COSC1107 Computing Theory

(We will commence soon. We are just allowing a few minutes for people to join and set up. *Please mute your microphone unless you are speaking.* You can raise your hand or use the chat at any time.)

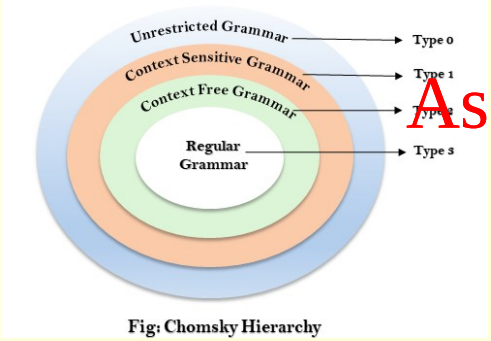
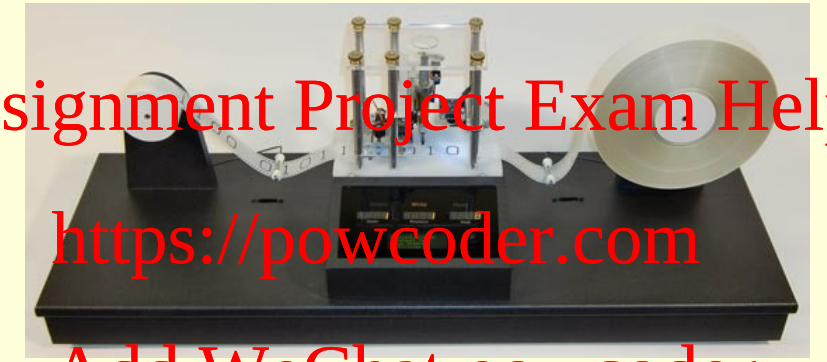
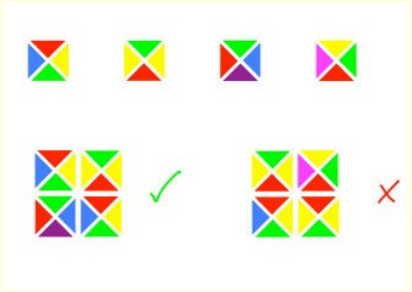
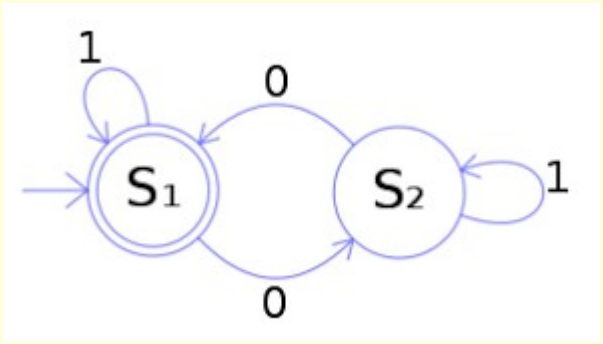
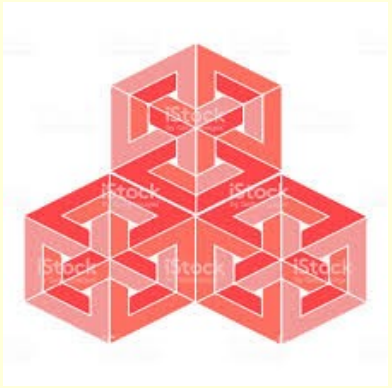


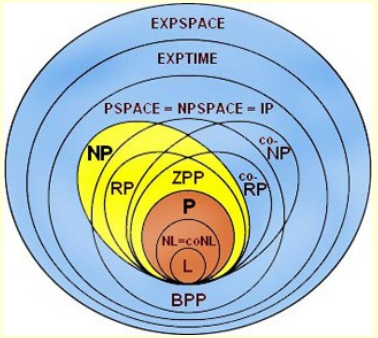
Fig: Chomsky Hierarchy

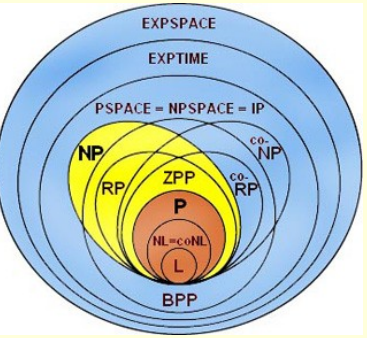


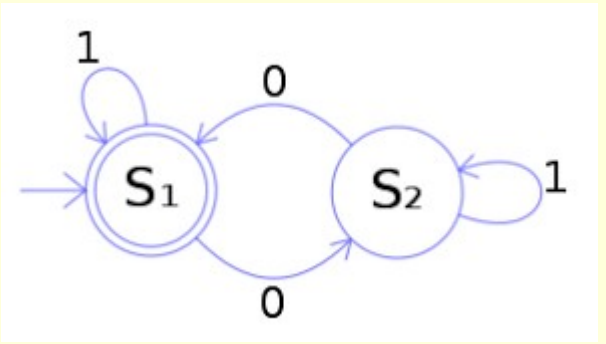










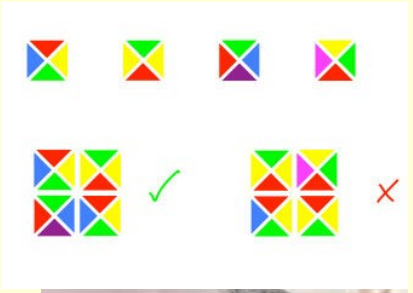


COSC1107

Computing Theory

Complexity & Intractability

Week 9



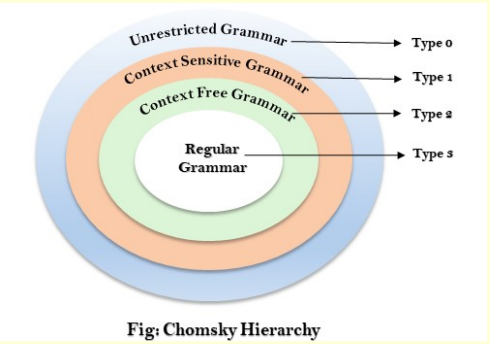


Fig: Chomsky Hierarchy

James Harland

james.harland@rmit.edu.au

* With thanks to Sebastian Sardina

Intro music 'Far Over' playing now …

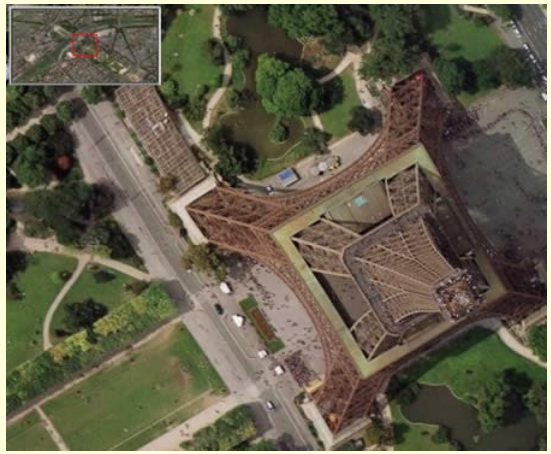
Acknowledgement

RMIT University acknowledges the people of the Woi wurrung and Boon wurrung language groups of the eastern Kulin Nations on whose unceded lands we conduct the business of the University. RMIT University respectfully acknowledges their Ancestors and Elders, past and present.

RMIT also acknowledges the Traditional Custodians and their Ancestors of the lands and waters across Australia where we conduct our business.

(add your name here to volunteer for this or email me)
(my personal Acknowledgement of Country is here)

Overview



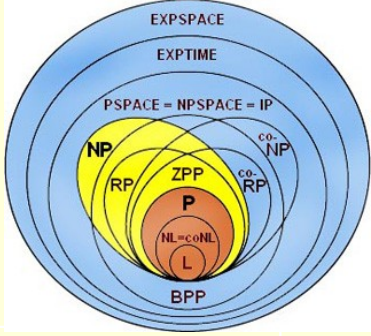
- Questions?
- Computational Limits
- Questions?
- Measuring Complexity
- Questions?
- Intractable problems
- Questions?
- Platypus Game ⟵ Of course!
- Questions?

Questions?

Questions?

Questions?

Computing Theory topics



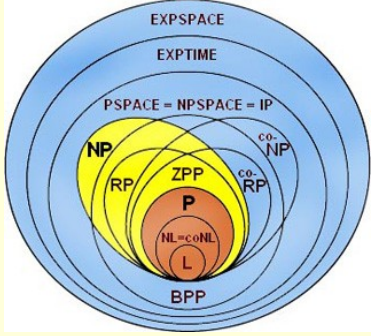
Languages	What do you want to do?
Grammars	What can you say?
Automata	What can you do?
Computability	What can't you do?
Complexity	**How hard can it be?**

Any attempt to solve an undecidable problem must be **incomplete**

- Sometimes 'yes'
- Sometimes 'no'
- Sometimes 'maybe' ⟵ Cannot be eliminated!

Any such attempt can only be an **approximate solution**

Computational limits



There are various limits on computation

Fundamental

- No (complete) algorithmic solutions exist
- Will always be beyond any technology
- **Example:** Halting problem

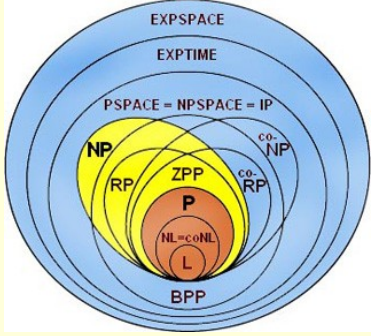
Practical

- (Complete) Algorithmic solutions exist
- Complexity is too high for problems beyond a smallish size
- **Example:** Hamiltonian circuit problem

Technological

- Any computing device has a finite memory, storage capacity, processing speed, bandwidth, …
- There is always a problem "just beyond" any technology
- **Example:** Platypus tournament

Computational limits



Beyond any algorithm **ever**

Undecidability

Beyond any technology but decidable

Intractability

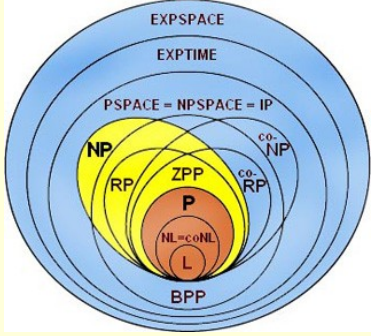
Beyond current technology but feasible

Technology

Within current technology

All processing power is finite!

Computational limits



Beyond any algorithm **ever**

Undecidability

Beyond any technology but decidable

Intractability

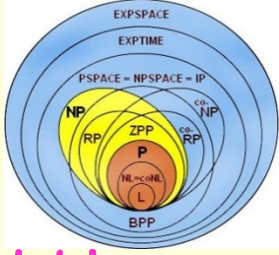
Beyond current technology but feasible

Technology

Within current technology

Asymptote!
(can never be reached)

Complexity



Undecidable

Undecidable

$L(G) =$

Assignment Project Exam Help

Decidable problems

https://powcoder.com

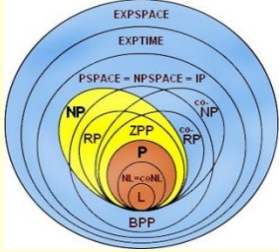
Add WeChat powcoder

How hard can they be?

Halting problem

$L(G_1) = L(G_2)$

Complexity



Turing machines

Our main focus

EXPSPACE

EXPTIME

PSPACE

??

??

P

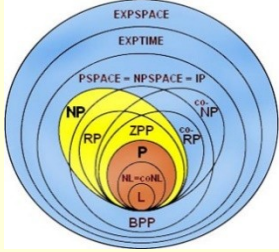
LOGTIME

Questions?

Questions?

Questions?

Complexity

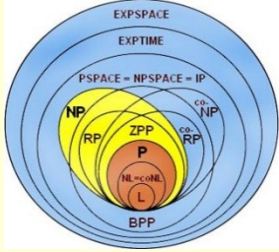


The ***White Council Band*** are touring the world!

- Spectacular stage show
- Tons of equipment to cart around
- Costs $ millions per day

Magic only works in Middle-Earth!

Complexity



Best route for **Australian** tour?

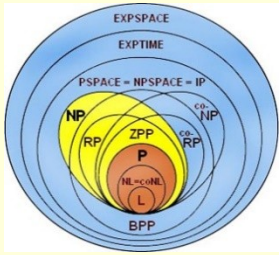
Melbourne, Sydney, Brisbane, Adelaide, Hobart, Perth, Darwin, Canberra (8 cities)

Best route for **US** tour? (25 cities)

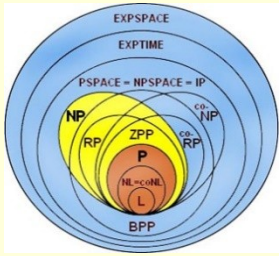
Best route for **World** tour? (100 cities)

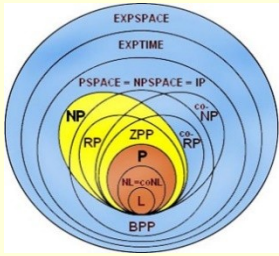
Need to find the **minimum cost** route in all cases …

White Council Tour

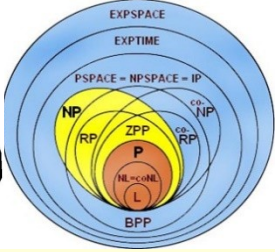


White Council Tour





White Council Tour

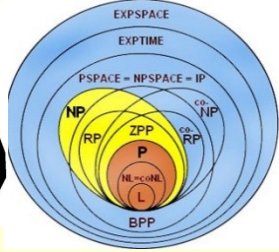


Simple programming problem

- Identify tour starting point
- Generate all tours starting from there
  - Calculate the cost for each tour
  - Keep the tour with the lowest cost so far
- Output minimum

Piece of cake!

White Council Tour



Tour	Cities	Routes
Australia	8	7! = 5,040
US	22	21! = 5.1×10^{19}
India	29	28! = 3.0×10^{29}
World	100	99! = 9.3×10^{155}

Tour	1 route s^{-1}	10^{12} route s^{-1}
Australia	1.4 hours	< 1 second
US	1.6 trillion years (!!)	1.6 years (!?)
India	9.6×10^{21} years (!!!)	9.6×10^{9} years (!!)
World	3.0×10^{148} years (!!!!!)	3.0×10^{136} years (!!!!)

1 year = 60x60x24x365.25 = 31,557,600 seconds

Problem Scales

n	n^2	2^n	n!
10	100	1024	3628800
20	400	1048576	$\sim 10^{18}$
30	900	1073741824	$\sim 10^{32}$
40	1600	$\sim 10^{12}$	$\sim 10^{47}$
50	2500	$\sim 10^{15}$	$\sim 10^{64}$
60	3600	$\sim 10^{18}$	$\sim 10^{81}$
70	4900	$\sim 10^{21}$	$\sim 10^{100}$
80	6400	$\sim 10^{24}$	$\sim 10^{118}$
90	8100	$\sim 10^{27}$	$\sim 10^{138}$
100	10000	$\sim 10^{30}$	$\sim 10^{157}$
200	40000	$\sim 10^{60}$	$\sim 10^{374}$
300	90000	$\sim 10^{90}$	$\sim 10^{614}$
400	160000	$\sim 10^{120}$	$\sim 10^{868}$
500	250000	$\sim 10^{150}$	$\sim 10^{1134}$

#atoms on Earth = $\sim 10^{50}$

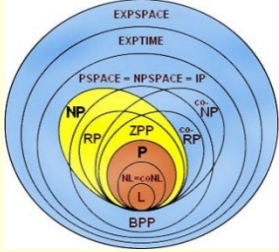
#particles in observable universe = $\sim 10^{80}$

Questions?

Questions?

Questions?

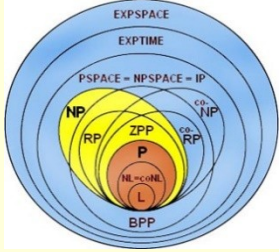
Measuring complexity



So how do we draw the 'line of intractability?'

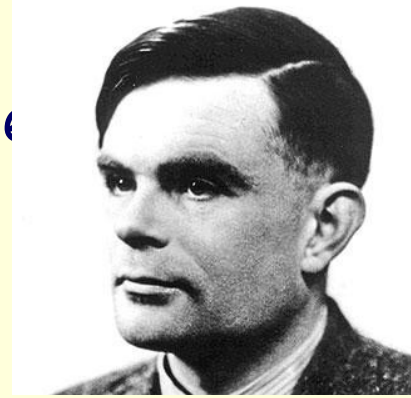
- How do we measure the resources required by a program?
- How do we do this independently of hardware?
- Complexity measure must
  - Not limit available memory or time
  - Allow for all computations
  - Not depend on a particular implementation

Measuring complexity



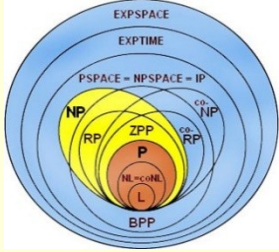
So how do we draw the 'line of intractability?'

- How do we measure the resources required by a program?
- How do we do this independently of hardware?
- Complexity measure must
  - Not limit available memory or time
  - Allow for all computations
  - Not depend on a particular implementation



"Ring any bells?" Does this sound familiar?

Measuring complexity

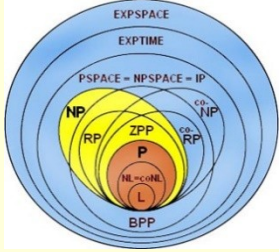


What exactly do we measure?

- Time?
- Space?
- Input size? Requirements?
- Code size?
- Conceptual difficulty?
- Effort to produce?
- Readability?
- Functionality?
- …

"The difference between time and space is that you can't reuse time"
— Merrick Furst

Execution time is often most critical
("Need more memory? Buy some!")

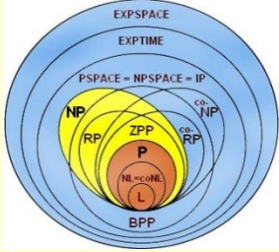


Measuring complexity

How exactly do we measure time?
Minimum? Maximum? Average? …

Typical: Worst-case (maximum) number of **a single critical operation** in terms of **input size**

- Using maximum gives guarantee
- May be misleading
- One-dimensionality simplifies analysis (perhaps too much!)
- Choice of operation can be critical (disk accesses, memory accesses, GPU calls, comparisons, multiplications, …)
- Average is more informative but generally much harder to find …



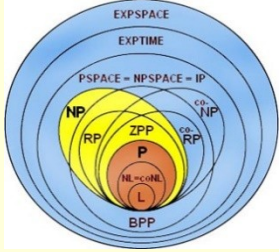
Measuring complexity

Algorithm	Critical operation
Sorting	Comparisons
Numerical calculations	Floating point operations
Integer calculations	Multiplications and divisions
Graphs	Edge traversals or vertices visited
Primality testing	Divisions
…	…

Note: the measurement is in the size of the **representation of the input**

Numeric input is represented in size **log n** (!!)

List of n integers to sort	Input size **n**
Integer n to factorise	Input size **log n**



Intractability

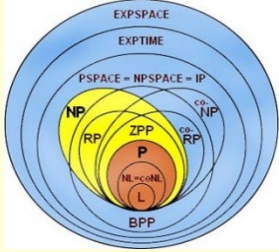
TRACTABLE

O(.)	10	20	30	40	50	60
n	0.00001s	0.00002s	0.00003s	0.00004s	0.00005s	0.00006s
n^2	0.0001s	0.0004s	0.0009s	0.0016s	0.0025s	0.0036s
n^3	0.001s	0.008s	0.027s	0.064s	0.125s	0.216s
n^5	0.1s	3.2s	24.3s	1.7 mins	5.2 mins	13.0 mins
2^n	0.001s	1.0s	17.9 mins	12.7 days	35.7 years	366 centuries
3^n	0.059s	58 mins	6.5 years	3855 centuries	2×10^8 centuries	1.3×10^{13} centuries

INTRACTABLE

Table from 'Computers and Intractability: A Guide to the theory of NP-completeness', Michael Garey & David Johnson, W.H. Freeman, 1979.

Intractability



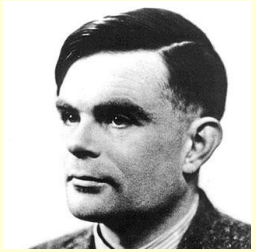
$O(1)$	Constant
$O(\log n)$	Logarithmic
$O(n)$	Linear
$O(n \log n)$	"n log n"
$O(n^2)$	Quadratic
$O(n^3)$	Cubic
$O(n^k)$	Polynomial
$O(2^n)$	Exponential
$O(n!)$	Factorial
$O(n^n)$	"Hyperfactorial"
$O(2\hat{}2^n)$	Double-exponential
Larger	***Go home!***

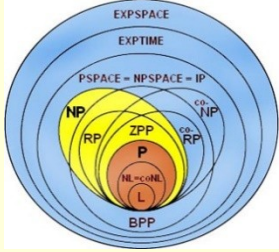
Polynomial (or less) is considered **tractable**
(ie $O(n^k)$ for some fixed integer k)

Twilight Zone:
Superpolynomial and Subexponential
[(n^k) for all k and $O(2^n)$]

Exponential (or worse) is considered **intractable**
(ie (2^n) or 2^n is $O(f)$)

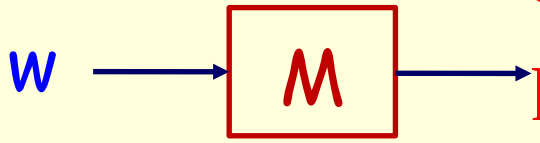
Measuring Complexity





Formally define what a **computation** is via Turing machines

Formally define what **complexity** is via Turing machines



M halts on w after at most
f(n) steps whe

Time complexity: If M takes at mos
to halt on input of size n

- Time depends on input size
- Rate of growth of f is of most interest …
- **M could be nondeterministic (!!)**

Questions?

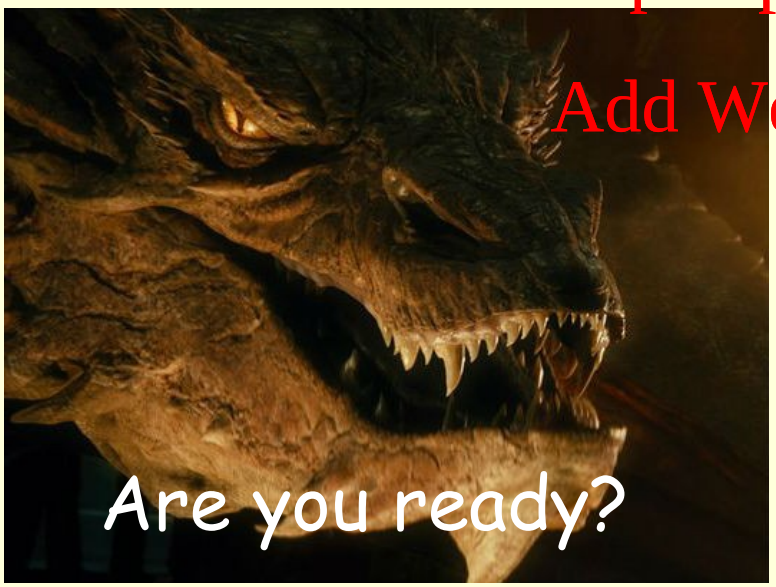
Questions?

Questions?

Quiz time!

Go to Canvas and find the quiz Lectorial 9 Question set

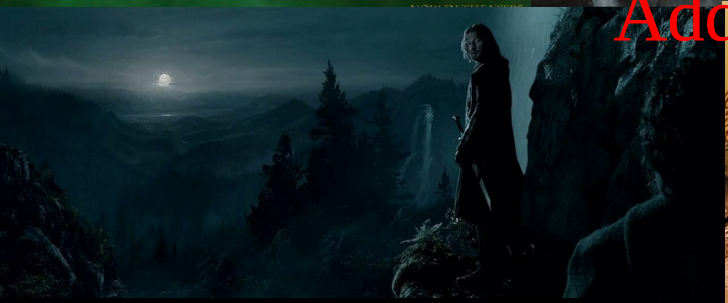
- Not worth any marks
- You can consult other students if you wish
- Time limit will be 10 minutes

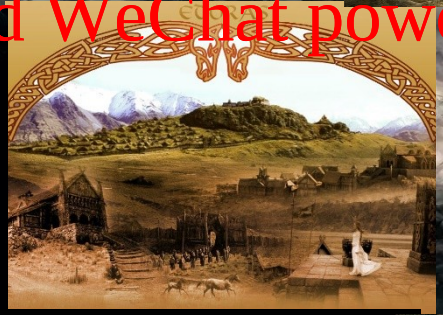


Go!

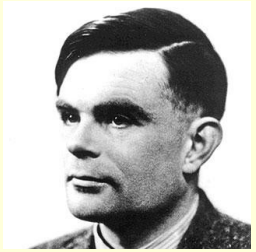
The pictures will take 10 minutes to disappear!

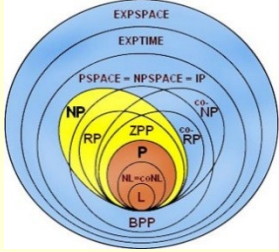
Thomas music means 1 minute left!





Difficult problems





Some problems have only exponential solutions known
Many important practical problems are in this class!

ie tour of minimal cost that visits every node exactly once?

Hamiltonian circuit problem

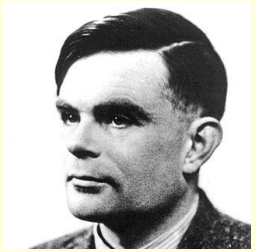
Given a graph G, is there a cycle that visits every node exactly once?

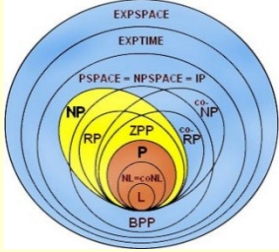
3 SAT problem

Given a set of clauses with exactly 3 Boolean variables each, is there a truth assignment that satisfies all the clauses?

http://www.cril.univ-artois.fr/~roussel/satgame/satgame.php?lang=eng

3 SAT





Given a set of clauses with exactly 3 Boolean variables each, is there a truth assignment that satisfies all the clauses?

Variables: x, y, z, which can be assigned true or false (1 or 0) (basically propositions)

Assignment: Function mapping every variable to true or false

Literal: variable or its negation (eg x or $\bar{x}$)

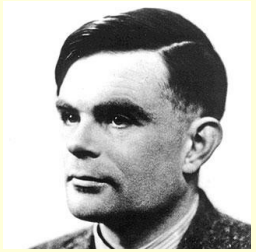
Clause: disjunction of literals (think CNF) (eg $x\ y\ z$, $\bar{x}\ y$, …)

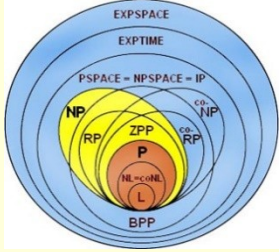
SAT problem: Given a set of clauses, is there an assignment that makes every clause true?

3 SAT problem: SAT problem where every clause has exactly 3 literals

"3 SAT is when SAT problems start getting difficult …"

3 SAT





Instance 1: C = { x, y }
Assign x = 1, y = 0

YES!

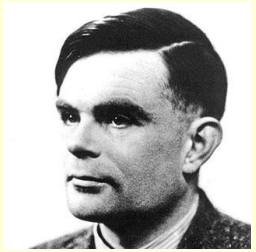
Instance 2: C = { x, w, y, w, x, y, w }
Assign x = 1, y = 1, w = 1

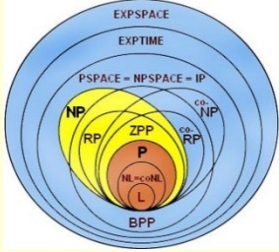
YES!

Instance 3: C = { x, w, y, w, x, y, w, x, y }
Must have x = 1, y = 0 (from last two clauses)
If w = 1, y ¬w is 0
If w = 0, x ¬w is 0

For n variables, there are 2^n possible assignme
Some applications have **thousands** of variable

3 SAT





Instance 4: $C = \{ p\ q\ r, p\ r\ q, p\ p\ r \}$

Assign p, r = 0, q, w = 1

0 0 0, 0 1 1, 0 1 0

0, 1, 1

No

Assign r = 0, p, q, w = 1

1 0 0, 1 1 1, 1 0 0

1, 1, 1

YES!

"You know, Mister Gandalf sir, it is much easier to check an assignment than to find one…"

"Now I see why they called you SamWISE Gamgee…"

- Finite number of possibilities …
- Only need one to succeed …

NDTMs!!

3 SAT & NDTMs

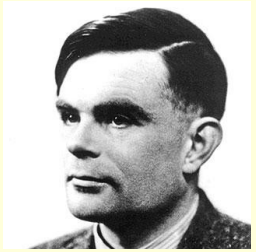
Construct an NDTM which d

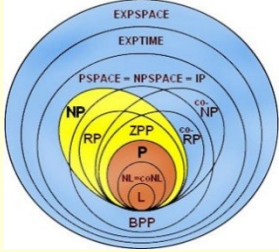
- For each variable in C, "gu
  somewhere on the tape
- Check whether the guess

- If the machine answers yes, there is at least one truth assignment that makes C true
- If the machine answers no, there is no truth assignment that makes C true

TM specifies a language … NDTM does too!

Checking vs Finding





The "Gamgee" property, ie it is **easier to check than to find**, holds for various problems

Factorisation: Given numbers A, B, C, check whether A x B = C, rather than find A and B from C

Hamiltonian circuit: Given a graph and a path, check the path is cyclic and visits every node rather than finding one

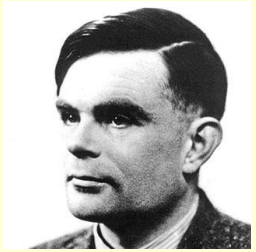
SAT, 3 SAT: Given an assignment, check whether all clauses in C are true, rather than finding such an assignment

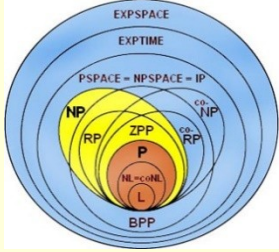
TSP: Given a cycle and a cost, check whether the cycle has total cost no more than the given cost, rather than finding one

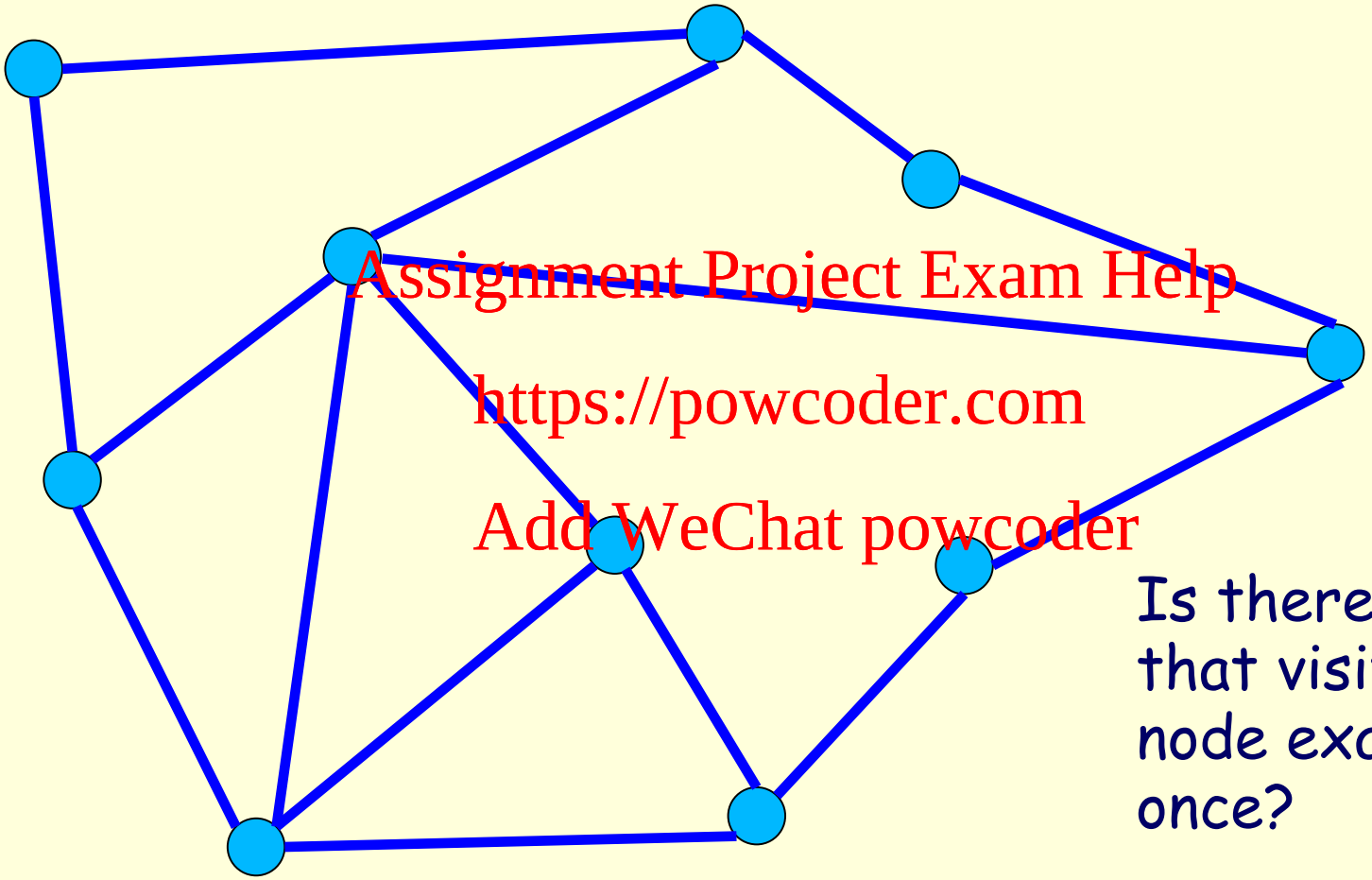
Password: Given a supposed PIN, check whether it is correct, rather than find one that is correct

...

Hamiltonian Circuit

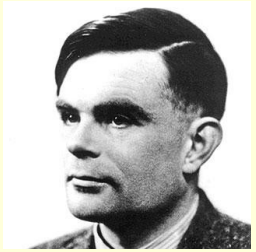


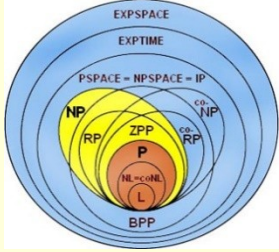


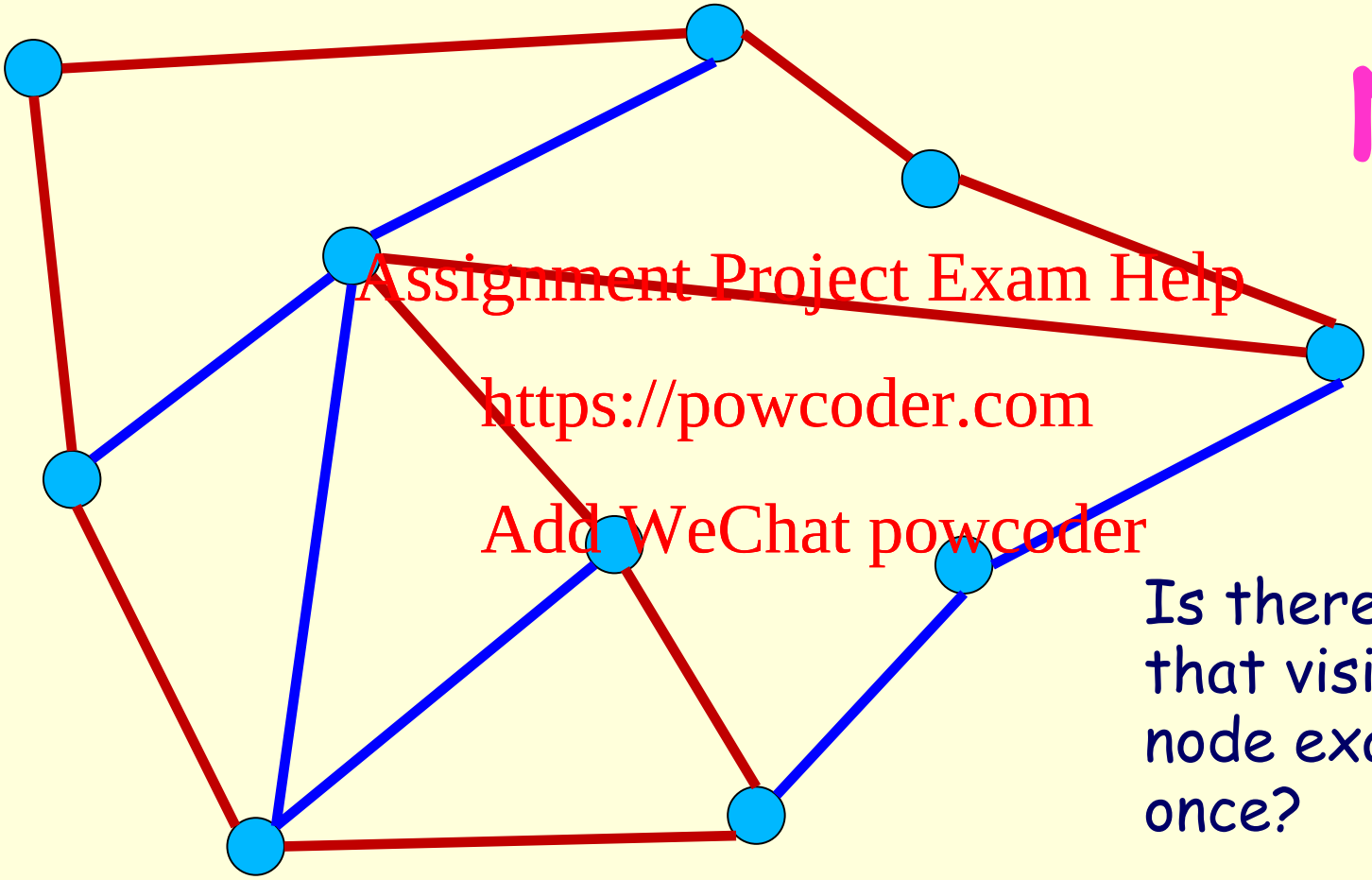


Is there a circuit that visits every node exactly once?

Hamiltonian Circuit



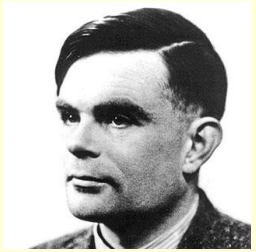


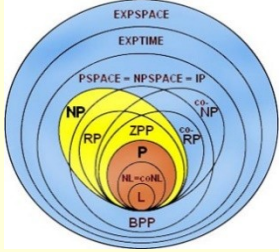


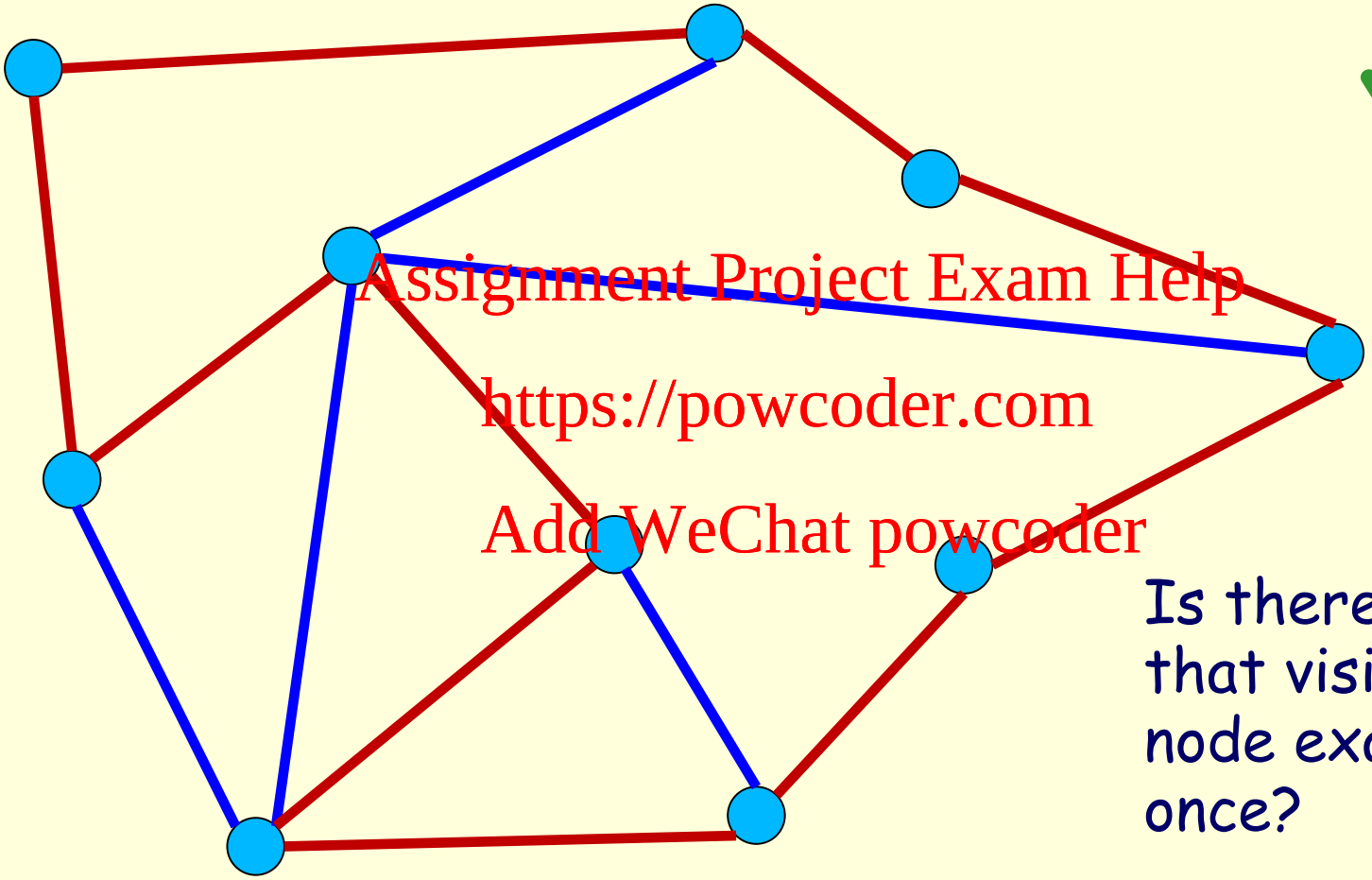
NO!

Is there a circuit that visits every node exactly once?

Hamiltonian Circuit



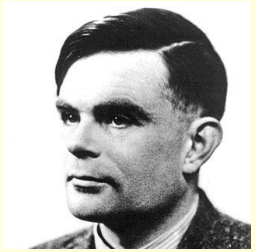


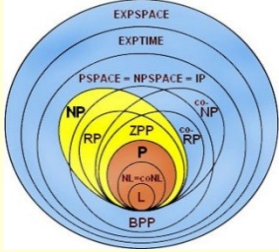


YES!

Is there a circuit that visits every node exactly once?

Hamiltonian Circuit





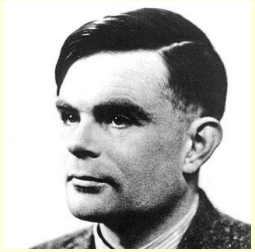
Construct an NDTM which does the following:

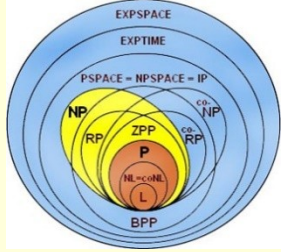
- Guess a path from a start node to any other node
- Check that the path
  - Visits every node exactly once
  - Is a cycle

- If the machine answers yes, the path is a Hamiltonian cycle
- If the machine answers no, there is no Hamiltonian cycle

"If there is such a path, some guess will work.
If no guess will work, there is no such circuit".

Nondeterminism



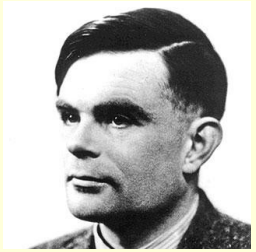


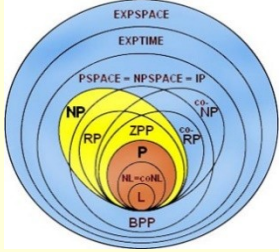
"Wait a second! Didn't you say you can't do better than a deterministic TM? We can just use them rather than this …this … witchcraft!"

"Calm yourselves! You can find an equivalent deterministic TM for any NDTM. But it may be exponentially larger and hence take similarly longer …"

"It was better before we had wizards …" ☺

Nondeterminism





For any nondeterministic TM, there is an equivalent deterministic TM, ie one that accepts the same language

- The deterministic TM cannot guess; it systematically searches through all possible guesses
- The deterministic TM terminate with success iff the nondeterministic TM terminates with success
- The deterministic TM may take exponentially longer (!!)

n steps

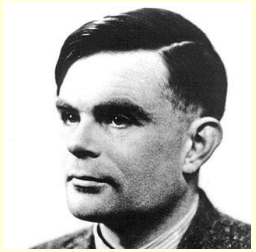
nondeterministic

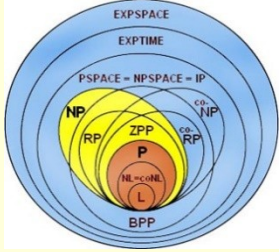
NO FREE LUNCH!

2^n steps

deterministic

Nondeterminism





For any nondeterministic TM, there is an equivalent deterministic TM, ie one that accepts the same language

- The deterministic TM cannot guess; it systematically searches through all possible guesses
- The deterministic TM terminate with success iff the nondeterministic TM terminates with success
- The deterministic TM may take exponentially longer (!!)

n steps

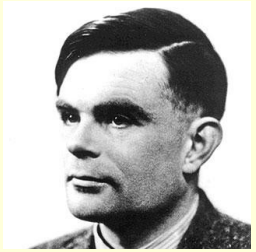
nondeterministic

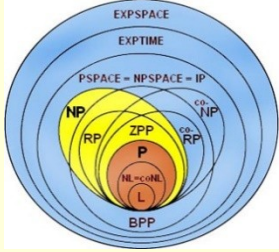
NO FREE LUNCH!

2^n steps

deterministic

Complexity classes





Two important complexity classes (there are many others!)

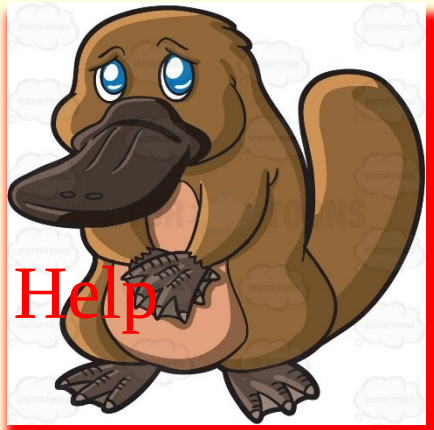
P: Decision problems that can be solved in polynomial time or less on a deterministic Turing machine

NP: Decision problems that can be solved in polynomial time or less on a nondeterministic Turing machine

P NP (deterministic TMs are trivially nondeterministic TMs)

Does P NP? **UNKNOWN!**

The Platypus Game



https://www.youtube.com/watch?reload=9&v=0gM5TjSOQ48
https://zoneringtones.com

The Platypus Game

3 player tournament

1 vs 1 vs 1
1 vs 1 vs 2
1 vs 1 vs 3
1 vs 2 vs 2
1 vs 2 vs 3
…
1 vs n vs n
2 vs 2 vs 2
2 vs 2 vs 3
…
3 vs 3 vs 3
…
(n-1) vs (n-1) vs (n-1)
(n-1) vs (n-1) vs n
n vs n vs n

$$\sum_{i=1}^{n} i(i+1)/2 = \left(\sum_{i=1}^{n} i^2 + \sum_{i=1}^{n} i\right)/2$$
$$= n(n+1)(2n+1)/12 + n(n+1)/4$$
$$= n(n+1)(n+2)/6$$

When n = 268,
this is **3,244,140**

Around 100 times more than a 2-player tournament!

The Platypus Game

4 player tournament

1 vs 1 vs 1 vs 1
1 vs 1 vs 1 vs 2
…
1 vs 1 vs 1 vs n
1 vs 1 vs 2 vs 2
…
1 vs 2 vs 2 vs 2
…
1 vs n vs n vs n
2 vs 2 vs 2 vs 2
2 vs 2 vs 2 vs 3
…
2 vs n vs n vs n
…
3 vs 3 vs 3 vs n
…
(n-1) vs (n-1) vs (n-1) vs n
n vs n vs n vs n

$$\sum_{i=1}^{n} i(i+1)(i+2)/6$$

$$= (\sum_{i=1}^{n} i^3 + 3\sum_{i=1}^{n} i^2 + 2\sum_{i=1}^{n} i)/6$$

$$= n^2(n+1)^2/4 + n(n+1)(2n+1)/2 + n(n+1)$$

$$= n(n+1)(n+2)(n+3)/24$$

When n = 268,
this is 219,790,485

Around 10,000 times more than a 2-player tournament!

When n = 90, this is 2,919,735

Assignment 2

- Detailed specification is out now
- Platypus tournament for 2,500 machines
- 'Second version' of Universality task from Assignment 1
- Research on at least two intractable problems

That's it!

I am out of here!

Break time! (We resume when all the pictures are gone! This will take 3 minutes!)