)$. Note that $P[w](x)$ has one of three outcomes:

- halting with an output
- failing to halt, or
- crashing.

We define the *runtime* $T_{P[w]}(x)$ to be the number of commands executed until P either halts or crashes (so $T_{P[w]}(x) = \infty$ if $P[w](x)$ fails to halt).

1.1 A 2D WordRAM model

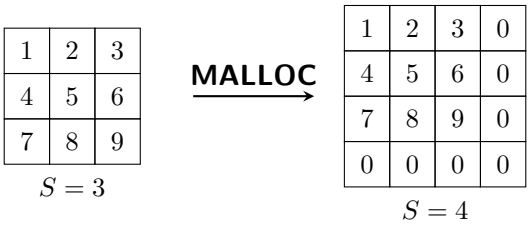
We define a 2-D Word-RAM model, as similar to the Word-RAM model, but the memory is a two-dimensional square array with side length S (hence total size $S \times S$), instead of a one-dimensional array. The changes in the definition are listed below; all other specifications stay the same.

- The commands to write to and read from memory use two variables for memory addresses:
  1. (read from memory) $\texttt{var}_0 = M[\texttt{var}_1][\texttt{var}_2]$ for variables $\texttt{var}_0, \texttt{var}_1, \texttt{var}_2 \in V$.
  2. (write to memory) $M[\texttt{var}_0][\texttt{var}_1] = \texttt{var}_2$ for variables $\texttt{var}_0, \texttt{var}_1, \texttt{var}_2 \in V$.

  As before, reads and writes to memory locations outside the $S \times S$ array have no effect.
- The parameter S is now called *memory width*. The crash conditions stay the same as in usual Word-RAM, including a crash if $S > 2^w$.
- Input is constrained to be within the first row and the variable `input_len` specifies the length of the input. We set $S =$ `input_len` in the initialization step, which suffices to include the input in the available memory.
- Output is defined to be in the first row, in the cells

$$M[0][\texttt{output\_ptr}], \ldots, M[0][\min\{\texttt{output\_ptr} + \texttt{output\_len} - 1, S - 1\}].$$

- MALLOC increases the memory width S by 1 (and consequently increases memory size from S^2 to $(S+1)^2 = S^2 + 2S + 1$).



$S = 3$ $\qquad$ $S = 4$

The goal of this exercise is to understand the proof of the following theorem.

Theorem 1.2. *For every 2-D Word-RAM program P, there is a 1-D Word-RAM program P' such that the following holds.*

1. *For every input x and every word length w such that $P[w](x)$ does not crash, there is a word length w' such that $P'[w'](2^w, \max_i x[i], x)$ halts without crashing on x iff $P[w](x)$ halts, and if they do halt,*
$$P'[w'](2^w, \max_i x[i], x) = P[w](x).$$

2. *If $P[w](x)$ crashes for some x, w, then $P'[w'](2^w, \max_i x[i], x)$ crashes for all possible w'.*

3. *If $P[w](x)$ halts without crashing in t steps, then there is a w' such that $P'[w'](2^w, \max_i x[i], x)$ halts without crashing in $O((n+t)^2)$ steps. Here n is the size of the input x.*

The Proof