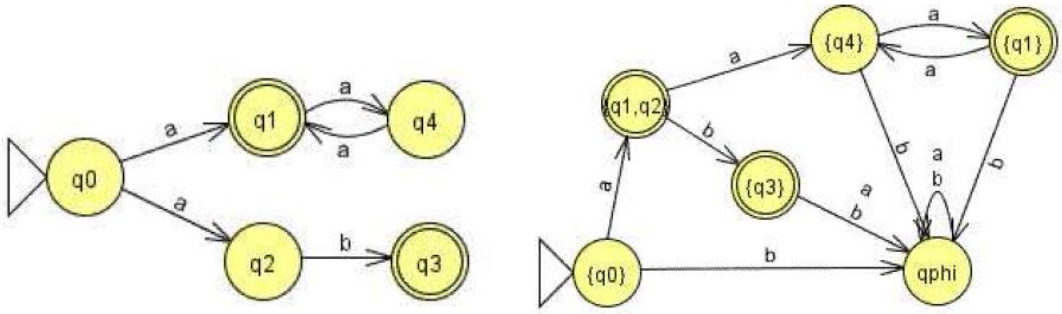
Challenge 3 –Non-Deterministic Finite Automata (NFAs and ε-NFAs) and the Conversion to DFAs

In order to implement regular expressions in software[1], it is common to first translate them to ε-NFAs. In practice, it depends on the size of the ε-NFAs, resultant DFAs, and on the characteristics of the target machine to decide about the kind of FA (Finite Automaton) to be used in the software implementation.

One of the problems of the DFAs are their possible state explosion in terms of the number of states of the equivalent ε-NFA (given an ε-NFA with N states we need a DFA of up to 2^N states).

However, one is intrigued by the fact that most DFAs we see have a number of states not much higher than those of the original ε-NFA. The example below is just one additional example where the resultant DFA (on the right) has just one more state than the original NFA (on the left).



Give examples of NFAs with N states (consider two examples, one for N = 2 and the other for N=3) that result in DFAs with 2^N states, when using the subset construction[2] method to convert.

[**To think more, but for now not needed to include in the answers to the challenge**: Is there a systematic approach you may use to generate such NFAs, given N and an appropriate alphabet?]

[1] There are also implementations of regular expressions in hardware!
[2] Also known as subset conversion.