Pumping Lemma for Regular Languages

Atonu Roy Chowdhury,
ext.atonu.roy@bracu.ac.bd

August 5, 2025

1 Are all languages regular?

Recall that a regular language is a language that is recognized by a DFA (or an NFA). So far we have seen multiple examples of regular languages. But are all languages regular? Are DFAs the all powerful machine that can solve all the problems? Unfortunately, the answer is no. We shall provide two intuitive explanations here.

Let Σ be a (finite) alphabet. Then a language over Σ is a set of strings. In other words, a language L is simply a subset of Σ^*. Therefore, the collection of **all** languages over Σ is $\mathcal{P}(\Sigma^*)$, the power set of Σ^*, i.e. the collection of all subsets of Σ^*.

Suppose we denote by Σ^n the set of all strings over Σ of length n. Then we have

$$\Sigma^* = \bigcup_{n=0}^{\infty} \Sigma^n = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \cdots \tag{1}$$

Notice that, each Σ^n is finite. In fact, $|\Sigma^n| = |\Sigma|^n$. Therefore, Σ^*, as a union of (countably) infinitely many finite sets, is (countably)[1] infinite[2]. Therefore, the collection of all languages over Σ is the power set of a (countably) infinite set. This is also an infinite set. But mathematicians have proved that this is an uncountably infinite set. That is, you cannot write out the elements in some sorted order and there is no notion of "next" element.

On the other hand, let's consider the set of all DFAs with n states. Call this set D^n. Then D^n is a finite set. In fact, one can show that $|D^n| \leq n^{n|\Sigma|+1}2^n$ (don't worry if you don't understand this weird-looking formula, just understand that this is a finite set). Therefore, the set of all DFAs is

$$D = \bigcup_{n=1}^{\infty} D^n = D^1 \cup D^2 \cup D^3 \cup D^4 \cup \cdots \tag{2}$$

Again, this is a union of (countably) infinitely many finite sets. So this set is (countably) infinite.

Now we have two different infinities. The set of all languages, $\mathcal{P}(\Sigma^*)$ is an uncountably infinite set. The set of all DFAs, D is a countably infinite set. Each regular language corresponds to one (or more) DFAs. Therefore, there can be at most as many regular languages as there are DFAs. Therefore, there are at most countably infinitely many regular languages.

As John Green said in *The Fault in Our Stars*, "Some infinities are bigger than other infinities"[3], the infinity of the set of all languages is larger than the infinity of the set of all regular languages.

[1] You can think of a countable infinite set as an infinite set where there is a notion of "next" element. For example, the set of non-negative integers $\mathbb{N} = \{0, 1, 2, 3, 4, \ldots\}$.

[2] In fact, $|\Sigma^*| = |\mathbb{N}|$. Think about it this way: you can list all the strings in Σ^* in a sorted manner, say sorted by length. So there is a notion of next element.

[3] https://www.goodreads.com/quotes/487705-there-are-infinite-numbers-between-0-and-1-there-s-1; mathematically speaking, the quote is correct, but the example he stated is wrong.

The former is uncountable, while the latter is countable. Therefore, there must exist non-regular languages.

The previous intuitive explanation was more mathematical. Now we shall present a more CS-motivated example. Suppose you have a DFA and you input a n-length string into it. Then it just reads all the characters one-by-one, changes states, and then as soon as it reads the whole string, it's ready with a yes-no answer. Therefore, we can say that a DFA operates in $\Theta(n)$ time.

However, not all yes-no problem can be solved in $\Theta(n)$ time (with n being the length of the input). Consider the primality checking problem for instance. You are given a number m as an input, and you have to check whether the number is prime or not. The naive approach of checking whether m is divisible by all the numbers from 2 to $\lfloor\sqrt{m}\rfloor$ is not very efficient. If m has n-many digits (in binary), i.e. $n \approx \lg m$, then the time complexity of this naive approach is

$$\Theta\left(\sqrt{m}\right) = \Theta\left(m^{\frac{1}{2}}\right) = \Theta\left(\left(2^{\lg m}\right)^{\frac{1}{2}}\right) = \Theta\left(\left(\sqrt{2}\right)^{n}\right). \tag{3}$$

So this is an exponential-time algorithm. In fact, there is no linear-time primality-checking algorithm that we know of yet. The best (deterministic) approach we know is AKS algorithm, which operates in $O\left(n^6\right)$.

So we know that not all yes-no problems can be solved in linear time. Even if a problem can be solved in linear time, we cannot really say that we can construct a DFA for it. Because DFAs have extremely limited memory. Therefore, not all languages can be recognized by some DFA, i.e. non-regular languages must exist!

2 Nonregularity and memory

Consider the following language over $\Sigma = \{\mathtt{0}, \mathtt{1}\}$:

$$L_1 = \{\mathtt{0}^n\mathtt{1}^n : n \geq 0\}. \tag{4}$$

If we want to construct a DFA that recognizes L_1, then the DFA needs to keep track of how many $\mathtt{0}$s it has read so far. But there is no bound on how many $\mathtt{0}$s the input or the accepting strings may have. So the DFA needs to keep track of an unlimited number of $\mathtt{0}$s, which is not possible since a DFA has only finitely many states.

We can consider each state of a DFA to carry some memory. But there are only finitely many states in a DFA. So in a sense, a DFA carries only a finite amount of memory. With this extremely limited memory, it's not possible to recognize a language as L_1.

But does this argument prove that L_1 is non-regular? Can we say that "well, since we need to keep track of an unbounded amount of $\mathtt{0}$s, we cannot construct a DFA for this language"? Actually, the answer is no. Even though this gives a heuristic of why L_1 might not be regular (in terms of memory usage), it doesn't prove that such a DFA doesn't exist. Just because we cannot find a DFA, doesn't really mean such a DFA is not out there. So we need to mathematically prove that such a DFA doesn't exist. We shall see the proof technique in the following section. But before that, let's consider the following example:

$$L_2 = \{w \in \{\mathtt{0}, \mathtt{1}\}^* : w \text{ has an equal number of occurences of } \mathtt{01} \text{ and } \mathtt{10} \text{ as substrings}\}. \tag{5}$$

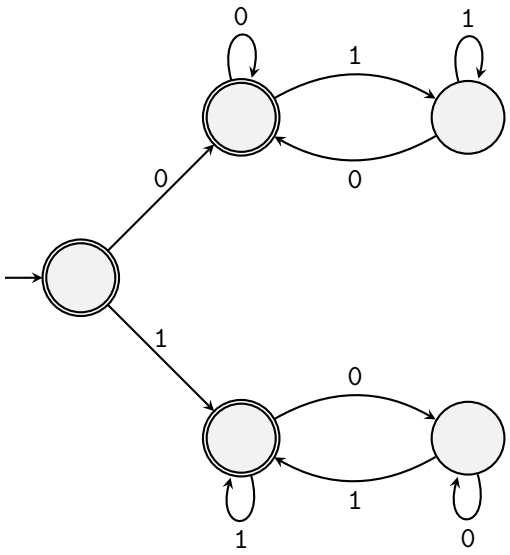
At a first glance, it appears that this language should be non-regular as well. Because it needs to keep track of how many times $\mathtt{01}$ and $\mathtt{10}$ appear as substrings, and there is no bound on this number. But our intuition on memory deceives us in this case. Let's look at some strings that are in L_2:

$$\varepsilon, \mathtt{0}, \mathtt{1}, \mathtt{010}, \mathtt{101}, \mathtt{0110}, \mathtt{1001}, \mathtt{010110}, \mathtt{101001}, \ldots$$

Do you see a common pattern? The strings are either ε, or start and end with the same symbol. Why? Because if the substring `01` appears, some part of the substring `10` is already produced. The next time `01` is appeared, we already have a `10`. So the quantity

number of times `01` appeard as substrings − number of times `10` appeard as substrings

is always either 1 or 0 or -1. In order for this quantity to be 0, it's necessary (and sufficient) that the strings start and end with the same symbol. So the DFA is

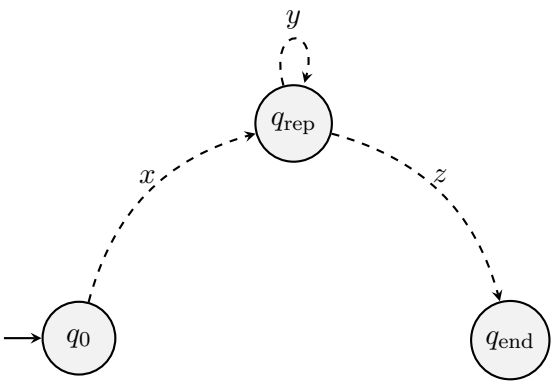


So even though our intuition on memory may give us some idea that some languages may be non-regular, that's not a concrete argument.

3 The pumping lemma for regular languages

In this section, we shall provide a mathematical tool to prove that certain languages are non-regular.

Suppose we have a DFA M, and we input a string of length n. As M processes the string, it goes through $n+1$ states in total (including the initial state before reading any symbol). Therefore, if the string is "very long", there must be repetitions in the sequence of states it visits. In particular, if the DFA has m-many states, and the string's length is greater than (or equal to) m, there will be a state which is repeated.



Suppose q_{rep} is a state that gets repeated. Say, upon reading the substring x, we go from the starting state q_0 to q_{rep}; from there, we read the substring y and stay at q_{rep}; finally we read z and go from q_{rep} to q_{end}.

Here, our input string is $w = xyz$. Now, the crucial observation is that if the string was $xyyz$, our ending state would've been the same. Because, in that case, our transition would look like

$$q_0 \xrightarrow{x} q_{\text{rep}} \xrightarrow{y} q_{\text{rep}} \xrightarrow{y} q_{\text{rep}} \xrightarrow{z} q_{\text{end}}$$

Not only that, upon every input xy^iz, for every $i \geq 0$, we would end up at the same state q_{end}. Now, in particular, if q_{end} is an accepting state, then the DFA accepts all the strings of the form xy^iz, for every $i \geq 0$.

Now we are ready to state the pumping lemma.

Lemma 1 (The Pumping Lemma for Regular Languages)**.**
If L is a regular language, then there is a number p (called **pumping length**) with the following property: for every $w \in L$ with $|w| \geq p$, we can divide w into 3 parts, $w = xyz$, such that

1. for each $i \geq 0$, $xy^iz \in L$;
2. $|y| > 0$;
3. $|xy| \leq p$.

We have already discussed the first condition, namely that $xy^iz \in L$ for every $i \geq 0$. The remaining two conditions are also important. The requirement $|y| > 0$ simply rules out the trivial choice $y = \varepsilon$. In other words, the repetition must occur after consuming at least one symbol. Furthermore, $|xy| \leq p$ ensures that we are considering the first repetition of states.

What the Pumping Lemma basically says is that every "sufficiently long" string in a regular language can be pumped. Meaning every string that has a length at least p can be divided into 3 pieces, $w = xyz$, such that we can "pump" y as many times as we want, and we are still in the language.

Some important remarks:

1. Note that while y cannot be ε, but x and z can be empty strings.
2. Let w be a very long string. There can be numerous ways of splitting it up into 3 pieces, $w = xyz$. But the condition 3 in Pumping Lemma guarantees us that we only need to look at those partitions where the first two part together has length at most p. And we have to account for **all** such partitions.
3. Suppose $w = xyz$ is such a partition. Then
$$\left|xy^iz\right| = |xyz| + \left|y^{i-1}\right| = |w| + (i-1)\,|y|\,. \tag{6}$$
We shall use this repeatedly while proving the nonregularity of languages.
4. As we have outlined, pumping length p is the number of states in a DFA. If a language is regular, there might be multiple DFAs that recognizes the language. So the number of states in a DFA that recognizes the language is not necessarily unique. In other words, pumping length is not unique. Pumping Lemma just guarantees that a pumping length exists for every regular language.
5. If p is a pumping length for a regular language L, then any number greater than p also works as a pumping length.

   Suppose $q > p$. Let w be a string in L with $|w| \geq q > p$. Then $|w| > p$. Since p is a pumping length, we can split w into 3 pieces, $w = xyz$ such that $|y| > 0$, $|xy| \leq p < q$, $xy^iz \in L$ for every $i \geq 0$.

   Therefore, every string with length at least q can be "pumped". Hence, q is also a pumping length.

6. In light of 5, we may assume, without loss of generality, that the pumping length p satisfies any additional property we find convenient. For example, if needed, we may take p to be even, or even to be a prime number.

   Indeed, if the original pumping length does not have the desired property, we can simply choose a larger number that does. Since any number greater than a pumping length is itself a valid pumping length, this replacement is always allowed.

   This works because there are infinitely many even numbers (and infinitely many primes). So given any p, there are even numbers (or prime numbers) larger than p.

4 Proving nonregularity of languages

The Pumping Lemma for Regular Languages is a strong tool to prove that certain languages are not regular. Notice that the statement of the pumping lemma gives us a property of regular languages. That means, every regular language has this property. So in order to prove the nonregularity of a language, we need to show that this property does not hold.

The usual prescription of such proofs are by contradiction. We assume that the given language is regular. Therefore, it has a pumping length p. Then we cleverly pick a string w with $|w| \geq p$. Then we show that no matter how we divide it into 3 pieces, there is some i for which xy^iz gets out of the language. Let's look at some examples.

Example 1. Consider the language from (4):

$$L = \{0^n 1^n : n \geq 0\}. \tag{7}$$

We want to show that this language is not regular. We want to use the Pumping Lemma. Let p be the pumping length for this language (assuming this language is regular). Then we need to pick a string with length at least p.

The most obvious choice is that we pick the first string of length at least p. That would be $w = 0^m 1^m$, with $m = \lceil \frac{p}{2} \rceil$. We have to partition w into 3 pieces, $w = xyz$. Then we have the following cases:

1. y consists of only 0s:

$$w = 0^m 1^m = \underbrace{000\ldots00}_{x}\underbrace{000\ldots00}_{y}\underbrace{0\ldots0111\ldots11}_{z}$$

   In this case, as we "pump" y once, in $xyyz$, the count of 0s and 1s will not be the same. It will have more 0s than 1s. So $xyyz$ gets out of the language.

2. y consists of only 0s:

$$w = 0^m 1^m = \underbrace{000\ldots0011\ldots1}_{x}\underbrace{111\ldots11}_{y}\underbrace{111\ldots11}_{z}$$

   In this case, as we "pump" y once, in $xyyz$, the count of 0s and 1s will not be the same. It will have more 1s than 0s. So $xyyz$ gets out of the language.

3. y has both 0s and 1s:

$$w = 0^m 1^m = \underbrace{000\ldots00}_{x}\underbrace{000\ldots00111\ldots11}_{y}\underbrace{111\ldots11}_{z}$$

   But now, if we pump y once, the format of $xyyz$ will not be $00\ldots011\ldots1$. So $xyyz$ gets out of the language.

Therefore, in all the cases, $xyyz$ gets out of the language. So the language is not regular.

We could've made our lives easier by choosing a different string so that we didn't have to consider multiple cases. For instance, say we chose $w = \mathtt{0}^p\mathtt{1}^p$. Then since xy altogether has length at most p, and the first p letters are all 0s, we know it for a fact that y consists of only 0s.

$$w = \mathtt{0}^p\mathtt{1}^p = \underbrace{\mathtt{000\ldots00}}_{x}\underbrace{\mathtt{000\ldots00}}_{y}\underbrace{\mathtt{0\ldots0111\ldots11}}_{z}.$$

Then if we "pump" y once, we get out of the language since there are more 0s than 1s. The formal way of writing this proof will be as follows.

Solution. Assume for the sake of contradiction that L is regular. Then L has a pumping length, say p. Consider the following string

$$w = \mathtt{0}^p\mathtt{1}^p \in L.$$

Then $|w| = 2p > p$. So w can be partitioned like $w = xyz$ such that $|y| > 0$, $|xy| \leq p$, and $xy^iz \in L$ for every $i \geq 0$.

Since $|xy| \leq p$, and the first p letters of w consists of 0s only, we can conclude that y consists of 0s only. Then for $i = 2$, $xyyz$ will be

$$xy^2z = xyyz = \mathtt{0}^{p+|y|}\mathtt{1}^p. \tag{8}$$

But $p + |y| \neq p$, so $xy^2z \notin L$. Therefore, L is not regular. ■

We shall use this trick of taking the first p characters to be the same often. Because in that case we don't have to consider multiple cases, and it just makes our lives a lot easier.

Example 2. Let's now consider the language

$$L = \left\{\mathtt{0}^{n^2} : n \geq 0\right\}. \tag{9}$$

It means the strings of L consists of 0s only, and the lengths of the strings are all perfect squares. Checking whether a number is perfect square or not also requires some notion of memory, so we can guess that this language is also non-regular.

Let's first look at how perfect squares are distributed:

$$0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, \ldots$$

You can see that the difference between two consecutive perfects squares is increasing. So large perfect squares cannot be near each other.

Now, assuming L is regular and p is the associated pumping length, we pick the string $w = \mathtt{0}^{p^2} \in L$. Then w can be partitioned into 3 pieces, $w = xyz$, such that $xy^iz \in L$ for every $i \geq 0$.

Now, what is the length of xy^2z? It's the $|xyz| + |y| = p^2 + |y|$, which is a number greater than p^2. What should we add to p^2 in order to get the next perfect square? It's $(p+1)^2 - p^2 = 2p + 1$.

So $|y|$ has to be at least $2p + 1$ to get to the next perfect square. But since $|xy| \leq p$, it's not possible. Again, we write the formal proof below.

Solution. Assume for the sake of contradiction that L is regular. Then L has a pumping length, say p. Consider the following string

$$w = \mathtt{0}^{p^2} \in L.$$

Then $|w| = p^2 > p$. So w can be partitioned like $w = xyz$ such that $|y| > 0$, $|xy| \leq p$, and $xy^iz \in L$ for every $i \geq 0$.

Since $|xy| \leq p$, $|y| \leq p < 2p+1$, so that

$$\left|xy^2z\right| = p^2 + |y| < p^2 + 2p + 1 = (p+1)^2 .$$

But this means $p^2 < \left|xy^2z\right| < (p+1)^2$, so $\left|xy^2z\right|$ cannot be a perfect square as it's strictly between two consecutive perfect squares. Therefore, $xy^2z \notin L$. Hence, L is not regular. ■

There's another way of writing the solution.

Alternate Solution. Assume for the sake of contradiction that L is regular. Then L has a pumping length, say p. Consider the following string

$$w = \mathtt{0}^{p^2} \in L.$$

Then $|w| = p^2 > p$. So w can be partitioned like $w = xyz$ such that $|y| > 0$, $|xy| \leq p$, and $xy^iz \in L$ for every $i \geq 0$.

Since $xy^2z \in L$ and $\left|xy^2z\right| > p^2$, we must have $\left|xy^2z\right| \geq (p+1)^2$. So we have,

$$\left|xy^2z\right| = p^2 + |y| \geq (p+1)^2 \implies |y| \geq 2p+1. \tag{10}$$

But we know $|y| \leq p$, since $|xy| \leq p$. Combining them, we have $p \geq |y| \geq 2p+1$, which is a contradiction! Therefore, L cannot be regular. ■

For some problems, you might need to "pump down", meaning you'd need to consider the $i = 0$ case in order to arrive at a contradiction.

Example 3. The language we are gonna consider now is

$$L = \{\mathtt{0}^m\mathtt{1}^n : m > n\}. \tag{11}$$

If p is the pumping length, for the string $w = \mathtt{0}^{p+1}\mathtt{1}^p$, y consists of all 0s. If we repeat y again and again, the number of 0s would get larger and larger than the number of 1s. So we don't really get a contradiction. However, if we remove y from w, since y has at least one 0, the number of 0s would get smaller than or equal to the number of 1s, and we get out of the language. The proof is as follows:

Solution. Assume for the sake of contradiction that L is regular. Then L has a pumping length, say p. Consider the following string

$$w = \mathtt{0}^{p+1}\mathtt{1}^p \in L.$$

Then $|w| = 2p+1 > p$. So w can be partitioned like $w = xyz$ such that $|y| > 0$, $|xy| \leq p$, and $xy^iz \in L$ for every $i \geq 0$.

Since $|xy| \leq p$, and the first p letters of w consists of 0s only, we can conclude that y consists of 0s only. Since $|y| > 0$, we can say $|y| \geq 1$. But now, for $i = 0$,

$$xy^0z = xz = \mathtt{0}^{p+1-|y|}\mathtt{1}^p. \tag{12}$$

Since $|y| \geq 1$, $p + 1 - |y| \leq p + 1 - 1 = p$. So $xy^0 \notin L$. Hence, L cannot be regular. ■

Example 4. Take the language

$$L = \{w \in \{\mathtt{0},\mathtt{1}\}^* \ : \ w \text{ has an equal number of } \mathtt{0}\text{s and } \mathtt{1}\text{s}\}.$$

If it were regular and p is a pumping length, we can take the string $w = \mathtt{0}^p\mathtt{1}^p \in L$, and then copy the solution of Example 1. But there's another way of showing that this language is not regular.

Assume that L is regular. $\mathtt{0}^*\mathtt{1}^*$ is a regular expression, so it generates a regular language. Therefore, $L \cap \mathtt{0}^*\mathtt{1}^*$ is also regular, being the intersection of two regular languages. However,

$$L \cap \mathtt{0}^*\mathtt{1}^* = \{\mathtt{0}^n\mathtt{1}^n \ : \ n \geq 0\}, \tag{13}$$

which we have proved to be non-regular in Example 1. Therefore, L cannot be regular.

5 Some more examples and solutions

Problem 1. Prove that $L_1 = \{w \in \{0,1,2\}^* : 0^n1^n2^n \text{ where } n \geq 0\}$ is not regular.

Solution. Assume for the sake of contradiction that L_1 is regular. Then let p be the pumping length for L_1. Now we take the string

$$w = 0^p1^p2^p \in L_1.$$

Then the length of w is $|w| = 3p \geq p$. So w can be split into xyz such that $|y| > 0$, $|xy| \leq p$, and $xy^iz \in L_1$ for each $i \geq 0$. Since $|xy| \leq p$, and the first p characters of w are all 0s, we can conclude that y consists of only 0s.

Then, for $i = 2$, xy^2z will be

$$xy^2z = xyyz = 0^{p+|y|}1^p2^p \notin L_1. \tag{14}$$

Thus we get a contradiction! Hence, L_1 is not a regular language. ■

Problem 2. Show that $L_2 = \{w \in \{0,1\}^* : 0^x1^y0^z \text{ where } z > x + y \text{ and } x, y \geq 0 \}$ is not regular.

Solution. Assume for the sake of contradiction that L_2 is regular. Then let p be the pumping length for L_2. Now we take the string

$$w = 0^p1^p0^{2p+1} \in L_2.$$

Then the length of w is $|w| = 4p + 1 > p$. So w can be split into xyz such that $|y| > 0$, $|xy| \leq p$, and $xy^iz \in L_2$ for each $i \geq 0$. Since $|xy| \leq p$, and the first p characters of w are all 0s, we can conclude that y consists of only 0s.

Then, for $i = 2$, xy^2z will be

$$xy^2z = xyyz = 0^{p+|y|}1^p0^{2p+1}. \tag{15}$$

This string is not in L_2, since $p + |y| + p \geq 2p + 1$. Thus we get a contradiction! Hence, L_2 is not a regular language. ■

Problem 3. Show that $L_3 = \{w \in \{0,1\}^* : w \text{ is a palindrome}\}$ is not a regular language.

Solution. Assume for the sake of contradiction that L_3 is regular. Then let p be the pumping length for L_3. Now we take the string

$$w = 0^p10^p \in L_3.$$

Then the length of w is $|w| = 2p + 1 > p$. So w can be split into xyz such that $|y| > 0$, $|xy| \leq p$, and $xy^iz \in L_3$ for each $i \geq 0$. Since $|xy| \leq p$, and the first p characters of w are all 0s, we can conclude that y consists of only 0s.

Then, for $i = 2$, xy^2z will be

$$xy^2z = xyyz = 0^{p+|y|}10^p = 0^p0^{|y|}10^p. \tag{16}$$

This string is not a palindrome, so it is not in L_3. Thus we get a contradiction! Hence, L_3 is not a regular language. ■

Problem 4. Is $L_4 = \{w \in \{a,b\}^* : \text{ numbers of a in } w \text{ is a prime number}\}$ a regular language? If so, show a DFA/NFA/RE, otherwise prove that it's not regular.

Solution. No, L_4 is not regular. Assume for the sake of contradiction that L_4 is regular. Then let p be the pumping length for L_4. Now we take the string

$$w = \mathtt{a}^q \in L_4,$$

where q is a prime number greater than p. Then the length of w is $|w| = q > p$. So w can be split into xyz such that $|y| > 0$, $|xy| \leq p$, and $xy^iz \in L_4$ for each $i \geq 0$. Now, for $i > 0$,

$$xy^iz = xyy^{i-1}z = \mathtt{a}^{q+(i-1)|y|}. \tag{17}$$

Since this string is in L_4, $q + (i-1)|y|$ is a prime number for each $i > 0$. But this is clearly not true, since choosing $i = q + 1$ gives

$$q + (i-1)|y| = q + q|y|, \tag{18}$$

which is divisible by q. Thus we get a contradiction! Hence, L_4 is not a regular language. ■

Problem 5. Show that $L_5 = \left\{\mathtt{0}^{3^n} \ : \ n \geq 0\right\}$ is not regular.

Solution. Assume for the sake of contradiction that L_5 is regular. Then let p be the pumping length for L_5. Now we take the string

$$w = \mathtt{0}^{3^p} \in L_5.$$

Then the length of w is $|w| = 3^p > p$. So w can be split into xyz such that $|y| > 0$, $|xy| \leq p$, and $xy^iz \in L_5$ for each $i \geq 0$.

Then, for $i = 2$, xy^2z will be

$$xy^2z = xyyz = \mathtt{0}^{3^p+|y|}.$$

Since this string is in L_5, $3^p + |y|$ is a power of 3 (which is, of course, larger than 3^p). The next power of 3 larger than 3^p is 3^{p+1}. So we have

$$3^p + |y| \geq 3^{p+1} \implies |y| \geq 3^{p+1} - 3^p = 2 \cdot 3^p. \tag{19}$$

On the other hand, $|xy| \leq p$ gives us that $|y| \leq p$. So

$$p \geq |y| \geq 2 \cdot 3^p. \tag{20}$$

This is clearly false since $3^p > p$. Thus we get a contradiction! Hence, L_5 is not a regular language. ■

Problem 6. Prove that $L_6 = \{w \in \{\mathtt{0}, \mathtt{1}\}^* : w = \mathtt{0}^{n!} \text{ where } n \geq 0\}$ is not a regular language.

Solution. Assume for the sake of contradiction that L_6 is regular. Then let p be the pumping length for L_1. Now we take the string

$$w = \mathtt{0}^{p!} \in L_6.$$

Then the length of w is $|w| = p! \geq p$. So w can be split into xyz such that $|y| > 0$, $|xy| \leq p$, and $xy^iz \in L_1$ for each $i \geq 0$. y consists of only 0s, so

$$xy^iz = \mathtt{0}^{p!+(i-1)|y|} \tag{21}$$

Then, for $i = 2$, xy^2z will be

$$xy^2z = xyyz = \mathtt{0}^{p!+|y|}. \tag{22}$$

Now, $|y| \leq p < p \cdot p!$, hence,

$$p! < p! + |y| < p! + p \cdot p! = p!\,(1 + p) = (p+1)! \tag{23}$$

So $p! < p! + |y| < (p+1)!$, and the length of xy^2z is strictly between two consecutive factorials. Hence, it cannot be a factorial. Thus we get a contradiction! Hence, L_6 is not a regular language. ■

Alternate Solution. Assume for the sake of contradiction that L_6 is regular. Then its complement language $\overline{L}_6$ is also regular.

$$\overline{L}_6 = \{w \in \{0,1\}^* : w = 0^m \text{ where } m \neq n! \text{ for any } n \geq 0\}.$$

Let p be the pumping length of $\overline{L}_6$. Now we take the string

$$w = 0^{p \cdot p!} \in \overline{L}_6.$$

Then the length of w is $|w| = p \cdot p! \geq p$. So w can be split into xyz such that $|y| > 0$, $|xy| \leq p$, and $xy^iz \in \overline{L}_6$ for each $i \geq 0$. y consists of only 0s, so

$$xy^iz = 0^{p \cdot p! + (i-1)|y|}. \tag{24}$$

Now, since $0 < |y| \leq p$, we have that $|y|$ divides $p!$. Now, choosing $i = 1 + \frac{p!}{|y|}$, we have

$$xy^iz = 0^{p \cdot p! + (i-1)|y|} = 0^{p \cdot p! + \frac{p!}{|y|}|y|} = 0^{p \cdot p! + p!} = 0^{(p+1)!} \notin \overline{L}_6. \tag{25}$$

Therefore, $\overline{L}_6$ is not a regular language. Thus we get a contradiction, and hence L_6 cannot be regular as well. ■

Problem 7. Show that $L_7 = \{w \in \{0,1\}^* : w = 0^a1^b1^c0^d \text{ where } a + b = c + d \text{ and } a, b, c, d \geq 0 \}$ is not a regular language.

Solution. Assume for the sake of contradiction that L_7 is regular. Then let p be the pumping length for L_7. Now we take the string

$$w = 0^p110^p \in L_7.$$

Then the length of w is $|w| = 2p + 2 \geq p$. So w can be split into xyz such that $|y| > 0$, $|xy| \leq p$, and $xy^iz \in L_7$ for each $i \geq 0$. $|xy| \leq p$, so y consists of only 0s, so

$$xy^iz = 0^{p+(i-1)|y|}110^p. \tag{26}$$

We choose $i = 4$, so that

$$xy^4z = 0^{p+3|y|}110^p. \tag{27}$$

Since this is in L_7, one can write it as $0^a1^b1^c0^d$ for $a + b = c + d$. By equating

$$0^{p+3|y|}110^p = 0^a1^b1^c0^d, \tag{28}$$

we get $a = p + 3|y|$, $d = p$ and $b + c = 2$. So $c - b \leq 2$. Furthermore,

$$c - b = a - d = 3|y| \geq 3, \tag{29}$$

as $|y| \geq 1$. So we get $c-b \leq 2$ and $c-b \geq 3$, which is a contradiction! Therefore, L_7 is not regular. ■

Alternate Solution. Assume for the sake of contradiction that L_7 is regular. Then let p be the pumping length for L_7. Now we take the string

$$w = 0^p1^p1^p0^p \in L_7. \tag{30}$$

Then the length of w is $|w| = 4p \geq p$. So w can be split into xyz such that $|y| > 0$, $|xy| \leq p$, and $xy^iz \in L_7$ for each $i \geq 0$. $|xy| \leq p$, so y consists of only 0s, so

$$xy^iz = 0^{p+(i-1)|y|}1^p1^p0^p. \tag{31}$$

We choose $i = 2p + 2$, so that

$$xy^{2p+2}z = 0^{p+(2p+1)|y|}1^p1^p0^p. \tag{32}$$

Since this is in L_7, one can write it as $\mathtt{0}^a\mathtt{1}^b\mathtt{1}^c\mathtt{0}^d$ for $a+b=c+d$. By equating

$$\mathtt{0}^{p+(2p+1)|y|}\mathtt{1}^p\mathtt{1}^p\mathtt{0}^p = \mathtt{0}^a\mathtt{1}^b\mathtt{1}^c\mathtt{0}^d, \tag{33}$$

we get $a = p+(2p+1)|y|$, $d=p$ and $b+c=2p$. So $c-b \leq 2p$. Furthermore,

$$c-b = a-d = (2p+1)|y| \geq 2p+1, \tag{34}$$

as $|y| \geq 1$. So we get $c-b \leq 2p$ and $c-b \geq 2p+1$, which is a contradiction! Therefore, L_7 is not regular. ■

Problem 8. Is the language $L_8 = \{\mathtt{0}^n\mathtt{1}^m : n \neq m\}$ regular? Why or why not?

Solution. Assume for the sake of contradiction that L_8 is regular. Then let p be the pumping length for L_8. Now we take the string

$$w = \mathtt{0}^p\mathtt{1}^{p+p!} \in L_8. \tag{35}$$

Then the length of w is $|w| = p+p+p! \geq p$. So w can be split into xyz such that $|y| > 0$, $|xy| \leq p$, and $xy^iz \in L_7$ for each $i \geq 0$. $|xy| \leq p$, so y consists of only 0s, so

$$xy^iz = \mathtt{0}^{p+(i-1)|y|}\mathtt{1}^{p+p!}. \tag{36}$$

$1 \leq |y| \leq p$, so $|y|$ divides $p!$. Take $i = 1+\frac{p!}{|y|}$. Then

$$xy^iz = \mathtt{0}^{p+(i-1)|y|}\mathtt{1}^{p+p!} = \mathtt{0}^{p+\left(1+\frac{p!}{|y|}-1\right)|y|}\mathtt{1}^{p+p!} = \mathtt{0}^{p+p!}\mathtt{1}^{p+p!} \notin L_8. \tag{37}$$

Therefore, L_8 is not a regular language. ■

Alternate Solution. Assume for the sake of contradiction that L_8 is regular. Then so is $\overline{L}_8$. Therefore, $\overline{L}_8 \cap \mathtt{0}^*\mathtt{1}^*$ is also regular, as the intersection of two regular languages. But

$$\overline{L}_8 \cap \mathtt{0}^*\mathtt{1}^* = \{\mathtt{0}^n\mathtt{1}^n \ : \ n \geq 0\}, \tag{38}$$

which we proved to be non-regular in Example 1. Therefore, L_8 is not regular. ■