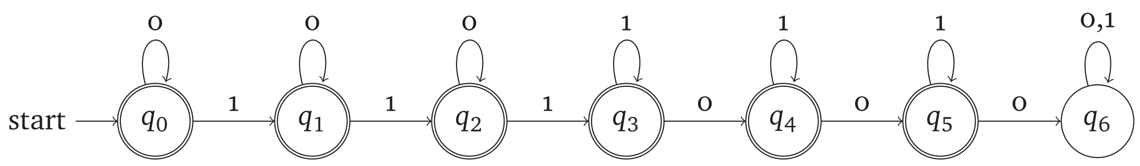
Problem type 1:

Provide the regular expression that describes the following NFA/DFA:

(See variants below)

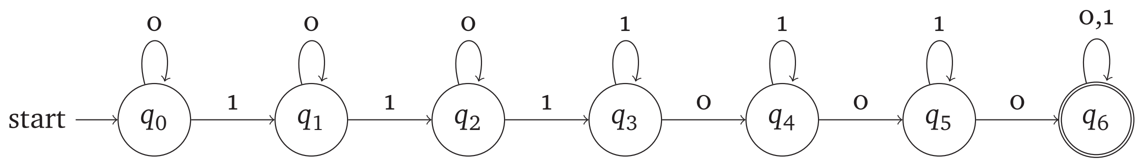
There is not enough time to go through Thompson's algorithm and such. You should simply attempt to look at the language that the DFA/NFA represents and write the regular expression for that.

a. **BYE**



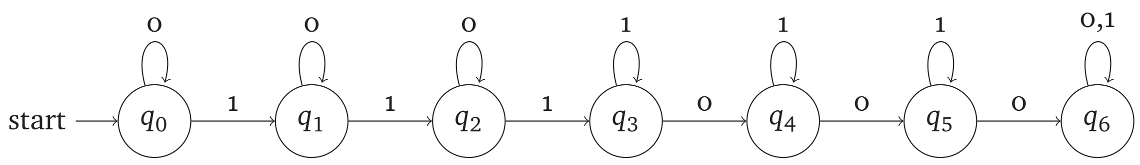
Solution:
All strings with*out* the subsequence **111000**. ■

b. **BYH**



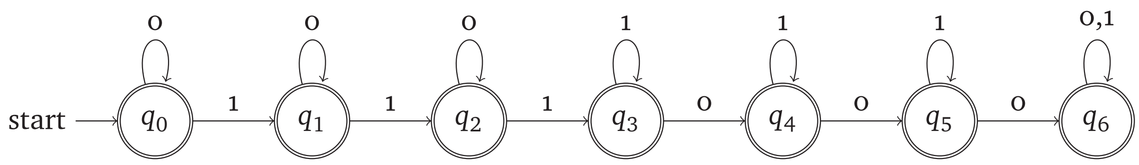
Solution:
All strings with the subsequence **111000**. ■

c. **BYC**



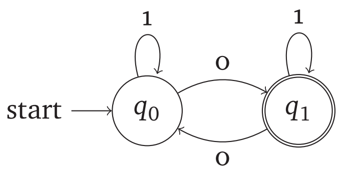
Solution:
Initially, I'm sure you're tempted to guess something with the subsequence **111000**. But wait a minute, where's the accept state? No accept state? Look like the regular expression $= \emptyset$. ■

d. **BYG**



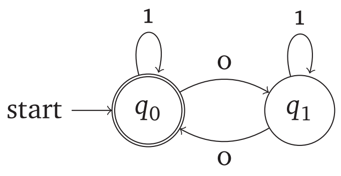
Solution:
Initially, I'm sure you're tempted to guess something with the subsequence **111000**. But wait a minute, they're all accept states? And it's a DFA? Look like the regular expression $= \Sigma^*$. ■

e. **BYB**



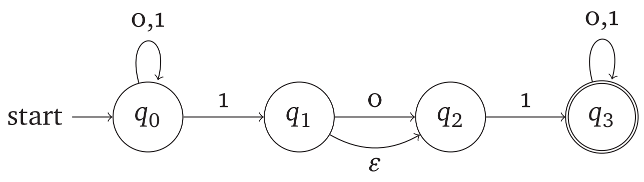
Solution:
$1^*0(01^*0+1)^*$ ■

f. **BYF**



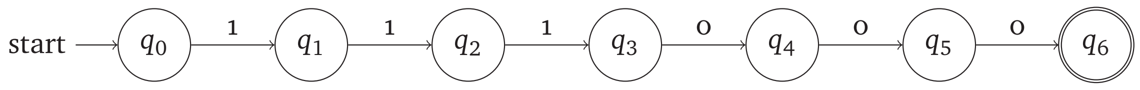
Solution:
$(01^*0+1)^*$ ■

g. **BYA**



Solution:
$(\mathtt{0}+\mathtt{1})^*\,\mathtt{1}\,(\varepsilon+\mathtt{0})\,\mathtt{1}\,(\mathtt{0}+\mathtt{1})^*$ ■

h. **BYD**



Solution: After seeing the previous iterations, I know the instinct is to say "blah blah subsequence **111000** blah blah.” But look again ... not all transitions are defined ... it’s a NFA. And there’s only one accept state and no loops. Only one string is accepted. Therefore the regular expression = **111000** ■