CSE 311 Section 09

Models of Computation

Administrivia

Announcements & Reminders

- HW8
  - Due **Friday** 5/31 @ 11:59
- Final Exam
  - Monday, 6/3 12:30-2:20pm KNE120
- Final Review Session
  - Thursday, 5/30 5:00-7:00pm
- 390z Final Review Exam
  - 5/28 12:30 & 2:30 Sections (2:30 section has more room)
  - Practice exam and solutions will be posted on the course website after

Deterministic Finite Automata

Deterministic Finite Automata

- A DFA is a finite-state machine that accepts or rejects a given string of symbols, by running through a state sequence uniquely determined by the string.

- In other words:
    - Our machine is going to get a string as input. It will read one character at a time and update “its state.”
    - At every step, the machine thinks of itself as in one of the (finite number) vertices. When it reads the character, it follows the arrow labeled with that character to its next state.
    - Start at the “start state” (unlabeled, incoming arrow).
    - After you’ve read the last character, accept the string if and only if you’re in a “final state” (double circle).

- Every machine is defined with respect to an alphabet Σ
- Every state has exactly one outgoing edge for every character in Σ
- There is exactly one start state; can have as many accept states (aka final states) as you want – including none.

Problem 1 – DFAs, Stage 1

Construct DFAs to recognize each of the following languages.
Let Σ = {0, 1, 2, 3}.

b) All strings whose digits sum to an even number.

c) All strings whose digits sum to an odd number.

Work on this problem with the people around you.

Problem 1 – DFAs, Stage 1

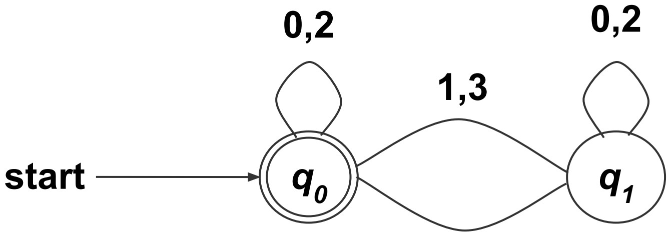
Let Σ = {0, 1, 2, 3}.

b) All strings whose digits sum to an even number.

Problem 1 – DFAs, Stage 1

Let Σ = {0, 1, 2, 3}.

b) All strings whose digits sum to an even number.



q_0: strings whose sum of digits is even
q_1: strings whose sum of digits is odd

Problem 1 – DFAs, Stage 1

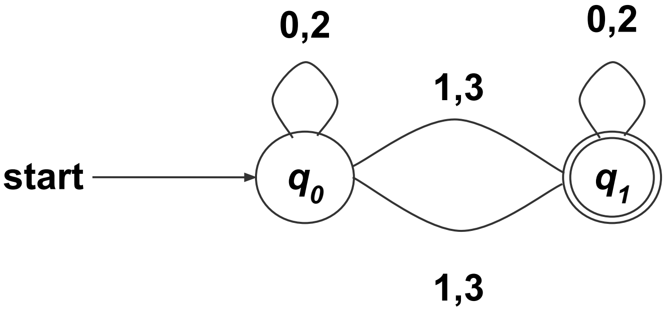
Let Σ = {0, 1, 2, 3}.

c) All strings whose digits sum to an odd number.

Problem 1 – DFAs, Stage 1

Let Σ = {0, 1, 2, 3}.

c) All strings whose digits sum to an odd number.



q_0: strings whose sum of digits is even

q_1: strings whose sum of digits is odd

DFA minimization

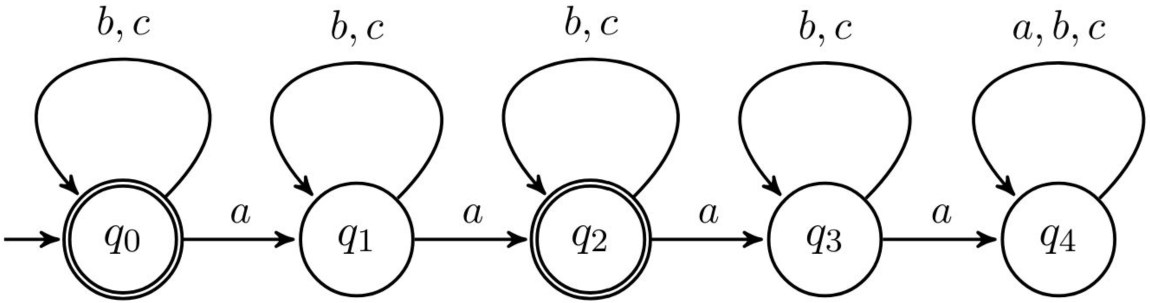
How does minimization work?

Reduce the number of states while maintaining the behavior (strings that the machine accepts)

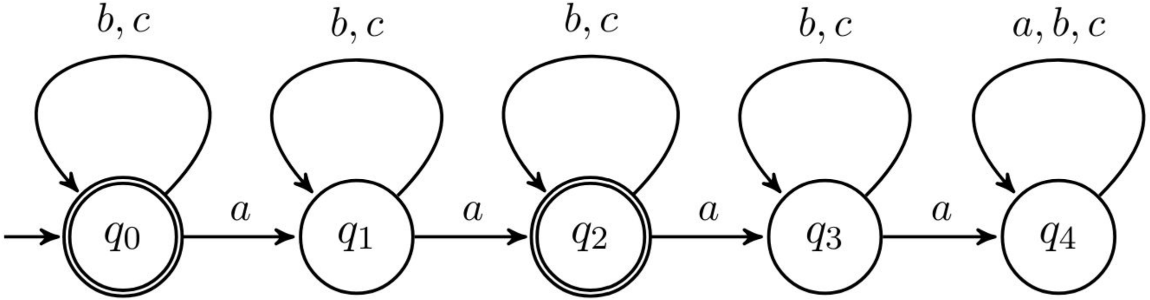
We will try to group equivalent states together so we know what states are safe to remove

Task 2:

For each step of the algorithm, write down the groups of states, which group was split in that step and the reason for splitting that group. At the end, write down the minimized DFA, with each state named by the set of states of the original machine that it represents

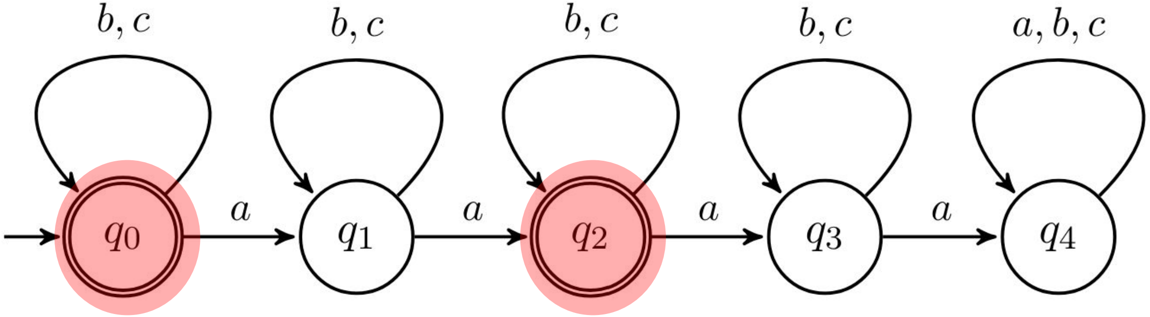


Task 2:



Step 1: Partition into final states and non-final states

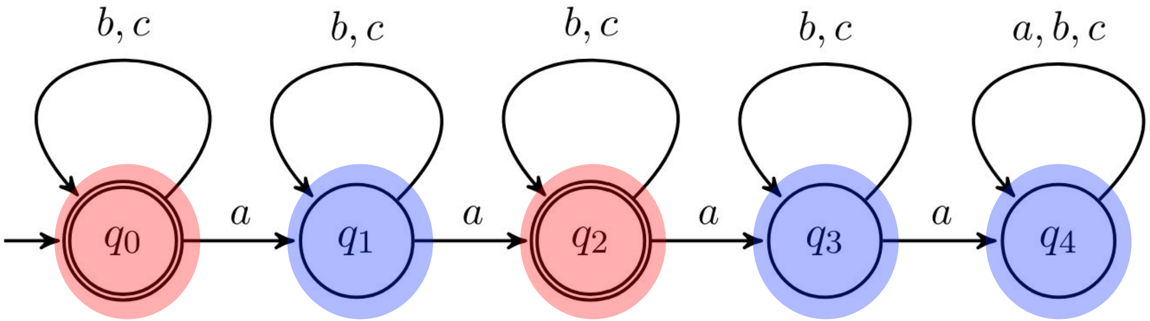
Task 2:



Step 1: Partition into final states and non-final states

Group 1: $\{q_0, q_2\}$

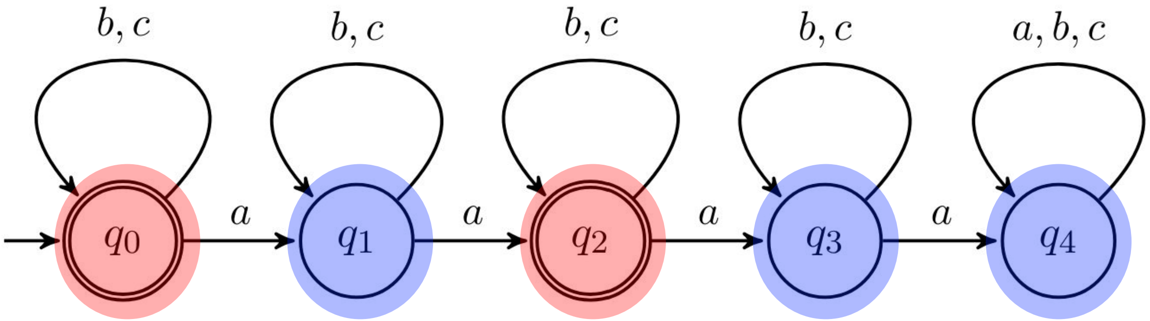
Task 2:



Step 1: Partition into final states and non-final states

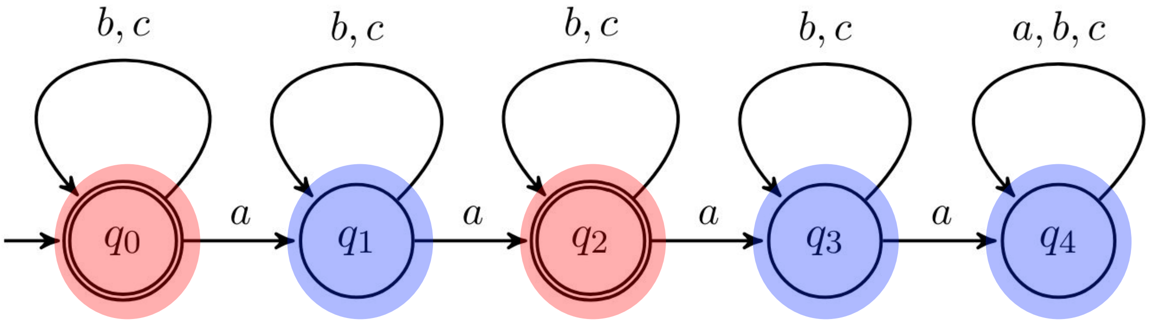
Group 1: $\{q_0, q_2\}$
Group 2: $\{q_1, q_3, q_4\}$

Task 2:



Step 2: Place states in groups based on transitioning to the same group from the same input symbol

Task 2:

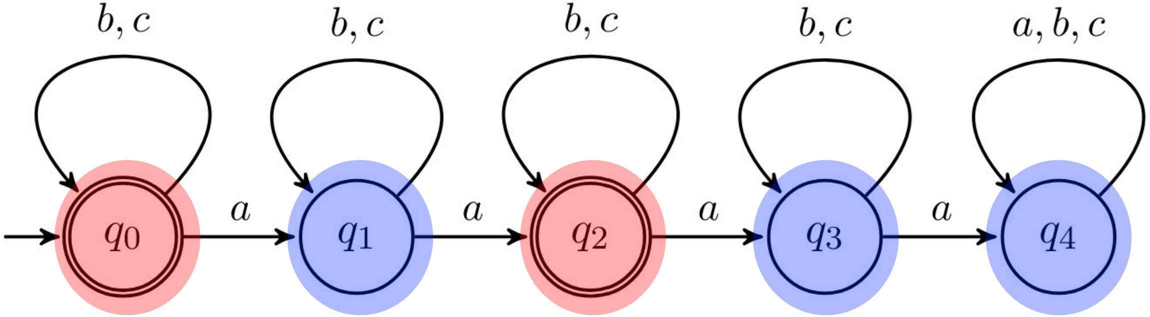


Step 2: Place states in groups based on transitioning to the same group from the same input symbol

{q_1, q_3 ,q_4 }

q_1 and q_3 transition to different groups on ‘a’

Task 2:

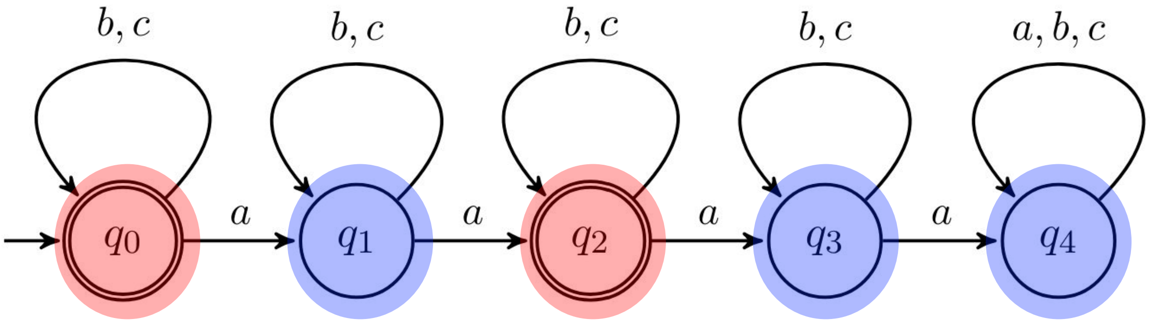


Step 2: Place states in groups based on transitioning to the same group from the same input symbol

Group 2: {q_1, q_3 ,q_4 }
q_1 and q_3 transition to different groups on ‘a’

Group 3: {q_1}

Task 2:



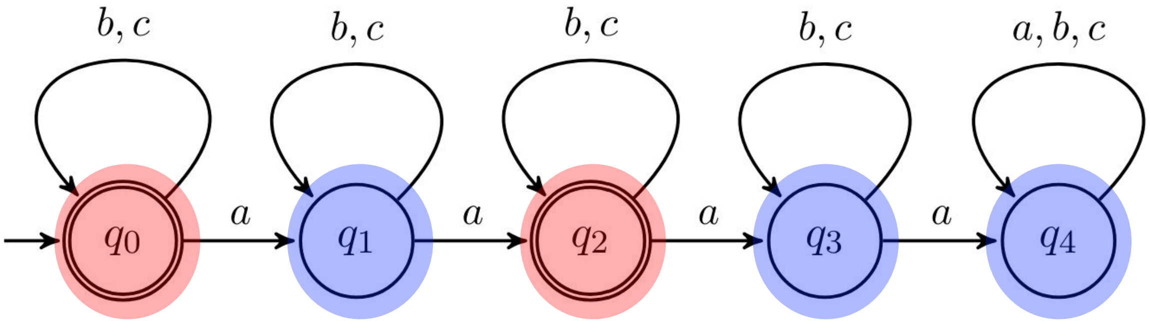
Step 2: Place states in groups based on transitioning to the same group from the same input symbol

Group 2: {q_1, q_3 ,q_4 }
q_1 and q_3 transition to different groups on ‘a’

Group 3: {q_1}

BUT q_3 and q_4 transition to the same groups on ‘a’ (blue group)

Task 2:



Step 2: Place states in groups based on transitioning to the same group from the same input symbol

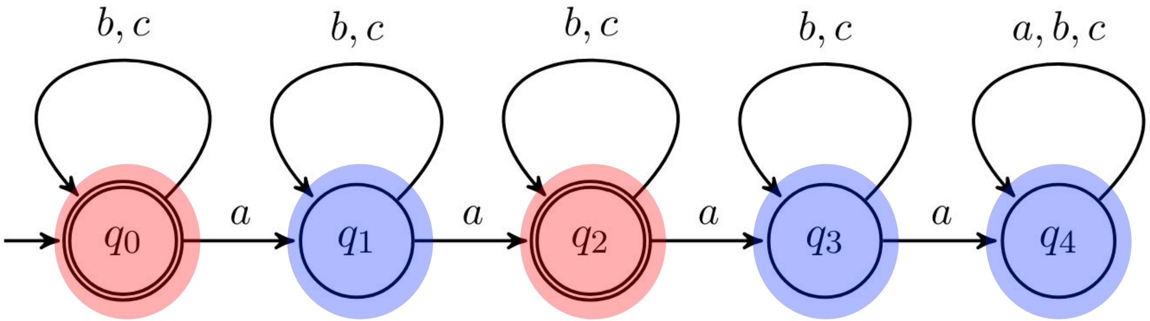
Group 2: {q_1, q_3 ,q_4 }
q_1 and q_3 transition to different groups on ‘a’

Group 3: {q_1}

BUT q_3 and q_4 transition to the same groups on ‘a’ (blue group)

Group 4: {q_3 ,q_4 }

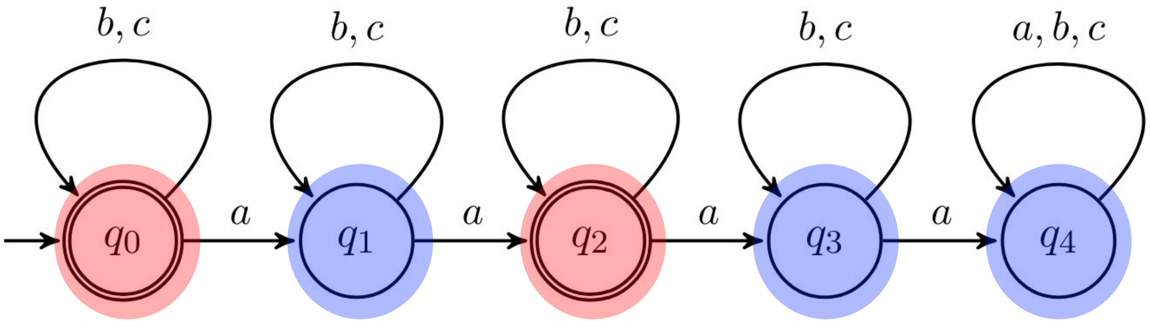
Task 2:



Step 2: Place states in groups based on transitioning to the same group from the same input symbol

From step 1: **{$\mathbf{q_0}$, $\mathbf{q_2}$}**
q_0, q_2 transition to <u>different groups for 'a'</u> (red group to blue group)

Task 2:

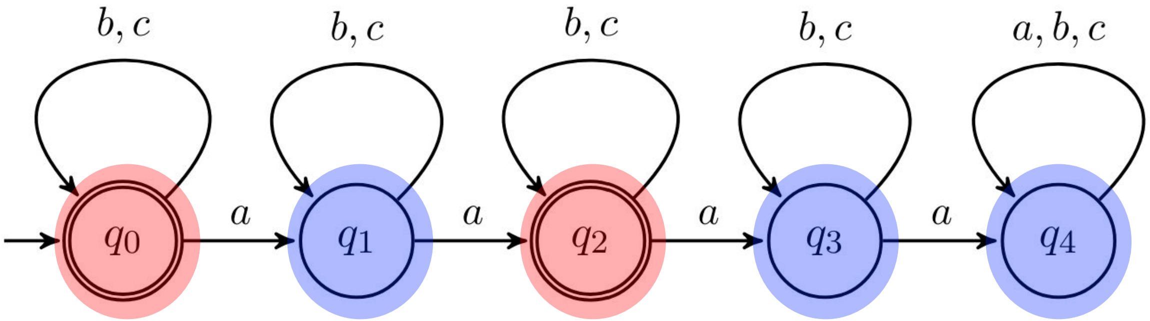


Step 2: Place states in groups based on transitioning to the same group from the same input symbol

From step 1: **{q_0, q_2}**
q_0, q_2 transition to different groups for ‘a’ (red group to blue group)

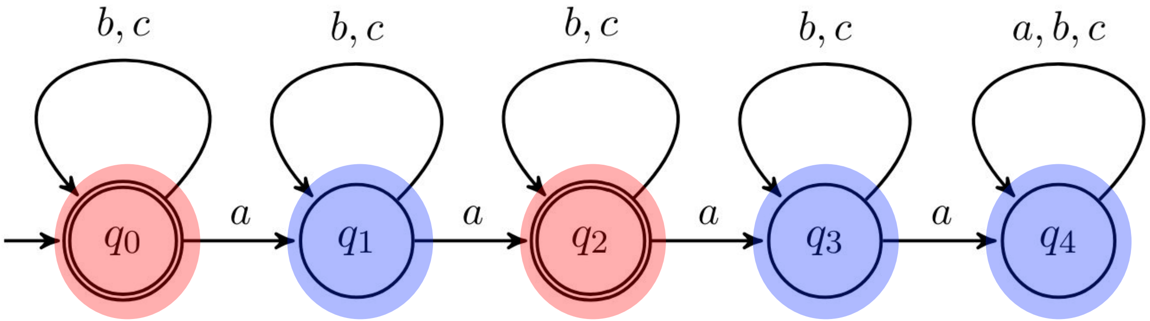
Group 1: {q_0}
Group 2: {q_2}

Task 2:



Step 2: Place states in groups based on transitioning to the same group from the same input symbol

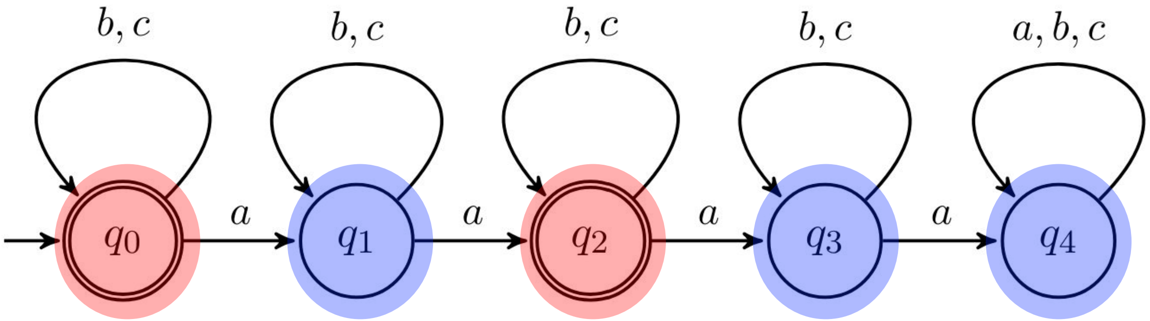
Task 2:



Step 3: Create the new minimized DFA with grouping

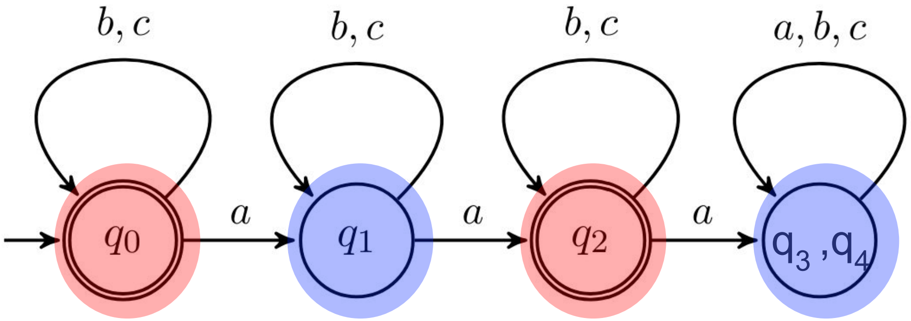
Group 1: $\{q_0\}$
Group 2: $\{q_2\}$
Group 3: $\{q_1\}$
Group 4: $\{q_3, q_4\}$

Task 2:



Step 3: Create the new minimized DFA with grouping

Group 1: $\{q_0\}$
Group 2: $\{q_2\}$
Group 3: $\{q_1\}$
Group 4: $\{q_3, q_4\}$



Nondeterministic Finite Automata

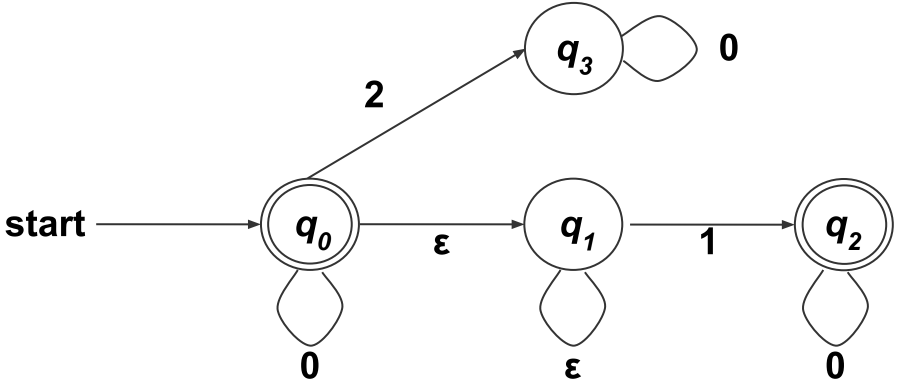
Nondeterministic Finite Automata

- Similar to DFAs, but with less restrictions.
  - From a given state, we’ll allow any number of outgoing edges labeled with a given character. (In a DFA, we have only 1 outgoing edge labeled with each character).
  - The machine can follow any of them.
  - We’ll have edges labeled with “ε” – the machine (optionally) can follow one of those without reading another character from the input.
  - If we “get stuck” i.e. the next character is a and there’s no transition leaving our state labeled a, the computation dies.

- An NFA still has exactly one start state and any number of final states.
- The NFA accepts x if there is some path from a start state to a final state labeled with x.
- From a state, you can have 0,1, or many outgoing arrows labeled with a single character. You can choose any of them to build the required path.

Task 3 – NFAs

a) What language does the following NFA accept?

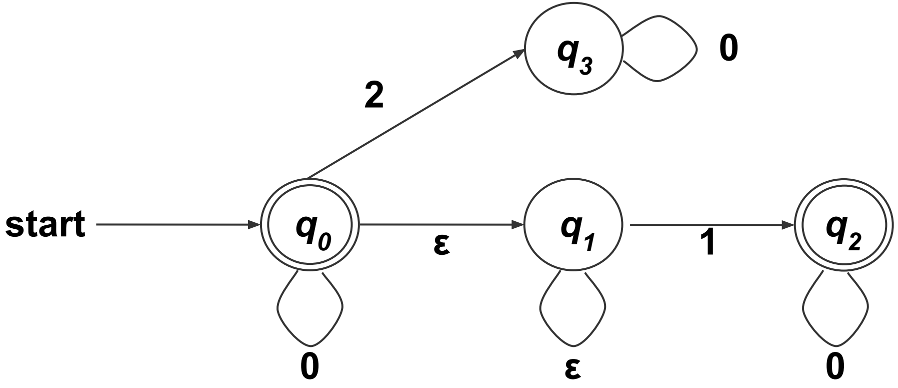


b) Create an NFA for the language "all binary strings that have a 1 as one of the last three digits".

Work on this problem with the people around you.

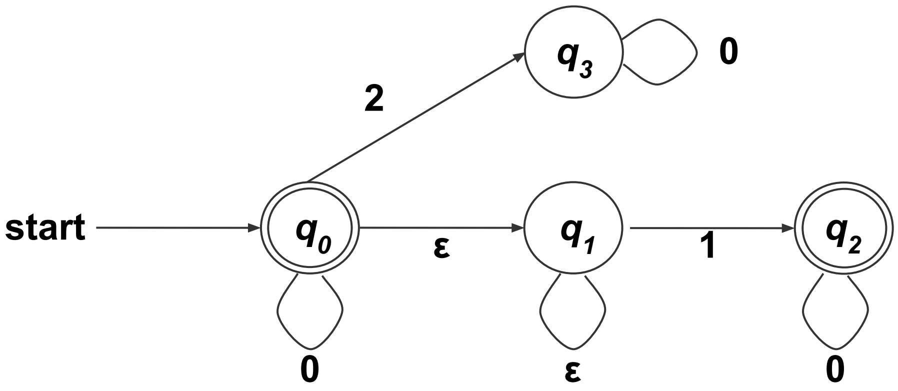
Task 3 – NFAs

a) What language does the following NFA accept?



Task 3 – NFAs

a) What language does the following NFA accept?



All strings of only 0’s and 1’s, not containing more than one 1.

Task 3– NFAs

b) Create an NFA for the language “all binary strings that have a 1 as one of the last three digits”.

Task 3– NFAs

b) Create an NFA for the language “all binary strings that have a 1 as one of the last three digits”.

Consider generating regex for:

All binary strings that have a 1 at the end

All binary strings that have a 1 at the second position from the end

All binary strings that have a 1 at the third position from the end

Task 3– NFAs

b) Create an NFA for the language “all binary strings that have a 1 as one of the last three digits”.

Consider generating regex for:

All binary strings that have a 1 at the end

(0 U 1)* **1**

All binary strings that have a 1 at the second position from the end

All binary strings that have a 1 at the third position from the end

Task 3– NFAs

b) Create an NFA for the language “all binary strings that have a 1 as one of the last three digits”.

Consider generating regex for:

All binary strings that have a 1 at the end

(0 U 1)* **1**

All binary strings that have a 1 at the second position from the end

(0 U 1)* **1** (0 U 1)

All binary strings that have a 1 at the third position from the end

Task 3– NFAs

b) Create an NFA for the language “all binary strings that have a 1 as one of the last three digits”.

Consider generating regex for:

All binary strings that have a 1 at the end

(0 U 1)* **1**

All binary strings that have a 1 at the second position from the end

(0 U 1)* **1** (0 U 1)

All binary strings that have a 1 at the third position from the end

(0 U 1)* **1** (0 U 1) (0 U 1)

Task 3 – NFAs

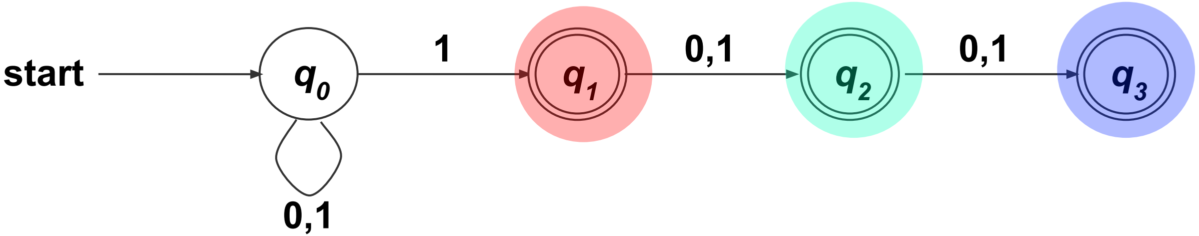
b) Create an NFA for the language “all binary strings that have a 1 as one of the last three digits”.

How can we use this to construct the NFA?

Task 3 – NFAs

b) Create an NFA for the language “all binary strings that have a 1 as one of the last three digits”.

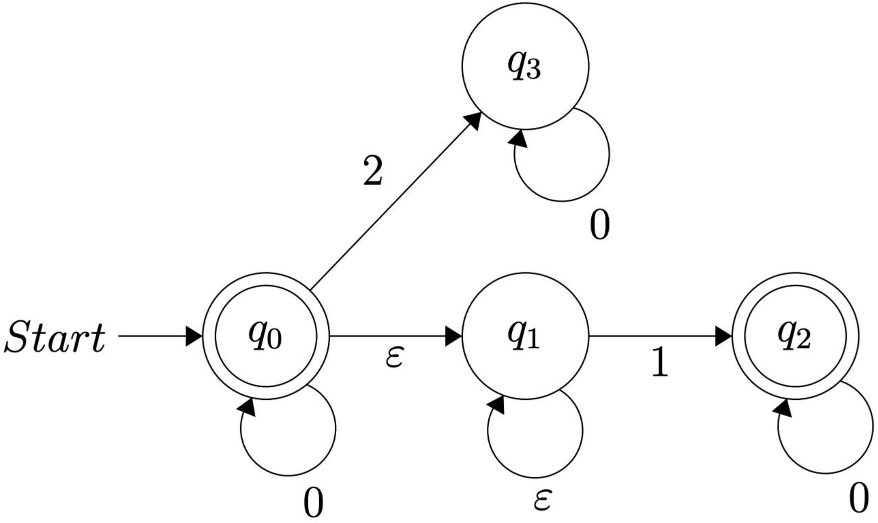
How can we use this to construct the NFA?



NFA to DFA

Task 5

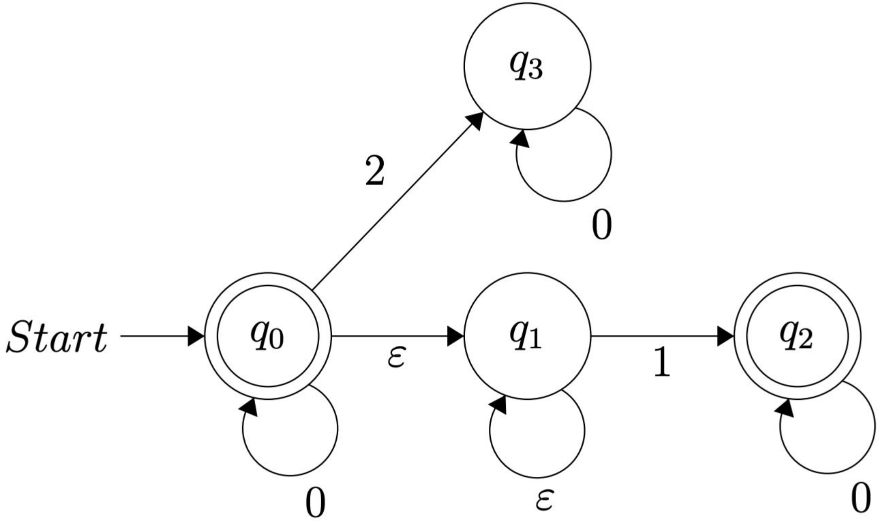
Convert the following NFA to a DFA of the same language



Task 5 - Documenting states in a table

Where can we get from the start state (q0) without reading input?

(DFA) state	0-transition	1-transition	2-transition

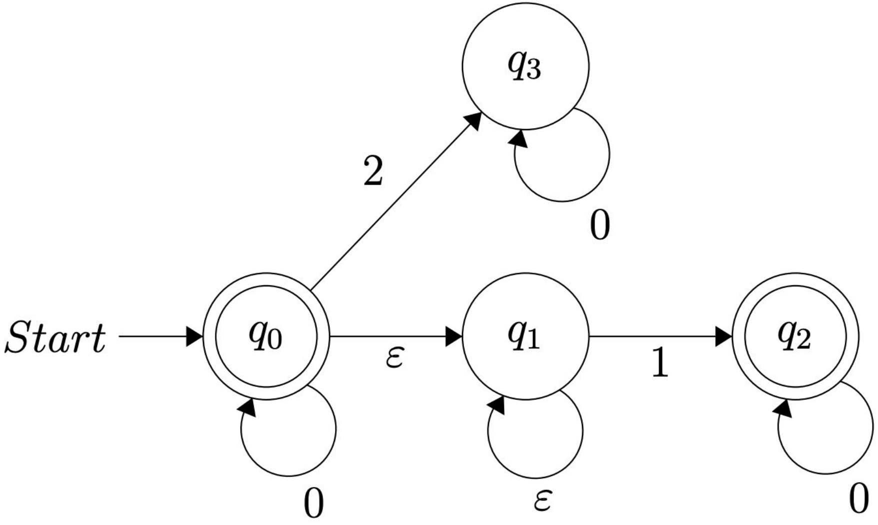


Task 5 - Documenting states in a table

Where can we get from the start state (q0) without reading input?

- q0
- take ε to q1

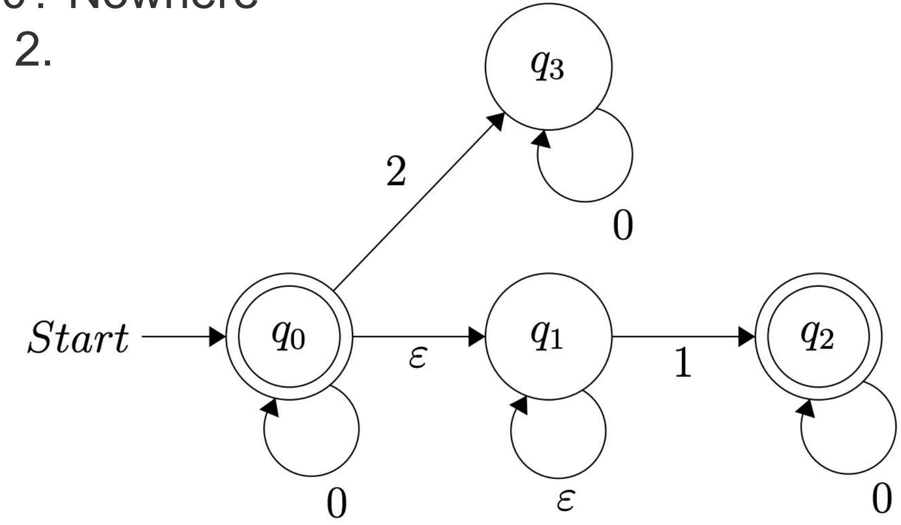
(DFA) state	0-transition	1-transition	2-transition
q0, q1			



Task 5 - Documenting states in a table

From q0, where can we get reading in 0? q0 and q1
From q1, where can we get reading in 0? Nowhere
Repeat for reading in 1, and reading in 2.
Then fill in the state table.

(DFA) state	0-transition	1-transition	2-transition
q0, q1	q0, q1	q2	q3

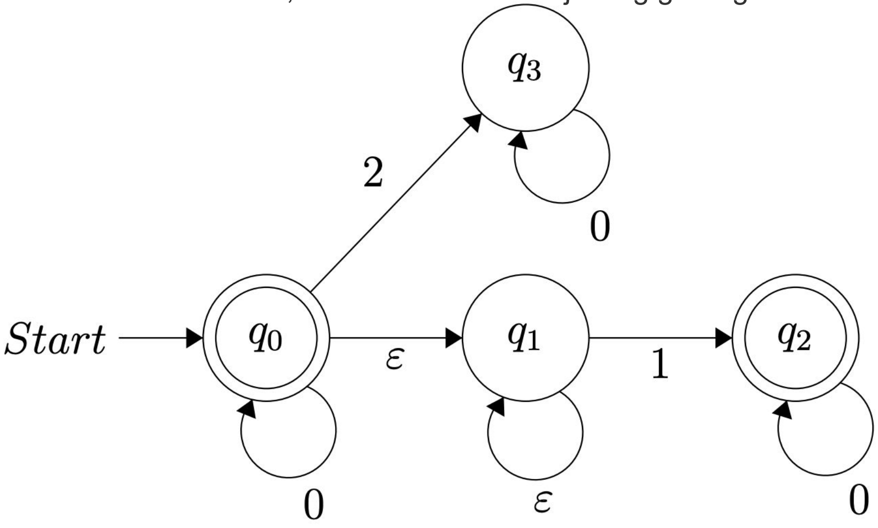


Task 5 - Documenting states in a table

We need to define the other states (q2, q3) in the table now.

- When there's no defined transition for an input, the computation dies in the NFA, so we send it to a rejecting garbage state ∅.

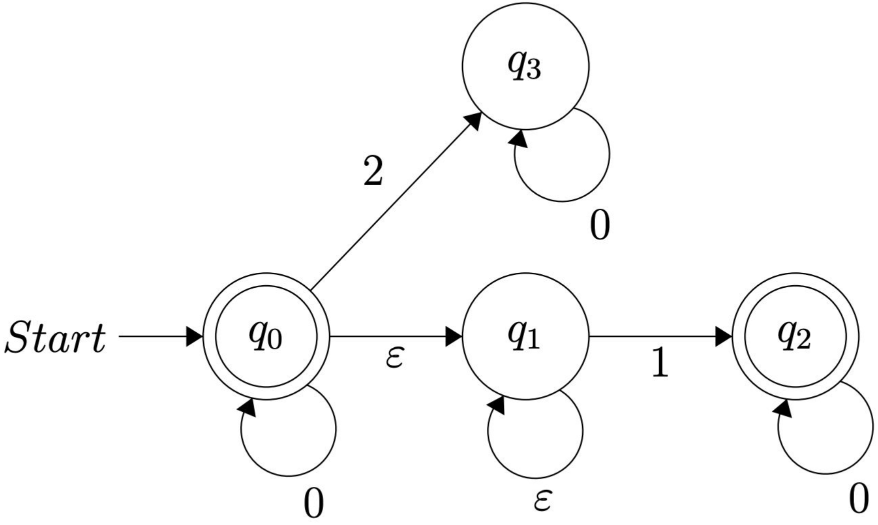
(DFA) state	0-transition	1-transition	2-transition
q0, q1	q0, q1	q2	q3
q2	q2	∅	∅
q3			



Task 5 - Documenting states in a table

We need to define the other states (q2 and q3) in the table now.

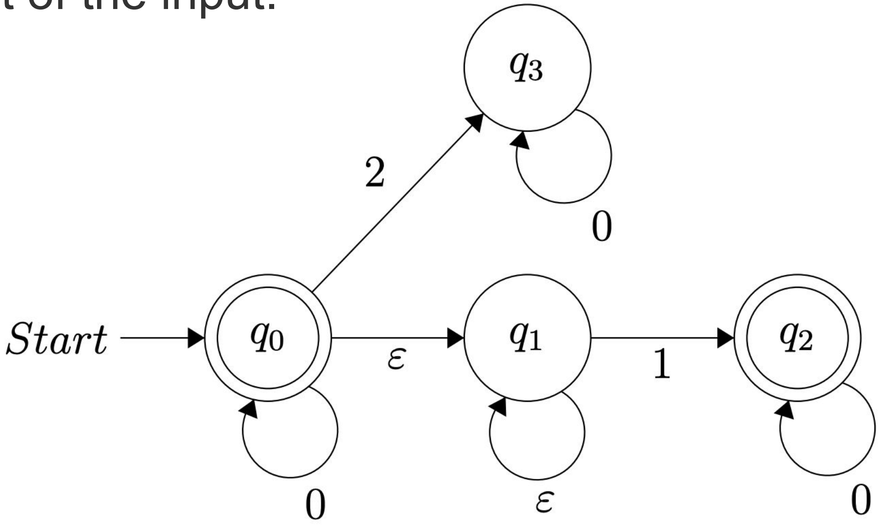
(DFA) state	0-transition	1-transition	2-transition
q0, q1	q0, q1	q2	q3
q2	q2	∅	∅
q3	q3	∅	∅



Task 5 - Documenting states in a table

The only state we have left to define is the “garbage” state. This one always is rejecting regardless of the rest of the input.

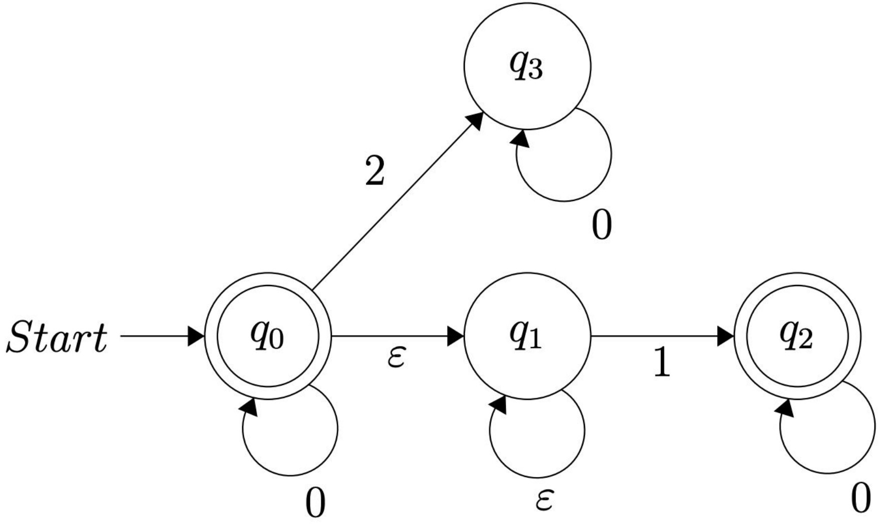
(DFA) state	0-transition	1-transition	2-transition
q0, q1	q0, q1	q2	q3
q2	q2	∅	∅
q3	q3	∅	∅
∅	∅	∅	∅



Task 5 - Documenting states in a table

Which states are accepting? (The rest are rejecting)

(DFA) state	0-transition	1-transition	2-transition
q0, q1	q0, q1	q2	q3
q2	q2	∅	∅
q3	q3	∅	∅
∅	∅	∅	∅

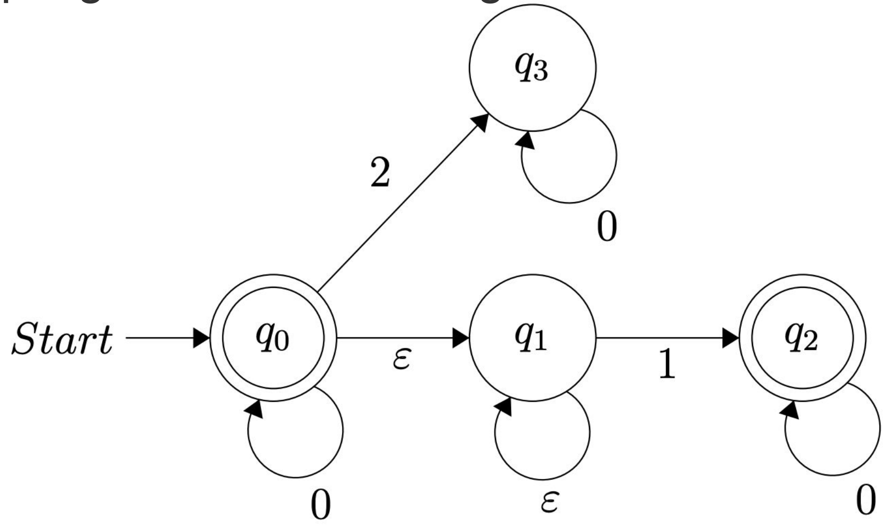


Task 5 - Documenting states in a table

Which states are accepting? (The rest are rejecting)

- All DFA states that include an accepting state from the original NFA.

(DFA) state	0-transition	1-transition	2-transition
q0, q1	q0, q1	q2	q3
q2	q2	∅	∅
q3	q3	∅	∅
∅	∅	∅	∅

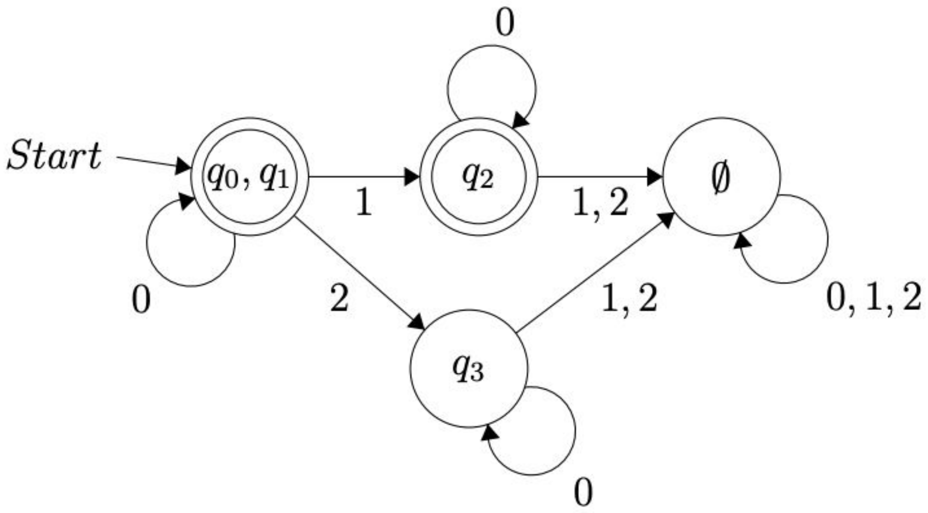


Now try drawing out the DFA using the state table.

- Don’t forget a start state!

Task 5

(DFA) state	0-transition	1-transition	2-transition
q0, q1	q0, q1	q2	q3
q2	q2	∅	∅
q3	q3	∅	∅
∅	∅	∅	∅



That’s All, Folks!

Thanks for coming to section this week!
Any questions?

By: Aruna & Alysa