Unequal Cardinalities

Outline for Today

- **Unequal Cardinalities**
  - Revisiting our first lecture with our new techniques!
- **Cantor's Diagonal Argument**
  - How do we formalize that pictorial proof from our first lecture?
- **The Pigeonhole Principle**
  - Proving results just by counting!

Recap from Last Time

Terminology

- A ***function*** f is a mapping from one set A to another set B such that every element of A is associated with a single element of B.
  - For each $a \in A$, there is some $b \in B$ with $f(a) = b$.
  - Evaluating the same function on the same inputs always produces the same output: if $f(a) = b_0$ and $f(a) = b_1$, then $b_0 = b_1$.
- If f is a function from A to B, we say that f is a ***mapping*** from A to B.
  - We call A the ***domain*** of f.
  - We call B the ***codomain*** of f.
- We write $\boldsymbol{f : A \to B}$ to indicate that f has domain A and codomain B.

Injective Functions

- A function $f : A \to B$ is called ***injective*** (or ***one-to-one***) if each element of the codomain has at most one element of the domain that maps to it.
  - A function with this property is called an ***injection***.
- Formally, $f : A \to B$ is an injection if this statement is true:

$$\forall a_1 \in A.\ \forall a_2 \in A.\ (f(a_1) = f(a_2) \to a_1 = a_2)$$

(*"If the outputs are the same, the inputs are the same"*)

- Equivalently:

$$\forall a_1 \in A.\ \forall a_2 \in A.\ (a_1 \neq a_2 \to f(a_1) \neq f(a_2))$$

(*"If the inputs are different, the outputs are different"*)

Surjective Functions

- A function $f : A \to B$ is called ***surjective*** (or ***onto***) if each element of the codomain is "covered" by at least one element of the domain.
  - A function with this property is called a ***surjection***.
- Formally, $f : A \to B$ is a surjection if this statement is true:

$$\forall b \in B.\ \exists a \in A.\ f(a) = b$$

(*"For every possible output, there's at least one possible input that produces it"*)

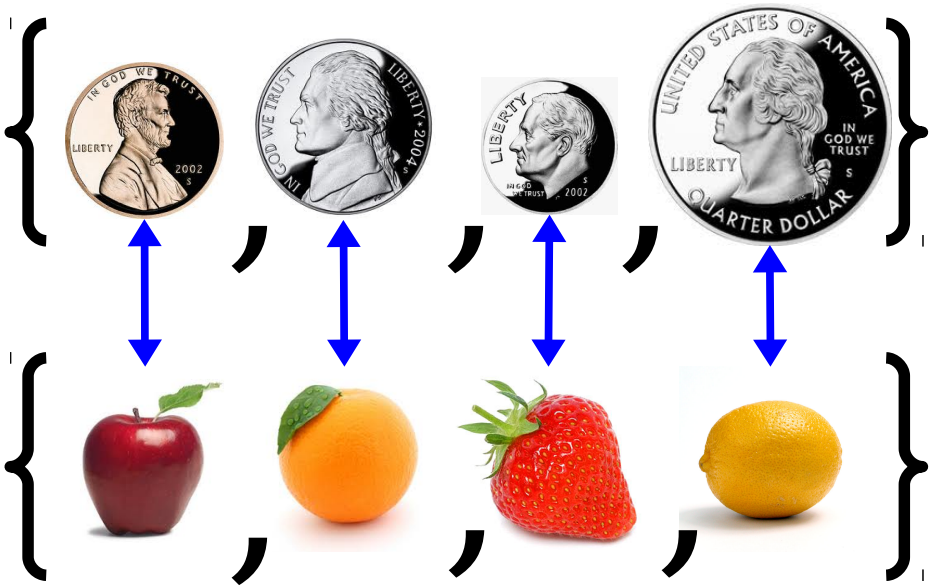
Bijections

- A function that associates each element of the codomain with a unique element of the domain is called ***bijective***.
  - Such a function is a ***bijection***.
- Formally, a bijection is a function that is both ***injective*** and ***surjective***.

Comparing Cardinalities

- The relationships between set cardinalities are defined in terms of functions between those sets.
- $|S| = |T|$ is defined using bijections.

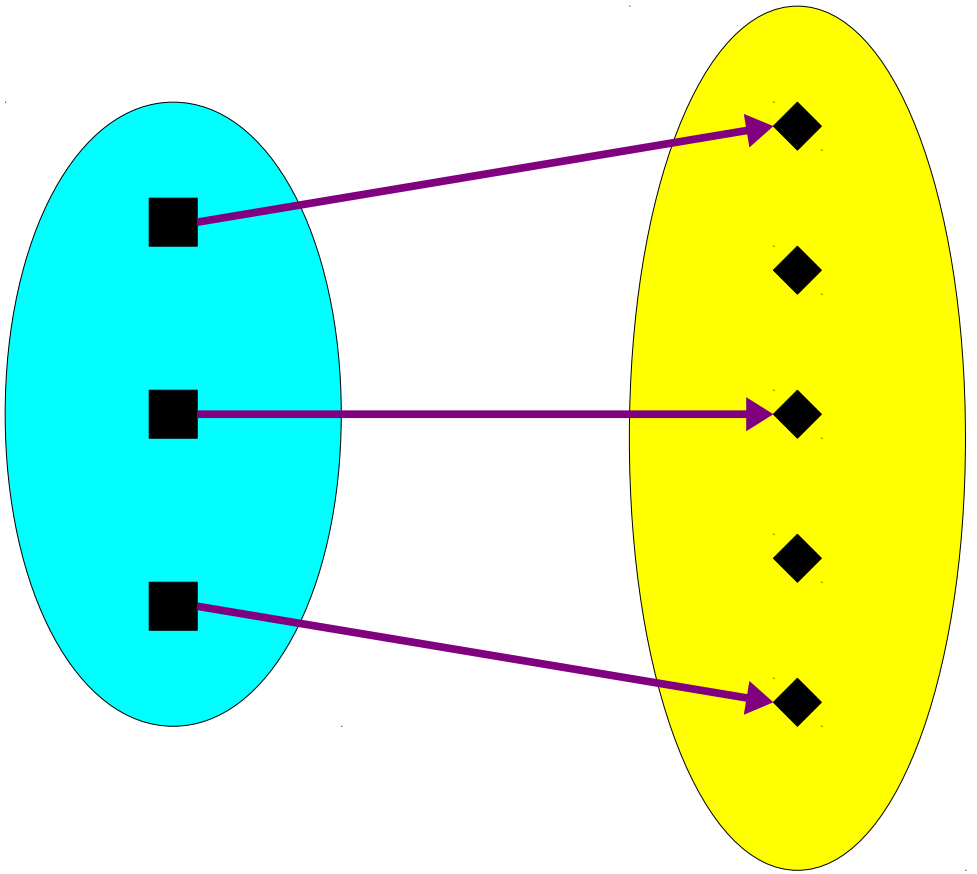
  $|S| = |T|$ if there exists a *bijection* $f : S \to T$



Ranking Cardinalities

- We define $|S| \leq |T|$ as follows:

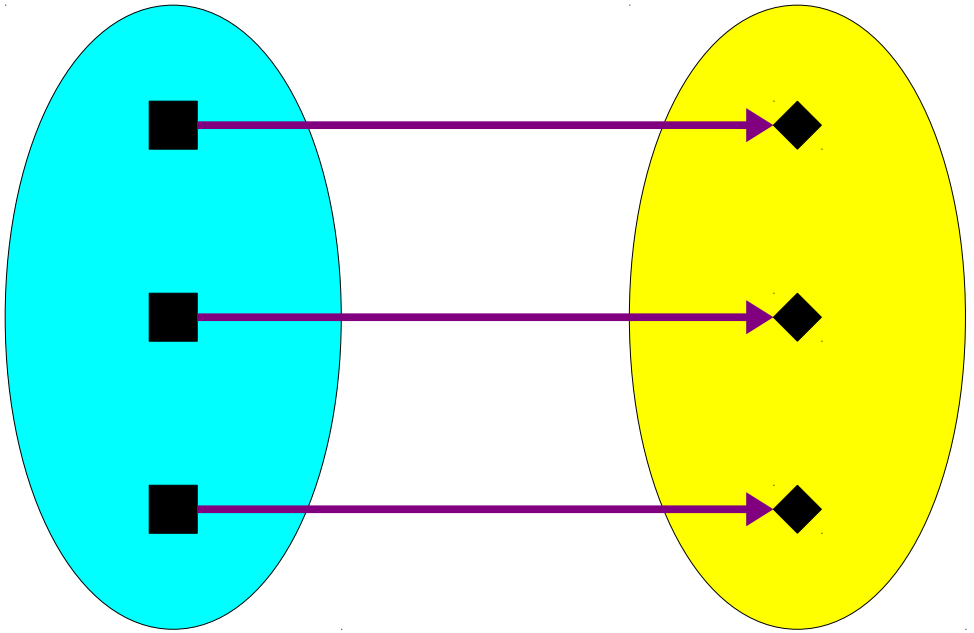
  $|S| \leq |T|$ if there is an injection $f : S \to T$



Ranking Cardinalities

- We define $|S| \leq |T|$ as follows:

  $|S| \leq |T|$ if there is an injection $f : S \to T$



Theorem (Cantor-Bernstein-Schroeder): If S and T are sets where $|S| \leq |T|$ and $|T| \leq |S|$, then $|S| = |T|$.

New Stuff!

Unequal Cardinalities

- Recall: $|A| = |B|$ if the following statement is true:

  There exists a bijection $f : A \to B$

- What does it mean for $|A| \neq |B|$?

  No function $f : A \to B$ is a bijection.

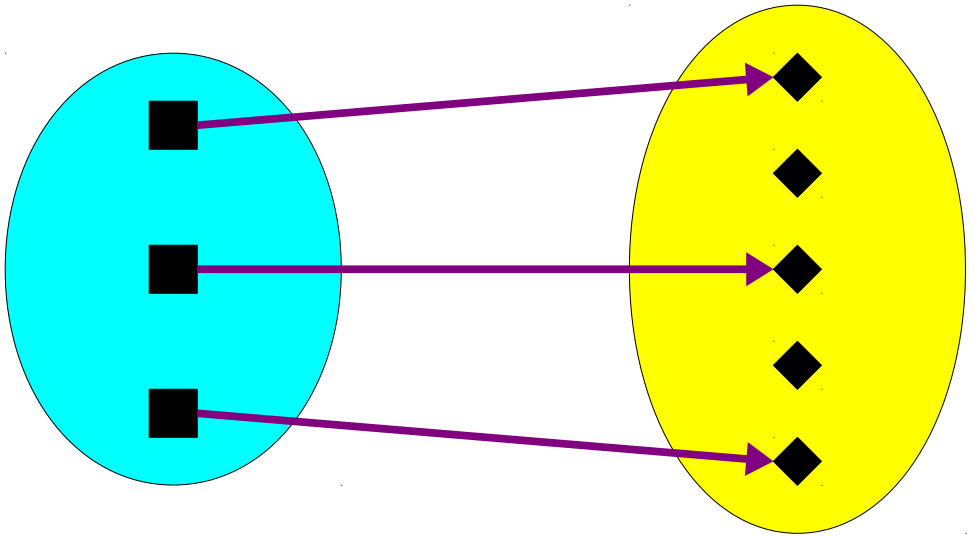
- Need to show that, out of all the (potentially infinitely many) functions from A to B, *not one of them* is a bijection.

Comparing Cardinalities

- Formally, we define < on cardinalities as

  $|S| < |T|$ iff $|S| \leq |T|$ and $|S| \neq |T|$

- In other words:
  - There is an injection from S to T.
  - There is no bijection between S and T.



Comparing Cardinalities

- Formally, we define < on cardinalities as

$$\mathbf{|S| < |T| \text{ iff } |S| \leq |T| \text{ and } |S| \neq |T|}$$

- In other words:
  - There is an injection from S to T.
  - There is no bijection between S and T.
- ***Theorem:*** For any sets S and T, exactly one of the following is true:

$$\mathbf{|S| < |T|} \qquad \mathbf{|S| = |T|} \qquad \mathbf{|S| > |T|}$$

Cantor's Theorem Revisited

Cantor's Theorem

- ***Cantor's Theorem*** is the following:

  If S is a set, then $|S| < |\wp(S)|$

- This is how we concluded that there are more problems to solve than programs to solve them.
- We informally sketched a proof of this in the first lecture.
- Let's now formally prove Cantor's Theorem.

The Key Step

- We need to show the following:

  If S is a set, then $|S| \neq |\wp(S)|$.

- To do this, we need to prove this statement:

  For any set S, no function $f : S \to \wp(S)$ is a bijection.

- Let's review the graphical intuition for this proof.

$x_0 \longleftrightarrow \{ x_0, x_2, x_4, \ldots \}$

$x_1 \longleftrightarrow \{ x_0, x_3, x_4, \ldots \}$

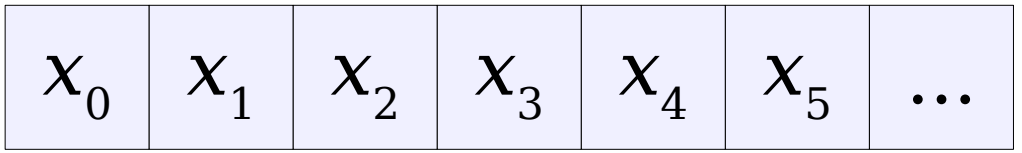
$x_2 \longleftrightarrow \{ x_4, \ldots \}$

$x_3 \longleftrightarrow \{ x_1, x_4, \ldots \}$

$x_4 \longleftrightarrow \{ x_0, x_5, \ldots \}$

$x_5 \longleftrightarrow \{ x_0, x_1, x_2, x_3, x_4, x_5, \ldots \}$

$\ldots$



$x_0 \leftrightarrow \{ x_0, x_2, x_4, \ldots \}$

$x_1 \leftrightarrow \{ x_0, x_3, x_4, \ldots \}$

$x_2 \leftrightarrow \{ x_4, \ldots \}$

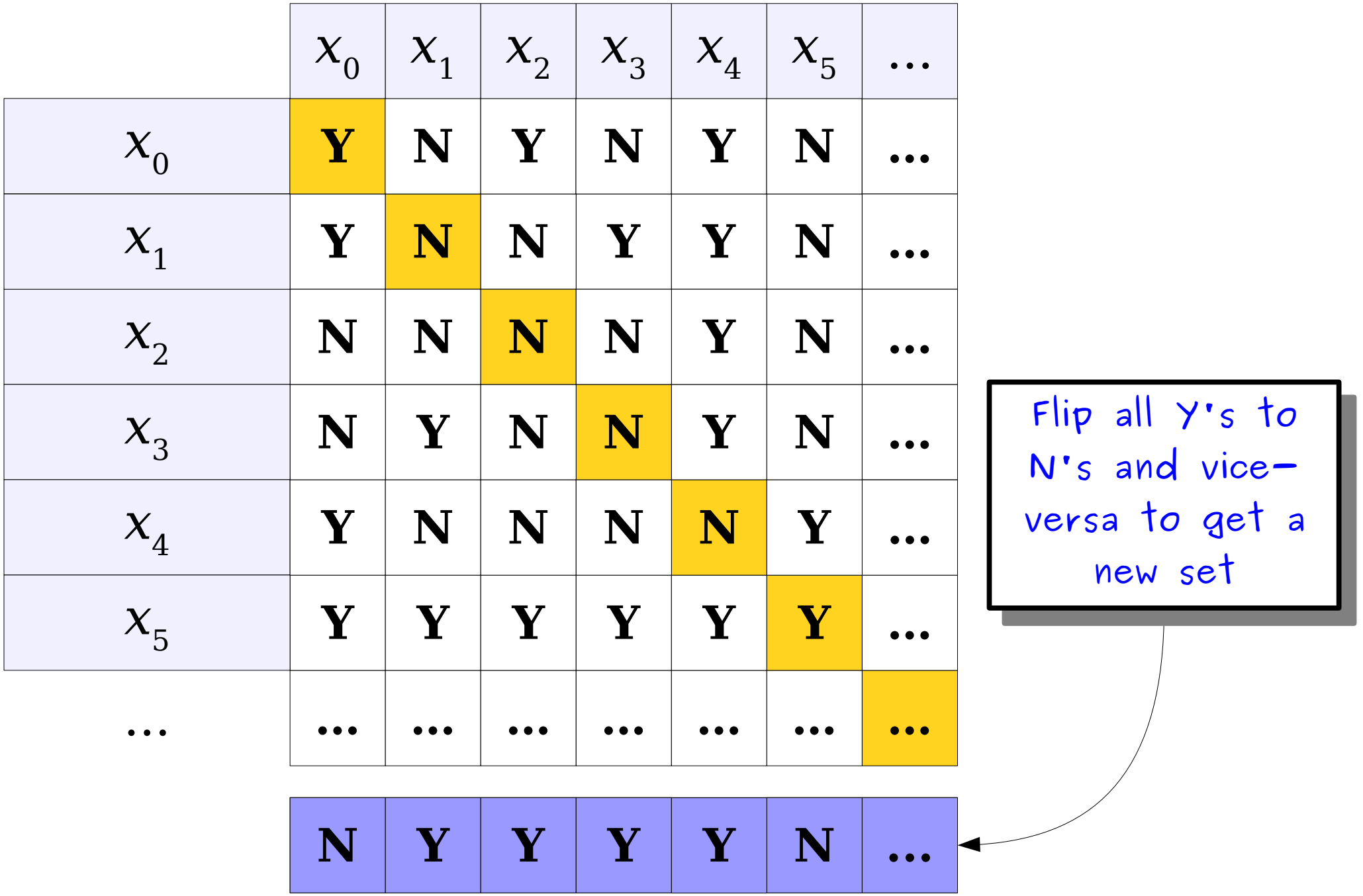
$x_3 \leftrightarrow \{ x_1, x_4, \ldots \}$

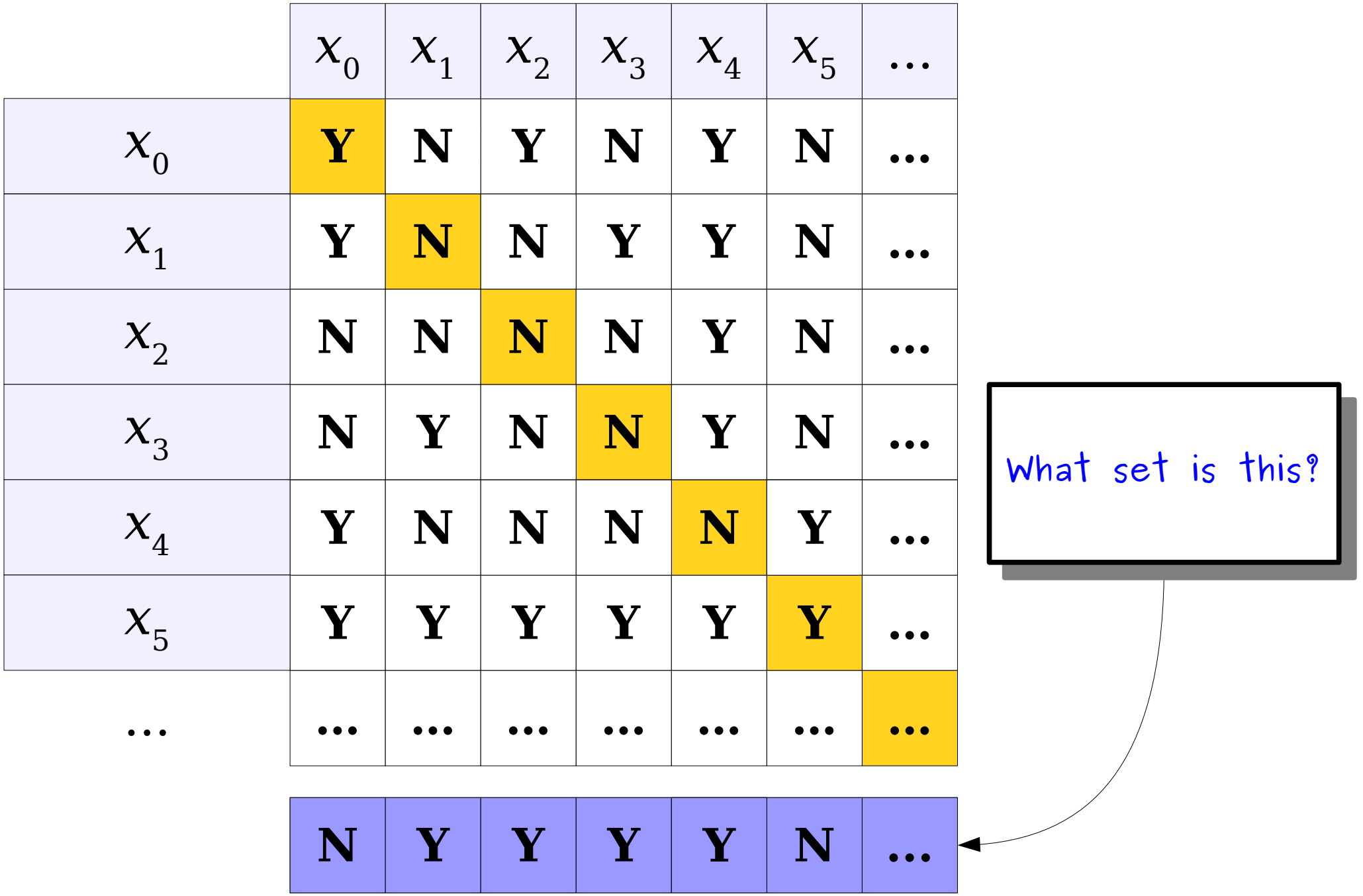
$x_4 \leftrightarrow \{ x_0, x_5, \ldots \}$

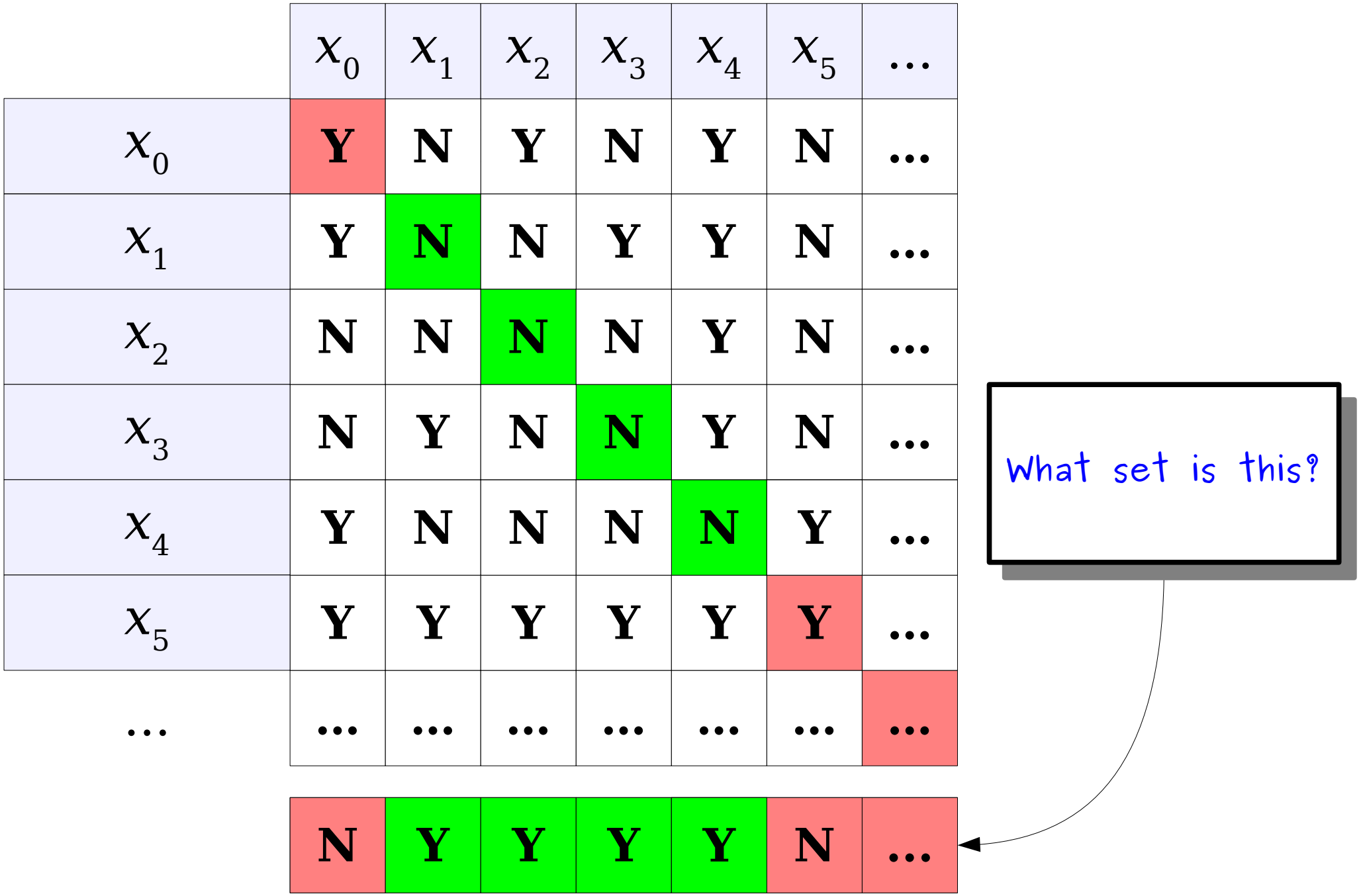
$x_5 \leftrightarrow \{ x_0, x_1, x_2, x_3, x_4, x_5, \ldots \}$

...

	x_0	x_1	x_2	x_3	x_4	x_5	…
x_0	**Y**	**N**	**Y**	**N**	**Y**	**N**	…
x_1	**Y**	**N**	**N**	**Y**	**Y**	**N**	…
x_2	**N**	**N**	**N**	**N**	**Y**	**N**	…
x_3	**N**	**Y**	**N**	**N**	**Y**	**N**	…
x_4	**Y**	**N**	**N**	**N**	**N**	**Y**	…
x_5	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	…
…	…	…	…	…	…	…	…


x_0 x_1 x_2 x_3 x_4 x_5 …
x_0 Y N Y N Y N …
x_1 Y N N Y Y N …
x_2 N N N N Y N …
x_3 N Y N N Y N …
x_4 Y N N N N Y …
x_5 Y Y Y Y Y Y …
… … … … … … … …
N Y Y Y Y N …
Flip all Y's to N's and vice-versa to get a new set


x_0 x_1 x_2 x_3 x_4 x_5 …
x_0 Y N Y N Y N …
x_1 Y N N Y Y N …
x_2 N N N N Y N …
x_3 N Y N N Y N …
x_4 Y N N N N Y …
x_5 Y Y Y Y Y Y …
… … … … … … … …
N Y Y Y Y N …
What set is this?


x_0 x_1 x_2 x_3 x_4 x_5 …
x_0 Y N Y N Y N …
x_1 Y N N Y Y N …
x_2 N N N N Y N …
x_3 N Y N N Y N …
x_4 Y N N N N Y …
x_5 Y Y Y Y Y Y …
… … … … … … … …
N Y Y Y Y N …
What set is this?

The Diagonal Set

- Let $f : S \to \wp(S)$ be an arbitrary function from S to $\wp(S)$.
- Define the set D as follows:

$$\boldsymbol{D = \{\, x \in S \mid x \notin f(x) \,\}}$$

("The set of all elements in S that aren't an element of the set they map to.")

- (A note on the notation: $x \notin f(x)$ means "the element x is not in the set $f(x)$," not "the element x is not mapped to by $f(x)$.")
- This is a formalization of the set we found in the previous picture.
- Using this choice of D, can we formally prove that no function $f : S \to \wp(S)$ is a bijection?

Theorem: If S is a set, then $|S| \neq |\wp(S)|$.

Proof: Let S be an arbitrary set. We will prove that $|S| \neq |\wp(S)|$ by showing that there are no bijections from S to $\wp(S)$.

Suppose for the sake of contradiction that there is a bijective function $f : S \to \wp(S)$. Starting with f, we define the set

$$D = \{ x \in S \mid x \notin f(x) \}. \qquad (1)$$

Since every element of D is also an element of S, we know that $D \subseteq S$, so $D \in \wp(S)$. Therefore, since f is surjective, we know that there is some $x \in S$ such that $f(x) = D$.

We can now ask under what conditions this element x happens to be an element of D. By definition of D, we know that

$$x \in D \text{ iff } x \notin f(x). \qquad (2)$$

By assumption, $f(x) = D$. Combined with (2), this tells us

$$x \in D \text{ iff } x \notin D. \qquad (3)$$

This is impossible. We have reached a contradiction, so our assumption must have been wrong. Therefore, there are no bijections between S and $\wp(S)$, and therefore $|S| \neq |\wp(S)|$, as required. ■

The Diagonal Argument

- ***This proof is tricky.*** It's one of the hardest proofs we're going to encounter over the course of this quarter.
- To help you wrap your head around how it works, we're going to ask you a few small questions about it on Problem Set Four.
- Don't panic if you don't get it immediately; you'll get a really good understanding of how it works if you play around with it.

Concluding the Proof

- We've just shown that $|S| \neq |\wp(S)|$ for any set S.
- To prove $|S| < |\wp(S)|$, we need to show that $|S| \leq |\wp(S)|$ by finding an injection from S to $\wp(S)$.
- Take $f : S \to \wp(S)$ defined as

$$\boldsymbol{f(x) = \{x\}}$$

- Good exercise: prove this function is injective.

Why All This Matters

- Proof by diagonalization is a powerful technique for showing two sets cannot have the same size.
- Can also be adapted for other purposes:
  - Finding specific problems that cannot be solved by computers.
  - Proving Gödel's Incompleteness Theorem.
  - Finding problems requiring some amount of computational resource to solve.
- We will return to this later in the quarter.

Time-Out for Announcements!

Edward Snowden Talk

- Edward Snowden (yes, *that* Edward Snowden) is the 2015 Symbolic Systems Distinguished Speaker.
- He'll be chatting over video with Prof. Ken Taylor and Prof. John Perry, and everything will be live-streamed into Cubberly Auditorium.
- Talk is May 15 at 12:00 noon.
- Interested? **Click here** to RSVP!

Problem Set Grading

- Problem Set Two is graded and will be returned electronically by the end of class.
- Problem Set Three checkpoints also are graded and will be returned electronically by the end of class.
- Have questions! Please ask!

Midterm Logistics

- First midterm exam is next **Thursday, April 30** from **7PM – 10PM**, location TBA.
- Covers material up through and including graphs, with a focus on topics covered in PS1 – PS3.
- Closed-book, closed-computer.
- You can bring one double-sided 8.5” × 11” sheet of notes with you when you take the exam.
- Need to take the exam at an alternate time? We're holding an alternate exam at 4PM on Thursday, April 30.
  - Please email us *immediately* if you need to take the exam at an alternate time.

Practice Midterm

- We will be holding a practice midterm exam next Monday, April 27 from 7PM – 10PM, location TBA.
- Completely optional, but highly recommended.
  - You don't want the actual midterm to be the first time you've had to solve proof-based math questions under time pressure!
- TAs will be available to answer your questions.
- Can't make the practice exam? We'll release it online as well.

Extra Practice Problems

- We'll be releasing sets of extra practice problems this week and early next week you can use to prepare for the exam.
- Need even more practice? Check out the CS103A website or read over the course notes.
- Feel free to stop by office hours with questions on these topics.

General Exam Thoughts

Your Questions

... next time

Back to CS103!

Changing Gears: The Pigeonhole Principle

The ***pigeonhole principle*** is the following:

If m objects are placed into n bins,
where $m > n$, then some bin contains
at least two objects.

(We sketched a proof in Lecture #02)

Why This Matters

- The pigeonhole principle can be used to show results must be true because they are “too big to fail.”
- Given a large enough number of objects with a bounded number of properties, eventually at least two of them will share a property.
- Can be used to prove some surprising results.

Using the Pigeonhole Principle

- To use the pigeonhole principle:
  - Find the m objects to distribute.
  - Find the $n < m$ buckets into which to distribute them.
  - Conclude by the pigeonhole principle that there must be two objects in some bucket.
- The details of how to proceed from there are specific to the particular proof you're doing.

A Surprising Application

Theorem: Suppose that every point in the real plane is colored either red or blue. Then for any distance $d > 0$, there are two points exactly distance d from one another that are the same color.

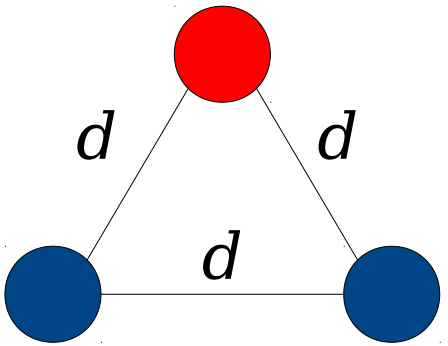
Thought: There are two colors here, so if we start picking points, we'll be dropping them into one of two buckets (red or blue).

How many points do we need to pick to guarantee that we get two of the same color?

A Surprising Application

Theorem: Suppose that every point in the real plane is colored either red or blue. Then for any distance $d > 0$, there are two points exactly distance d from one another that are the same color.

Any pair of these points is at distance d from one another. Since two must be the same color, there is a pair of points of the same color at distance d!



A Surprising Application

Theorem: Suppose that every point in the real plane is colored either red or blue. Then for any distance $d > 0$, there are two points exactly distance d from one another that are the same color.

Proof: Consider any equilateral triangle whose side lengths are d. Put this triangle anywhere in the plane. Because the triangle has three vertices and each point in the plane is only one of two different colors, by the pigeonhole principle at least two of the vertices must have the same color. These vertices are at distance d from each other, as required. ■

The Hadwiger-Nelson Problem

- No matter how you color the points of the plane, there will always be two points at distance 1 that are the same color.
- Relation to graph coloring:
  - Every point in the real plane is a node.
  - There's an edge between two points that are at distance exactly one.
- ***Question:*** What is the chromatic number of this graph? (That is, how many colors do you need to ensure no points at distance one are the same color?)
- This is the ***Hadwiger-Nelson problem***. It's known that the number is between 4 and 7, but no one knows for sure!

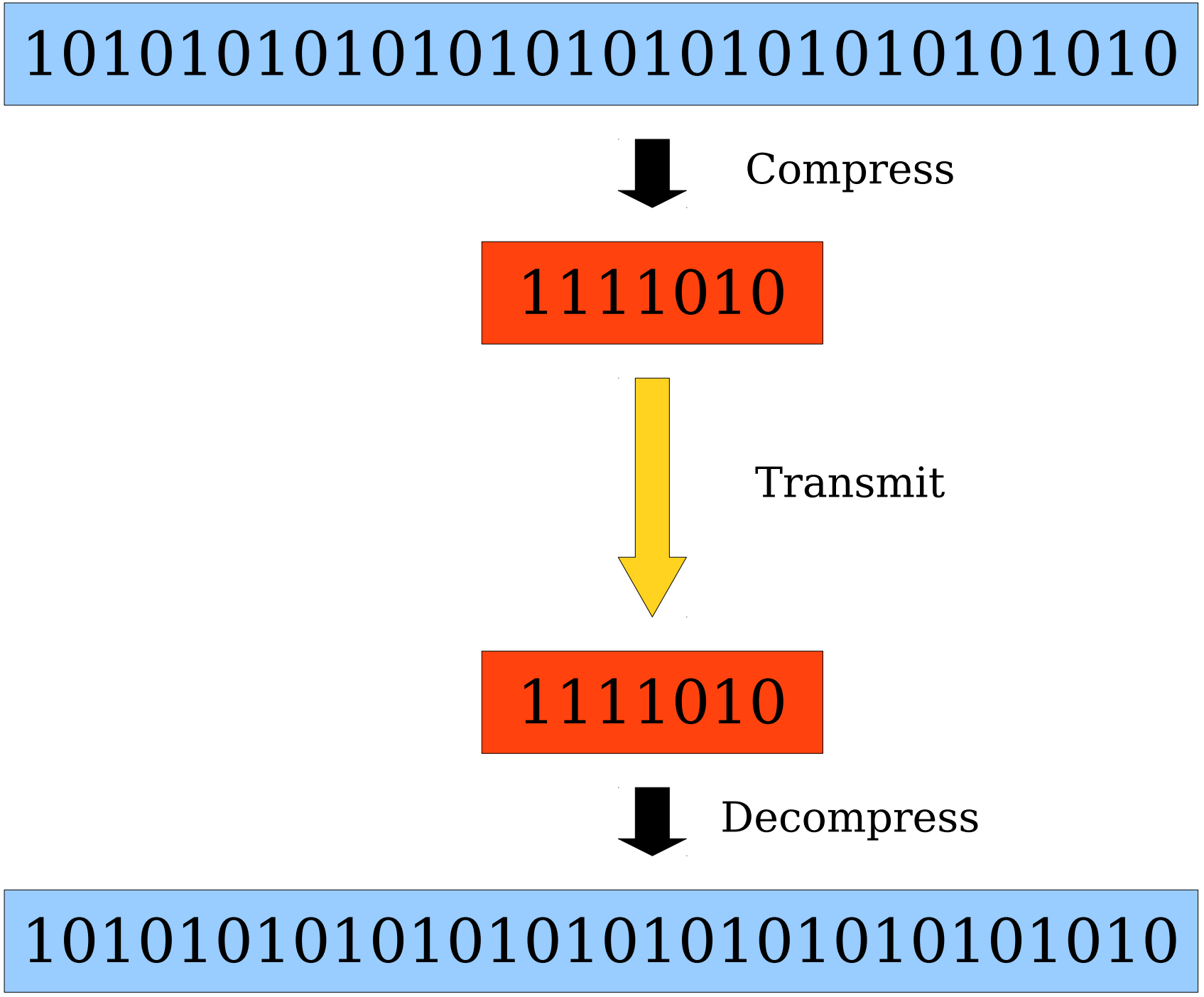
The Limits of Data Compression

Bitstrings

- A ***bitstring*** is a finite sequence of 0s and 1s.
- Examples:
  - 11011100
  - 010101010101
  - 0000
  - ε (the ***empty string***)
- There are 2^n bitstrings of length n.

Data Compression

- Inside a computer, all data are represented as sequences of 0s and 1s (bitstrings)
- To transfer data (across a network, on DVDs, on a flash drive, etc.), it is useful to reduce the number of 0s and 1s before transferring it.
- Most real-world data can be compressed by exploiting redundancies.
  - Text repeats common patterns (“the”, “and”, etc.)
  - Bitmap images use similar colors throughout the image.
- ***Idea:*** Replace each bitstring with a *shorter* bitstring that contains all the original information.
  - This is called ***lossless data compression***.


10101010101010101010101010101010
Compress
1111010
Transmit
1111010
Decompress
10101010101010101010101010101010

Lossless Data Compression

- In order to losslessly compress data, we need two functions:
  - A ***compression function*** C, and
  - A ***decompression function*** D.
- These functions must be inverses of one another: $D(C(x)) = x$.
  - Otherwise, we can't uniquely encode or decode some bitstring.
- ***Claim:*** if $D(C(x)) = x$ for all x, then C must be injective.

A Perfect Compression Function

- Ideally, the compressed version of a bitstring would always be shorter than the original bitstring.
- ***Question***: Can we find a lossless compression algorithm that always compresses a string into a shorter string?
- To handle the issue of the empty string (which can't get any shorter), let's assume we only care about strings of length at least 10.

A Counting Argument

- Let $\mathbb{B}^n$ be the set of bitstrings of length n, and $\mathbb{B}^{<n}$ be the set of bitstrings of length less than n.
- How many bitstrings of length n are there?
  - **Answer**: 2^n
- How many bitstrings of length *less than* n are there?
  - **Answer:** $2^0 + 2^1 + \ldots + 2^{n-1} = 2^n - 1$
- By the pigeonhole principle, no function from $\mathbb{B}^n$ to $\mathbb{B}^{<n}$ can be injective – at least two elements must collide!
- Since a perfect compression function would have to be an injection from $\mathbb{B}^n$ to $\mathbb{B}^{<n}$, ***there is no perfect compression function!***

Why this Result is Interesting

- Our result says that no matter how hard we try, it is ***impossible*** to compress every string into a shorter string.
- No matter how clever you are, you cannot write a lossless compression algorithm that always makes strings shorter.
- In practice, only highly redundant data can be compressed.
- The fields of ***information theory*** and ***Kolmogorov complexity*** explore the limits of compression; if you're interested, go explore!