CS 4510 Automata and Complexity	3/30/2022
Lecture 15	
Lecturer: Frederic Faulkner	*Scribe(s):*

1 To Infinity and Beyond

Which would you say that there are more of, positive integers or positive even integers? The question might even sound silly; all the positive even integers are already included in the positive integers. Surely a set is larger than one of its subsets? On the other hand, we know each set is infinitely large. Infinity doesn't behave nicely as a number. Does it even make sense to compare the "sizes" of two infinite sets?

Yes it does! In the early 1900's, George Cantor pioneered the foundations of the mathematical framework for discussing infinite sets. In particular, he invented a method known as "diagonalization" which plays an important role in complexity theory. We will use it by the end of the lecture to show that there is at least one language which cannot be recognized by any Turing machine.

1.1 Countable Sets

How would you prove that two, say, bowls of fruit contain the same number of objects?

Presumably you would just count them each, and show that you end up with the same number each time. However, there is another proof you can do that doesn't involve advanced mathematical concepts like "numbers" and "counting". It is simple, direct, and visual: you could just pair the objects up, one from each set, and show that there are none left over.

The "count and compare numbers" technique doesn't work for infinite sets, but the "pair things up and check for leftovers" technique works just fine! The following is a proof that there is exactly the same number of positive even integers as positive integers:

1	2
2	4
3	6
4	8
5	10
...	...

The left hand column contains every positive integer, while the right hand column contains every positive even integer. Each element of each set shows up in the table once, and there are no leftovers, so the two sets are the same size.

Using mathematical language, what we have done is establish a *bijection* between the two sets. This leads to the following definitions.

Recall. A bijection is a function that is both onto and one-one.

Definition 1 *We say that two sets have the same cardinality if we can establish a bijection between them.*

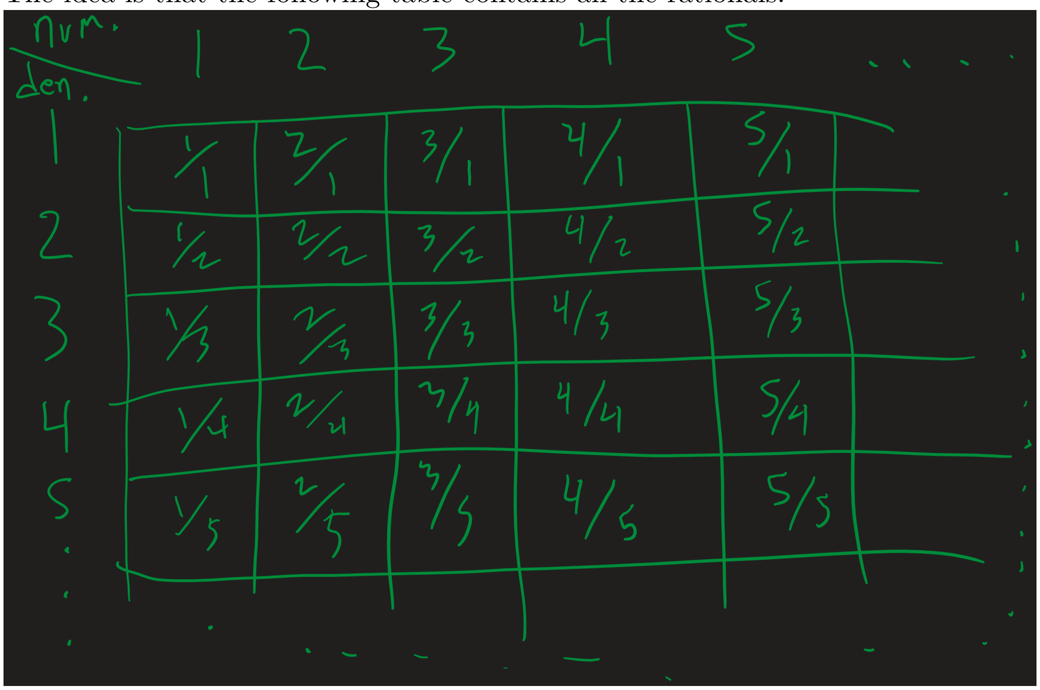
Definition 2 *We say a set is countable if it is finite or has the same cardinality as the positive integers.*

So the even positive integers are countable. The name makes sense: countable sets are the ones where you can order all the elements in a line and then "count" them, with the caveat that you could never stop counting. But you could count them, and every element would get a unique number.

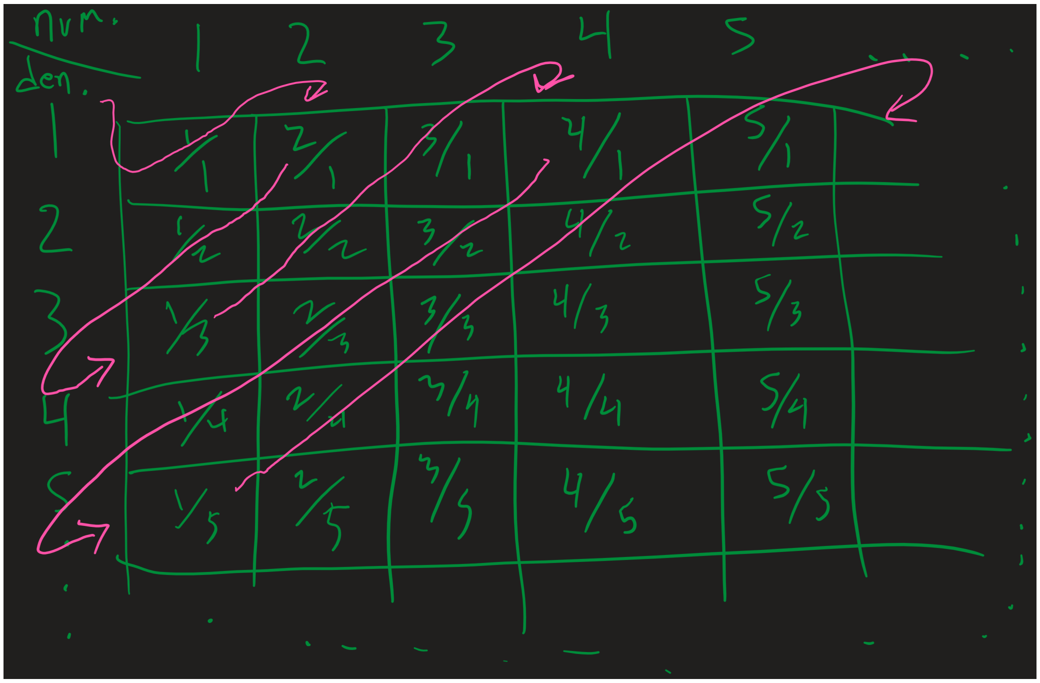
With some thought, you should be able to see that the set of all integers (positive, negative, and zero) is countable.

What may be less obvious is that the set of rational numbers (fractions) is also countable. After all, how would you order them? You cannot put them in ascending order! (What is the "next" rational after 1/2? In between any two rationals on the number line there are infinitely many other rationals.) In fact, the result was surprising to many when Cantor published it.

The idea is that the following table contains all the rationals.



You can traverse it along the diagonals to order them (skipping any that have already appeared):



This allows us to list the rationals like so (note that the fifth element in the list skips $\frac{2}{2}$ because $\frac{1}{1}$ has already appeared):

1	$\frac{1}{1}$
2	$\frac{2}{1}$
3	$\frac{1}{2}$
4	$\frac{1}{3}$
5	$\frac{3}{1}$
6	$\frac{4}{1}$
...	...

Since we can list out every rational, the set of rationals is countable. At this point, you might be wondering whether all sets are countable. After all, it doesn't seem unreasonable to think that perhaps you could list out the elements of any set. That you cannot do so is one of the Cantor's major results.

1.2 Uncountable sets

Definition 3 *A set is uncountable if it is not countable.*

Theorem 4 *The set of real numbers is uncountable.*

Proof.

We proceed by contradiction. Suppose that the real numbers are countable. Then you could list them out. The list might look something like the following:

1	0.1234567891011...
2	3.1415926535...
3	1.11111111111
4	7.13282993413822....
5	11.00000000000000....
...	...

Perhaps you stayed up all night and wrote this list yourself, and you are certain that you have remembered to include every real number. But you are wrong! There is at least one real number that you have missed, which we will call x. We define x to be the following: the n'th digit of x is a 2 if n'th digit of the n'th item on your list is a 1; else the n'th digit is a 1.

In other words, walk down the diagonal of your table, and create x by doing something different than what you see.

1	0.**1** 234567891011...
2	3.1 **4** 15926535...
3	1.11 **1** 11111111
4	7.132 **8** 2993413822....
5	11.0000 **0** 000000000....
...	...
	x = 0.21211....

We claim that x is not in your list. What position could it be at? It cannot be at position 87, because x has a different digit 87 places after the decimal place than the number in your list. (If #87 has a 1 in the 87th place after the decimal place, x has a 2 there instead by definition; if #87 has anything that is not a 1 there, then x has a 1.) So your "list" of all the real numbers was a not list of all the real numbers after all, since it is missing x. (In fact, it is missing infinitely many real numbers; can you say why?)

Any time that you try to pair up the positive integers and the reals, there are always some reals (infinitely many, in fact) that are leftover. So there are more real numbers than positive integers, and the reals are thus uncountable.

2 Diagonalization and Turing machines

Theorem 5 *There is some language that is not recognized (or decided) by any Turing machine.*

Proof. The crux of this proof is that there are uncountably many languages, but only countably many Turing machines. Since there are more languages then Turing machines, some languages have no corresponding Turing machine.

First, you should convince yourself that you can represent a Turing machine by a binary string. But then you can order all of the Turing machines by writing them as binary strings and ordering them first by length and then lexicographically (ε, 0, 1, 00, 01, 10, 11, 000, 001, 010, etc.) dropping out any that are not valid machines. Let M_1, M_2, etc. be the list of Turing machines.

We can now build a table with all the machines going down the left and inputs going across the top. In each cell we mark whether that machine accepts that input. For example, the table might look like this:

	ε	0	1	00	01	10	...
M_1	No	No	No	Yes	Yes	Yes	
M_2	No	Yes	Yes	Yes	Yes	Yes	
M_3	Yes	No	Yes	No	Yes	No	
M_4	No	No	No	No	No	No	
...	...						

Once again, we can find a language that is not in this table. Let us create L, by walking down the diagonal and adding the i'th binary string to L if and only if it is not accepted by M_i. Since M_1 rejects ε we put it in L. However, 0 and 1 are accepted by M_2 and M_3 so they are not added to L. 00 is rejected by M_4 so it is added to L and so on.

	ε	0	1	00	01	10	...
M_1	**No**	No	No	Yes	Yes	Yes	
M_2	No	**Yes**	Yes	Yes	Yes	Yes	
M_3	Yes	No	**Yes**	No	Yes	No	
M_4	No	No	No	**No**	No	No	
...	...						

$$\mathrm{L} = \{\varepsilon, 00, ...\}$$

No Turing machine accepts L, because every Turing machine appears in the list on the left hand side of the table, but L is different from the language of each machine in the table.

(For example, L is not the language of M_5, because if M_5 accepts 01, 01 is not in L, while if M_5 does not accept 01, then 01 is in L.)

This is not a constructive proof, since it shows that *some* language is not recognizable but it doesn't say which one. Next class we will discuss some specific non-recognizable languages.