Lecture 2

Yash Mehan

21 Aug, 2021

Idea for today: are there problems that computers cant solve? How many?

If at all a problem is a computational problem we can pose it as membership queries in languages

Membership query: Is this certain thing a member of this certain subset? Gives 0 for no, 1 for yes.

1. Suppose a problem is `is 2 a prime number`. then the output is either a 0 or a 1. it can framed as membership query: `is 2 a member of the set of prime numbers?` This is a bit stretch, this is a decision problem, and this point shows that all decision problems can essentially be restated as membership problems.
2. Suppose there is a sorting problem: `given a list of numbers, sort them`, we can make it a membership query too: `given the input list [3, 1, 2], is`[1, 3, 2] a member of the set conataining the correct answer?`set containing the correct answer will itself be`{[1,2,3]}`The answer for this problem will be 0, because`[1,3,2]`is not a member of the set`{[1,2,3]}‘.
3. a problem can be rounding off a number, i.e. `round off 3.14 to the nearest unit's digit` the answer in this case is `3`, but we can model it into another question: `is the one's digit in binary representation of 3.14 rounded off to nearest units digit 1?` The answer is this case is yes, i.e. `1`, similarly, this can be asked for the `2s place, is it 1?` since $\lfloor 3.14 \rfloor = 3 = (11)_2$. Thus a problem can give multiple bits as output, and each bit can be made a to a problem itself.

Language is collection of all finite-lengthed 0-1 bitstrings.

subset of infinitely long bitstrings of 0 and 1, i.e. $\{0,1\}^*$

Countability and computability

In due course of the lecture it will turn out that there are uncountably many computational problems, but there are only countably many programs, so there must be uncountably many unsolvable problems.

Countable Sets

An infinite set S is countable if there exists a bijection from $f : \mathbb{N} \to S$

Intuition: Storing data in an array: this analogy gives a feel of indices ($\in \mathbb{Z}$) being mapped to the content stored at the location in the array. If we are asked to retrieve the content at say, 42nd index, and it makes sense, i.e. those elements have been/can be *enumerated* and indexed, then it is a countable set.

A set is uncountable if it's not countable.

Plan

1. We start by proving that number of computer programs is countable.
2. Prove that number of problems is uncountable

Part 1

Number of computer programs is countable

There is no loss in generality in working with, say, C programs. Any and every computer program can be made to work in C. Turing machine has 1 instruction. yet we can claim the same as above.

C programs are just some binary strings, not all binary strings are C programs. A program can be represented as a set of 0s and 1s. And we can show that the set of all finite length 0 1 bitstrings, is countable. The set of infinite length bitstrings won't be countable, though, but then we don't have computer programs that are infinitely long, and this was one of the hypothesis we started with (in this course), because we cannot store infinitely long programs in finite amount of memory.

Proof of countability of $\{0, 1\}^*$

natural number mapped to the strings (in a sorted lexicographical order)

$1 \to$ `\0` (length 0 string)

$2 \to 0$

$3 \to 1$

$4 \to 00$

$5 \to 01$

$6 \to 10$

$7 \to 11$

$8 \to 000$

and so on

You can give me any finite-lengthed string of 0 and 1 and I can tell you which index it corresponds to, and you can give me any index and I can tell what string it maps to. Although observing the pattern and ascertaining the bijection function is fairly simple and obvious, but we don't need to.

So this means number of C programs / computer programs is countable.

Part 2

Number of problems is uncountable

A problem is defined as a language, which was itself $\{0,1\}^*$. Then what is the set of all possible problems? $\mathbf{P}(\{0,1\}^*)$ The total number of problems exactly equals the cardinality of this power set. ($\mathbf{P}$ is the powerset). As it turn out, $\mathbf{P}(\{0,1\}^*)$ is uncountable. Proof:

Cantor's diagonalisation and showing a contradiction.

Each member of the power set is basically a combination of binary strings, sourced from the aforementioned $\{0,1\}^*$.

Thus for each combination, we can list out the entire $\{0,1\}^*$ and attach a 0 if that string isn't to be included in a member of the subset, or a 1 if that string is.

e.g. $\{0,10,11\}$ can be a member of the powerset $\mathbf{P}(\{0,1\}^*)$ and can be shown as

```
1 emptystring => 0 (not present in {0,10,11})
2 0 => 1 is present in {0,10,11}
3 1 => 0
4 00 => 0
5 01 => 0
6 10 => 1
7 11 => 1
8 000 => 0
9 001 => 0
```

so $\{0,10,11\}$ can be denoted by another bitstring (lets call it `presencestring`) 0100011000000000....

For all such members a presence string can be calculated. Assume that all these `presencestrings` are countable as well. So there must exist a bijection from $\mathbb{N}$ to this set pf `presencestrings` Now using Cantor's diagonalisation argument.

For creating the new `presencestring`s, **flip the bit** from a previously existing `presencestring` to be sure that our newly created `presencestring` is different from all the already existing `presencestring`s.

There is always a new `presencestring`, which is unaccounted for, because each `presencestring` was assigned a natural number. That means there are more `presencestrings` i.e. images for the mapping than inputs i.e. from $\mathbb{N}$ (which were countable). so $\mathbf{P}(\{0,1\}^*)$ is uncountable.

There are more computational problems than there are programs, but can you give an example of one such unsolvable problem?

Or maybe those unsolvable program aren't realizable in real life anyway?

> Write a program which takes in 2 programs and checks whether these programs solve the same problem or not.