# Computing Theory

COMP 147 (4 units)

## Chapter 3: The Church-Turing Thesis
## Section 3.3

# Alonzo Church and Alan Turing

Image: http://geopolicraticus.wordpress.com/2012/11/23/the-church-turing-thesis-and-the-asymmetry-of-intuition/

# The Church-Turing Thesis (1936)

"computability" ⇔ λ-calculus ⇔ Turing machines

# Hilbert's 10th Problem (1900)

- Devise a process of finite steps that tests if a polynomial has an integral root
  - (Hilbert didn't actually use the word "algorithm")
- Issues at the time:
  - What is an "algorithm"?
  - Can non-existence of an algorithm be demonstrated?
  - If we don't understand precisely what an algorithm is, how can we argue about whether one exists for a particular problem?

# Defining Algorithms

- Turing and Church independently came up with definitions of an algorithm

- Church: an algorithm is anything that can be expressed in the λ-calculus

- Turing: an algorithm is anything that can be performed by a Turing machine

- Gödel: an algorithm is anything that can be expressed by primitive recursive functions*

*This may be slightly inaccurate terminology.

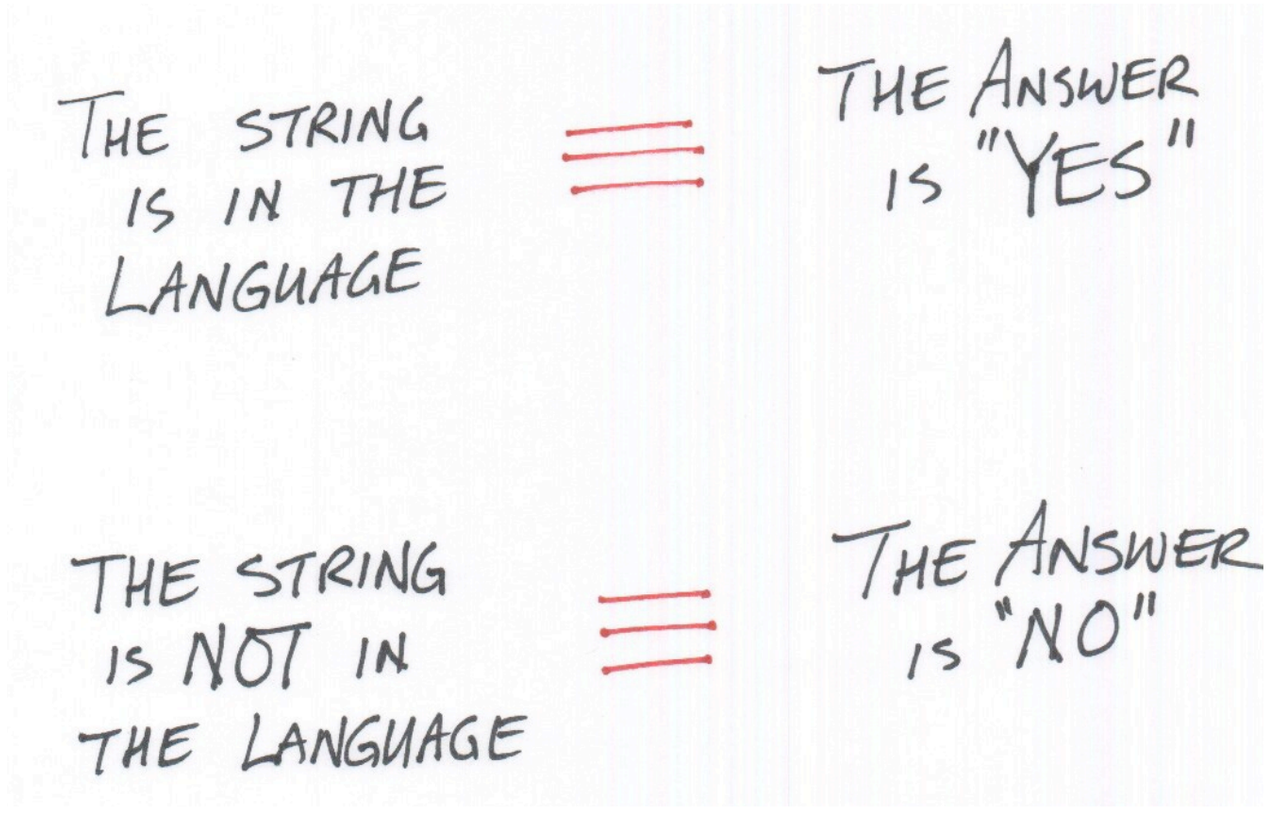
# Defining Algorithms

- Importantly, all the preceding notions of an algorithm were proven to be equivalent
  - They all state that the same class of problems is “effectively computable” in the sense that they have an algorithmic solution
  - The fact that they are all equivalent is *strong evidence* that they are correct

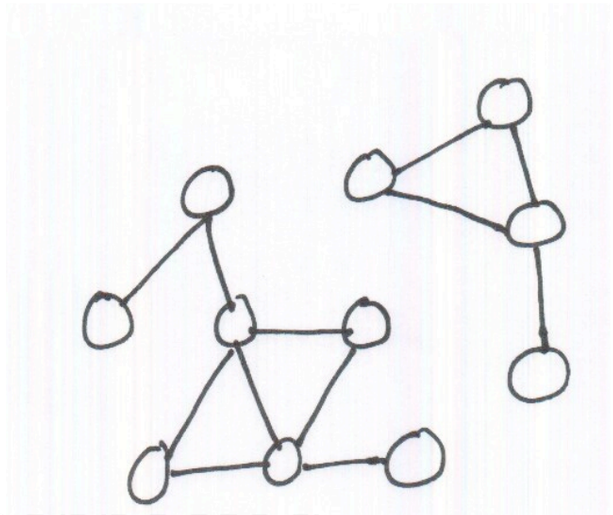
- Effectively computable ⇔ an algorithm exists ⇔ Turing-decidable

# TM as Problem Solvers

- TM can decide or recognize languages
- Arbitrary problems can be expressed as languages
- An instance of the problem can be encoded into a string



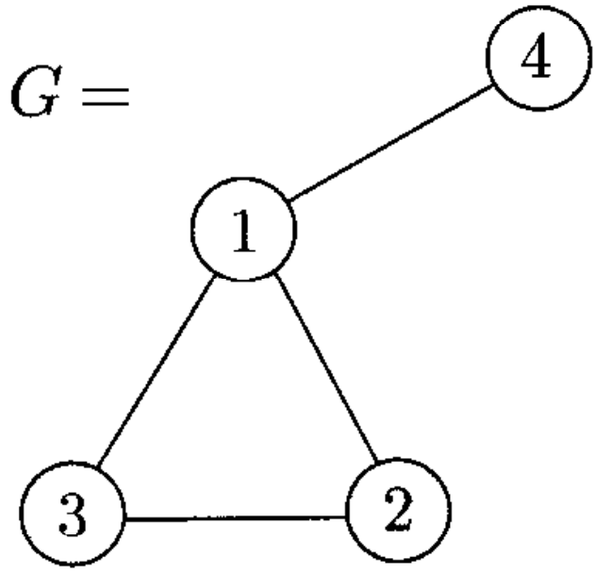
# Example



- Problem: Determine if an undirected graph is connected

- Expressed as a language:
  $A = \{ \langle G \rangle \mid G$ is a connected undirected graph $\}$

- We would like to find a TM that decides that language
  - Accepts if input string represents a connected graph
  - Rejects if graphs is not connected (or if input is not a valid representation a of a graph)
  - Loop? No, TM will always halt

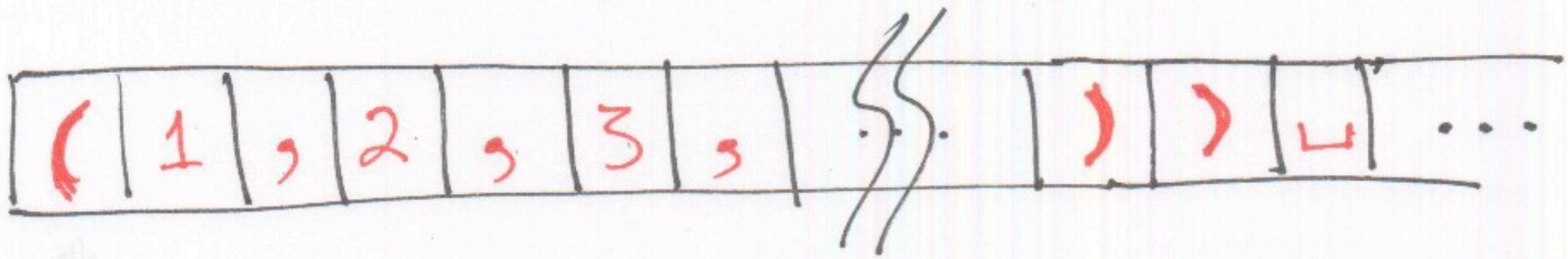



$\langle G \rangle =$ `(1,2,3,4)((1,2),(2,3),(3,1),(1,4))`

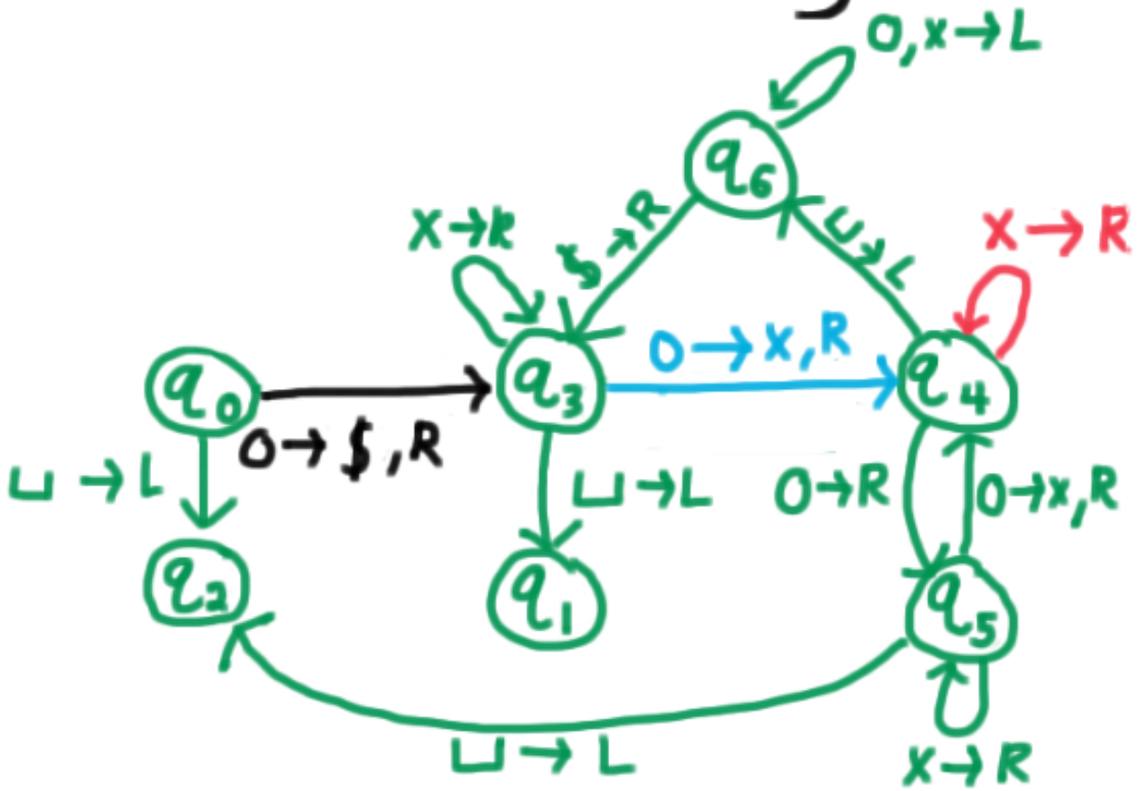
list of nodes  list of edges

**FIGURE 3.10**
A graph $G$ and its encoding $\langle G \rangle$

$\Sigma = \{ (, ), ,, 1, 2, 3, 4, \cdots, 9 \}$



# Encoding a Turing Machine

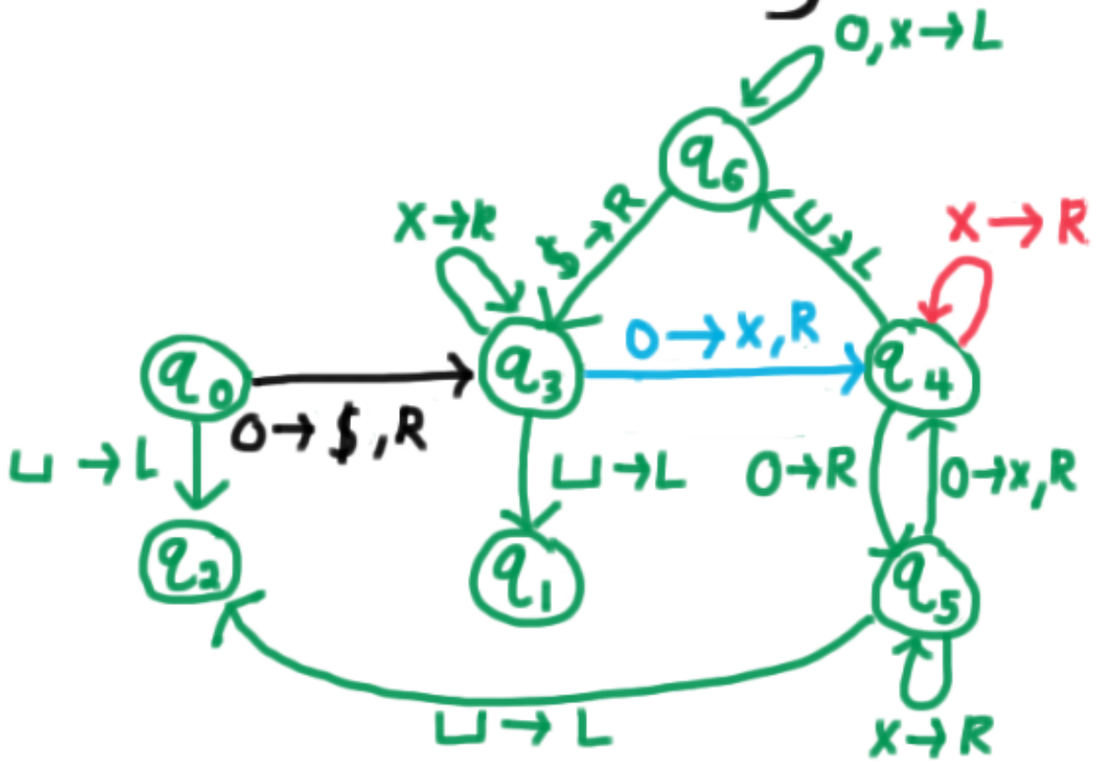



| Symbol | Representation |
|---|---|
| b | a 00 |
| 0 | a 01 |
| X | a 10 |
| $ | a 11 |

<M> = (q000, a01, q011, a11, R)

- q000 → Input state
- a01 → Input Symbol
- q011 → Output state
- a11 → Output Symbol
- R → Direction

# Encoding a Turing Machine




| Symbol | Representation |
|---|---|
| b | a 00 |
| 0 | a 01 |
| X | a 10 |
| $ | a 11 |

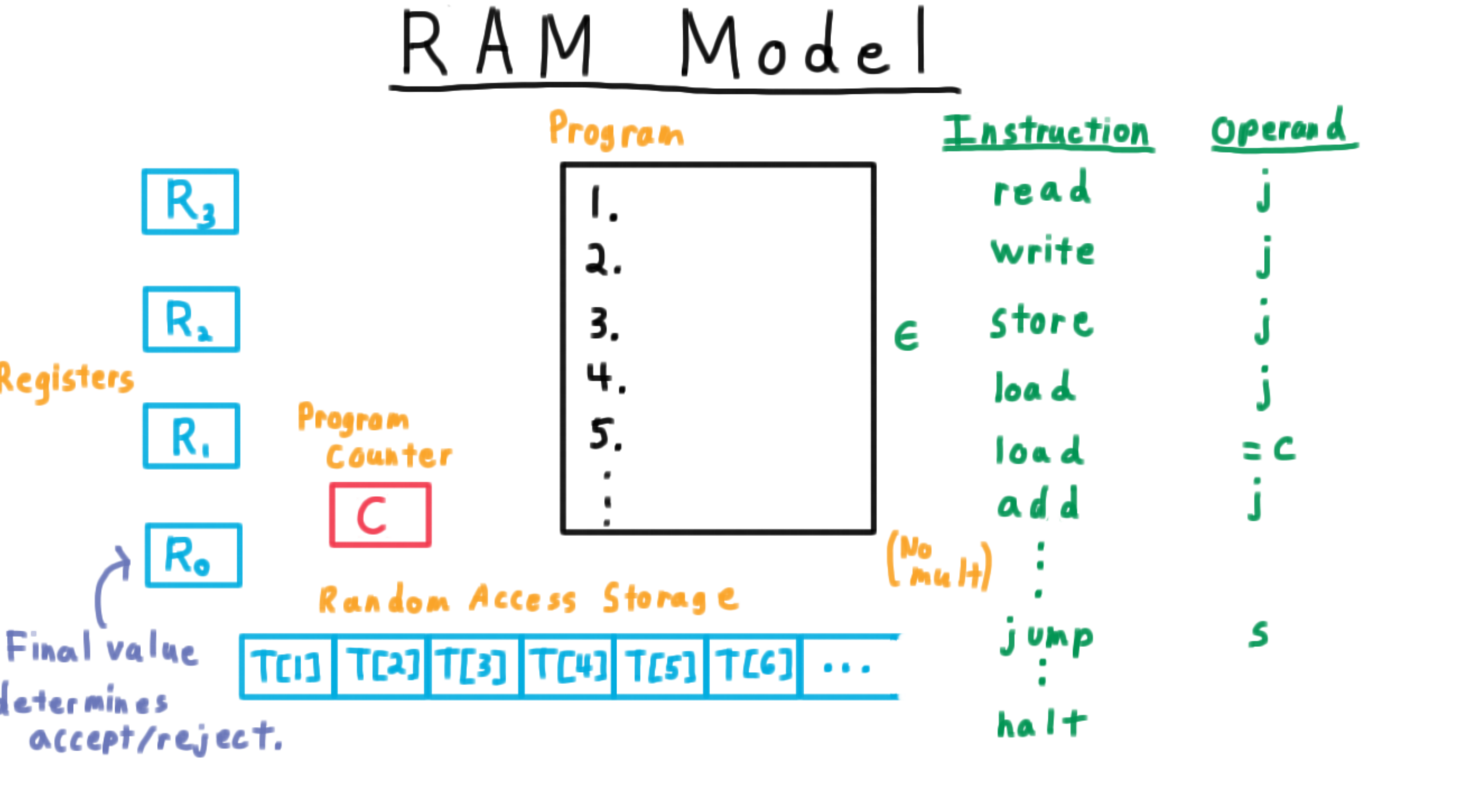
<M> = (q000, a01, q011, a11, R)

(q011, a01, q100, a10, R)

Input state — Input Symbol — Output state — Output Symbol — Direction

# TMs are as Powerful as Computers

Random Access model, very closely resembles the basic CPU/register/ memory paradigm behind the design of modern computers.



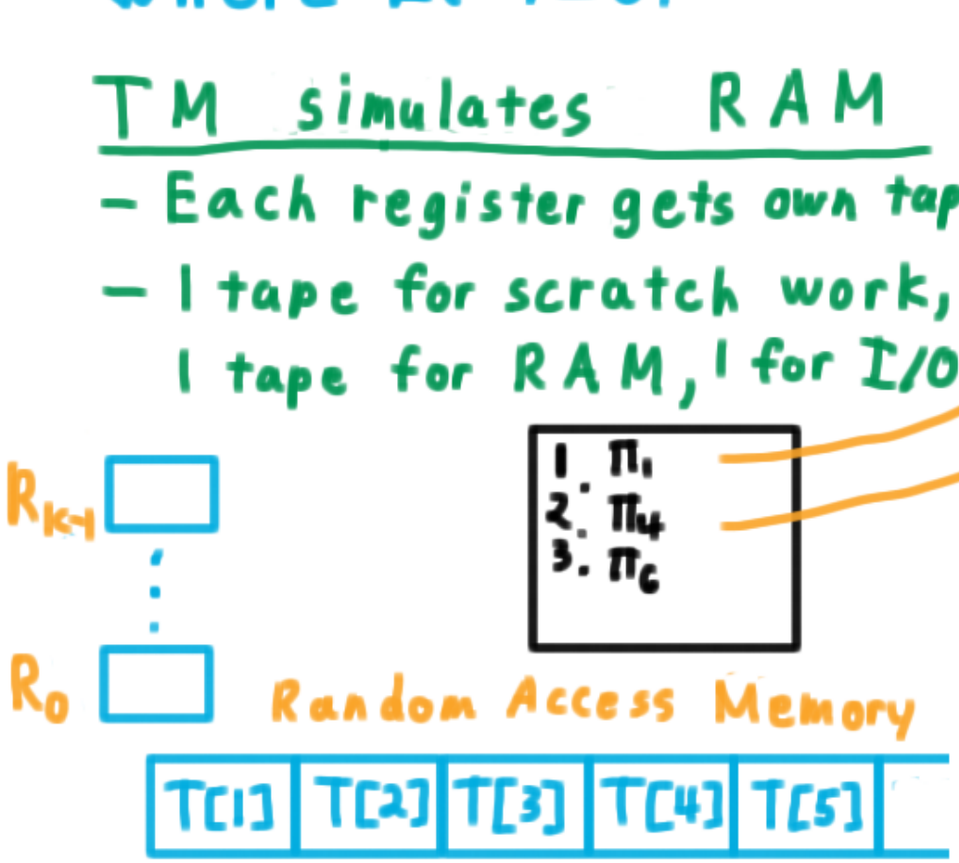
# TMs are as Powerful as Computers

## Equivalence of RAM and Turing Machines

Let $E: \{\sqcup\} \cup \Sigma \to \{0, .., |\Sigma|\}$ be a 1-to-1 correspondence where $E(\sqcup) = 0$.

### TM simulates RAM

- Each register gets own tape.
- 1 tape for scratch work,
  1 tape for RAM, 1 for I/O



# Algorithm=Turing Machine

- high-level specification: pseudocode, expressed in programming language

- implementation-level: Contents of the tape, data representation, tape head movement etc..

    - more details but sill abstract

- specification of implementation:

    - state, alphabet, transition function etc…

# Example: Graph Connectedness

$A$ = { <$G$> | $G$ is a connected undirected graph }
Turing machine $M$ decides $A$

$M$ = "On input $\langle G \rangle$, the encoding of a graph $G$:

1. Select the first node of $G$ and mark it.
2. Repeat the following stage until no new nodes are marked:
3. For each node in $G$, mark it if it is attached by an edge to a node that is already marked.
4. Scan all the nodes of $G$ to determine whether they all are marked. If they are, *accept*; otherwise, *reject*."