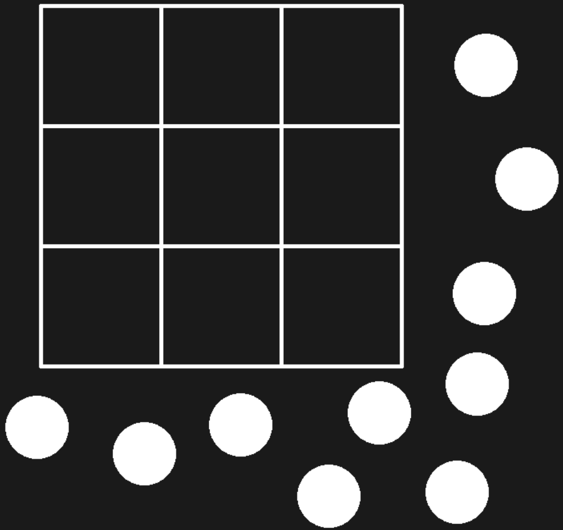
An Introduction to Mathematical Proofs

Pigeonhole Principle

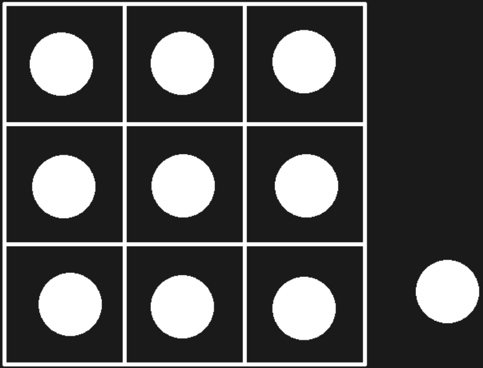
Who? Fahad Hossaini

When? Whenever You Watch

Pigeonhole Principle



Pigeonhole Principle



What's The Point?

While the statement is super general and very intuitive, it's helpful in multiple ways.

First, we see that 'size' plays a role when it comes to the possibility of bijections between sets. We can *never* form a bijection between a set with 9 elements and a set with 10 elements.

Bijections Between Sets?

Let A, B be sets. If there exists some function $f : A \to B$ such that f is a bijection, then we say that there's a bijection between A and B. Or, A and B are bijective.

Likewise, we can say A and B are injective/surjective when there exists a function $f : A \to B$ that is injective/surjective respectively.

'Size'

We use absolute values around sets to denote their 'size.' So, if $A = \{1, 2, 4\}$, then $|A| = 3$.

Formally, the word we use is cardinality, or, the cardinality of A is 3. There's a reason we use cardinality and not 'size' which we'll talk about in the next video.

Pigeonhole Principle And Cardinality

If $A = \{1, 2, 3, 4, 5\}$ and $B = \{a, b, c, d\}$, then by the Pigeonhole Principle (or PHP), there can't be an injective function from A to B.

Notice that: $|A| > |B|$.

Now consider a function from B to A. Clearly, such a function can never be surjective.

In this case: $|B| < |A|$.

Or In Other Words

We notice the following:

If $|A| \leq |B|$, then we can find an injective function from A to B. Likewise, if $|A| \geq |B|$, we can find a surjective function from A to B.

Then, we see that if $|A| = |B|$, we can find a bijection from A to B. And indeed, this logic works backward!

If we can find a bijection between A and B, we can match up elements one-to-one in a unique way, so they must have the same cardinality.

Okay, Back To PHP

That's enough cardinality, let's talk about some cool PHP consequences.

I claim that I can come up with a program that compresses any file by 5% and decompresses that file back to the original.

Does such a program exist?

The Answer: No!

Proof.

Why? PHP! Assume for the sake of contradiction that such a program exists. Then, consider the set of all files of size $100x$ with D. They're 2^{100x} many possible files of this size.

After compression, the files are now of size $95x$ and therefore 2^{95x} many possible files of this size. Call this set C.

Now, our contradiction. Our decompression algorithm can't map uniquely all the compressed files to the decompressed files. Why? Since $2^{95x} < 2^{100x}$, we have that $|C| < |D|$ and no surjection can exist from C to D. Thus, our decompression algorithm can't exist so such a compression machine can't exist. □

Here's Another Cool One

Consider the following numbers:

$647, 647^2, 647^3, \ldots, 647^{10}, 647^{11}$

Without any calculation, what can we say about the last digits of all these numbers?

At least two of these numbers will share their last digit.

While we don't know which numbers and which digit, we have some bound. Some cap. Some limit. And while this lower bound is much stricter and with some elementary math, we can see exactly what these last digits look like, we have some conditions.

Bounds

PHP is really good at finding some bound or limit. While the bounds we develop is super primitive, we can prove that bounds exist. Prove!