Exam 1 a week from today
- practice exam on website
- office hours updated
- Review session Monday

# CSE 105
# THEORY OF COMPUTATION

"Winter" 2018

http://cseweb.ucsd.edu/classes/wi18/cse105-ab/

# Today's learning goals

Sipser Section 2.2

- Define push-down automata informally and formally
- Trace the computation of a push-down automaton
- Describe the language recognized by a push-down automaton

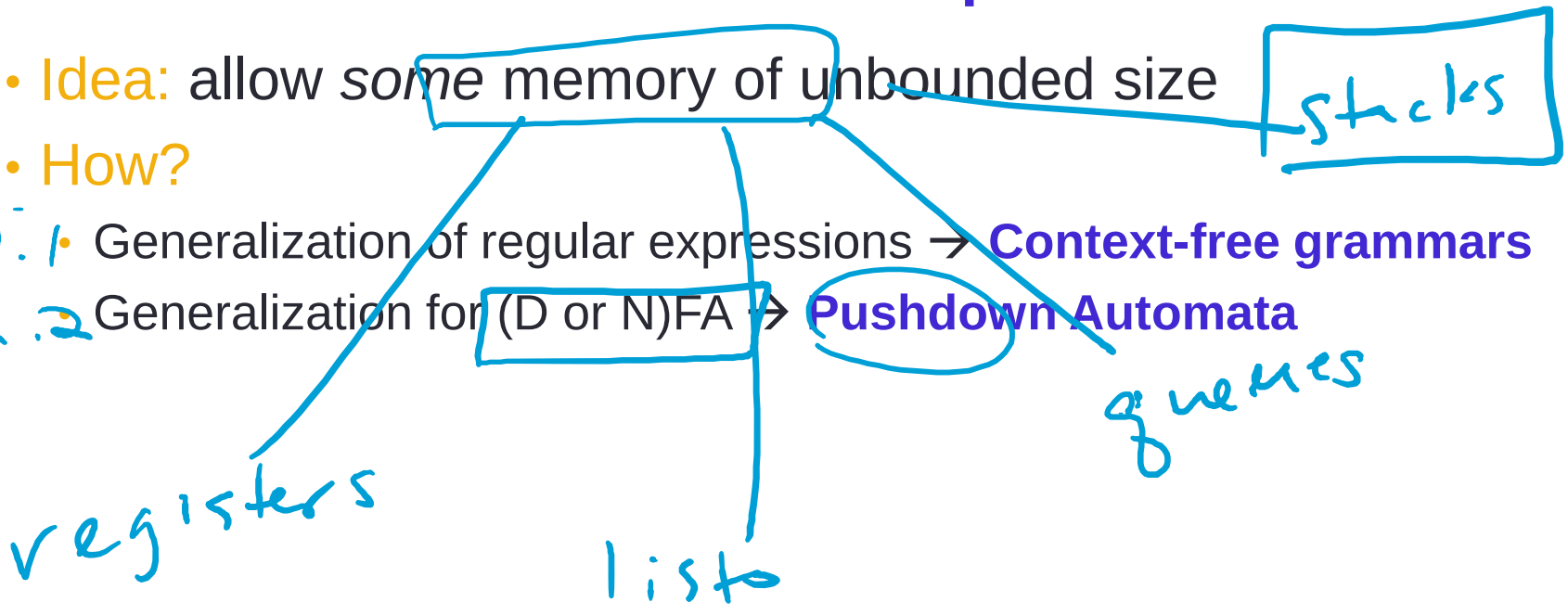
# Regular sets: not the end of the story

- Many **nice / simple / important** sets are not regular
- Limitation of the finite-state automaton model
  - Can't "count"
  - Can only remember finitely far into the past
  - Can't backtrack
  - Must make decisions in "real-time"
- We know computers are more powerful than this model…

*Which conditions should we relax?*

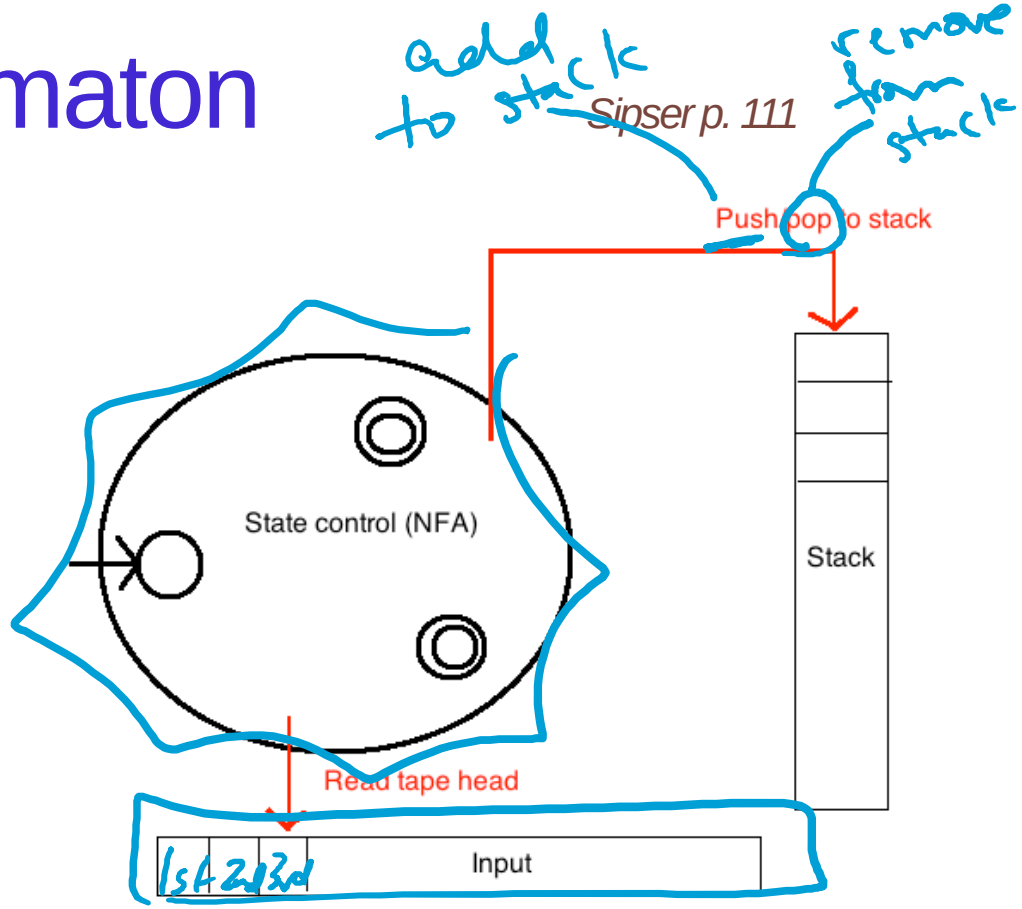
# The next model of computation

- Idea: allow *some* memory of unbounded size
- How?
  - Generalization of regular expressions → **Context-free grammars**
  - Generalization for (D or N)FA → **Pushdown Automata**



# Push-down automaton

- NFA + stack



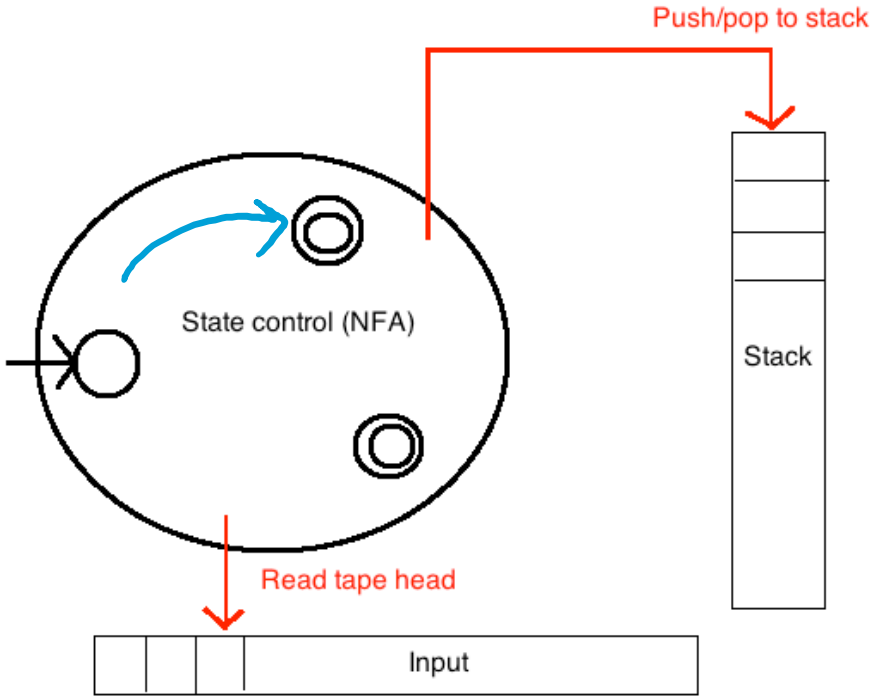
# Push-down automaton

*Sipser p. 111*

- NFA + stack

*At each step*

1. **Transition** to new state based on current state, letter read, and top letter of stack.
2. (Possibly) **push or pop** a letter to (or from) top of stack



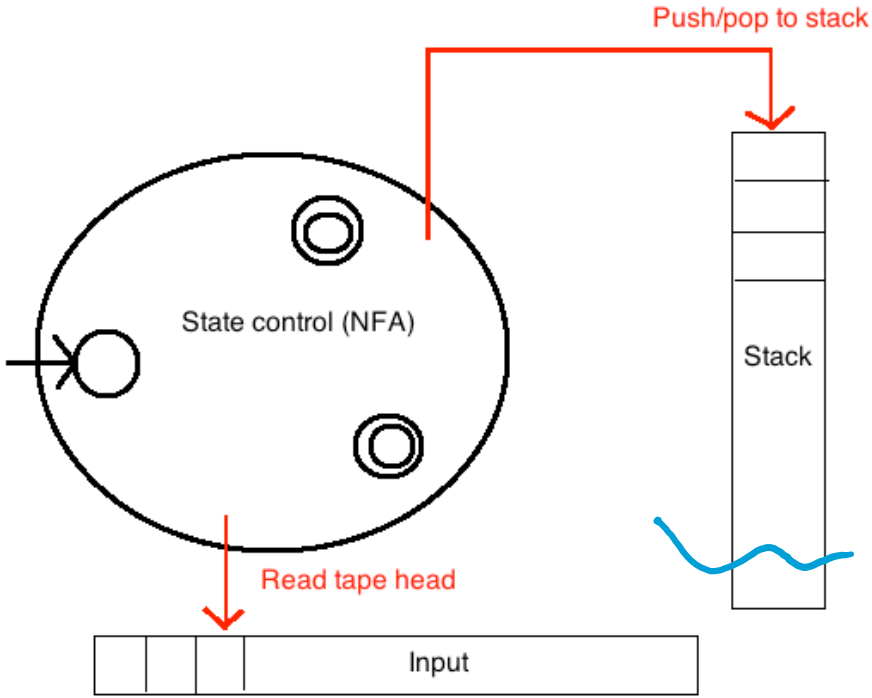
# Push-down automaton

*Sipser p. 111*

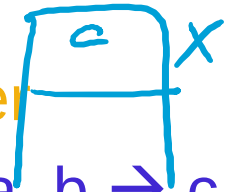
- NFA + stack

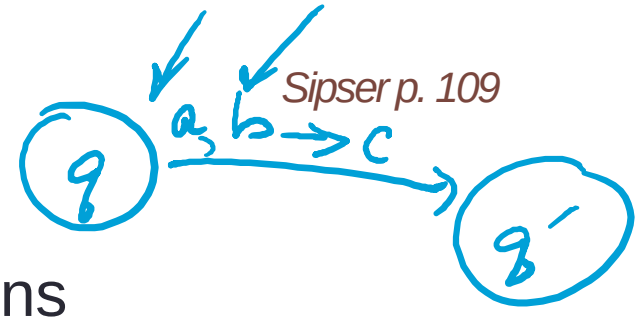
*Accept a string iff*

there is **some** sequence of states and **some** sequence of stack contents which processes the entire input string and ends in an accepting state.



# State diagram for PDA




If hand-drawn or in Sipser

State transition labelled a, b → c means

"read an a from the input, pop b from the stack, push c to the stack."

In JFLAP use ; instead of →

# State diagram for PDA

*Sipser p. 109*

how do we know when stack empty?

## If hand-drawn or in Sipser

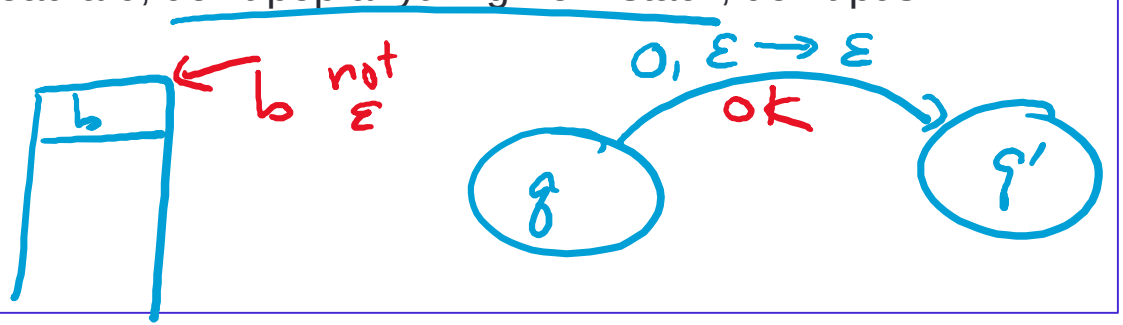
State transition labelled a, b → c means

"read an a from the input, pop b from the stack, push c to the stack."

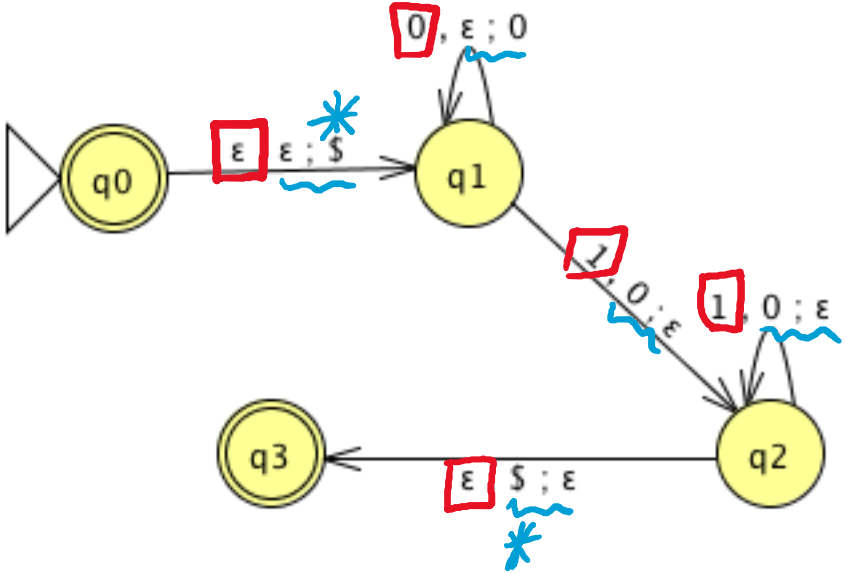
input

What edge label would indicate "Read a 0, don't pop anything from stack, don't push anything to the stack"?

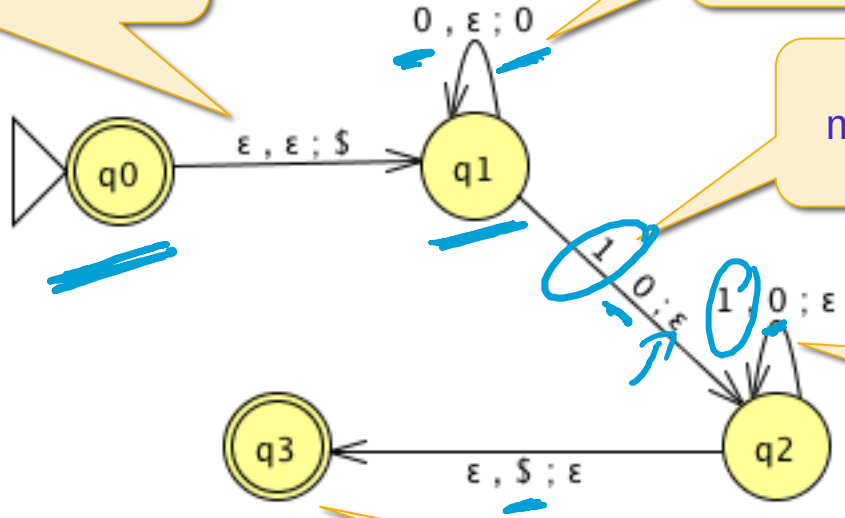
A. 0, ε → ε
B. ε, 0 → ε
C. ε, ε → 0
D. ε → ε, 0
E. I don't know.



# Push down automaton ex.




stack alphabet {-, -, -}




Without reading any input, push a $ to the stack

While we read 0s, push each one to stack (to keep count)

When we read a 1, match it with one of the 0s in stack

For each 1 we read, match it to a 0 that was on the stack.

If have reached the end of the input and the stack is empty (only $ remains), then #0s = #1s and the machine accepts the string.

$

$L(\cdots) = \{0^n 1^n \mid n \geq 0\}$

# Formal definition of PDA

*Sipser Def 2.13 p. 113*

A PDA is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$ where $Q, \Sigma, \Gamma, F$ are all finite sets and

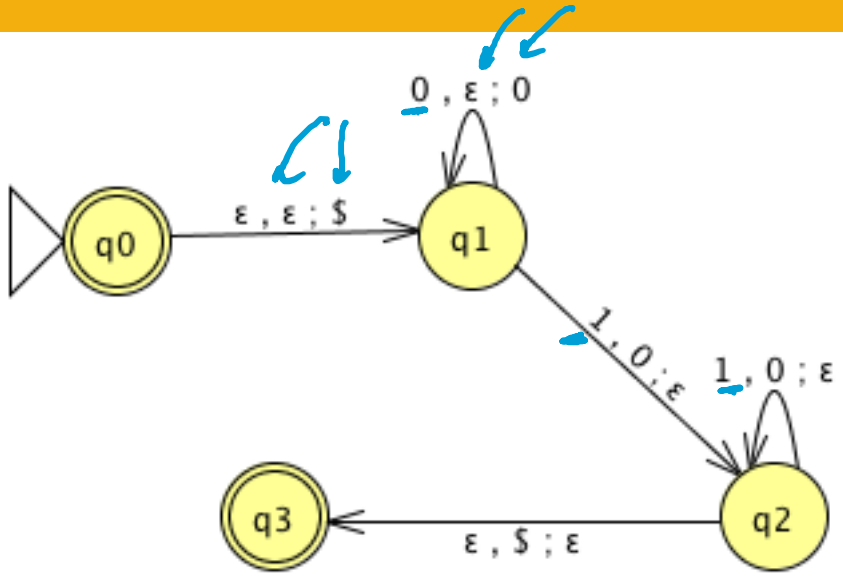
1. $Q$ is the set of states
2. $\Sigma$ is the input alphabet
3. $\Gamma$ is the stack alphabet (no constraints about $\Sigma$ vs $\Gamma$)
4. $\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \to \mathcal{P}(Q \times \Gamma_\varepsilon)$ is the transition function
5. $q_0 \in Q$ is the start state
6. $F \subseteq Q$ is the set of accept states.

$\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$

$\Gamma_\varepsilon = \Gamma \cup \{\varepsilon\}$

# Formal definition

What is Σ and what is Γ for this PDA?




A. Σ = Γ = {0, 1}
B. Σ = {0, 1}, Γ = {$, ε}
C. Σ = {0, 1}, Γ = {$, 0}
D. Σ = {0, 1, ε}, Γ = {$, 0, 1, ε}
E. None of the above.

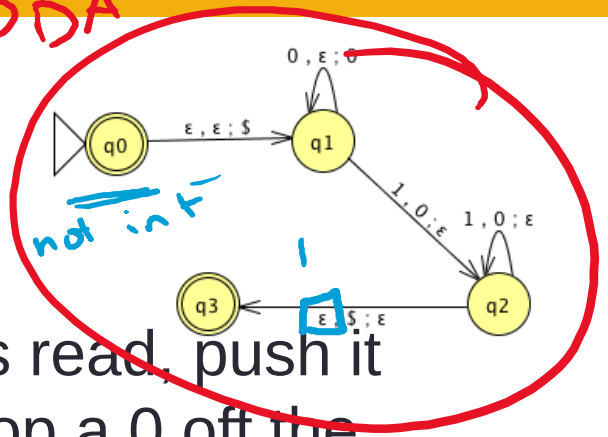
# Designing a PDA

$$L = \{ 0^i 1^{i+1} \mid i \geq 0 \}$$

*Informal description of PDA:*

Read symbols from the input. As each 0 is read, push it onto the stack. As soon as 1s are seen, pop a 0 off the stack for each 1 read. If the stack becomes empty and there is exactly one 1 left to read, read that 1 and accept the input. If the stack becomes empty and there are either zero or more than one 1s left to read, or if the 1s are finished while the stack still contains 0s, or if any 0s appear in the input following 1s, reject the input.

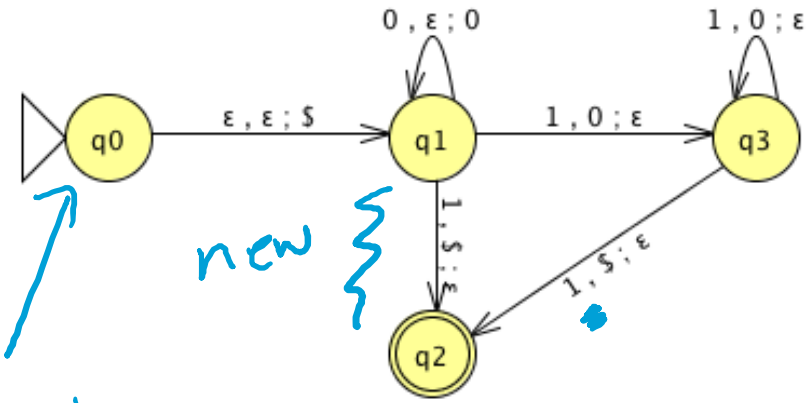
# Informal description




Read symbols from the input. As each 0 is read, push it onto the stack. As soon as 1s are seen, pop a 0 off the stack for each 1 read. If the stack becomes empty and there is exactly one 1 left to read, read that 1 and accept the input. If the stack becomes empty and there are either zero or more than one 1s left to read, or if the 1s are finished while the stack still contains 0s, or if any 0s appear in the input following 1s, reject the input.

# Designing/Tracing a PDA

$$L = \{ 0^i 1^{i+1} \mid i \geq 0 \}$$




new

not accepting anymore

Sample computations?

- nondeterminism
- stack contents

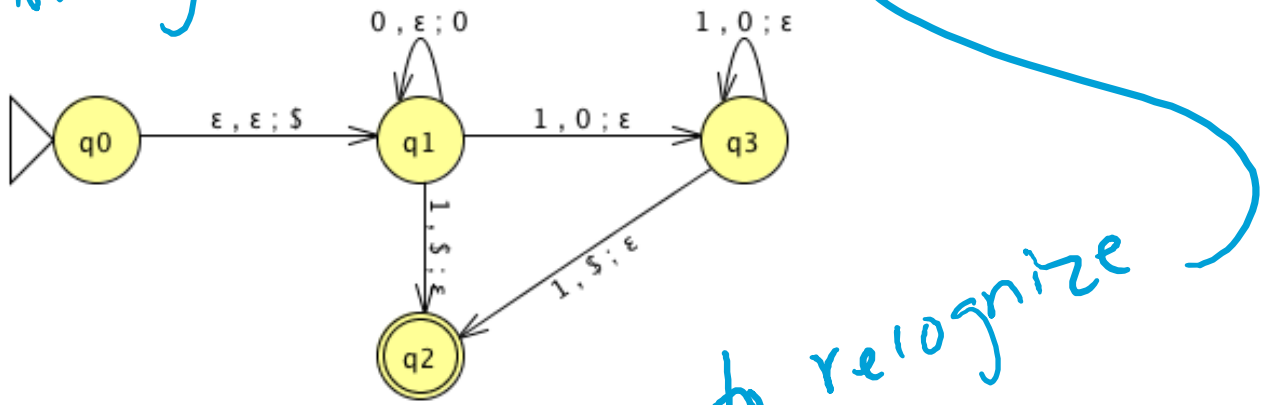
- add another state
- change transition when pop $
- add one more 0 to stack

# Designing/Tracing a PDA

BONUS EX:

$L = \{ 0^i 1^j \mid j \geq i \geq 0 \}$

Modify this




to recognize

# Example

$$L = \{ a^i b^j c^k \mid i=j \text{ or } i=k, \text{ with } i,j,k \geq 0 \}$$

Which of the following strings are **not** in L?

A. b
B. abc
C. abbcc
D. aabcc
E. I don't know.

# Designing a PDA

$$L = \{ a^i b^j c^k \mid i=j \text{ or } i=k, \text{ with } i,j,k \geq 0 \}$$

*Informal description of PDA:*

How would you design an algorithm that, given a string, decides if it is in this set?

- What information do you need to track?
- How much memory do you need?
- Are you using non-determinism?

# Designing a PDA

$$L = \{ a^i b^j c^k \mid i=j \text{ or } i=k, \text{ with } i,j,k \geq 0 \}$$

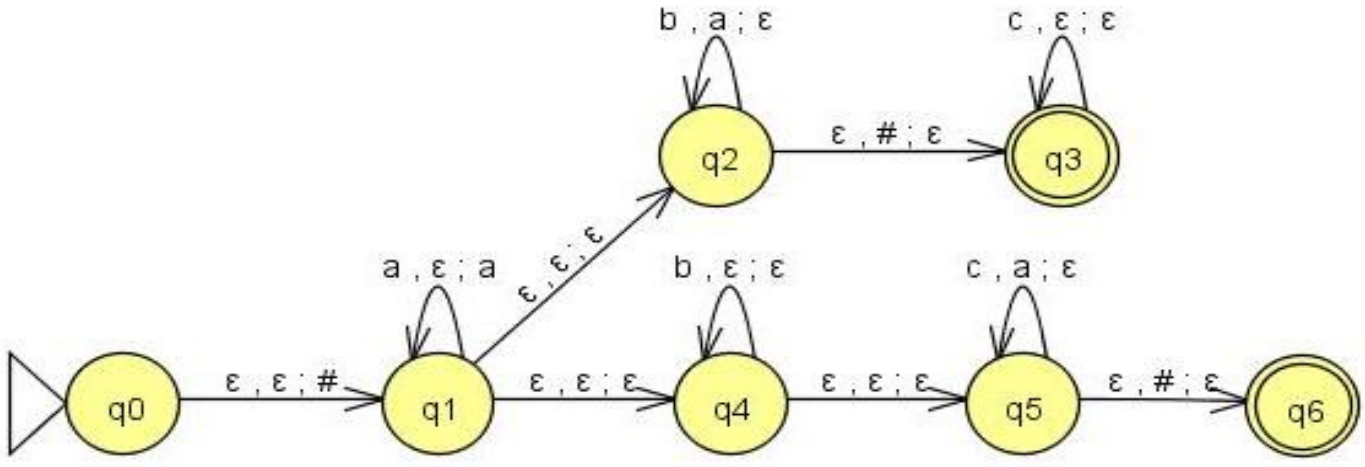
*Informal description of PDA:*

- The PDA pushes a $ to indicate the top of the stack, then starts reading a's, pushing each one on to the stack.
- The PDA guesses when it's reached the end of the a's and whether to match the number of a's to the number of b's or the number of c's.
- If trying to match number of b's with number of a's: PDA pops off a's for each b read. If there are more a's on the stack but no more b's being read, reject. When the end of the stack ($) is reached, the number of a's matches the number of b's. If this is the end of the input or if any number of c's is read at this point, accept; otherwise, reject.
- If trying to match the number of c's with number of a's: first read any number of b's without changing stack contents and then nondeterministically guess when to start reading c's. For each c read, pop one a off the stack. When the end of the stack ($) is reached the number of a's and c's so far match.

# Designing a PDA

$$L = \{ a^i b^j c^k \mid i=j \text{ or } i=k, \text{ with } i,j,k \geq 0 \}$$

*State diagram of PDA:*



# For next time

- Work on Group HW3 **due Saturday**

**Exam 1** one week from today

Pre class-reading for Friday: Examples 2.16, 2.18