(1) Spears & Dragons with a Genie. The main point is that the state of having killed two dragons without getting a spear (and thus having no spears) is now different from the start state. More subtle, the state after killing a dragon and still having one spear in hand is different from the state after picking up a first spear (whether from the genie or not). So besides the states of 0 spears, 1 spear, and 2 spears shown in class, we will need these two additional states, plus the dead state as before. Any state in which one is not dead counts as accepting.

![Diag](https://example.com/diagram.png)

(A potential ambiguity in the rules is whether getting a spear from the genie counts as "picking up" a spear. But as the diagram makes clear, it doesn't matter on the first go---because the genie's spear can only be the first of two spears before killing two dragons, and the second such spear was "picked up" from the floor of a room in the original sense. However, suppose you get $x = \$$DD\ell D\ell$. If another $D$ comes along, you would need to rub the lamp to kill it. But does this count as "having killed two dragons since you last picked up a spear"? Did you "pick up" the previous spear from the genie? If you count $xD$ as a surviving case, then you need to change the DFA and it becomes more complex.)

(2) Convert the following NFA $N$ into a DFA $M$ such that $L(M) = L(N)$. The first thing to note is that the $\epsilon$-arc means "whenever state 1, then also 2" and makes the start state $\delta$ of the DFA equal $\{1,2\}$, not just $\{1\}$. This is vital here because $N$ accepts $\epsilon$ so we need the start state of $M$ to be accepting, and a set-state $R$ of $M$ will be accepting if and only if it includes $2$. At right of $N$ is the "delta-underbar" function $\delta(p,c) = \{ r : (\exists q)[(p,c,q) \in \delta \land \epsilon \in L_{qr}] \}$.

![Diag](https://example.com/diagram.png)

$\delta(1,a) = \{3\}$ $\delta(1,b) = \emptyset$

$\delta(2,a) = \{2\}$ $\delta(2,b) = \{4\}$

$\delta(3,a) = \emptyset$ $\delta(3,b) = \{2,4\}$

$\delta(4,a) = \{4\}$ $\delta(4,b) = \{1,2\}$
Then the rule for the DFA $M = (Q, \Sigma, \Delta, S, F)$ is $\Delta(P, c) = \bigcup_{p \in P} \delta(p, c)$. This can be expanded in a breadth-first search beginning with $P = S$. With the $\delta$ table arranged as above, this only needs unioning the sets in the column for the char $c$ and the rows for states in $P$. So we get:

$\Delta(S, a) = \delta(1, a) \cup \delta(2, a) = \{3\} \cup \{2\} = \{2, 3\}$. This is a new state of $M$.

$\Delta(S, b) = \delta(1, b) \cup \delta(2, b) = \varnothing \cup \{4\} = \{4\}$. This is also a new state. Expand $\{2, 3\}$ next:

$\Delta(\{2, 3\}, a) = \delta(2, a) \cup \delta(3, a) = \{2\} \cup \varnothing = \{2\}$. Another new state.

$\Delta(\{2, 3\}, b) = \delta(2, b) \cup \delta(3, b) = \{4\} \cup \{2, 4\} = \{2, 4\}$. Also new. But let's expand in order.

$\Delta(\{4\}, a) = \delta(4, a) = \{4\}$. Same old state.

$\Delta(\{4\}, b) = \delta(4, b) = \{1, 2\}$. Back to start, not new. So keep popping the set-state queue.

$\Delta(\{2\}, a) = \delta(2, a) = \{2\}$. A self-loop in $M$.

$\Delta(\{2\}, b) = \delta(2, b) = \{4\}$. Also not new. This leaves one more state to pop. Will we be done?

$\Delta(\{2, 4\}, a) = \delta(2, a) \cup \delta(4, a) = \{2\} \cup \{4\} = \{2, 4\}$. Another self-loop in $M$.

$\Delta(\{2, 4\}, b) = \delta(2, b) \cup \delta(4, b) = \{1, 2\} \cup \{4\} = \{1, 2, 4\}$. Aaugh! A new state. Continue:

$\Delta(\{1, 2, 4\}, a) = \delta(1, a) \cup \delta(2, a) \cup \delta(4, a) = \{3\} \cup \{2\} \cup \{4\} = \{2, 3, 4\}$. New.

$\Delta(\{1, 2, 4\}, b) = \delta(1, b) \cup \delta(2, b) \cup \delta(4, b) = \{1, 2, 4\}$. We just read it off the rows. Not new.

$\Delta(\{2, 3, 4\}, a) = \delta(2, a) \cup \delta(3, a) \cup \delta(4, a) = \{2, 4\}$. Not new.

$\Delta(\{2, 3, 4\}, b) = \delta(2, b) \cup \delta(3, b) \cup \delta(4, b) = \{1, 2, 4\}$. Not new, so: the BFS has closed.

Footnotes [not required for full credit]: Note that of the sixteen subsets of $\{1, 2, 3, 4\}$, four of them that include 1 but not 2 could be ruled out right away thanks to the epsilon-arc. Of the remaining twelve, we had to use seven---not too bad. Of note is that we never got the state $\varnothing$. This means that there is no string that $N$ cannot process from its start state. We also never got the "everything state" $\{1, 2, 3, 4\}$. This means that there is no string $x$ such that $N$ can process $x$ to any one of its states. This is different from noting e.g. that the string $abbb$ can be cycled through all the states of $N$ on the fly. (Note also that the word "any" is dubious here: it means "for all" but is talking "any one". More precise is to say "there is no string $x$ such that for all states $q$, $N$ can process $x$ from $s$ to $q$," but that's nerdier.)

Note also that the algorithm, even while dedicated to economizing, does not always give you a minimal DFA. In this case, the three states at right can all be condensed into one "eternal" state:
In this case, the other char does keep us in the "everything" state, and that closes the BFS.

The start state is just \( S = \{ s \} \) this time, since there are no \( \epsilon \)-arcs out of \( s \). Expanding the BFS:

\[
\begin{align*}
\Delta(S, a) &= \delta(s, a) = \{ q \} \quad \text{This is new.} \\
\Delta(S, b) &= \delta(s, b) = \{ f, q \}. \quad \text{Also new.} \\
\Delta(\{ q \}, a) &= \delta(q, a) = \{ f, q \}. \quad \text{Same as a new state we already found.} \\
\Delta(\{ q \}, b) &= \delta(q, b) = \{ s \}. \quad \text{Back to start. So only \{ f, q \} left to expand at this point.} \\
\Delta(\{ f, q \}, a) &= \delta(f, a) \cup \delta(q, a) = \{ s \} \cup \{ f, q \} = \{ s, f, q \}. \quad \text{We got all three states!} \\
\Delta(\{ f, q \}, b) &= \delta(f, b) \cup \delta(q, b) = \emptyset \cup \{ s \} = \{ s \}. \quad \text{We still have to expand \{ s, f, q \}.} \\
\Delta(\{ s, f, q \}, a) &= \{ s, f, q \} \because a \text{ is the char that took us to the "everything" state.} \\
\Delta(\{ s, f, q \}, b) &= \delta(s, b) \cup \delta(f, b) \cup \delta(q, b) = \{ f, q \} \cup \emptyset \cup \{ s \} = \{ s, f, q \}. \\
\end{align*}
\]

Study note: \( \delta(f, a) = \{ s, q, f \} \) and \( \delta(f, b) = \{ s \} \) would not come out wrong here, but they would make more work by taking \( \epsilon \)'s on the front end. We will get \{ s, q, f \} from the \( \Delta \) rule anyway---and in this case we won't ever get \( \emptyset \).

[IMHO, when you're doing NFA-to-DFA as a timed problem, once you have the "underline-delta" table done, you can safely do the BFS on-the-fly while drawing \( M \) without writing the \( \Delta \) rules longhand as I've done in this key. But trying to do so from a "delta" table that still has a column for "\( \epsilon \)" or "\( \epsilon^* \)" is more error-prone.]

In this case, the other char does keep us in the "everything" state, and that closes the BFS. The
accepting states are \( \{f, q\} \) and \( \{s, f, q\} \) because they include \( f \). These become the rejecting states of the complemented DFA \( M' \), which is shown to the right of \( M \).

The previous "eternally good" state has now become a dead state. Hence, it can never be involved in an accepting computation used in building the regular expression, so we can just delete it. This technically leaves an NFA, but that doesn't matter---we did the complementation step we needed. We note that this NFA \( N' \) has only one accepting state different from its start state, so we can execute the algorithm without the prior step of adding a new accepting state that has \( \epsilon \)-arrows from all the old accepting states.

Eliminating state 3 gets down to a 2-state base case.

In: 1 on \( a \), 2 on \( a \).

Out: Only 1, on \( b \). Hence update \( T(1, 1) \) and \( T(2, 1) \).

\[
T(1, 1)_{\text{new}} = T(1, 1)_{\text{old}} + T(1, 3)T(3, 3)'T(3, 1) = \emptyset + b \cdot \emptyset^* \cdot b = bb.
\]

\[
T(2, 1)_{\text{new}} = T(2, 1)_{\text{old}} + T(2, 3)T(3, 3)'T(3, 1) = b + a \cdot \emptyset^* \cdot b = b + ab.
\]

\[
L' = L_{11} \cup L_{12} \quad \text{where}
\]

\[
L_{11} = (bb + a(b + ab))' = (bb + ab + aab)'
\]

\[
L_{22} = L_{11} \cdot a \quad \text{So one form of the final answer is}
\]

\[
L' = (ab + bb + aab)'(\epsilon + a).
\]