28 August 2019

An algorithm a repeatable finite series of steps/instructions to accomplish an intended task.

* finiteness
* well-defined /repeatable /non-ambiguous
* has zero or more inputs of specific types
* there is an output/result with defined relationship to the input.

Analysis of Algorithms:

1. Is it correct?
2. Does it terminate?
3. Time Complexity? (Worst case, average case)
4. Space Complexity?

⋮

1

```
for i = 1...10
| print i
end for
```

- assume: $i = 1 \ldots 10$ iterates i through $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\} =: S$
- i takes values in a finite set S & takes each of those values exactly once.

2

```
j = 10
while j > 0
| j ← j/2
end while
```

loop guard

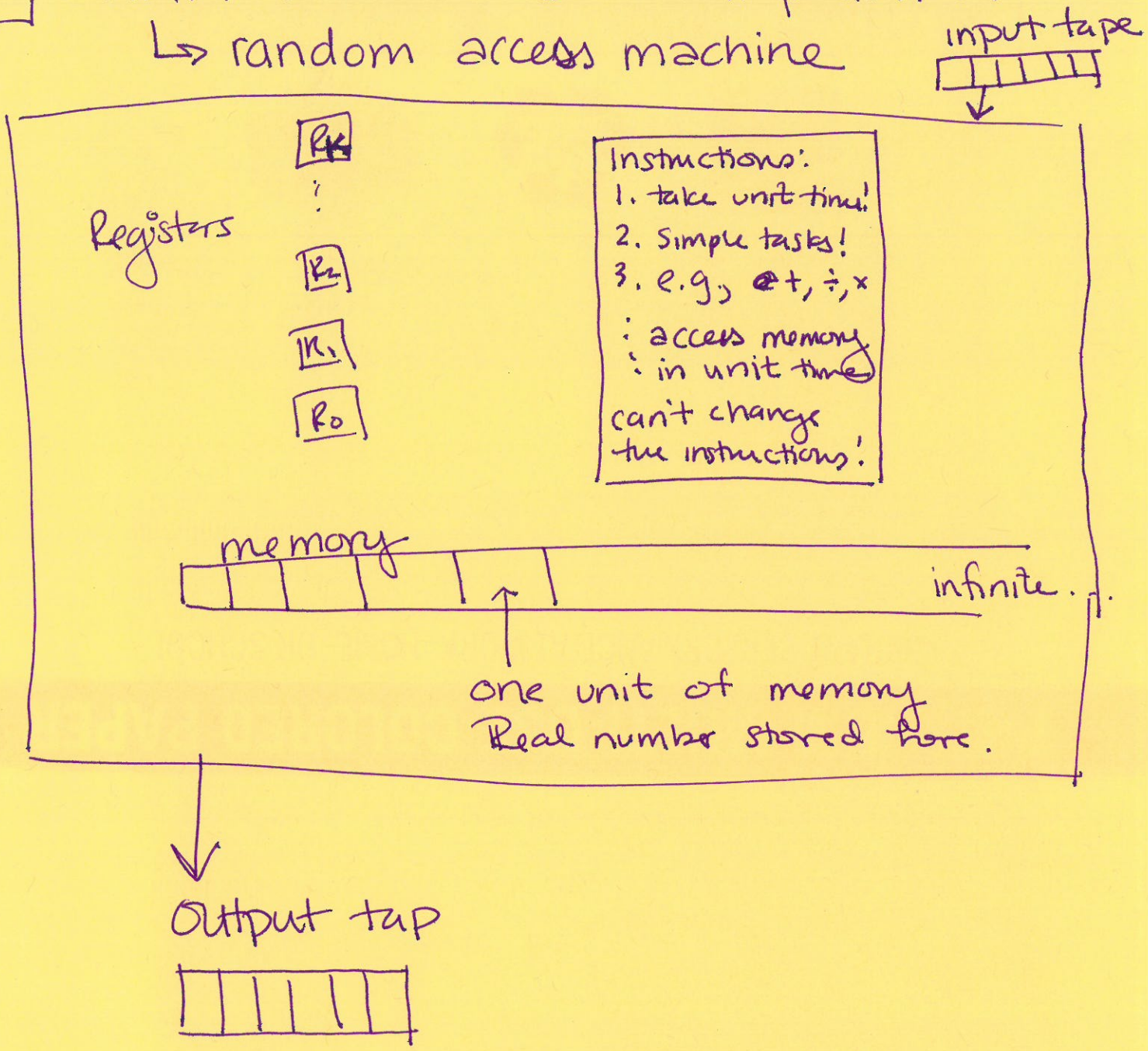
- if $j \in \mathbb{Z}$, terminates if rounds $\frac{1}{2}$ to 0 but not if rounds $\frac{1}{2}$ to 1.
- if j F.P., then probably terminates
- if $j \in \mathbb{R}$, then never terminates
- Suppose j is an int and $1/2 = 0$, then how do we show it terminates?

$D: \{\text{state spaces}\} \to \mathbb{N}$

$X \longmapsto j$

real - RAM model of computation

↳ random access machine



- Time complexity = # of basic instructions in the program, use O or Θ notation
- Space complexity = # of memory cells needed, usu. using O or Θ notation

Let S be a set.

We say that S is well-ordered if

$\forall\ A \subseteq S$, $\exists$ a smallest element in A.

i.o.w: $\inf A \in A$.

$\mathbb{N}$ are a well-ordered set!

TO prove an ~~algo~~ (loop) terminates, we define a fcn:

$D: \{\text{state space}\} \longrightarrow \mathbb{N}$

such that

1. D is well-defined when entering the loop for the first time.
2. each time through the loop, D decreases
3. when D reaches 1, the loop terminates

↑ so, the loop guard should check if we have reached the smallest element.

notations:

$\forall$ = for all

$\in$ = is an elt. of

$\exists$ = there exists

$\min A$ = if A finite, $\min A$ is the smallest elt.

$\inf A$ = allows for inf. sets.

i.o.w = in other words

$\inf\{(0,1)\} = 0 \notin (0,1)$