Nine Errors Students Commonly Make When Applying the Pumping Lemma

The pumping lemma for regular languages is the following:

**Lemma.**

For all regular languages $L$, there exists a constant $n$ (depending on $L$) such that for all $z \in L$ with $|z| \geq n$, there exists a factorization $z = uvw$, with $|uv| \leq n$, $|v| \geq 1$, such that for all $i \geq 0$ we have $uv^iw \in L$.

Note that the pumping lemma states a property of regular languages. Hence one cannot use it directly to prove that a language is regular, but one can use the contrapositive (or proof by contradiction) to prove that a language is not regular. The contrapositive is as follows:

If for all $n$, there exists a $z \in L$ with $|z| \geq n$ such that for all factorizations $z = uvw$ satisfying the conditions $|uv| \leq n$ and $|v| \geq 1$, there exists an $i \geq 0$ such that $uv^iw \not\in L$, then $L$ is non-regular.

One common way people think about the pumping lemma is as follows: you are playing a four-step game against an adversary. The adversary is all-powerful, knows everything, but cannot cheat. Your goal is prove the language non-regular; your adversary is trying to prevent your proof from going through. You take turns choosing various objects:

- **Step 1:** adversary chooses $n$.
- **Step 2:** you choose $z \in L$ with $|z| \geq n$.
- **Step 3:** adversary chooses a factorization $z = uvw$ with $|uv| \leq n$ and $|v| \geq 1$.
- **Step 4:** you choose $i$. You “win” and show $L$ is not regular if $uv^iw \not\in L$, no matter what the adversary did in steps 1 and 3. Otherwise you lose: your proof didn’t work.

The following are the nine errors students commonly make in applying the pumping lemma:

**Error 1. Choosing a string $z$ that is not in $L$.** For example, suppose $L = \{ww : w \in \{a,b\}^*\}$.

You might incorrectly choose $z = a^n b^n$, which is not in $L$. At this point it’s easy to “win” — just pick $i = 1$; then $z = uv^1w \not\in L$. 

1
Error 2. Not handling all possible factorizations of the string $z$ as $uvw$. For example, consider

$$L = \{ww : w \in \{a, b\}^\ast\}$$

again. Suppose the adversary chooses $n$ and you choose $z = a^{2n}$. Then the adversary is supposed to choose a factorization $z = uvw$. If, by mistake, you do not think about all possible factorizations of $z$, you might wrongly choose to look only at the factorization specified by $u = \epsilon, v = a, w = a^{2n-1}$. In this case, you could choose $i = 0$, to get the string $uv^i w = a^{2n-1} \notin L$ and “win”. But you haven’t really “won”, because you didn’t handle all possible ways the adversary could factor $z$. The adversary could have chosen $u = \epsilon, v = aa, w = a^{2n-2}$, in which case $uv^i w \in L$ for all $i \geq 0$. In fact, if you choose $z = a^{2n}$, then you cannot possibly “win” the game. You need to choose a different $z$ here.

Error 3. Choosing a string $z$ that is not specific enough. Remember: you get to choose any string in $L$, based on $n$, that is longer than $n$ in length. Why make the adversary’s job easy? The adversary wants to defeat you by picking a bad factorization. Usually, the more specific you choose your string, the less freedom the adversary will have to respond.

For example, in the language $L$ above, you might have been tempted to choose $z = xx$, where $x$ was any string of length $\geq n$. Then you let the adversary break the string up as $z = uvw = xx$. By picking $i = 0$, you might conclude that $uw \neq xx$, and so obtain a “contradiction”. But this is simply not true! It does not suffice to show that $uv^i w \neq xx$ for a particular $x$; you must show it for all possible $x$, since that is the meaning of not being in $L$.

In fact, this kind of argument cannot succeed with such a general choice of $z$. For if your string was, say, $z = a^n a^n$, then the adversary can choose $u = \epsilon, v = aa$, and $w = a^{2n-2}$. In this case, no matter what $i$ you choose, the resulting string $uv^i w \in L$, and you cannot “win”.

Moral of the story: construct your string $z$ with care.

Error 4. Choosing a string $z$ that does not depend on $n$. For example, in the language $L$ above, suppose you picked $z = abab$. The problem is that you don’t know what $n$ is; you must be able to account for all possible values of $n$ picked by the adversary. If the length of the string you picked is not a function of $n$, you are in trouble.

Error 5. Choosing a negative or fractional $i$. This is not allowed by the statement of the pumping lemma. In looking at $uv^i w$, you must choose an $i$ that is a non-negative integer. Furthermore, since $z \in L$, considering the case $i = 1$ will never win.

Error 6. Applying the pumping lemma to a regular language. For example, consider

$$L = \{0^x 1^y : x + y \equiv 0 \pmod{4}\}.$$ 

This language is regular, but you might be tempted to try to prove it is not regular via the pumping lemma. You might pick, for example, the string $z = 0^{4n+3}1$. Then let the adversary factor $z$ as $z = uvw$. Hence $u = 0^n, v = 0^i, and w = 0^i1$, where $a + b + c = 4n + 3$. Then
you might assert, “We can choose \( i \) such that \( uv^i w = 0^{4n+3+i}b_1 \), and then clearly for all \( b \) we have that \( x + y = 4n + 3 + ib + 1 \) is not a multiple of 4.”

The problem with this claim is that it is false. For example, if \( b = 4 \), then \( 4n + 3 + ib + 1 \) is a multiple of 4 for all \( i \).

Moral here: be careful about what you assert, and be fairly confident that the language is indeed non-regular before you begin your proof.

Error 7. Assuming that all long strings in a regular language \( L \) can be written as \( uv^i w \) for some \( i \geq 2 \). This is not necessarily true. For example, if \( L = \{0, 1, 2\}^* \), then you might be tempted to conclude that there exist words \( u, v, w \) such that all sufficiently long strings in \( L \) can be written as \( uv^i w \) for some \( i \geq 2 \). This is simply false, as there exist strings in \( L \) that contain no substring of the form \( vv \) — this was first proved by the Norwegian mathematician Axel Thue in 1906.

Thue’s example also kills the same “theorem” when \( u, v, \) and \( w \) are allowed to lie in some finite set.

Error 8. Trying to use the pumping lemma to prove that a language is regular. The pumping lemma is a statement about a property of regular languages. It says, “If \( L \) is regular, then \( L \) has the following property.” Hence one cannot use the pumping lemma to prove that a language is regular; one can only use it to prove a language is non-regular.

In fact, there are languages which are non-regular, but nevertheless satisfy the conclusions of the pumping lemma! One example is the following language:

\[ L = \{a^i b^j c^k : i = 0 \text{ or } j = k\}. \]

Suppose \( z \in L \) is the string chosen to pump. There are two cases.

Case 1: \( z = b^j c^k \) for some integers \( j, k \). Pick \( n = 1 \); hence we may assume either \( j \geq 1 \) or \( k \geq 1 \). Then there exists a factorization \( z = uvw \), where \( u = \epsilon, v = b \) (if \( j \geq 1 \)) or \( v = c \) (if \( j = 0 \)) \( w \) is the rest of the string, and then \( uv^i w \in L \) for all \( i \geq 0 \).

Case 2: \( z = a^i b^j c^j \), for some integers \( i, j \) with \( i \geq 1 \). Pick \( n = 1 \). Then there exists a factorization \( z = uvw \), where \( u = \epsilon, v = a \), and \( w \) is the rest of the string, and \( uv^i w \in L \) for all \( i \geq 0 \).

The moral of the story is that the ordinary pumping lemma is not powerful enough to be able to directly prove the non-regularity of certain non-regular languages. Other techniques are needed.

Error 9. Choosing a string \( z = z(n) \), depending on \( n \), in such a way that

\[ \{z(n) : n \geq 1\} \]

is a regular language.
If you choose the string $z = z(n)$ to depend on $n$ in such a way that

$$L_z = \{z(n) : n \geq 1\}$$

is itself regular, then the pumping lemma cannot succeed in proving $L$ non-regular. For suppose it did. Then for each way of factoring $z = uvw$ with $|uv| \leq n$ and $|v| \geq 1$, there would be a choice of $i \geq 0$ such that $uv^iw \notin L$. But since $L_z \subseteq L$, $uv^iw \notin L_z$. Hence by the pumping lemma, $L_z$ itself would not be regular. But $L_z$ is in fact regular — a contradiction.

Hence one must choose the string $z = z(n)$ in a sufficiently “irregular” way to ensure that $L_z$ itself is not regular. As an example, consider the language

$$L = \{ww : w \in \{a, b\}^*\}.$$  

One might be tempted to choose the string $z = z(n) = a^{2n}$, which is certainly in $L$. However, the associated language is

$$L_z = \{a^{2n} : n \geq 1\} = \{a\}^+,$$

which is regular, so this choice for $z$ cannot possibly succeed in proving that $L$ is non-regular. So instead you would need to pick something like $z = a^n ba^n b$. 

4