37181: WEEK 6: COUNTABLE, BIG O, PIGEONHOLE PRINCIPLE

A/Prof Murray Elder, UTS

Wednesday 28 August 2019

PLAN

- Countable
- Big O
- comparing speed of algorithms
- pigeonhole principle

COUNTABLE

Definition: if a set X is in bijection with a finite set or $\mathbb{N}$ then we say it is *countable*.

Last lecture and workshop we proved $\mathbb{Z}, \mathbb{Q}, 2\mathbb{N}$ are countable: recall:

Definition: if a set X is in bijection with $\mathbb{N}$ then we also say it is *countably infinite*.

Is there any set that is "bigger" than $\mathbb{N}$?

BIJECTION

Claim: $\mathbb{N}$ and $\mathbb{R}$ are not in bijection. That is, $\mathbb{R}$ is "strictly bigger" than $\mathbb{N}$.

Proof:

PROOF

Here is my proof again, in case my first “live” attempt is too messy.

Suppose (for contradiction) that $\mathbb{R}$ is the same size as $\mathbb{N}$.

This means that there is some bijection from one set to the other.

Let’s suppose this bijection is $f : \mathbb{N} \to \mathbb{R}$, and write $f(0), f(1), f(2), \ldots$

for example

$$\begin{aligned} f(0) &= 376.72333\ldots \\ f(1) &= -0.1111111\ldots \\ f(2) &= -0.5432100\ldots \\ f(3) &= 17.0000000\ldots \\ &\vdots \end{aligned}$$

Now I will tell you a real number that f has missed. So f is not onto. (Contradiction).

Here is my real number. It is the decimal number $0.x_0x_1x_2x_3x_4\ldots$ where I have to tell you what each x_i is.

For each $i \in \mathbb{N}$, I choose x_i to be a digit that is *not* the i-th digit in $f(i)$. (Say add 1 to it and reduce mod 10).

Now, tell me where my number is on the list?

This famous proof (due to Cantor) is known as a *diagonalisation argument*.

The same idea is used to prove that the Halting Problem is undecidable (see 41080 Theory of Computing Science).

So we have $\mathbb{N}, \mathbb{Z}, \mathbb{Q}$ are all countably infinite, and $\mathbb{R}$ is uncountable.

Question: is there any set of size strictly between these?

TEAM ASSIGNMENT

For any set A (think infinite), you will prove on the team assignment that A is *not* in bijection with $\mathscr{P}(A)$.

Note, it is possible to think of $\mathbb{R}$ as (in bijection with) $\mathscr{P}(\mathbb{N})$: idea:

So what your assignment question will imply is quite amazing: there are many different sizes of infinity.

```
power_slow(a real; n positive integer)
x = 1.0
for i = 1 to n do
    x = x*a
return x
```

power_slow(3, 6):

i	x

```
power_slow(a real; n positive integer)
x = 1.0
for i = 1 to n do
    x = x*a
return x
```

power_slow(3, 100):

i	x

```
power_fast(a real; n positive integer)
x = 1.0
i = n
while i > 0 do
    if i is odd then
        x = x*a
    i = floor(i/2)
    if i > 0 then
        a = a*a
return x
```

power_fast(3, 6):

x	i	a

```
power_fast(a real; n positive integer)
x = 1.0
i = n
while i > 0 do
    if i is odd then
        x = x*a
    i = floor(i/2)
    if i > 0 then
        a = a*a
return x
```

power_fast(3, 100):

x	i	a

How can we *formally* capture the idea that the second algorithm is *faster*? Can we estimate that on input of size n, the algorithm will take roughly some function of n steps?

Look back at the two algorithms and try to roughly guess a function for each one.

Luckily, someone already thought of how to capture the idea of algorithm speed/complexity, using this definition.

Definition

Let $f, g : \mathbb{N}_+ \to \mathbb{R}$. We say that g *dominates* f if there exist constants $m \in \mathbb{R}^+$ and $k \in \mathbb{N}_+$ such that

$$|f(n)| \leqslant m|g(n)|$$

for all $n \in \mathbb{N}, n \geqslant k$.

We use the notation $f \in O(g)$ and read this as "f is in Big O of g".

$O(g)$ is the set of all functions from $\mathbb{N}_+$ to $\mathbb{R}$ which are dominated by g.

Defn: *g dominates f* if there exist constants $m \in \mathbb{R}^+$ and $k \in \mathbb{N}_+$ such that $|f(n)| \leqslant m|g(n)|$ for all $n \in \mathbb{N}$, $n \geqslant k$.

Ex: who dominates who: $f(n) = \log_2 n + 5$ versus $g(n) = n$:

Defn: *g dominates f* if there exist constants $m \in \mathbb{R}^+$ and $k \in \mathbb{N}_+$ such that $|f(n)| \leqslant m|g(n)|$ for all $n \in \mathbb{N}$, $n \geqslant k$.

Ex: who dominates who: $f(n) = \log_2 n + 5$ versus $g(n) = n$:

Here is my prepared solution in case my "live" one is too messy:

Claim 1: $n \leqslant 2^n$. Let $P(n)$ be the statement $n \leqslant 2^n$. $P(1)$ true. Assume $P(k)$ then $n + 1 \leqslant n + n = 2n \leqslant 2.2^n = 2^{n+1}$ so $P(k + 1)$ is implies, so by PMI true for all $n \geqslant 1$.

Take $\log_2$ both sides gives $\log_2 n \leqslant n$, so

$$\log_2 n + 5 \leqslant n + 5 \leqslant n + n$$

(if $n \geqslant 5$)

$$= 2n$$

so $m = 2, k = 5$ gives the result.

EG

Defn: *g dominates f* if there exist constants $m \in \mathbb{R}^+$ and $k \in \mathbb{N}_+$ such that $|f(n)| \leqslant m|g(n)|$ for all $n \in \mathbb{N}$, $n \geqslant k$.

Some careful analysis of the two algorithms above shows that, up to some constants, power_fast takes about $\log_2 n$ steps (because each step divides i by 2, roughly) and power_slow takes n steps.

Because of Big O, if you count steps slightly differently, and/or count the initial and final lines of the code, the time complexity function changes a bit but is **the same** up to Big O.

Defn: *g dominates f* if there exist constants $m \in \mathbb{R}^+$ and $k \in \mathbb{N}_+$ such that $|f(n)| \leqslant m|g(n)|$ for all $n \in \mathbb{N}$, $n \geqslant k$.

Recall the formula

$$\log_b y = \frac{\log_a y}{\log_a b}.$$

If $b = 2$ and your calculator only has a button for $\log_{10}$ or $\log_e = \ln$ then you can use this formula:

$$\log_2 n = \frac{\log_e n}{\log_e 2}.$$

Since $\log_e 2 \approx 4.6$ is a fixed number, it doesn't matter which log we use because of the m in the definition.

EG

Defn: *g dominates f* if there exist constants $m \in \mathbb{R}^+$ and $k \in \mathbb{N}_+$ such that $|f(n)| \leqslant m|g(n)|$ for all $n \in \mathbb{N}$, $n \geqslant k$.

Let $f(n) = 6n$ and $g(n) = n^2$. Show that g dominates f, that is, $6n \in O(n^2)$.

EG

Defn: *g dominates f* if there exist constants $m \in \mathbb{R}^+$ and $k \in \mathbb{N}_+$ such that $|f(n)| \leqslant m|g(n)|$ for all $n \in \mathbb{N}$, $n \geqslant k$.

Let $f(n) = 6n$ and $g(n) = n^2$. Show that g dominates f, that is, $6n \in O(n^2)$.

Here is my prepared solution. It might be different that the one I just did "live" – many choices for m, k can work.

There exist $m = 1, k = 6$ so that

$$6n \leqslant n.n = n^2$$

for all $n \geqslant 6$.

Defn: *g dominates f* if there exist constants $m \in \mathbb{R}^+$ and $k \in \mathbb{N}_+$ such that $|f(n)| \leqslant m|g(n)|$ for all $n \in \mathbb{N}$, $n \geqslant k$.

Show that $g(n) = n^2$ is not dominated by $f(n) = 6n$. That is, $n^2 \notin O(6n)$.

We need the *negation* of the Big O definition.

$$\forall m \in \mathbb{R}_+ \forall k \in \mathbb{N}_+ \exists n \in \mathbb{N}_+ [(n \geqslant k) \wedge (|f(n)| > m|g(n)|)].$$

Here is my prepared solution.

Given m, k fixed numbers (positive), there is a number $n = \mathbf{max}\{6m, k\}$ so that $n > 6m$ so

$$n^2 > m6n.$$

EG

Defn: *g dominates f* if there exist constants $m \in \mathbb{R}^+$ and $k \in \mathbb{N}_+$ such that $|f(n)| \leqslant m|g(n)|$ for all $n \in \mathbb{N}$, $n \geqslant k$.

Let $f(n) = 6n^2 + 5n + 2$ and $g(n) = n^2$. Show that $f \in O(g)$ and $g \in O(f)$. So they are (up to Big O equivalence) the "same".

Defn: *g dominates f* if there exist constants $m \in \mathbb{R}^+$ and $k \in \mathbb{N}_+$ such that $|f(n)| \leqslant m|g(n)|$ for all $n \in \mathbb{N}$, $n \geqslant k$.

Show that $f(n) = n^3$ is dominated e^n.

My prepared solution:

Let $P(n)$ be the statement $n^3 \leqslant e^n$. Need to find a value k so that $P(k)$ is true, then

assume for $s \geqslant k$ that $P(s)$ is true, then $P(s+1)$:

$$(n+1)^3 = n^3 + 3n^2 + 3n + 1 \leqslant n^3 + n^3$$

(by a Lemma I will prove below)

$$= 2n^3 \leqslant 2e^n < e.e^n = e^{n+1}$$

since $e = 2.7... > 2$.

Lemma: $3n^2 + 3n + 1 \leqslant n^3$ for n big enough.

CHALLENGE QUESTION

Defn: *g dominates f* if there exist constants $m \in \mathbb{R}^+$ and $k \in \mathbb{N}_+$ such that $|f(n)| \leqslant m|g(n)|$ for all $n \in \mathbb{N}$, $n \geqslant k$.

Show that $f(n) = n^c$ is dominated e^n where c is any positive integer.

So polynomials are "slower" than exponentials.

Your proof might use the Binomial Theorem:

$$(x+y)^c = x^c + \binom{c}{1}x^{c-1}y + \binom{c}{2}x^{c-2}y^2 + \cdots + \binom{c}{c-1}x^1y^{c-1} + y^c$$

$$(n+1)^c = n^c + \binom{c}{1}n^{c-1} + \binom{c}{2}n^{c-2} + \cdots + \binom{c}{c-1}n + 1$$

In your proof, c is a fixed number, so all the $\binom{c}{i}$ terms are bounded above by some fixed number.

EG

Defn: *g dominates f* if there exist constants $m \in \mathbb{R}^+$ and $k \in \mathbb{N}_+$ such that $|f(n)| \leqslant m|g(n)|$ for all $n \in \mathbb{N}$, $n \geqslant k$.

Which is faster (who dominates who?) – exponentials or factorials?

That is, show that one of $f(n) = c^n$, $g(n) = n!$ dominates the other (for any $c \in \mathbb{R}_+$ say).

If you do more computer science, you will use Big O. You count (roughly, since constant multiples don't matter) the number of steps an algorithm takes on inputs of size $n \in \mathbb{N}$.

The algorithm is "fast" if the number of steps is $O(n)$ or maybe $O(n \log n)$, and "bad" if it takes $O(c^n)$ or worse steps.

Issues: is it bad if it takes this many steps on *all* inputs, or some inputs, or most inputs.

See Worksheet 6 for a table of standard functions and their names used in time complexity.

Exercise: prove that the worst case running time for the Euclidean algorithm is when the input is two of the Fibonacci numbers (worksheet 3)

Standard exercise in comp sci: compare running times for algorithms which sort a list of numbers: bubble sort versus merge sort.

Recall:

Lemma

Let A, B be finite sets. If $f : A \to B$ is

- *1-1 then $|A| \leqslant |B|$.*
- *onto then $|B| \leqslant |A|$.*

Proof: ?

To prove the 1-1 rigorously, we need another **Axiom**

Axiom (Pigeonhole principle)

If m pigeons occupy n pigeonholes and $m > n$ then some pigeonhole has at least two pigeons in it.

Remember, this is an *axiom* like well ordering (and induction) – they seem obvious, but we can't derive them from other things.

Axiom (Pigeonhole principle)

If m pigeons occupy n pigeonholes and $m > n$ then some pigeonhole has at least two pigeons in it.

Out of 13 people, what is the chance two of them have the same Western Zodiac sign?

Out of 367 people, what is the chance two of them have the same birthday?

Axiom (Pigeonhole principle)

If m pigeons occupy n pigeonholes and $m > n$ then some pigeonhole has at least two pigeons in it.

Out of 145 people, what is the chance two of them have the same Western Zodiac sign AND Chinese Zodiac?

Let $A \subseteq \mathbb{N}_+$ with $|A| = 28$. Then A contains at least two elements with the same remainder $\mod 27$.

Proof: the pigeons are ...

and the pigeonholes are ...

EX

If 11 integers are chosen from $\{1, 2, 3, \ldots 100\}$ then at least two, say x and y, are such that

$$|x - y| \leqslant 9$$

Proof: the pigeons are ...

and the pigeonholes are ...

EX

If 11 integers are chosen from $\{1, 2, 3, \ldots 100\}$ then at least two, say x and y, are such that

$$|x - y| \leqslant 9$$

Proof: the pigeons are ... the 11 integers chosen

and the pigeonholes are ... boxes $[1, 10], [11, 20] \ldots, [91, 100]$.

Place a number in the box labeled by the interval containing it.

EX

If 11 integers are chosen from $\{1, 2, 3, \ldots 100\}$ then at least two, say x and y, are such that

$$|\sqrt{x} - \sqrt{y}| < 1$$

Proof: the pigeons are ...

and the pigeonholes are ...

Recall:

Lemma

Let A, B be finite sets. If $f : A \to B$ is

- *1-1 then $|A| \leqslant |B|$.*
- *onto then $|B| \leqslant |A|$.*

Proof:

Axiom (Pigeonhole principle)

If m pigeons occupy n pigeonholes and $m > kn$ then some pigeonhole has more than k pigeons in it.

Ex: Use PHP to show that if $S \subseteq \mathbb{N}_+$ and

1. $|S| \geqslant 3$ then S contains two distinct elements x, y such that $x + y$ is even.

Ex: Use PHP to show that if $S \subseteq \mathbb{N}_+$ and

2. $|S| > 6$ then S contains three distinct elements x, y, z such that $x + y + z$ is a multiple of 3.

NEXT

Next lecture time:

- Team assignment: meet your team several times including **next Wednesday 4-6pm**.
- This room and CB07.03.010BG upstairs (with whiteboards) with tutors and lecturer to discuss.
- Workshops tomorrow, Friday and Monday. Then no workshops until 19 September.