*** A key simplifying assumption is that operations are *unit-cost*. Each basic instruction takes one "unit" of time, and we count the number of basic instructions executed to measure time.
(RAM model continued)

**Not all algorithms are appropriate for the unit-cost model ...**

Consider the following program that repeatedly multiplies by 10:

1. Set the variable $x$ equal to 10.
2. Repeat for $n$ steps the operation of multiplying $x$ by 10.

Since each multiplication adds another digit to $x$, this program computes a value with over $n$ digits.
(RAM model continued)

**Another Example**

Repeatedly square a value:

1. Set the variable $x$ equal to 10.
2. Repeat for $n$ steps multiplying the variable $x$ times itself.

Since each multiplication about doubles the number of base 10 digits to represent $x$ (i.e., $x$ is a 1 followed by a number of 0's that doubles with each multiplication), this program computes a value with roughly $2^n$ digits.

For example, even when $n = 25$, more than 30 million digits.
NOTE:

It is possible to strengthen the RAM model by charging a cost per operation that depends on the size of the numbers involved.

However, the extra complication is really not necessary if we are willing to exercise good judgement to understand when it would not be practical to manipulate in a fixed time unit numbers that are involved in an algorithm.
Arrays

The only data structure that we consider to be part of the basic RAM model is the standard one dimensional array, or equivalently the ability to perform *indirect addressing*.

For an integer $x$, not only are there instructions to read or write to memory location $x$, but we can also read or write to memory location $y$, where $y$ is the value stored in memory location $x$.

For example, $x = A[i]$ does not place the contents of the memory location $i$ into $x$, but rather, it copies the contents of a memory location that is determined by the value of $i$. 