Problem 2. (10 marks)

For each of the following automata over $\Sigma = \{a, b\}$, state 5 strings that are in the language of the automaton, and 5 strings that are not. (You score max(0, 5 - num_errors) points per part)

```
Sigma = a b
Q = q0 q1 q2 q3
start = q0
F = q2
q0 a q1
q0 b q2
q1 a q0
q1 b q0
q2 a q3
q2 b q3
q3 a q3
q3 b q3
```

```
Sigma = a b
Q = q0 q1 q2
start = q0
F = q1 q2
q0 a q1
q0 b q1
q0 a q2
q0 b q2
q1 a q2
q1 b q2
q2 a q2
q2 b q0
```

Problem 3. (30 marks)

For each of the following three languages over $\Sigma = \{a, b\}$, provide a regular expression and a DFA for the language. For full marks, your automata should have at most 10 states. Partial marks may be awarded for larger automata. (5 marks per regular expression and 5 marks per DFA)

1. The set of strings that contain *abbab*, in that order, but not necessarily consecutively.

   Said differently, the set of strings such that one can delete some amount of letters, possibly zero, to obtain *abbab*.

   For example, *aaababaababba* is in the language, while *bbbababbbaa* is not.

2. The set of strings of length 4 that contain exactly two *b*s.

For example, *abba* is in the language, while *abbb* and *abb* are not.

3. The set of strings that contain exactly three *a*s and an even number of *b*s. For example, *abababb* is in the language, while *ababab* is not.

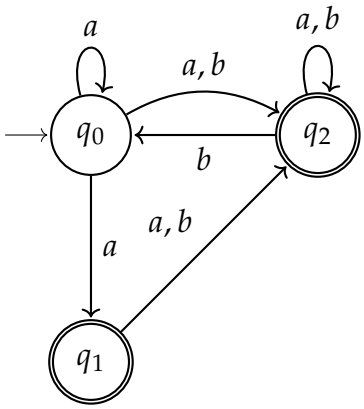
Problem 4. (20 marks)

Always eager to help students with their assignments, the new teaching assistant Chat2022 introduces a "clipping" mechanism for determinising NFAs. It's like trimming a bonsai – cutting away the superfluous to reveal a clean, aesthetic structure. But here's the catch: just as shortcuts might miss out on scenic routes, "clipping" might not always capture the true essence of an NFA. While some students happily accept Chat2022's clipped solutions, others notice discrepancies in behaviour. Your assignment: delve deep into Chat2022's method.

In this problem, we only consider automata without epsilon transitions.

A *clipping* of an NFA N is a DFA obtained by the following process: for each state-letter pair (q, x) where $q \in Q, x \in \Sigma$, if $|\delta(q, x)| > 1$, choose one element of it to be $\delta(q, x)$ and ignore the rest. Note that an NFA may have many possible clippings.

For example, here is an NFA N:



There are two pairs (q, x) where $|\delta(q, x)| > 1$, i.e., (q_0, a) and (q_2, b). Hence there are 6 possible clippings of N, corresponding to fixing one of the 3 choices for (q_0, a) and one of the 2 choices for (q_2, b).

1. Consider the following claim: "For every epsilon-free NFA N and string x, the NFA N accepts x if and only if some clipping of N accepts x." This claim is false. Show that it is false by providing a counterexample. (10 marks)

   (Aside: It turns out that deciding whether N accepts x is in P, while deciding whether some clipping of N accepts x is NP-complete, which you will learn about later in the course. Hence, if the claim were true, it would imply that P = NP.)

An automaton A over the alphabet Σ is *universal* if it accepts every string, that is, if $L(A) = \Sigma^*$.

2. Consider the following claim: "For every epsilon-free NFA N, the NFA N is universal if and only if some clipping of N is universal." This claim is also false. Show that it is false by providing a counterexample. (10 marks)

In both of these questions, each counterexample should:

- be over the alphabet $\Sigma = \{a, b\}$,
- have no more than 10 states, and
- have no more than 10 clippings.

Problem 5. (20 marks)

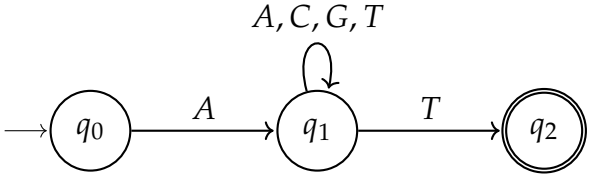
You work at *AutoGenoma*, a facility that designs automata for pattern matching DNA strings. They use an in-house variant of Nucleic Acid Notation. This includes the letters A, C, G and T which represent DNA bases, as well as a new letter X. A string that contains the letter X represents strings in which every occurrence of X is replaced by the same base letter. For instance, the string $XXCXT$ represents the strings $AACAT, CCCCT, GGCGT$ and $TTCTT$, but not $AACTT$ (since X needs to be replaced by the same base letter), and not $AXCAT$ (since every occurrence of X must be replaced). On the other hand, the string $AACAT$ is represented by $XXCXT, AAXAT$ and $AACAX$.

Your job is to convert an automaton M over $\{A, C, G, T\}$ into an automaton M' that accepts exactly the strings over $\{A, C, G, T, X\}$ that represent some string accepted by M. We say that M' *represents* M. For instance, if M accepts $AACAT$ then M' must accept all of the following strings:

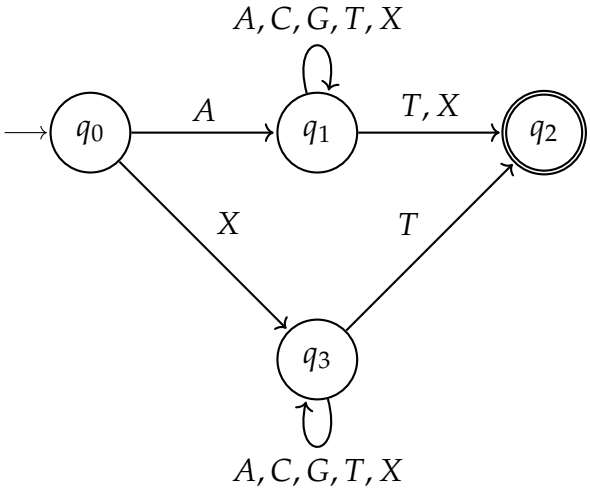
- $AACAT$ (zero occurrences of X),
- $XACAT, AXCAT, AAXAT, AACXT, AACAX$ (one occurrence of X),
- $AXCXT, XACXT, XXCAT$ (two occurrences of X),
- $XXCXT$ (three occurrences of X);

and if M' accepts $XXCXT$ then M must accept at least one of the following strings: $AACAT, CCCCT, GGCGT, TTCTT$.

For example, suppose M is the following automaton:



Then the following automaton represents M:



1. Provide a program to convert an NFA M over alphabet $\{A, C, G, T\}$ into an NFA M' over alphabet $\{A, C, G, T, X\}$ that represents M. (10 marks)
2. Provide a program to convert a DFA M over alphabet $\{A, C, G, T\}$ into a DFA M' over alphabet $\{A, C, G, T, X\}$ that represents M. (10 marks)

The conversions should ensure that the number of states in M' is polynomial in the number of states of M.