CS 3313
Foundations of Computing:

Math Review:
Countable & Uncountable Sets
Diagonalization

http://gw-cs3313-2021.github.io

(Discete) Math Review

- Encoding integers…and enumeration (ordering)
- Cardinality of sets
    - Countable and Uncountable Sets

- Diagonalization technique (proof of contradiction)

Integers, Strings, and Other data

- Data types are important as a programming tool.
  - Program manipulates a data type…operations are defined on data types
- But at another level, there is only one type, which you may think of as integers or strings.
  - A string of 0’ and 1’s !
- *Key point: Strings that are programs are just another way to think about the same one data type.*

- Recall data types and encodings from Architecture course…..

Example: Text

- Strings of ASCII or Unicode characters can be thought of as binary strings, with 8 or 16 bits/character.
- Binary strings can be thought of as integers.

Enumeration..i.e., ordering of strings

- Enumeration/ordering: It makes sense to talk about "the i-th binary string."
  - Goal: to list all binary strings in some order
    - So we have the first string, second string, … n-th string, etc…
- Consider set of all strings over $\{0,1\}$
  - $\{ \lambda, 0, 1, 00, 10, 000, \ldots\ldots\ldots \}$
- Is the ordering just the decimal equivalent
  - Using the weighted positional binary representation of a decimal number ?
    - 101 is the number 5, etc.

Enumeration: Binary Strings to Integers

- There's a small glitch:
  - If you think simply of binary integers, then strings like 101, 0101, 00101,… all appear to be "the fifth string" since their decimal equivalent is the integer 5
- Fix by prepending a "1" to the string before converting to an integer (decimal equivalent).
  - Thus, 101 -> 1101, 0101 -> 10101, and 00101 -> 100101
  - And therefore 101, 0101, and 00101 are the 13^{th}, 21^{st}, and 37^{th} strings, respectively.
- { λ, 0, 1, 00, 01,…} become {1, 10, 11, 100, …}
  - λ is first string, 0 is second string, 1 is third string,…..

Example: Images

- Represent an image in (say) GIF.
- The GIF file is an ASCII string.
- Convert string to binary.
- Convert binary string to integer.
- Now we have a notion of "*the i-th image*."

Example: Enumerations of Proofs

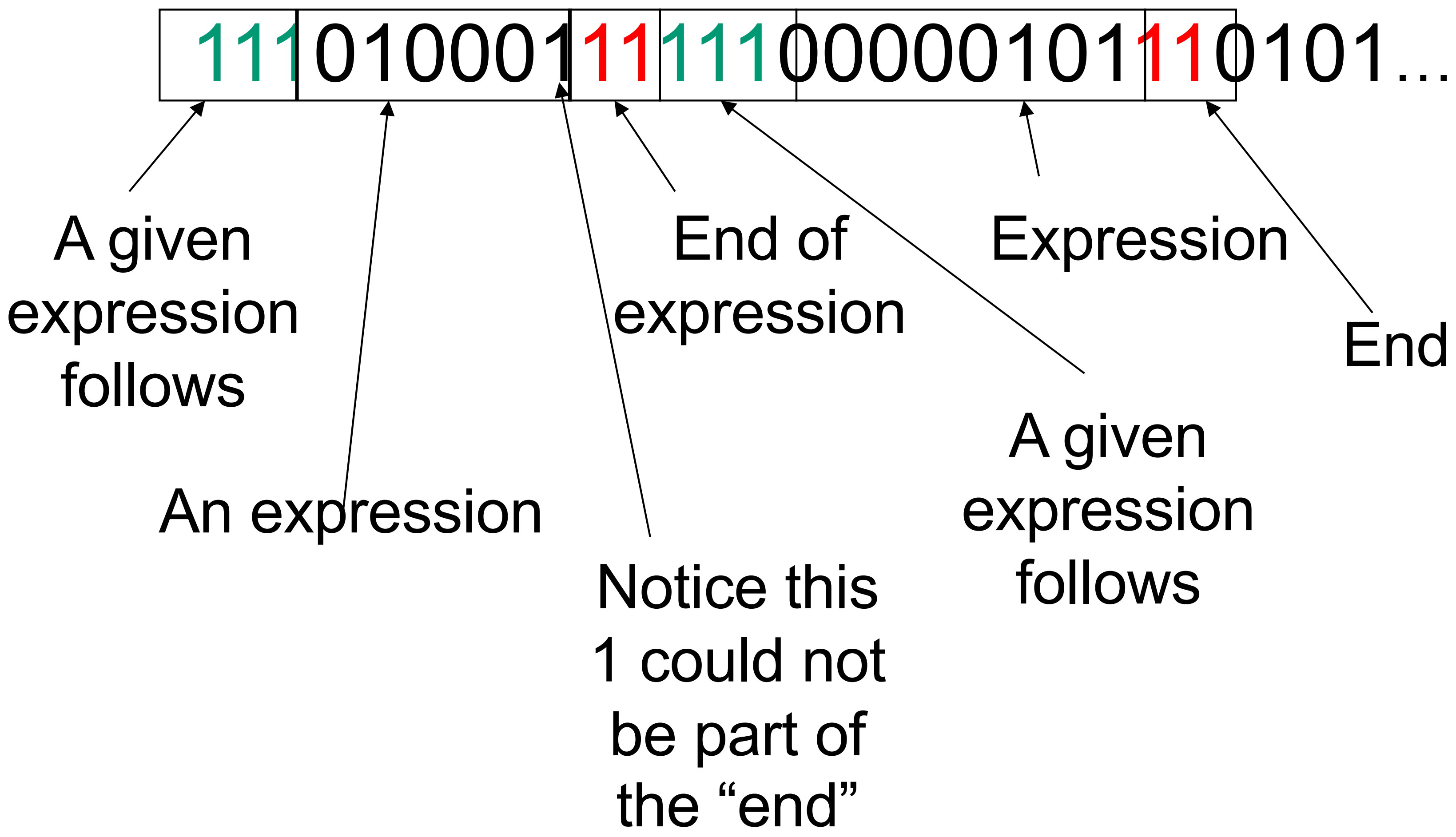
- A formal proof is a sequence of logical expressions, each of which follows from the ones before it….therefore:
- 1. Encode mathematical expressions of any kind in Unicode.
- 2. Convert expression to a binary string and then an integer.

Proofs – (2)

- But a proof is a sequence of expressions, so we need a way to *separate* them.
- Also, we need to indicate which expressions are given and which follow from previous expressions.
- Quick-and-dirty way to introduce new symbols into binary strings:
  1. Given a binary string, precede each bit by 0.
     - Example: 101 becomes 010001.
  2. Use strings of two or more 1's as the special symbols.
     - Example: 111 = "the following expression is given";
     - Example: 11 = "end of expression."

Key takeaway: remember this concept of using a specific pattern/string as a separator between fields

Example: Encoding Proofs



11101000111111000001011110101…

A given expression follows

An expression

Notice this 1 could not be part of the “end”

End of expression

A given expression follows

Expression

End

Example: Programs

- Programs are just another kind of data.
- Represent a program in ASCII.
- Convert to a binary string, then to an integer.
- Thus, it makes sense to talk about “the i-th program.”
- *Hmm…There aren’t all that many programs*.

What’s our takeaway?

- Languages, datatypes, programs, proofs can be encoded (as 0’s and 1’s) and *enumerated*
    - We can talk of the *i-th* program or the *i-th* proof etc.

- Next we formalize the notion of counting/enumeration….and the size of sets

Next….concept of Countable and diagonalization

- Arguments about the size of a set
- Diagonalization technique – to prove (by contradiction)

- Why review this……
- When we get to discussing properties of computability – specifically, undecidability (unsolvable problems), we need diagonalization technique to prove a problem is undecidable

Set Cardinality

- Cardinality of a set is the number of elements in the set
- Set can be finite or infinite
- Two sets A,B have the same cardinality if there is a one-to-one correspondence (mapping) from A to B
- A={0,1,2,3,4,5} and B={a,b,c,d,e,f}
  - f(0)=a, f(1)=b, f(2)=c, f(3)=d, f(4)=e, f(5)=f

Countable and Uncountable Sets

- Intuition: if we can arrange the elements of set in a manner where we can speak of “first element”, “second element”,etc.

- An infinite set A is countably infinite *if and only if* it has the same cardinality as the set of Natural numbers (positive integers)
  - There is a one to one correspondence (one to one and onto) from A to *N*.
- A set is countable iff it is finite or is countably infinite
- A set that is not countable is said to be uncountable
- Useful Theorems:
  - 1. If A ⊆ B and B is countable then A is countable
  - 2. If A ⊆ B and A is uncountable then B is uncountable

Enumerations

- An *enumeration* of a set is a 1-1 correspondence between the set and the positive integers.
- Thus, we have seen enumerations for strings, programs, proofs, and pairs of integers.

Countably Infinite Sets…Example

- Two sets have the same cardinality if there is a one to one correspondence between them
- Set of all integers Z is a countably infinite set: $f: Z \text{->} N$
  - $f(0) = 1$ $f(-i)= 2i$ $f(+i)= 2i+1$
  - $0 \text{->} 1, -1 \text{->} 2, 1 \text{->} 3, -2 \text{->} 4, \ldots$
  - *"ordering"* $0, -1, 1, -2, 2, -3, 3, \ldots..$
- Set of even integers Z_2 is countably infinite $f_2: N \text{->} Z_2$
  - $f_2 (n) = 2n$
- Set of primes P is countably infinite – f_3
  - f_3: (p) : p is the i-th prime.
    - Recall: we proved earlier that set of primes is infinite

Rational Numbers *Q*

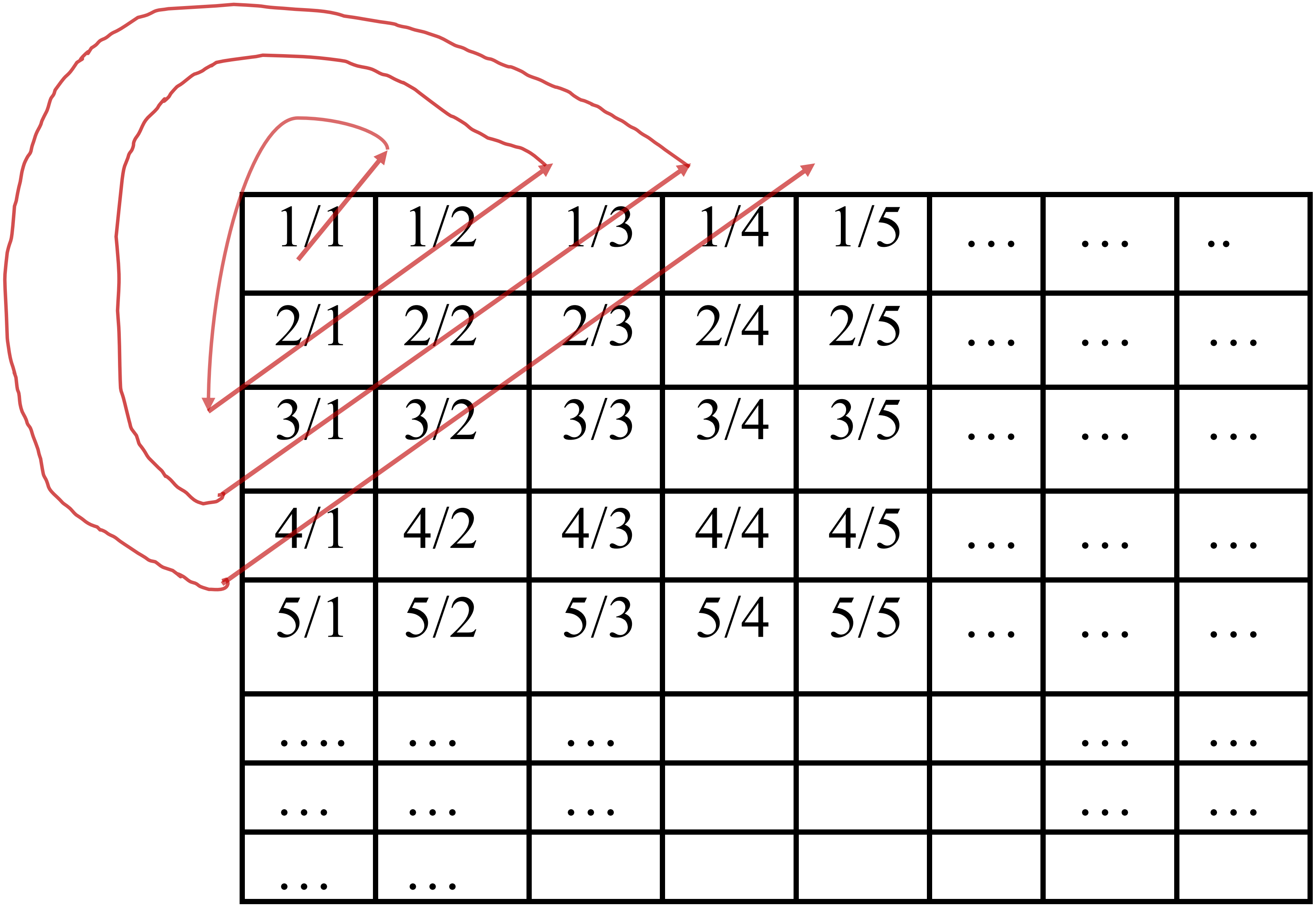
- Rational number *p/q* *p,q* are integers

- Theorem: Set of positive rational numbers Q is countable
- Intuition: list the rational numbers "in order"
  - Find a way to "label" the rational numbers to get the first rational number, the second rational number, etc.
    - For simplicity, let's work with *p,q* positive
  - Observe: We can view the number *p/q* as a pair of integers *[p,q]* and then order them first by sum and then by first component
    - [1,1], [2,1], [1,2], [3,1], [2,2], [1,3], [4,1], [3,2],..[1,4],[5,1]…

Ordering of (positive) Rational Numbers Q

- Make an infinite matrix containing all positive rationals.
  - The *i-th* row has all rational numbers with ***i*** in the numerator
  - The *j-th* column has all rational numbers with ***j*** in the denominator
  - Next turn this matrix into a list (ordered list)
- A bad way to do this would be to begin the list with all elements in the first row
  - Is not a good approach because first row is infinite and the list would never get to the second row!
- Instead, we list the elements on the diagonals
  - First element contain 1/1, i.e., p/q where $p+q=2$
  - Second diagonal contains numbers p/q where $p+q=3$
  - Third diagonal contains numbers p/q where $p+q=4$
  - ……

Ordering of (positive) Rational Numbers Q



1/1	1/2	1/3	1/4	1/5	…	…	..
2/1	2/2	2/3	2/4	2/5	…	…	…
3/1	3/2	3/3	3/4	3/5	…	…	…
4/1	4/2	4/3	4/4	4/5	…	…	…
5/1	5/2	5/3	5/4	5/5	…	…	…
….	…	…				…	…
…	…	…				…	…
…	…						

Rational Numbers *Q*

- Deciphering the ordering…
- Observe: We can view the number *p/q* as a pair of integers *[p,q]* and then order them first by sum and then by first component
  - [1,1], [2,1], [1,2], [3,1], [2,2], [1,3], [4,1], [3,2],..[1,4],[5,1]…
  - Known as filing order for a 2-tuple

Set of Real Numbers *R*: An countable set

- A real number has a decimal representation
  - π = 3.1415926…..
  - √2 = 1.4142135…
- A trick similar to rationals does not work here….but will use it to proof by contradiction
- Theorem: Set of real number *R* is uncountable
- Proof by contradiction….
  - The technique, developed by Cantor, is known as *diagonalization*

- *We will prove that the set of reals between 0 and 1 (0,1) is uncountable*
  - This is a subset of R, therefore R is uncountable

Uncountable Set…..concept of Diagonalization

- Proof by contradiction: Assume the set (0,1) is countable.
- Any number in this set can be represented as $0.d_1d_2d_3....$
- Suppose the set of countable, then we can write a list of real numbers and count them from 1 *to* n: $x_1, x_2, ...x_n,..$
  - *Each* $x_i = 0.d_1d_2.....$ *Where* d_i *is a digit*
- Notation: for each number x_i in the list (this appears in the *i-th* position) we can list the values of the digits a_{ij} in each position j after the decimal

$x_1 = 0.\ a_{11}\,a_{12}\,a_{13}\,a_{14}...$ Ex: $x_1 = 0.1342..$ then $a_{11}=1,\ a_{12}=3,\ a_{13}=4..$

$x_2 = 0.\ a_{21}\,a_{22}\,a_{23}\,a_{24}...$

..

$x_i = 0.a_{i1}\,a_{i2}\,a_{i3}....a_{ij}$

Uncountable set: (0,1) of reals

- Now view this concept as a matrix
    - Infinite number of columns and rows
    - *i-th* row is real number x_i
    - *j-th* column is value of a_{ij} – the value in *j-th* decimal position of x_i

a_{11}	a_{12}	a_{13}	a_{14}	a_{14}	…	…	..
a_{21}	a_{22}	a_{23}	a_{24}	a_{25}	…	…	…
a_{31}	a_{32}	a_{33}	a_{34}	a_{35}	…	…	…
a_{41}	a_{42}	a_{43}	a_{44}	a_{45}	…	…	…
…	…	…	…	…	…	…	…
a_{i1}	a_{i2}	a_{i3}				a_{ij}	…
…	…	…				…	…
…	…						

Example

- Illustrate construction (of matrix) with an example..
- $x_1 = 0.23117....\ x_2=0.11654...\ \ x_3=0.14142..\ \ x_4=0.40375...$

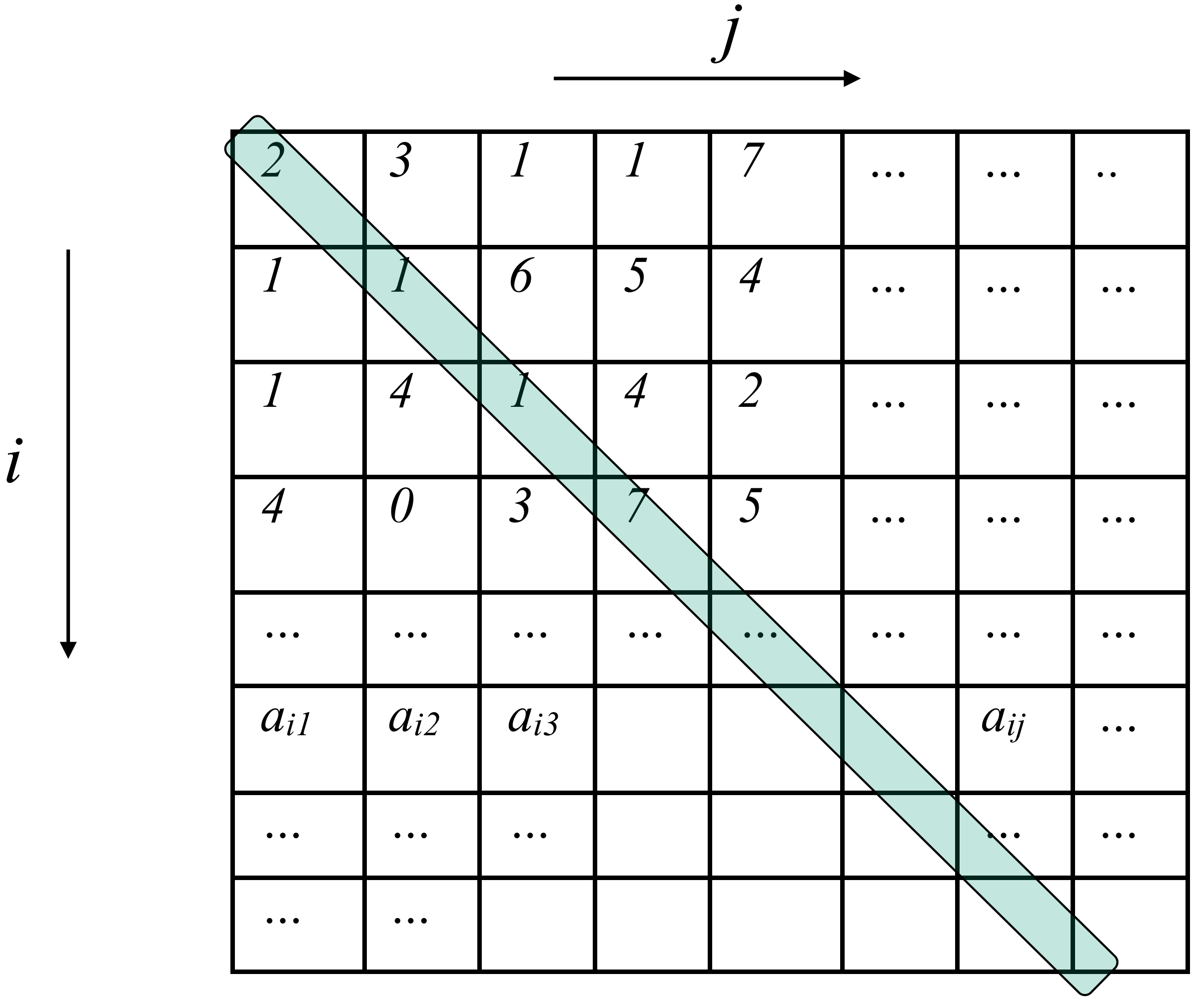
j →

i ↓

2	3	1	1	7	…	…	..
1	1	6	5	4	…	…	…
1	4	1	4	2	…	…	…
4	0	3	7	5	…	…	…
…	…	…	…	…	…	…	…
a_{i1}	a_{i2}	a_{i3}				a_{ij}	…
…	…	…				…	…
…	…						

Example

Consider the entries on the diagonal a_{ii} and construct y such that y is not in this (infinite) matrix – **contradiction** since we said every real number in (0,1) can be listed in this manner.



2	3	1	1	7	…	…	..
1	1	6	5	4	…	…	…
1	4	1	4	2	…	…	…
4	0	3	7	5	…	…	…
…	…	…	…	…	…	…	…
a_{i1}	a_{i2}	a_{i3}				a_{ij}	…
…	…	…				…	…
…	…						

- $x_1= 0.23117….$ $x_2=0.11654…$ $x_3=0.14142..$ $x_4=0.40375…$

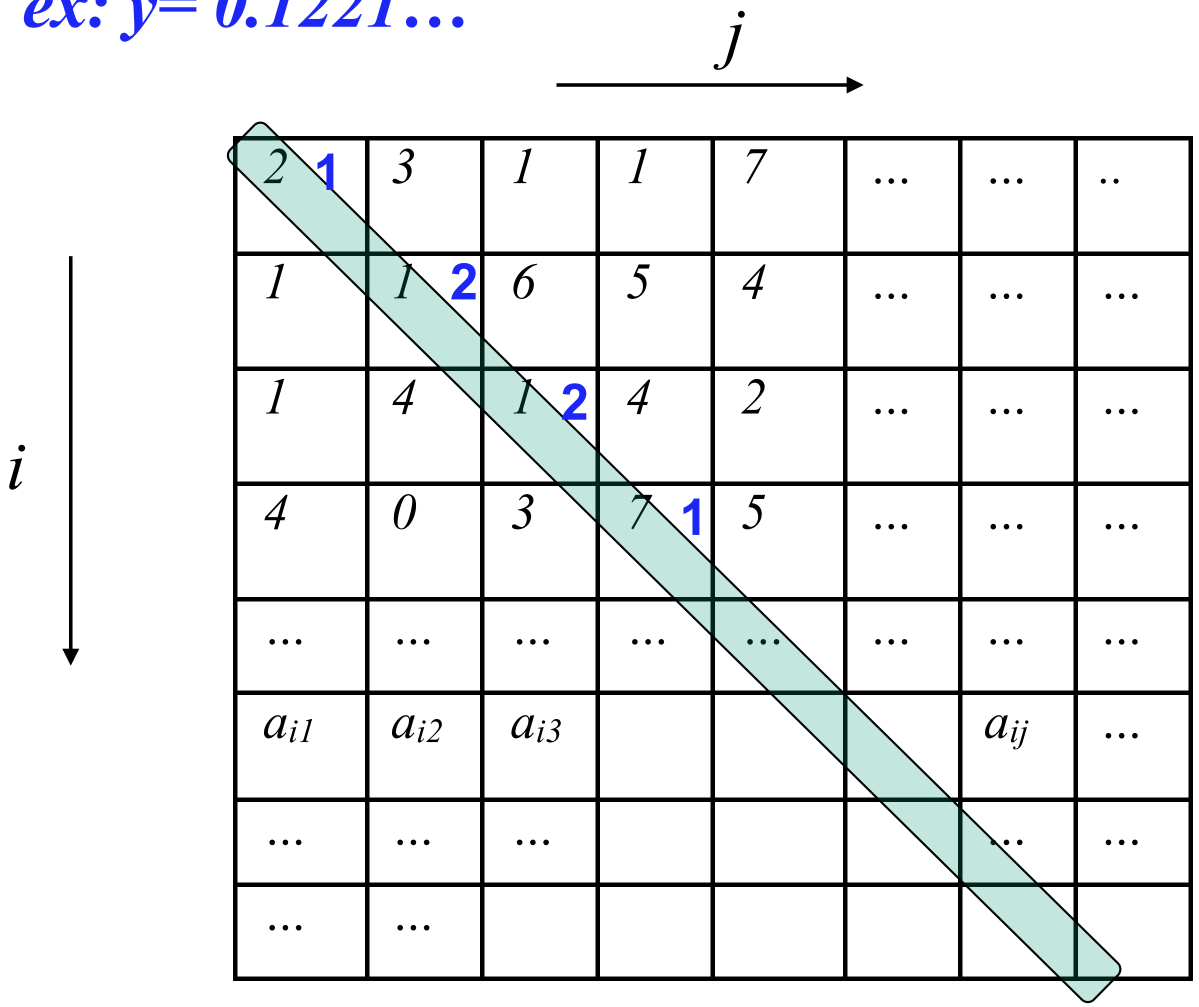
Example- Contradiction

If y is listed in this matrix then $y= x_k$, for some k, and $x_k= 0.a_{k1}\, a_{k2}\, a_{k3}..\, a_{kj}\ldots$

Pick y such that $a_{kj} \neq a_{jj}$ for all j

How? Ex: define $a_{kj} =2$ if $a_{jj} =1$ else $a_{kj} =1$

ex: $y= 0.1221\ldots$



2 1	3	1	1	7	…	…	..
1	1 2	6	5	4	…	…	…
1	4	1 2	4	2	…	…	…
4	0	3	7 1	5	…	…	…
…	…	…	…	…	…	…	…
a_{i1}	a_{i2}	a_{i3}				a_{ij}	…
…	…	…				…	…
…	…						

- $x_1= 0.23117\ldots.\ x_2=0.11654\ldots\ \ x_3=0.14142..\ \ x_4=0.40375\ldots$

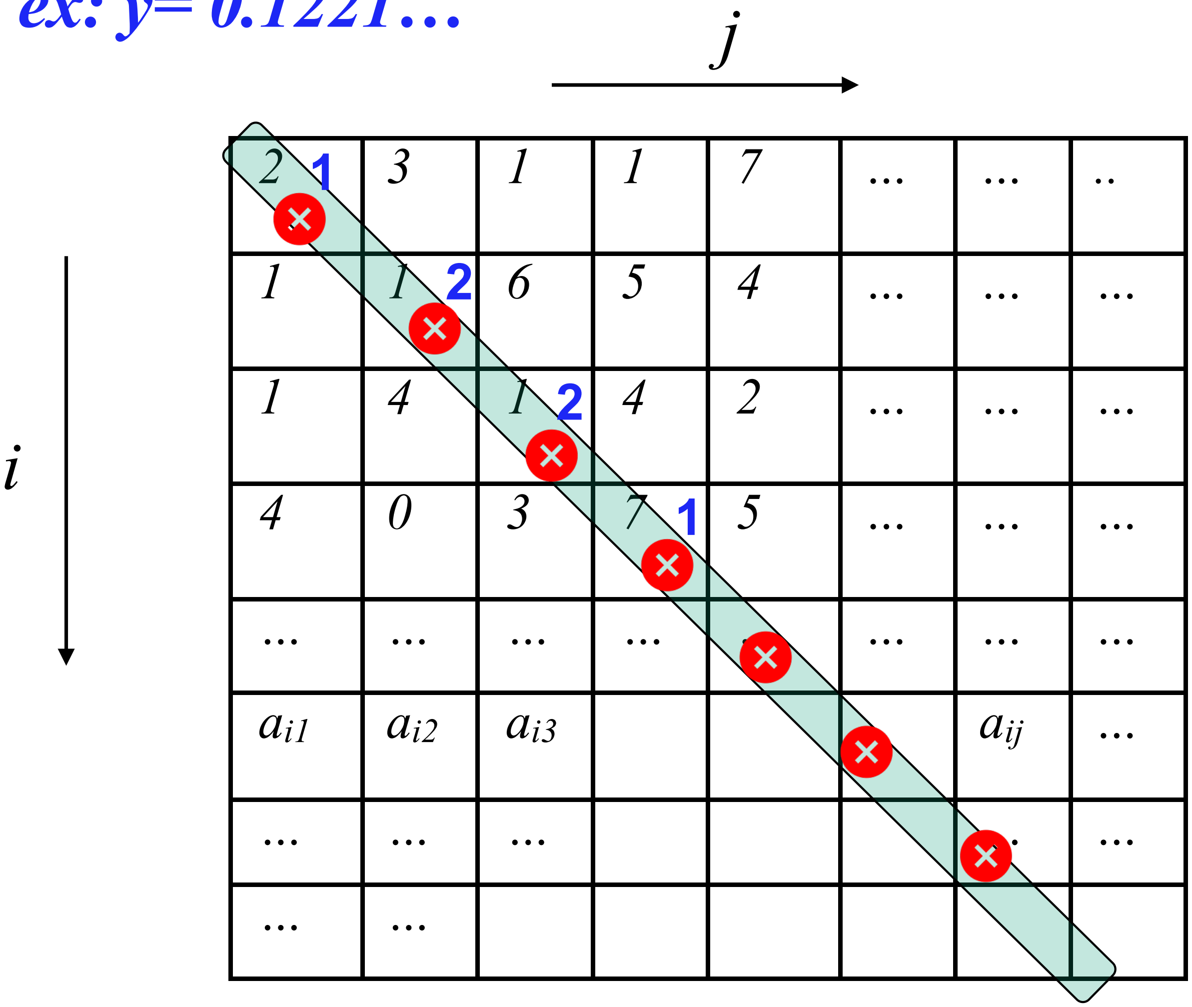
Example- Contradiction

If y is listed in this matrix then $y = x_k$, for some k, and $x_k = 0.a_{k1}\,a_{k2}\,a_{k3}..\,a_{kj}...$

Pick y such that $a_{kj} \neq a_{jj}$ for all j

How? Ex: define $a_{kj} = 2$ if $a_{jj} = 1$ else $a_{kj} = 1$

ex: $y = 0.1221...$



2 **1**	3	1	1	7	…	…	..
1	1 **2**	6	5	4	…	…	…
1	4	1 **2**	4	2	…	…	…
4	0	3	7 **1**	5	…	…	…
…	…	…	…		…	…	…
a_{i1}	a_{i2}	a_{i3}				a_{ij}	…
…	…	…				…	…
…	…						

0.1221… cannot be in the matrix:
cannot be in row 1
cannot be in row 2
cannot be in row 3
etc….

- $x_1 = 0.23117....\ x_2 = 0.11654...\ \ x_3 = 0.14142..\ \ x_4 = 0.40375...$

Proof - contradiction

- Consider the number y, where y= $0.a_{k1}\, a_{k2}\, a_{k3}..\, a_{kj}\ldots$ *where* $\boldsymbol{a_{kj} \neq a_{jj}}$ ***for all j***
- As one instance: pick $a_{kj} = 2$ *if* $a_{jj} = 1$ else $a_{kj} = 1$
- **Claim**: this number y cannot appear in the matrix as any x_k
- Proof: if $y = x_k$ for some k, then we have at least one decimal position j where a_{kj} is not equal to value in row k, column j in the matrix. Contradiction
  - Example: ***y= 0.1221…***
    - *Cannot be in the matrix…cannot be in any row because at least one column j (the diagonal entry) the value is not the same as value in the complete matrix.*

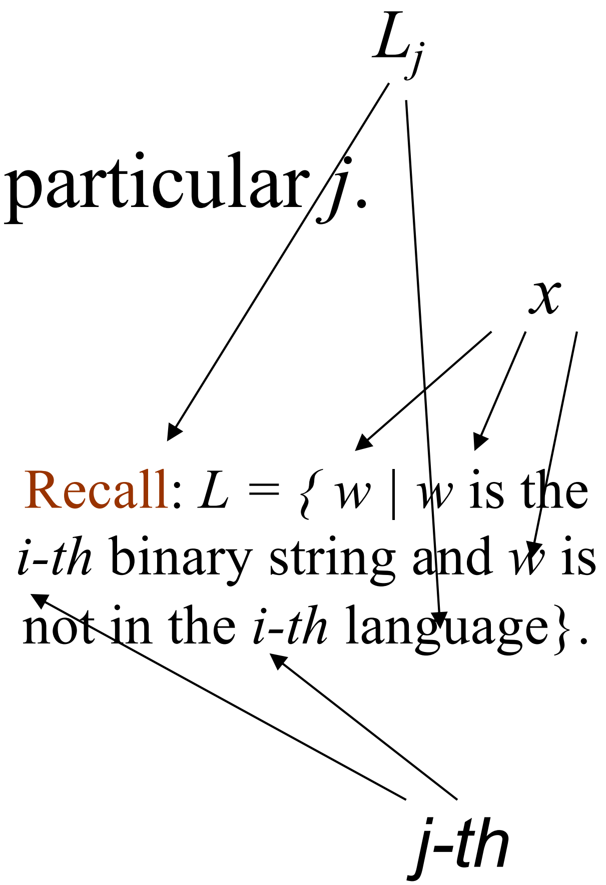
This type of proof construction – diagonalization is due to Cantor

Another example of an uncountable set: How Many Languages?

- Are the languages over {0,1} countable?
  - Recall: A language over {0,1} is a subset of {0,1}*
    - Set of strings over {0,1}
- No – here's a proof.
- Suppose we could enumerate all languages over {0,1} and talk about "the *i-th* language."
- Consider the language $L = \{ w \mid w$ is the *i-th* binary string and w is not in the *i-th* language$\}$.
  - We discussed at the start of the lab, one way to enumerate binary strings
    - So we can talk about the *i-th* binary string

Proof – Continued

- Clearly, L is a language over $\{0,1\}$.
- Thus, it is the *j-th* language for some particular j.
- Let x be the j-th string.
- Is x *in* L?
  - If so, x is not in L by definition of L.
  - If not, then x is in L by definition of L.



Recall: $L = \{ w \mid w$ is the *i-th* binary string and w is not in the *i-th* language$\}$.

Diagonalization Picture

Imagine an in finite matrix:

rows correspond to languages and columns to strings

a 1 in the entry for row *i*, column *j* means *j-th* string is in the *i-th* language

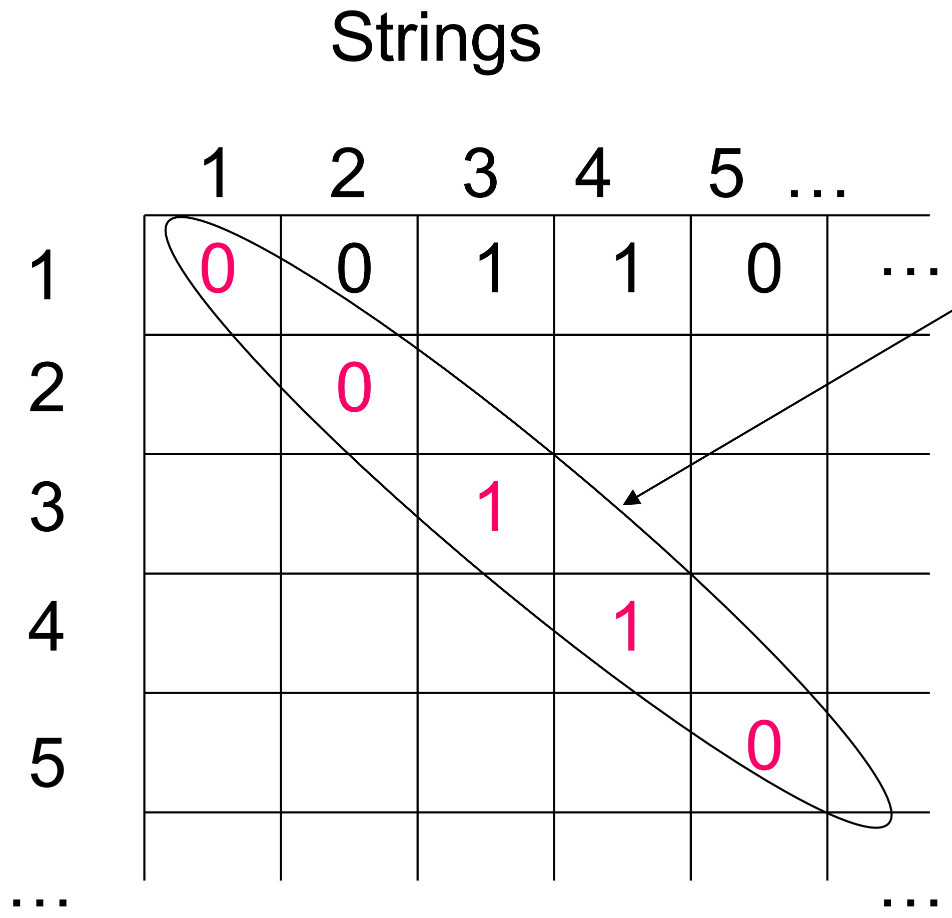
If we could enumerate languages, we could create such a table

Strings

		1	2	3	4	5	…
	1	1	0	1	1	0	…
	2		1				
Languages	3			0			
	4				0		
	5					1	
	…						…

Diagonalization Picture

Flip each
diagonal
entry



	1	2	3	4	5	…
1	0	0	1	1	0	…
2		0				
3			1			
4				1		
5					0	
…						…

Can’t be
a row –
it disagrees
in an entry
of each row.
Disagrees with
i-th row in
i-th position

Proof – Concluded

- We have a contradiction: x is neither in L nor not in L, so our sole assumption (that there was an enumeration of the languages) is wrong.
- Comment: This is really bad; there are more languages than programs.
- E.g., there are languages with no membership algorithm.

Why is this proof important:
this proof is used to provide a proof of a problem that cannot be solved by a turing machine !!