Regular Expressions

and grep

and sed

and more?

Regular Expressions

Globs = Regular Expressions

Differences

Regular Expressions

- Patterns that match against certain strings
- Different from globs
- Compatible with many applications
- But why are they called regular expressions?
  - For interesting theoretical reasons
  - That you will learn later

Example: Phone numbers

- Multiple possible strings
  - 123-456-7890
  - 1234567890
  - 456-789-1234
- But the formats follow a few patterns
  - ###-###-###
  - ##########

Solution: Regular expressions

- Create a pattern that specifies which strings to match
- `(\d{3}-?){2}\d{4}` – matches a phone number

Examples

- `gpi` – matches "gpi"
- `[hjkl]` – matches "h", "j", "k", and "l"
- `07-?131` – matches "07131" and "07-131"
- `item[1-3]` – matches "item1", "item2", "item3"
- `codes*` – matches "code", "codes", "codess", "codesssss", etc.

Parts of a regular expression

- Normal characters
  - `gpi – matches "gpi"`
- Quantifiers
  - repeating* – matches "repeatin", "repeating", " repeatinggg", etc.
  - ab{1,3} – matches "ab", "abb", or "abbb"
- Character classes
  - [hjkl] – matches "h", "j", "k", "l"
  - \d – matches and digit
  - . – matches any character
- Use parentheses for grouping

Quantifiers

Quantifier	Matches
`a?`	Zero or one
`a*`	Zero or more
`a+`	One or more
`a{3}`	Exactly 3
`a{3,}`	3 or more
`a{3,6}`	Between 3 and 6

Character classes

Class	Matches
`[abc]`	a or b or c
`[^abc]`	not any of a, b, c
`[a-z]`	A lowercase letter
`[^A-Za-z]`	Not a letter
`\s`	Whitespace
`\d`	Digit
`.`	Any single character

Example

```
(\d{3}-?){2}\d{4}
```

Matches any digit

Example

```
(\d{3}-?){2}\d{4}
```

Matches any 3 digits

Example

```
(\d{3}-?){2}\d{4}
```

Matches an optional hyphen

Example

`(\d{3}-?){2}\d{4}`

Matches 2 groups of 3 digits

Ex:
123-456-
123456-
123456

Example

```
(\d{3}-?){2}\d{4}
```

Matches 2 groups of 3 digits,
then 4 more digits

Special sequences

- $ - End of string
- ^ - Start of string
- Parentheses for grouping

Cheat sheet

- a* – Matches zero or more times
- a? – Matches one or zero times
- a{3} – Matches three times
- . – Matches any single character
- [a-z0-9] – Matches a digit or lowercase character
- [^xy] – Matches anything other than x and y.
- ^ - Matches start of string
- $ - Matches end of string

Quiz!

Matches	Regex
ababab or **abab**	`abab(ab)? or (ab){2,3}`
ab any number of times	`(ab)*`
[any letter][any number] ex: A4	`[A-Za-z]\d`
example.com website.com etc.	`[a-z]*\.com`

Regex vs Globs and ranges

Regex	Glob/Range equivalent
`.`	`?`
`file[1-7]\.txt`	`file{1..7}.txt`
`.*`	`*`
`(ab)*`	`Not possible`

Grep

- Search files and directories using regular expressions!
- Prints lines that include a match
- Name comes from g/re/p command in the UNIX text editor ed
- **`$ grep 'evidence' largefile.txt`**
  - Searches largefile.txt for "evidence".
- **`$ grep -r 'secrets' path/to/directory`**
  - Searches recursively for "secrets".

Sed (look familiar????)

- Stands for "stream editor"
- Can perform find and replace on a file
- sed 's/find/replace/g' path/to/file
  - Prints result of replacement to the command line, leaving input untouched
- sed -i 's/find/replace/g' path/to/file
  - "In place"
  - Edits the file

How does grep work?

- It seems like some guessing is necessary
  - Imagine matching "abc" against a?b?c?abc
  - Lots of guessing would be exponential time.
- But grep is fast
  - For deep theoretical reasons. Involving finite state machines.

<Extra Content>

Super Stylish TA Merch

```
FgrUeNatCprTa
ctIicOaNliSdA
eaRsfEoBrcoAm
puSteHrsSciCe
nRtIiPsTtSs<3
 :%s/[A-Z]//g
```

Super Stylish TA Merch

FgrUeNatCprTa

ctIicOaNliSdA

eaRsfEoBrcoAm

puSteHrsSciCe

nRtIiPsTtSs<3

:%s/[A-Z]//g

Super Stylish TA Merch

greatpra

cticalid

easforcom

puterscie

ntists<3

:%s/[A-Z]//g

Super Stylish TA Merch

great practical

ideas for

computer scientists

<3

Super Stylish TA Merch

FgrUeNatCprTa

ctIicOaNliSdA

eaRsfEoBrcoAm

puSteHrsSciCe

nRtIiPsTtSs<3

:%s/[a-z]//g

Super Stylish TA Merch

FgrUeNatCprTa

ctIicOaNliSdA

eaRsfEoBrcoAm

puSteHrsSciCe

nRtIiPsTtSs<3

:%s/[a-z]//g

Super Stylish TA Merch

FUNCT

IONSA

REBA

SHSC

RIPTS<3

:%s/[a-z]//g

Super Stylish TA Merch

FUNCTIONS

ARE

BASH

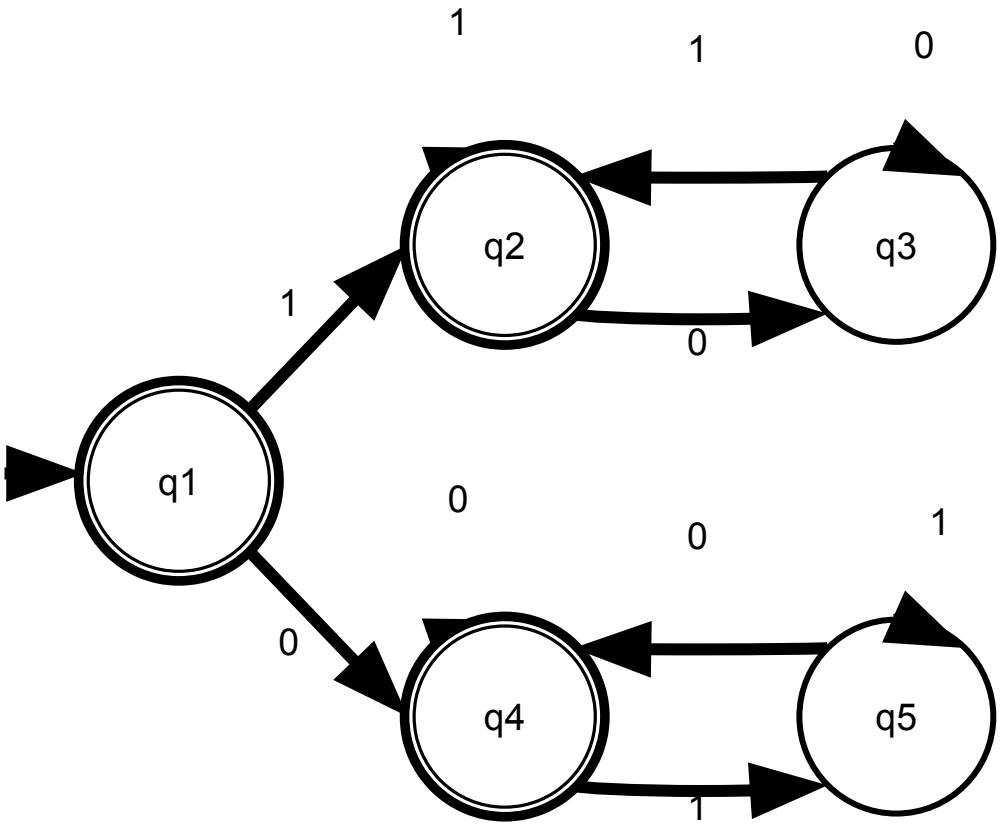
SCRIPTS

<3

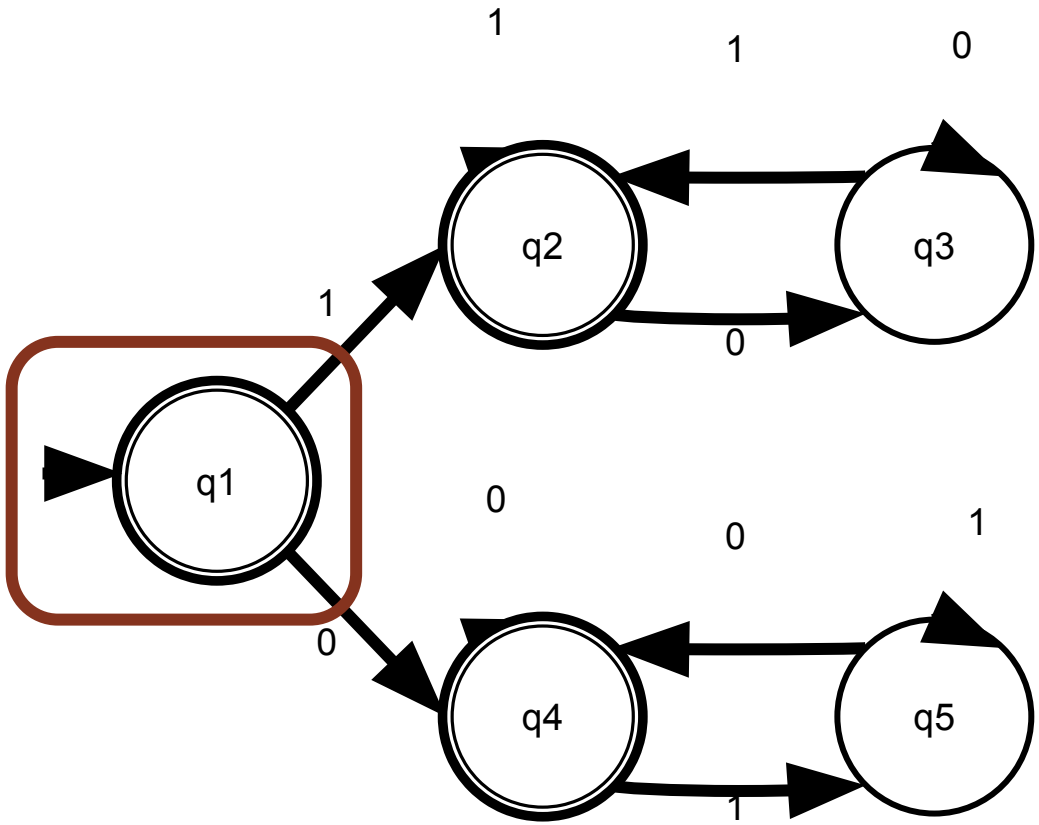
DFA = Deterministic Finite-state Automaton

DFA $= (Q, \Sigma, \delta, q_0, F)$

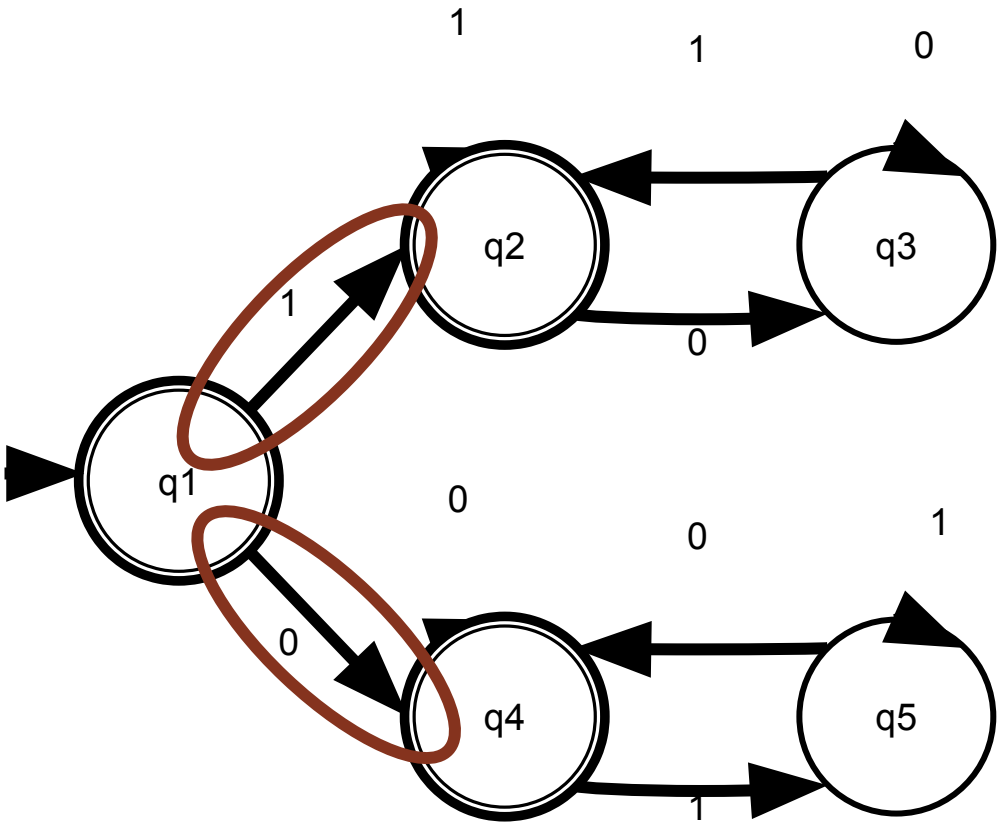
DFA =



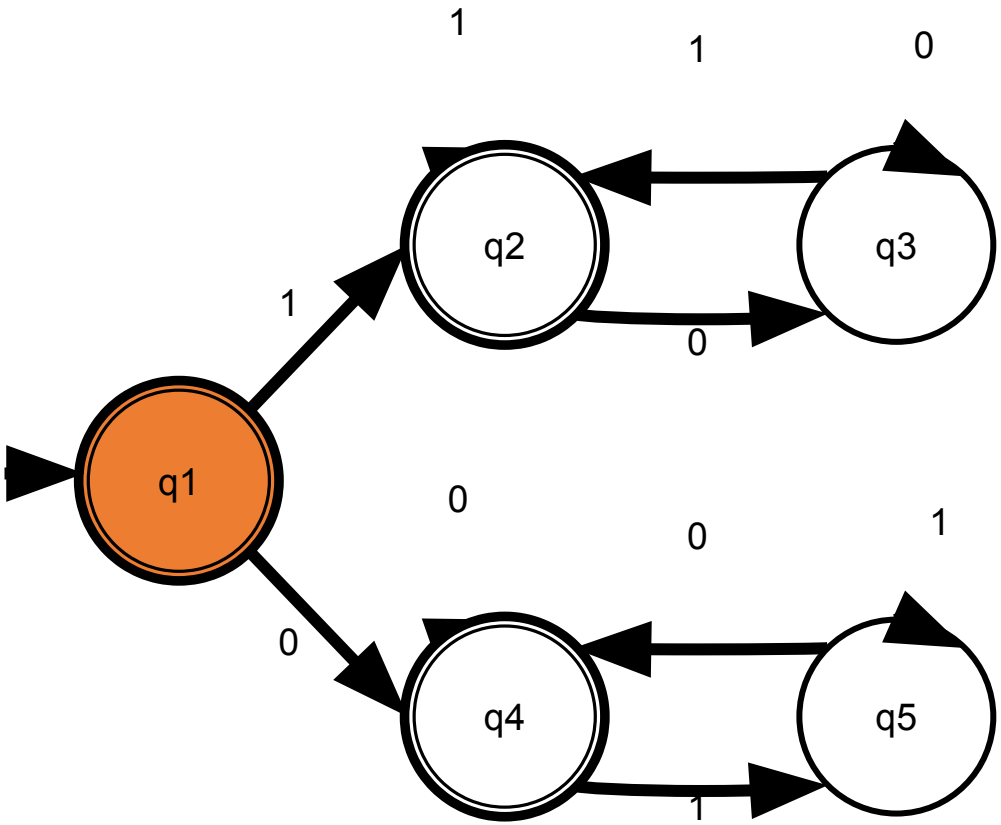
DFA =



DFA =

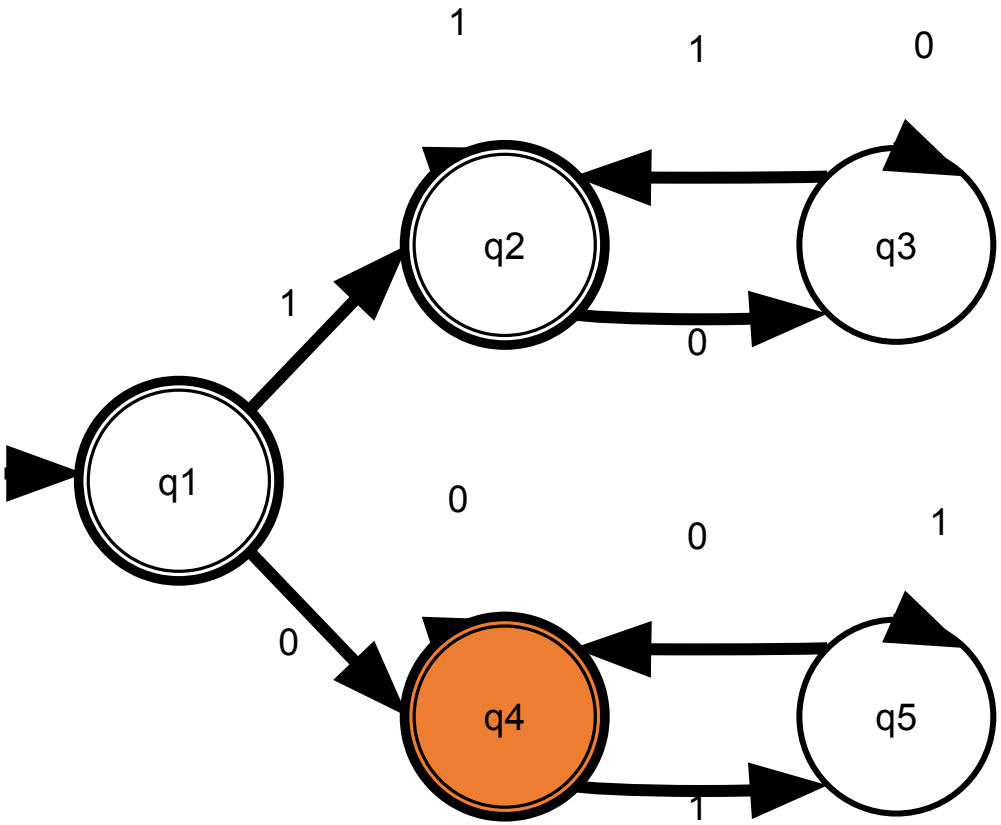


DFA =



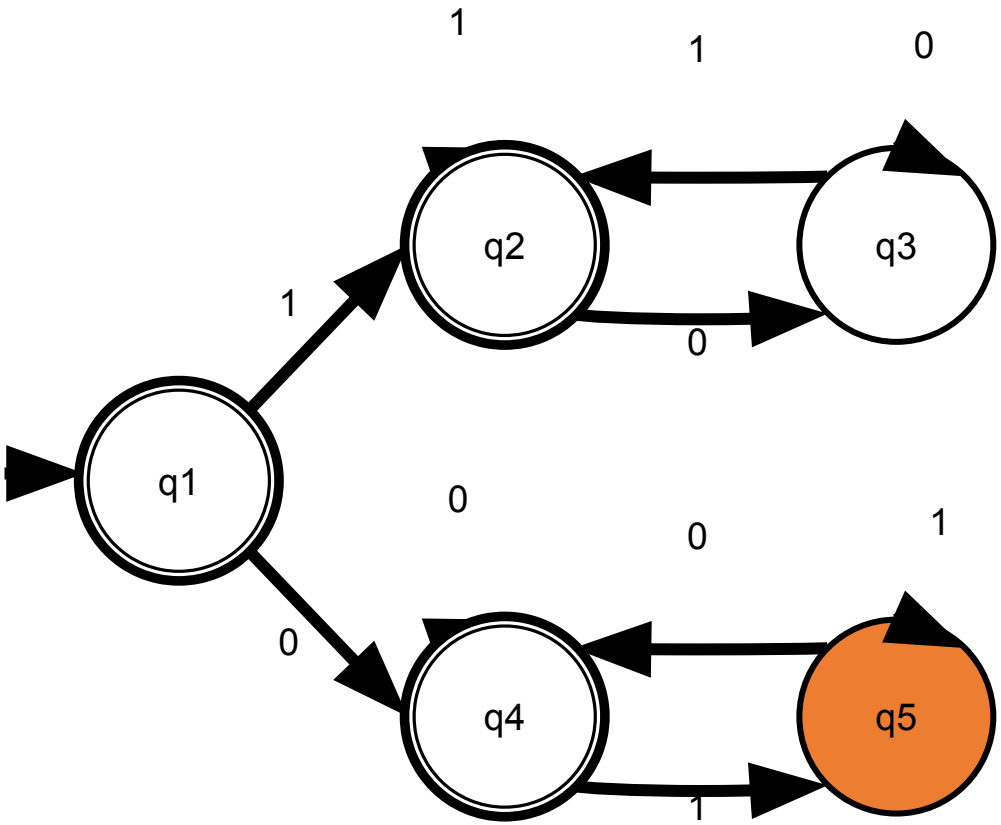
0100

DFA =



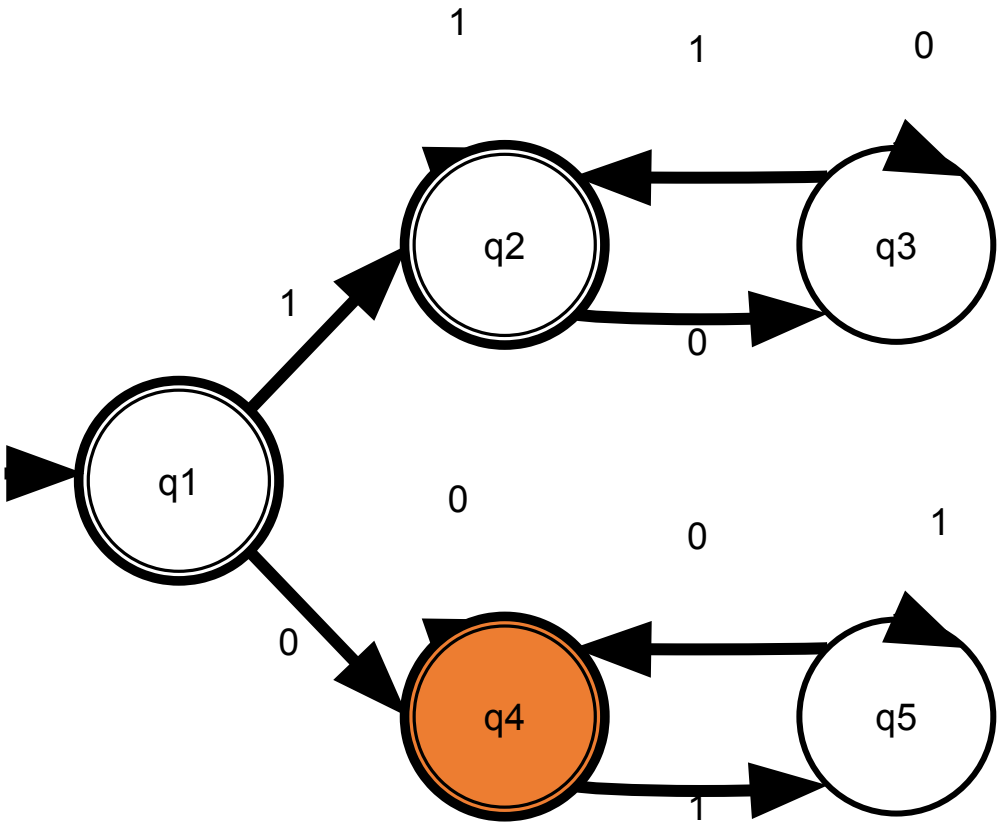
0101

DFA =



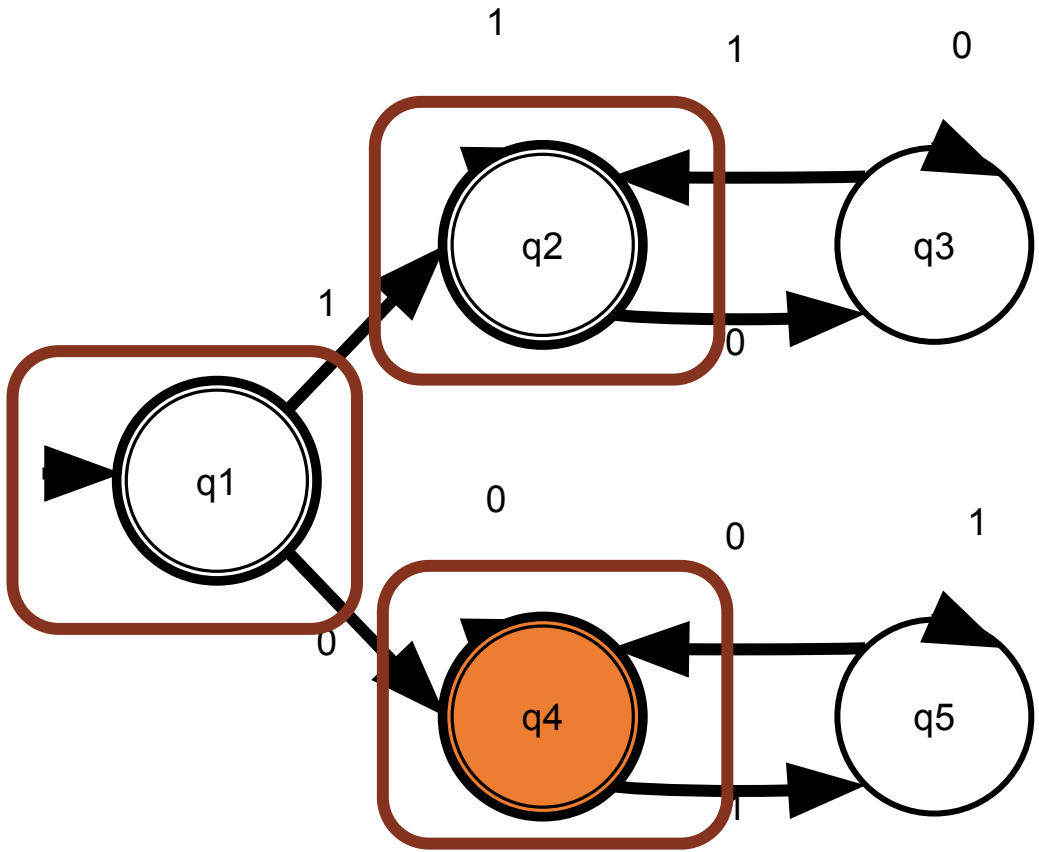
0100

DFA =



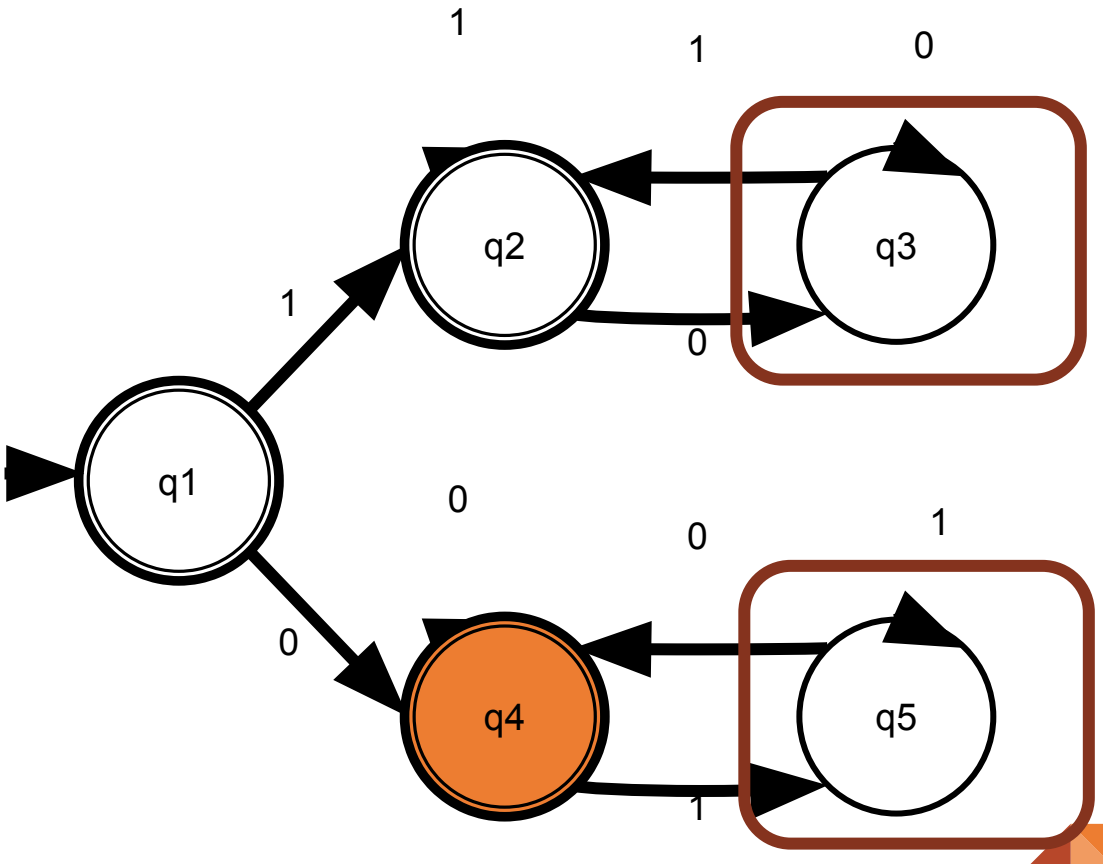
0100

DFA =



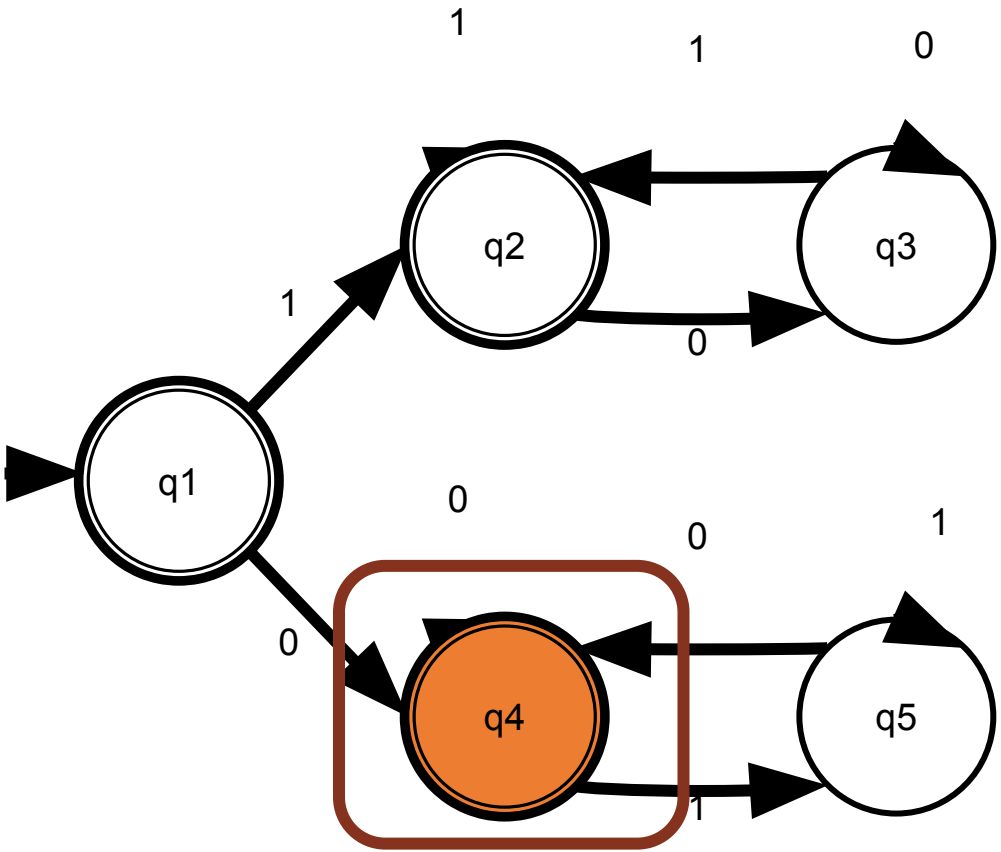
1001

DFA =



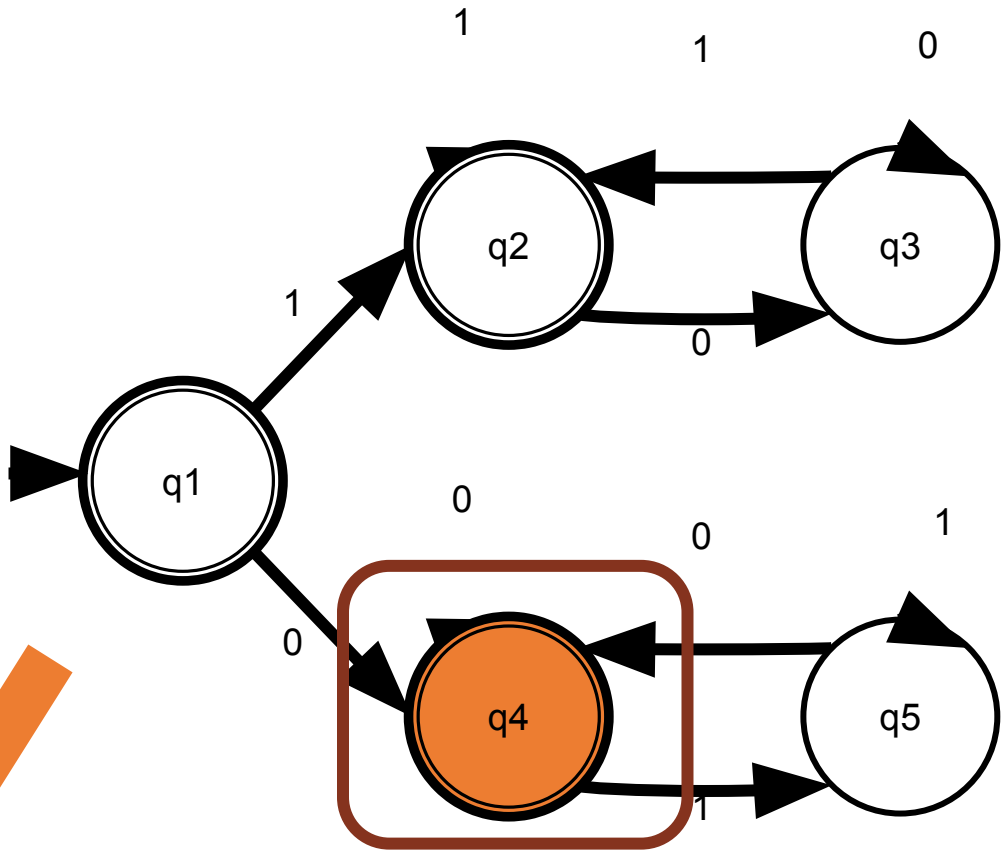
1001

DFA =



1001

DFA =

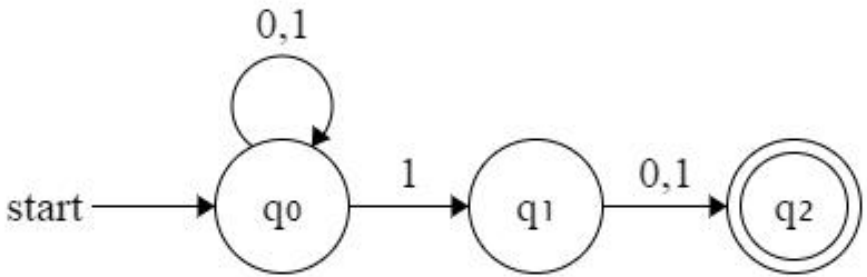


1001

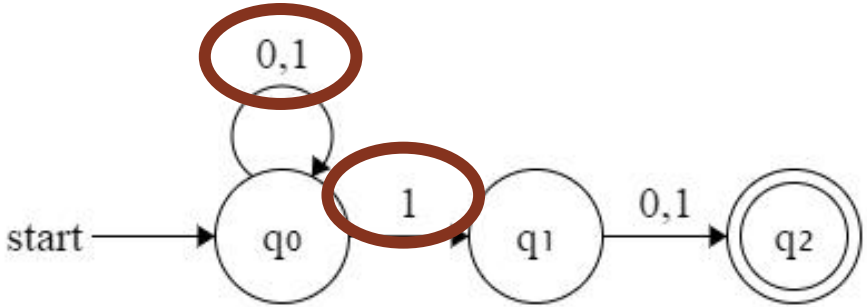

Regex
Efficiently Converted
NFA

NFA = Nondeterministic Finite-state Automaton

NFA =



NFA =



- Accept if there exists any path to an accepting state
- Computed by updating the set of reachable states

grep

</Extra Content>

Just one more thing…

If we call the states “slides” and the transitions “hyperlinks”…

Cheat sheet

- a* – Matches zero or more times
- a? – Matches one or zero times
- a{3} – Matches three times
- . – Matches any single character
- [a-z0-9] – Matches a digit or lowercase character
- [^xy] – Matches anything other than x and y.
- ^ - Matches start of string
- $ - Matches end of string

Bash scripting summary

- Bash scripts end in a .sh extension
- Always start with a shebang
  - `#!/usr/bin/env bash`
- Add permissions with **`chmod +x script.sh`**

Lab pro tips

- Lab is up! zombielab
- Be careful with escaping correctly. Both bash and regex have characters that must be escaped
- Don't forget to do **chmod +x script.sh** and add **#!/usr/bin/env bash**
- scottylabs.org/wdw/register