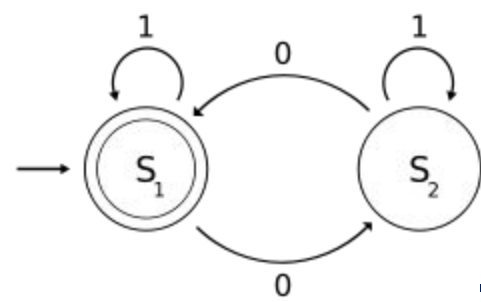
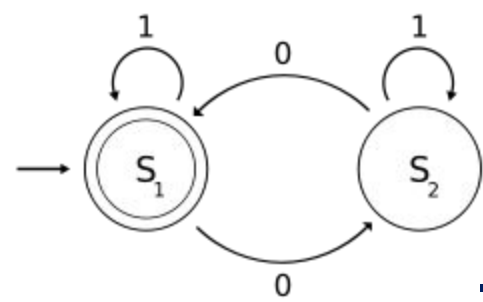




Measuring Time Complexity

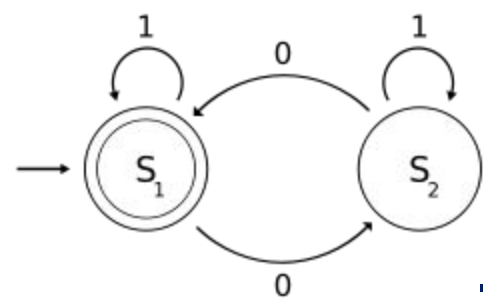


Today's adventures

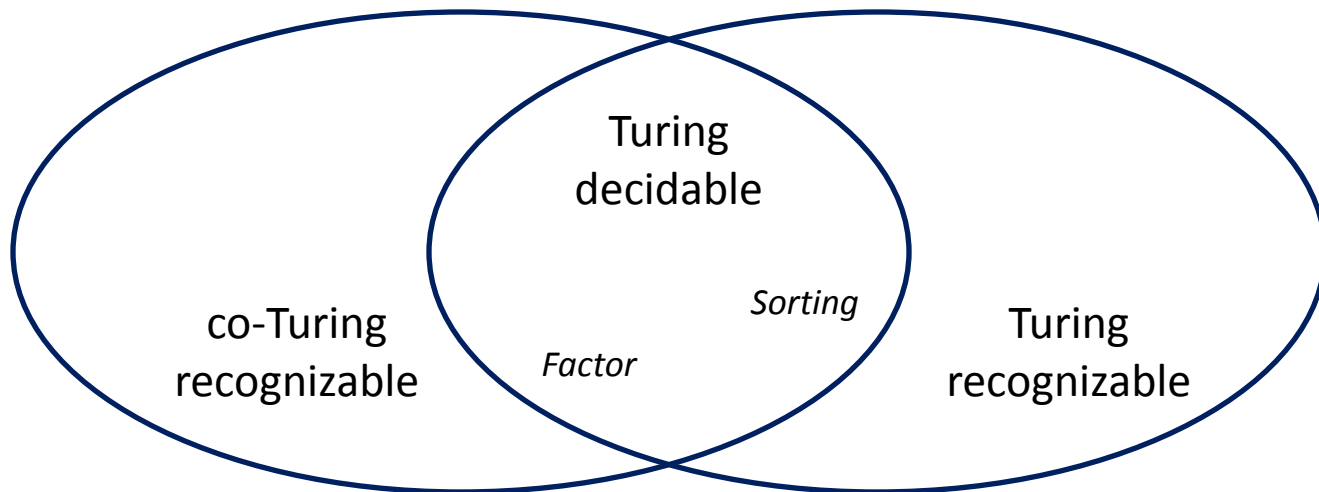


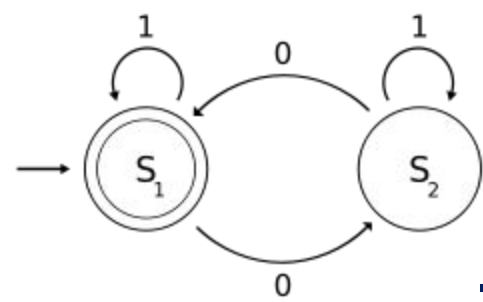
- Define time complexity,
- define big O, little o & examples
- relationships with multitape and nondeterministic machines

Weekly tip: You are the architect and designer of your own success



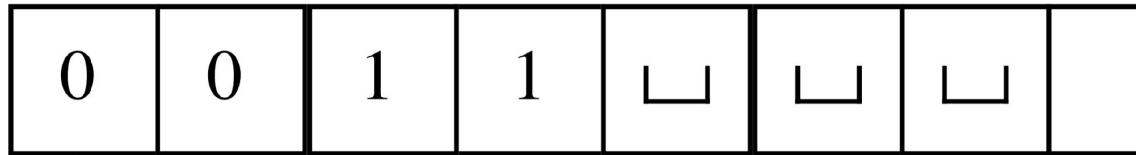
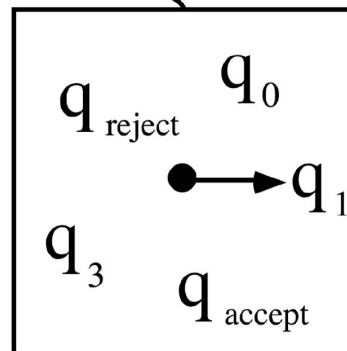
Solvable in Theory ...

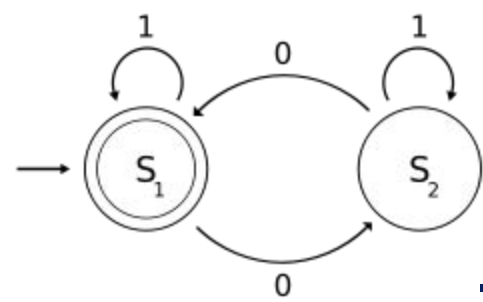




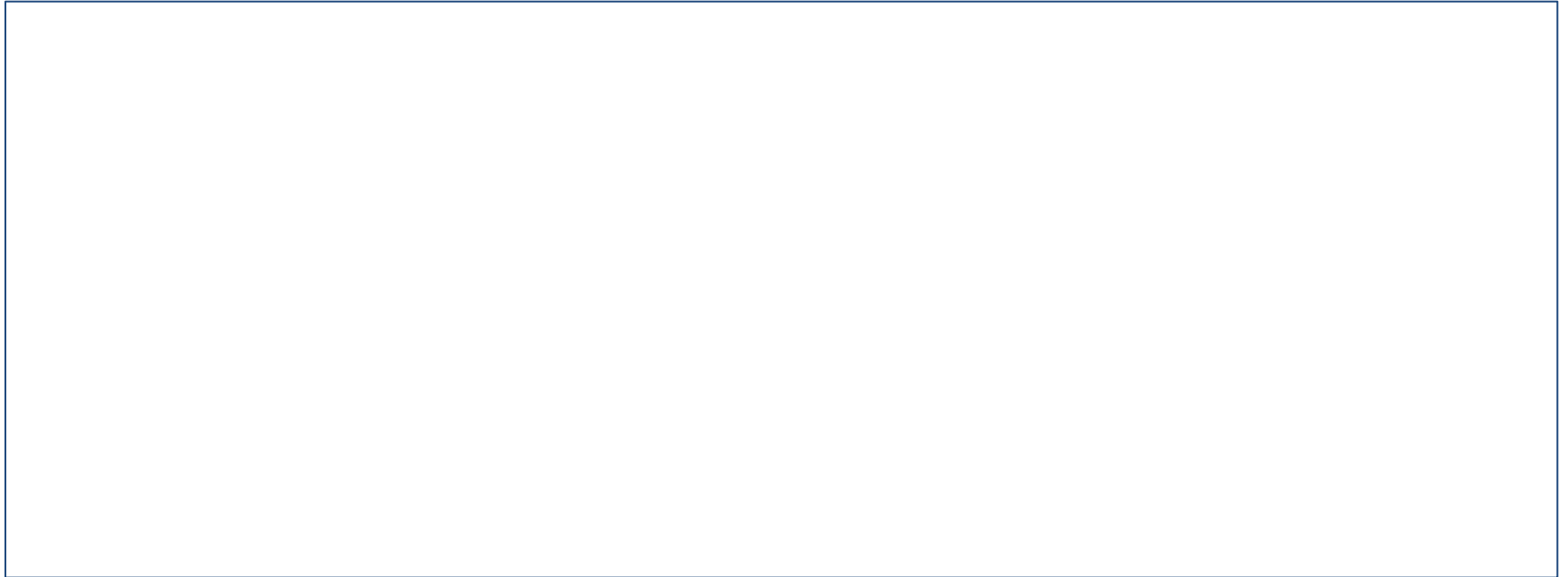
Measuring Difficulty

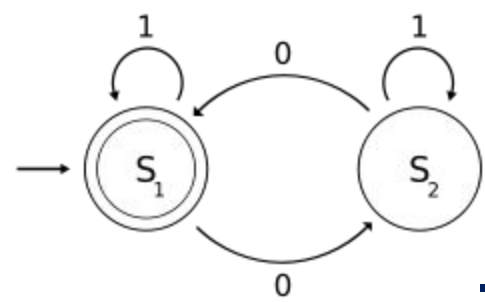
How hard is it to recognize $\{ 0^k 1^k \mid k \geq 0 \}$?



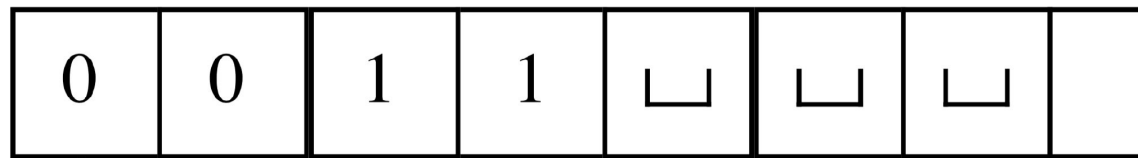
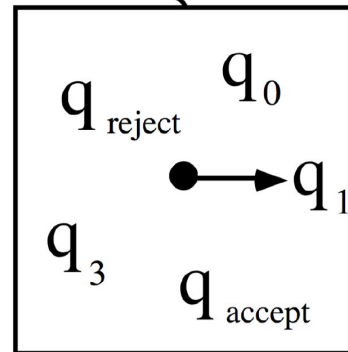


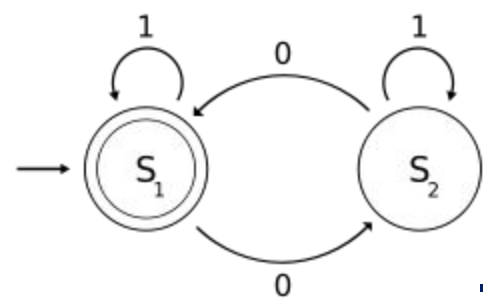
An Algorithm



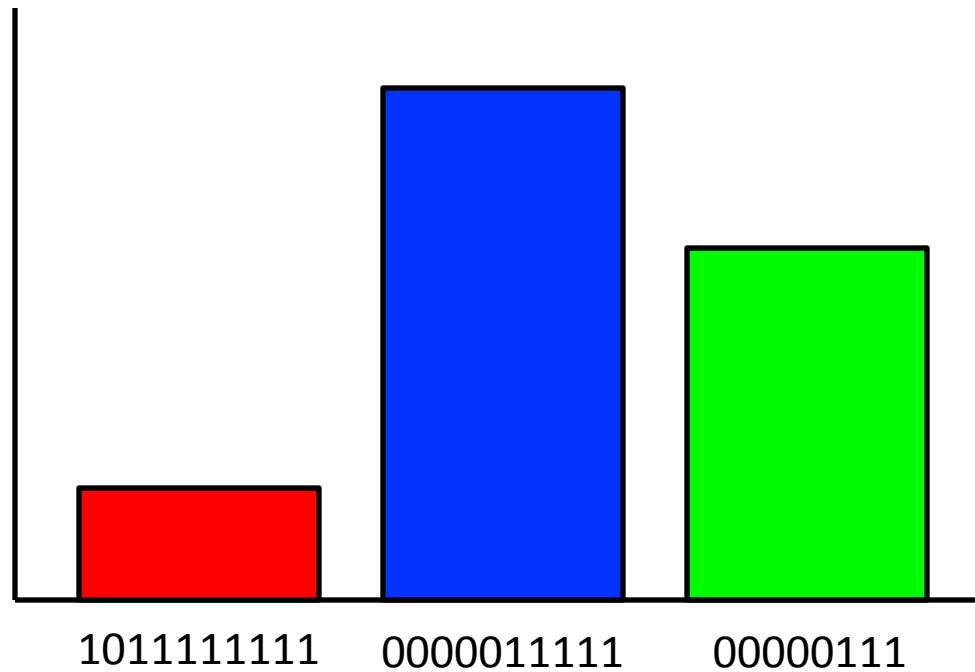


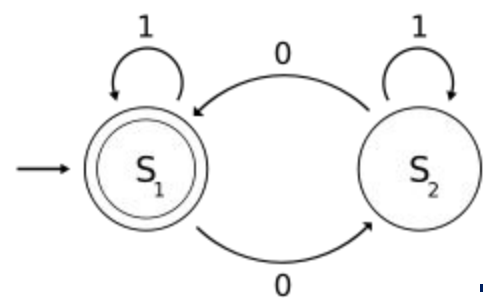
Number of Steps Depends on Size of Input





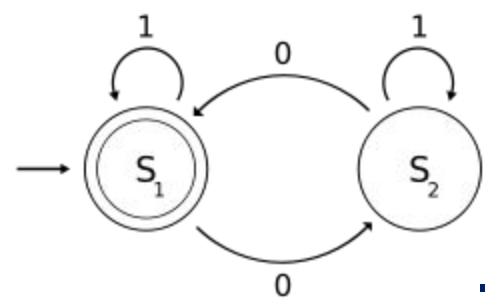
The Good, the Bad, and the Average



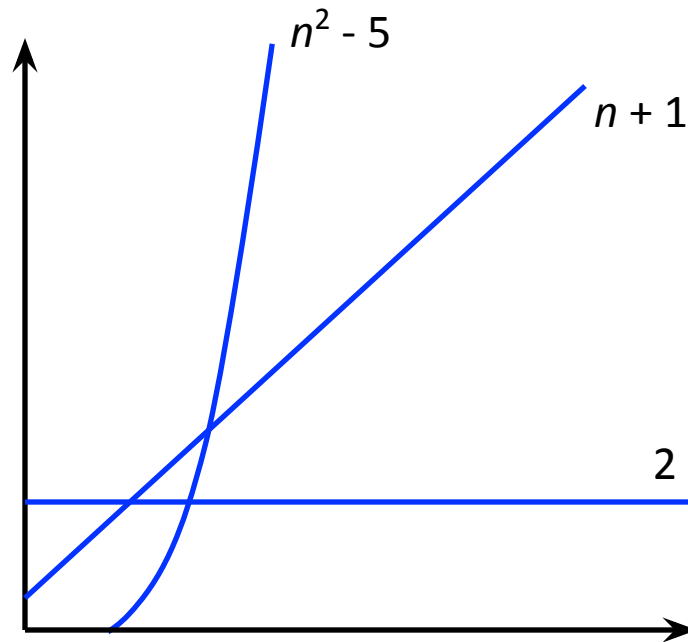


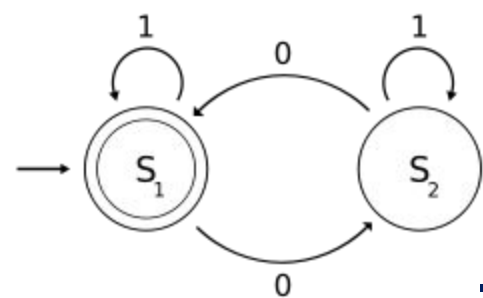
Time Complexity

Definition. The *time complexity* of TM M is the function $f: \mathbb{N} \rightarrow \mathbb{N}$, where $f(n)$ is the maximum number of steps that M uses on any input of length n .



Asymptotic Analysis





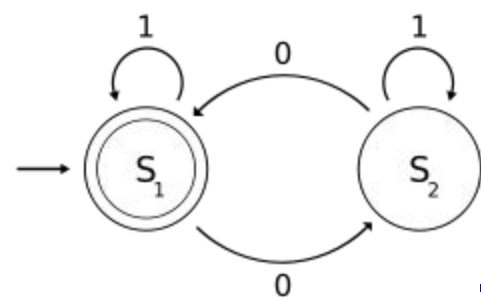
Big O, Little o

Definition. Consider $f, g: \mathbb{N} \rightarrow \mathbb{R}^+$. Say that $f(n) = O(g(n))$ if positive integers c and n_0 exist such that for every $n \geq n_0$

$$f(n) \leq c g(n)$$

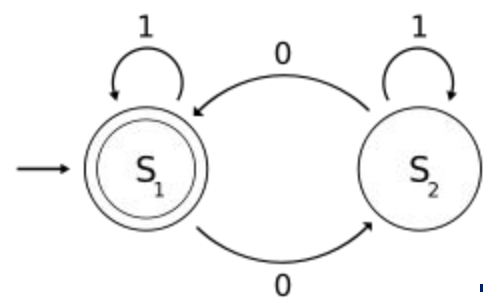
Definition. Consider $f, g: \mathbb{N} \rightarrow \mathbb{R}^+$. Say that $f(n) = o(g(n))$ if

$$\lim_{n \rightarrow \infty} f(n)/g(n) = 0$$

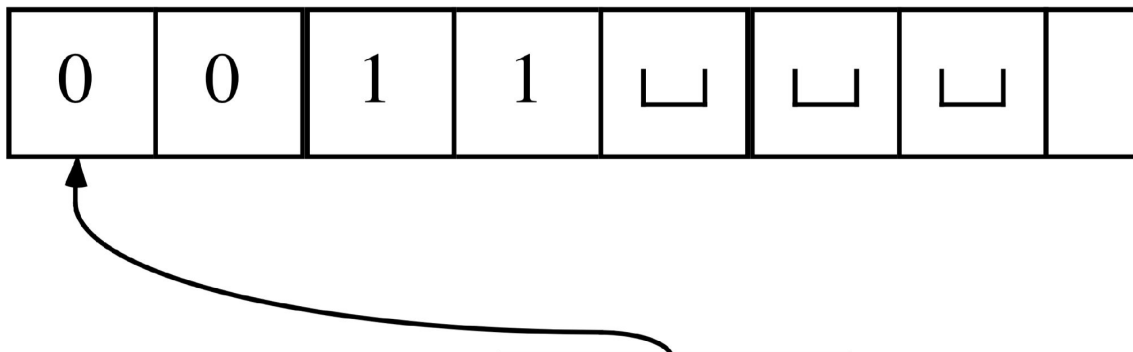


True or False?

1. $8n + 5 = O(n)$
2. $8n + 5 = o(n)$
3. $\sqrt{n} = o(n)$
4. $\log_2 n = o(\ln n)$
5. $n \log \log n = o(n \log n)$
6. $n^2 = o(n \log n)$

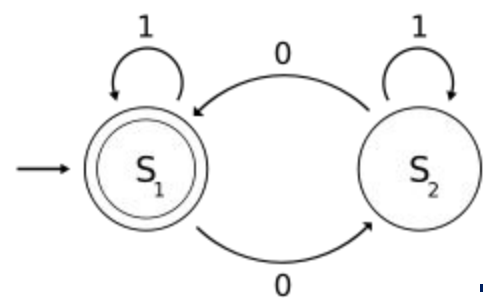


The Verdict Please ...



M_1 = “On input string w :

1. Scan across the tape and *reject* if a 0 is found to the right of a 1.
2. Repeat the following if both 0s and 1s remain.
3. Scan across tape, crossing off a single 0 and a single 1.
4. If either 0 or 1 remains, *reject*. Otherwise, *accept*.”



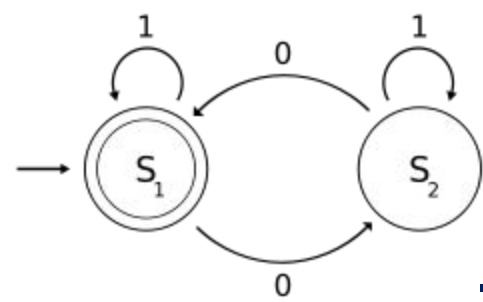
Time Complexity Classes

Definition. Let $t: \mathbb{N} \rightarrow \mathbb{N}$ be a function. Define the time complexity class, $\text{TIME}(t(n))$, to be

$$\text{TIME}(t(n)) = \{ L \mid L \text{ is a language decided by } O(t(n)) \text{ time TM} \}$$

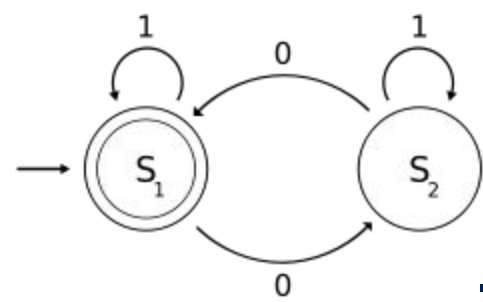
Example. The language $\{ 0^k 1^k \mid k \geq 0 \} \in \text{TIME}(n^2)$.*

* And $\text{TIME}(n^3)$ and $\text{TIME}(n^4)$ and ...



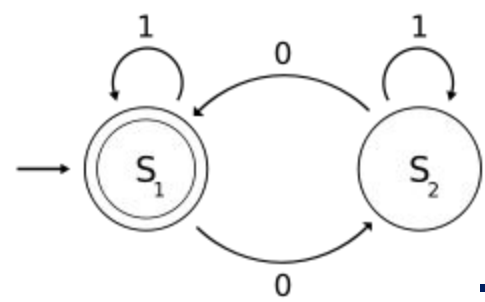
Losing TIME Complexity



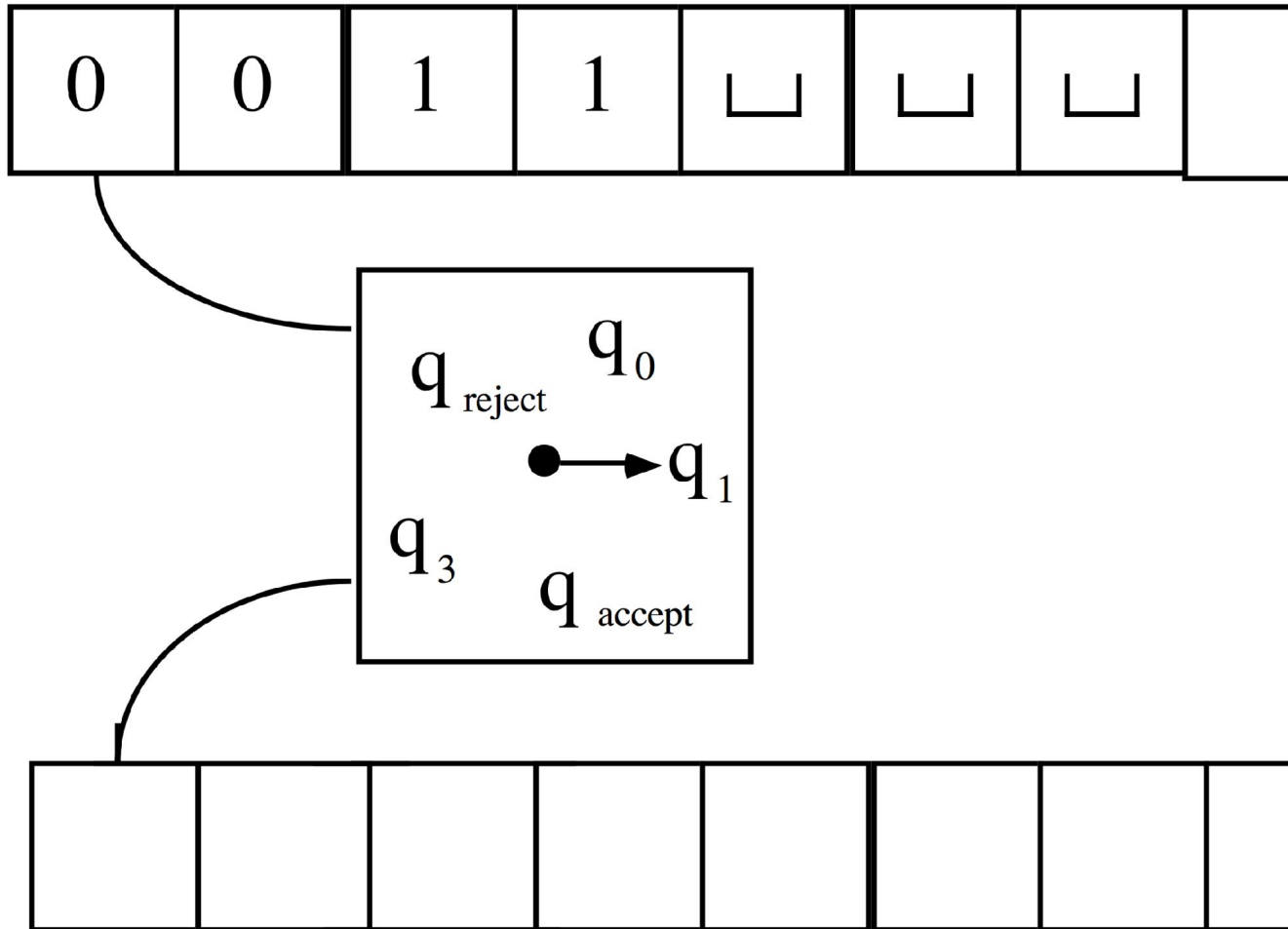


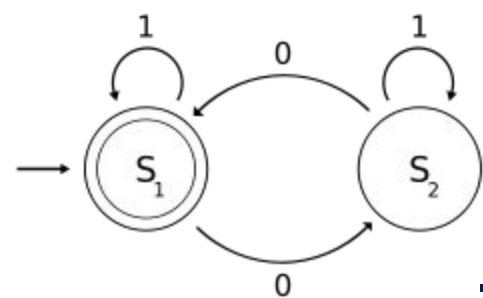
Better Still???

Theorem. Let $f: \mathbb{N} \rightarrow \mathbb{N}$ belong to $o(n \log n)$. If language $A \in \text{TIME}(f)$, then A is regular.



Ah, But What If We Had Two Tapes ...



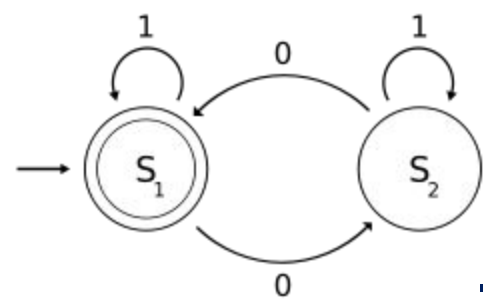


Well, Then We Could Do Better Still!*

Theorem. Let $t(n)$ be a function, where $t(n) \geq n$. Then every $t(n)$ time multitape Turing machine has an equivalent $O(t^2(n))$ time single-tape Turing machine.

Proof Idea. Compare the time complexity of the given multitape machine with the equivalent single tape machine that simulates it.

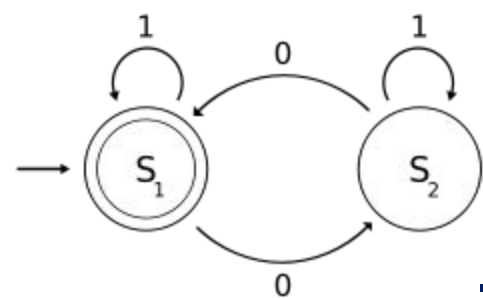
* But only slightly.



Blast from the past

If we have a binary tree of height t ,
what is the maximum number of nodes it can have?
What about the minimum?

If we have an n -ary tree of height t ,
what is the maximum number of nodes it can have?



How Costly Is Nondeterminism?

Theorem. Let $t(n)$ be a function, where $t(n) \geq n$. Then every $t(n)$ time nondeterministic single-tape Turing machine has an equivalent $2^{O(t(n))}$ time deterministic single-tape Turing machine.

Proof Idea.

Compare the time complexity of the given nondeterministic machine with the equivalent single tape deterministic machine that simulates it.

