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.)

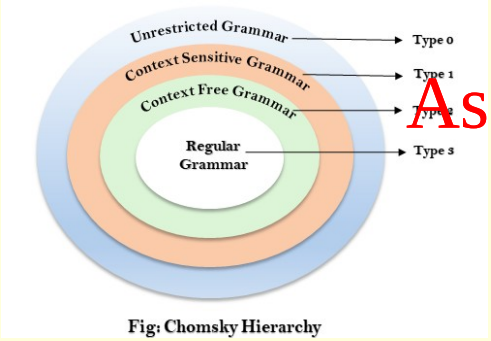
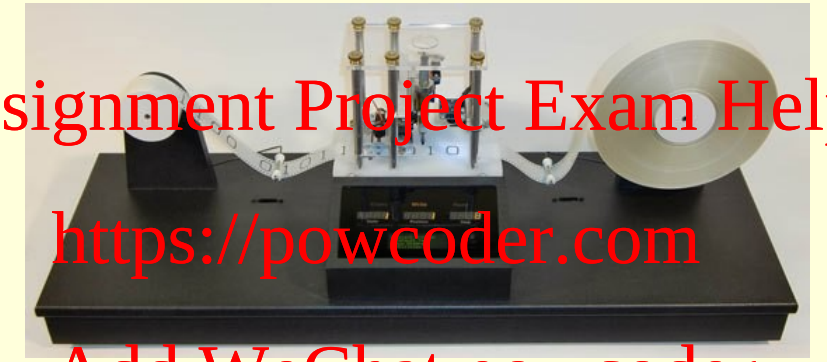
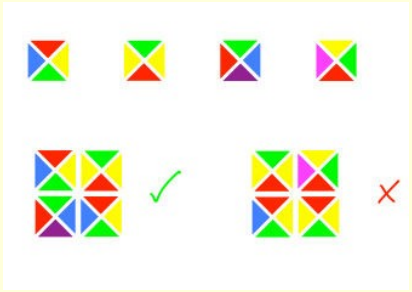
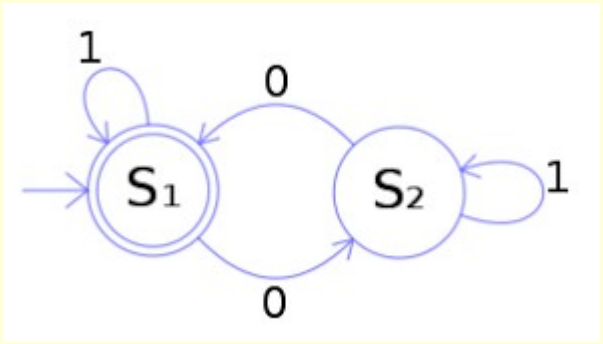
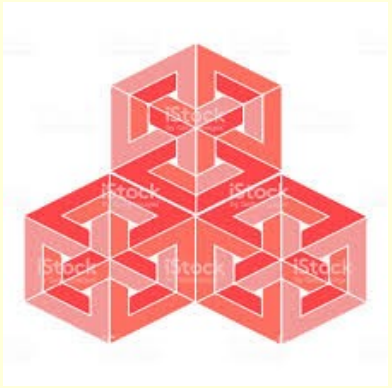


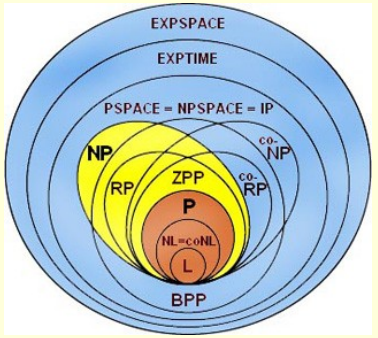
Fig: Chomsky Hierarchy

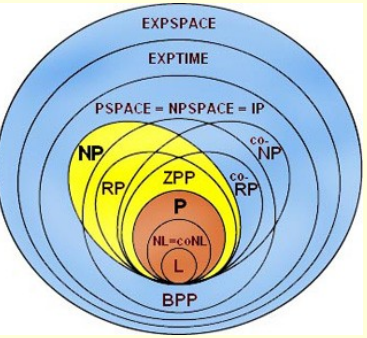


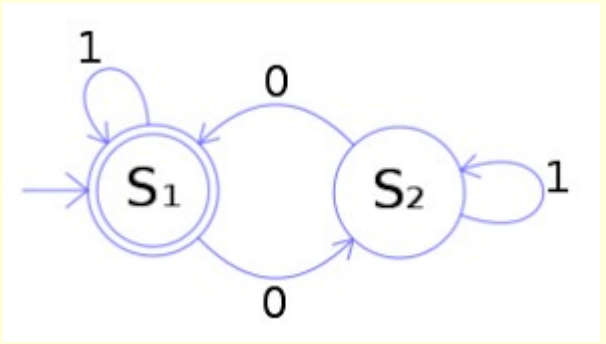












COSC1107

Computing Theory

Analysing Complexity

Week 10

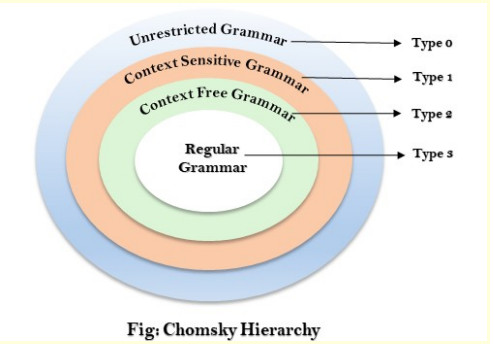


Fig: Chomsky Hierarchy

James Harland

james.harland@rmit.edu.au

* With thanks to Sebastian Sardina

Intro music 'Far Over' playing now …

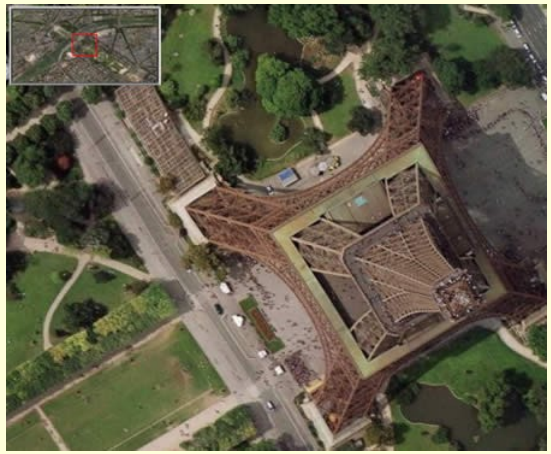
Acknowledgement

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



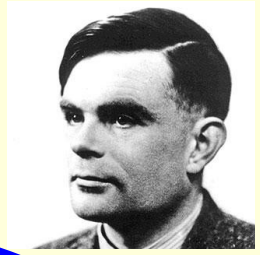
- Questions?
- NP-completeness
- Questions?
- RSA Cryptosystem
- Questions?
- Probabilistic algorithms
- Questions?
- Platypus Game ⟵ Of course!
- Questions?

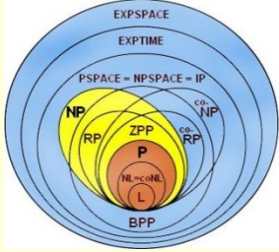
Questions?

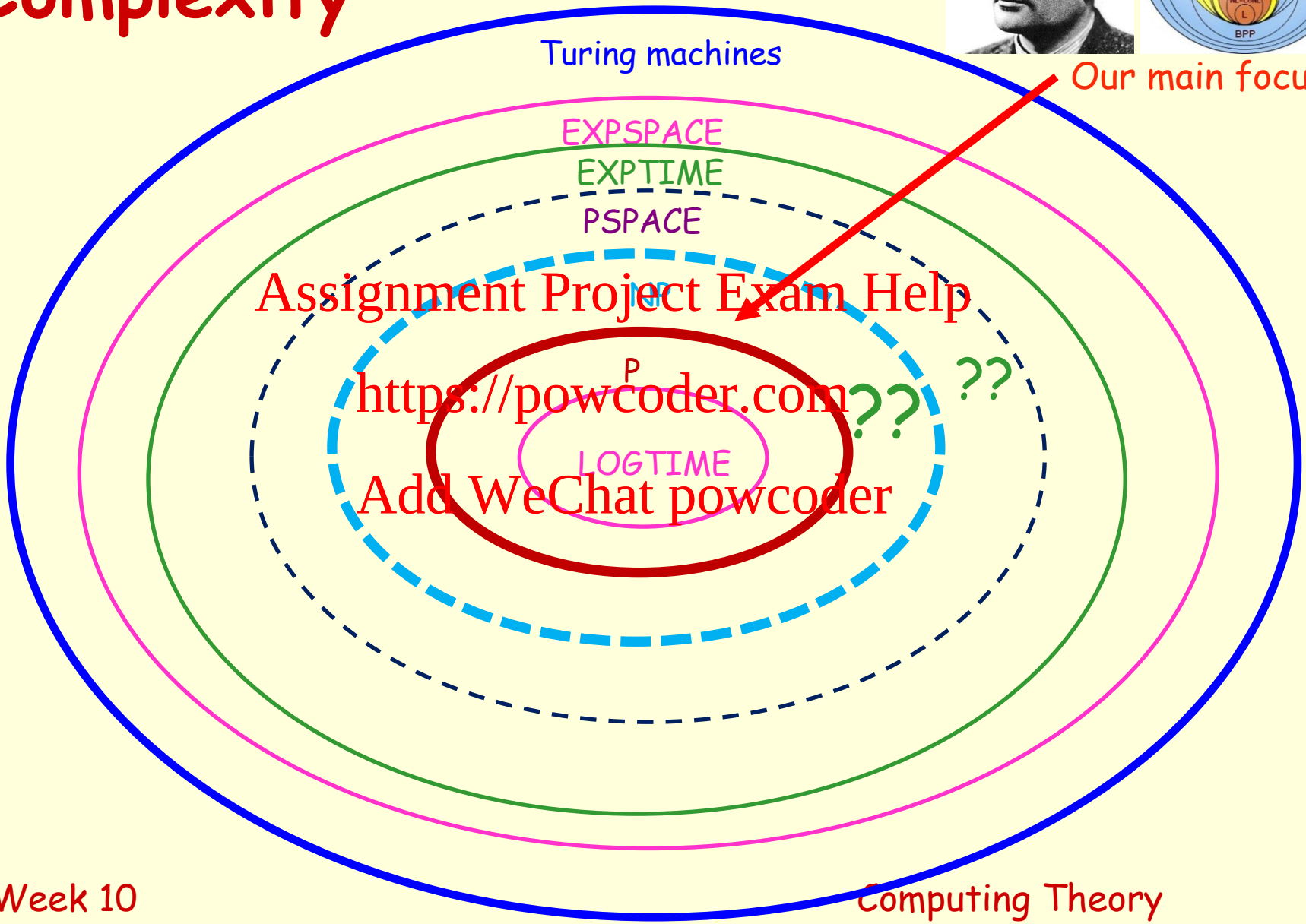
Questions?

Questions?

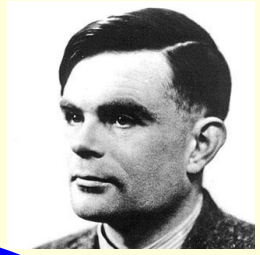
Complexity

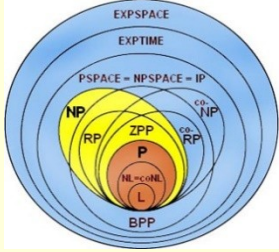



EXPSPACE
EXPTIME
PSPACE = NPSPACE = IP
NP
co-NP
RP
ZPP
co-RP
P
NL=coNL
L
BPP


Turing machines
Our main focus
EXPSPACE
EXPTIME
PSPACE
P
LOGTIME
??
??
??
Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Complexity





Checking a solution is hard

NP

Assignment Project Exam Help

https://powcoder.com

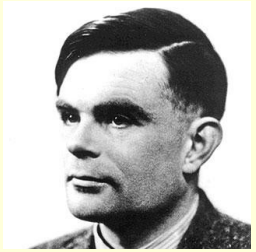
P

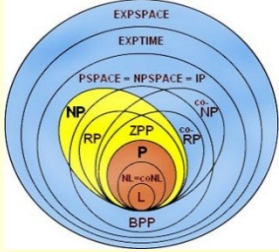
Finding a solution is easy

Add WeChat powcoder

Checking a solution is easy;
Finding a solution is hard.

Complexity





NP

3SAT

HC

TSP

…

Checking a solution is easy;
Finding a solution is hard.

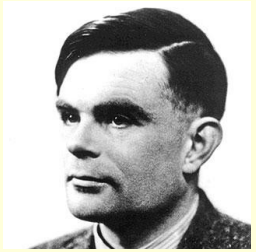
P

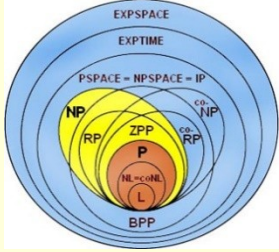
Finding a solution is easy

Primality, sorting, …

"Nobody is certain that P and NP are different, but many experts believe so …"

P = NP?

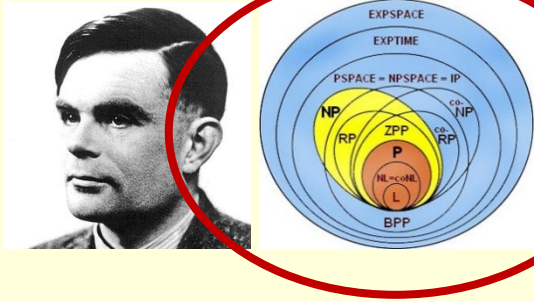


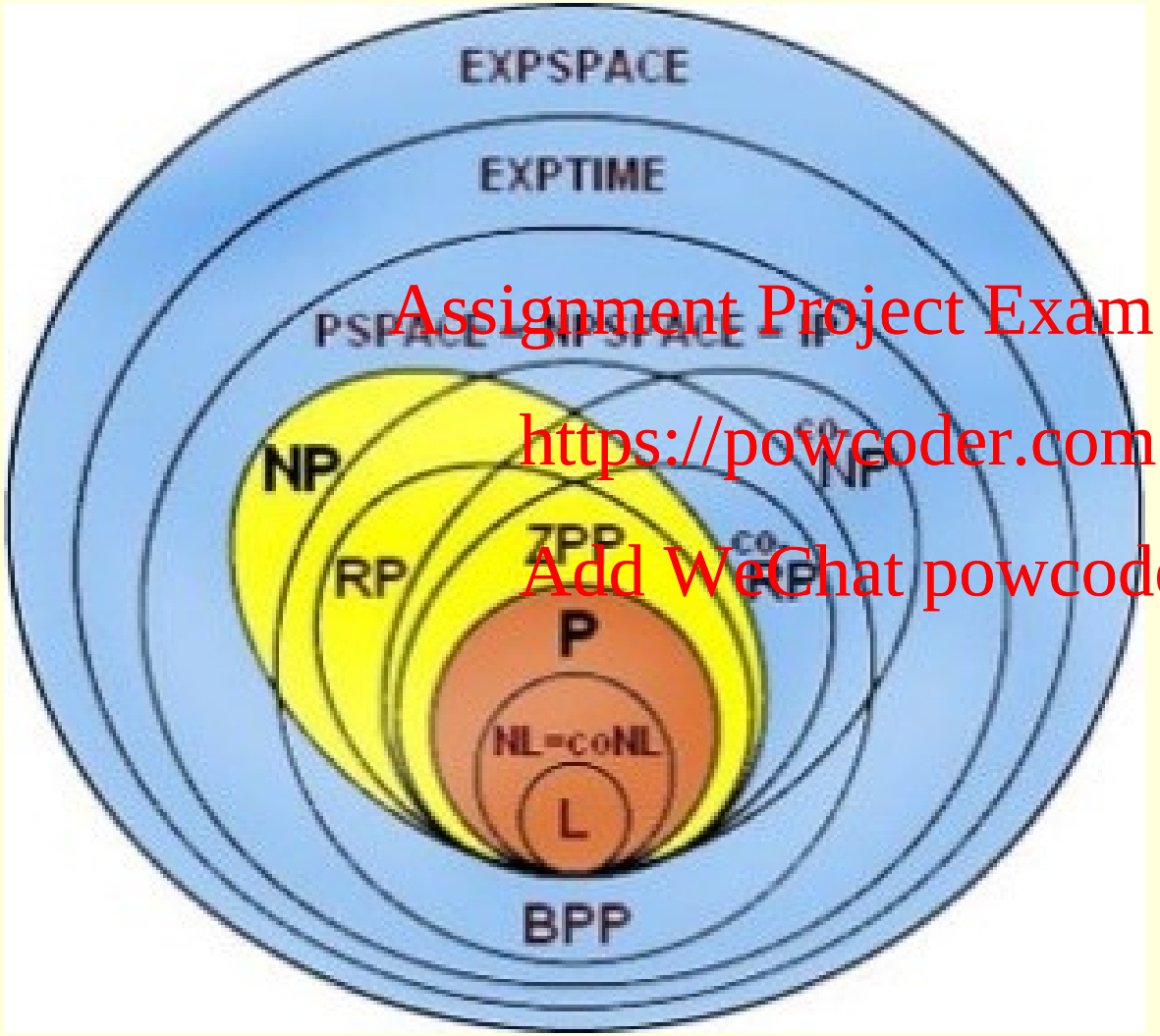


- Clearly P ⊆ NP. Is NP ⊆ P? No proof either way!
- Clay Institute in USA will give **US$1 million** to anyone who can settle this question!
- Same prize for 6 other problems
- First offered in 2000, only one has been claimed
- See http://www.claymath.org/millennium-problems and
- http://www.claymath.org/millennium-problems/p-vs-np-problem

- How could you prove P = NP?
  - *Find a polynomial-time algorithm on a (deterministic) TM for an **NP-complete** problem*
- How could you prove P ⊆ NP?
  - *Reason about all possible algorithms? (!!)*
- Neither has been done so far …

P = NP?





Many sub-classes of NP and PSPACE

No need to know all of these!

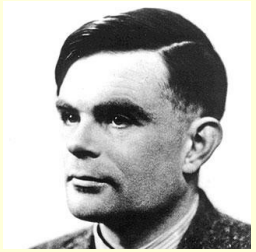
All could be **empty if P = NP**

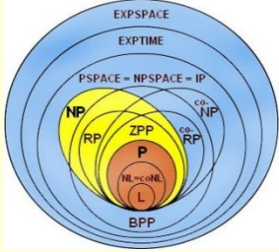
Questions?

Questions?

Questions?

NP-completeness





A problem R is **NP-complete** if

- R is in NP
- If a **polynomial-time** algorithm exists for R, then **a polynomial-time algorithm exists for every problem in** NP

3SAT is NP-complete (!!!)

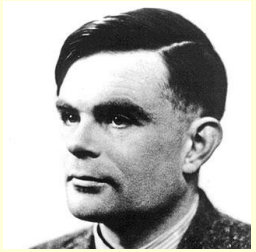
(Stephen Cook, 1971. Richard Karp 1972. Leonid Levin 1973. Cook & Karp won Turing Awards for this work)

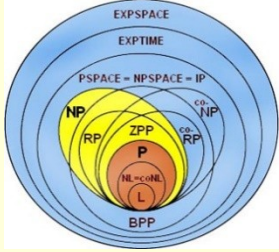
NP-complete

- "hardest" problems in NP
- solve one and you solve them all …

2SAT is in P (!!)

Complexity





NP

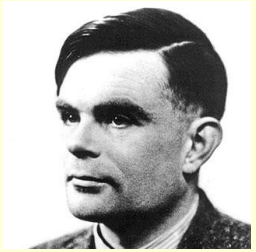
NP-complete

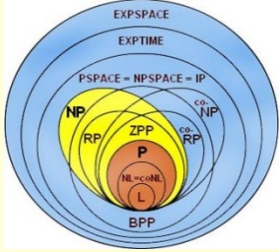
3SAT HC TSP

P

2SAT SORTING PRIMALITY

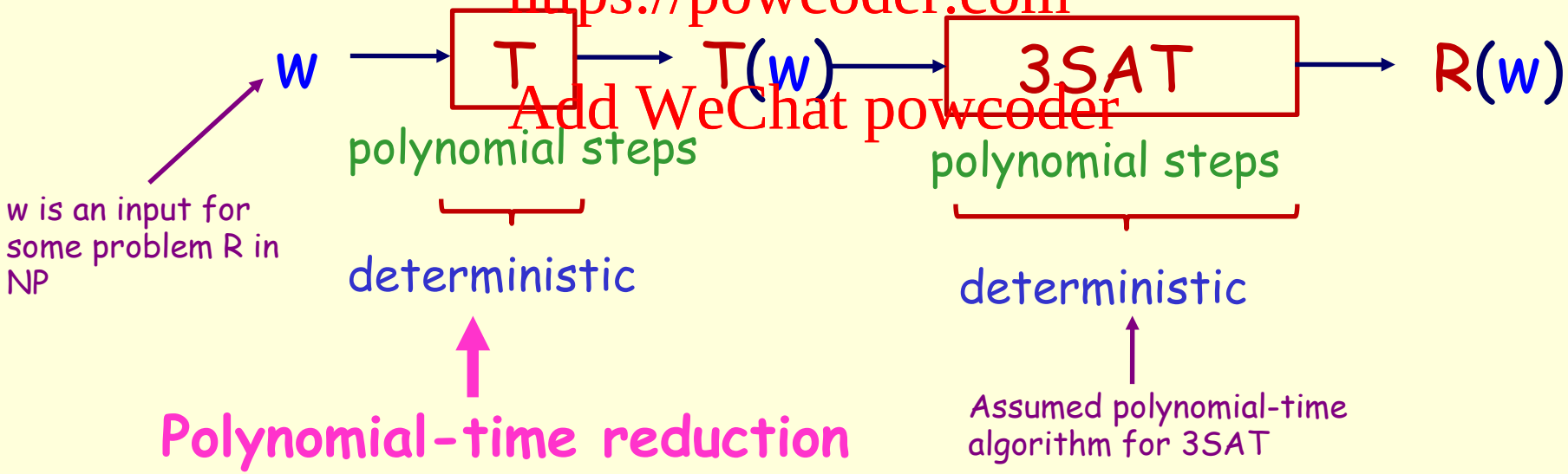
NP-completeness



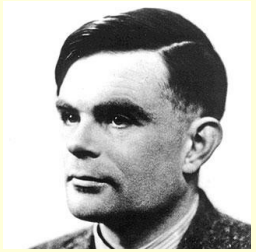


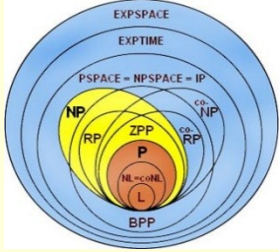
3SAT is NP-complete, i.e.

- 3SAT is in NP
- If a polynomial-time algorithm exists for 3SAT, then a polynomial-time algorithm exists for every problem in NP



NP-completeness

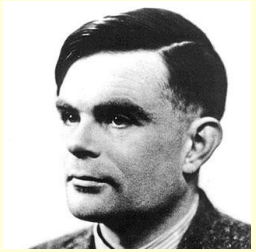


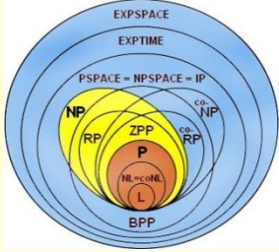


- 3SAT is NP-complete
- Hamiltonian Circuit is NP-complete
- Travelling Salesperson is NP-complete
- Vertex Cover is NP-complete
- … is NP-complete (☺)

Thousands of problems are NP-complete (!!!)
(reduce 3SAT to your favourite problem in NP, and …)

NP-completeness





Reduce problem *A* to problem B means

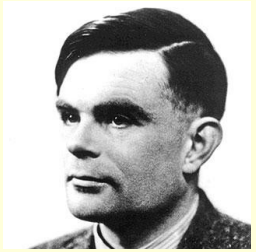
- You can solve *A* quickly if you can solve B quickly
- You can solve *A* in polynomial time if you can solve B in polynomial time
- B is at least as hard as *A*
- *A* is no harder than B

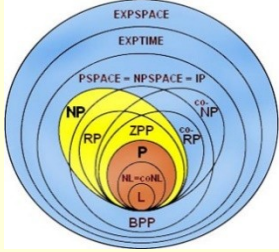
A problem R is **NP-hard** if every problem in NP can be reduced to it in polynomial time

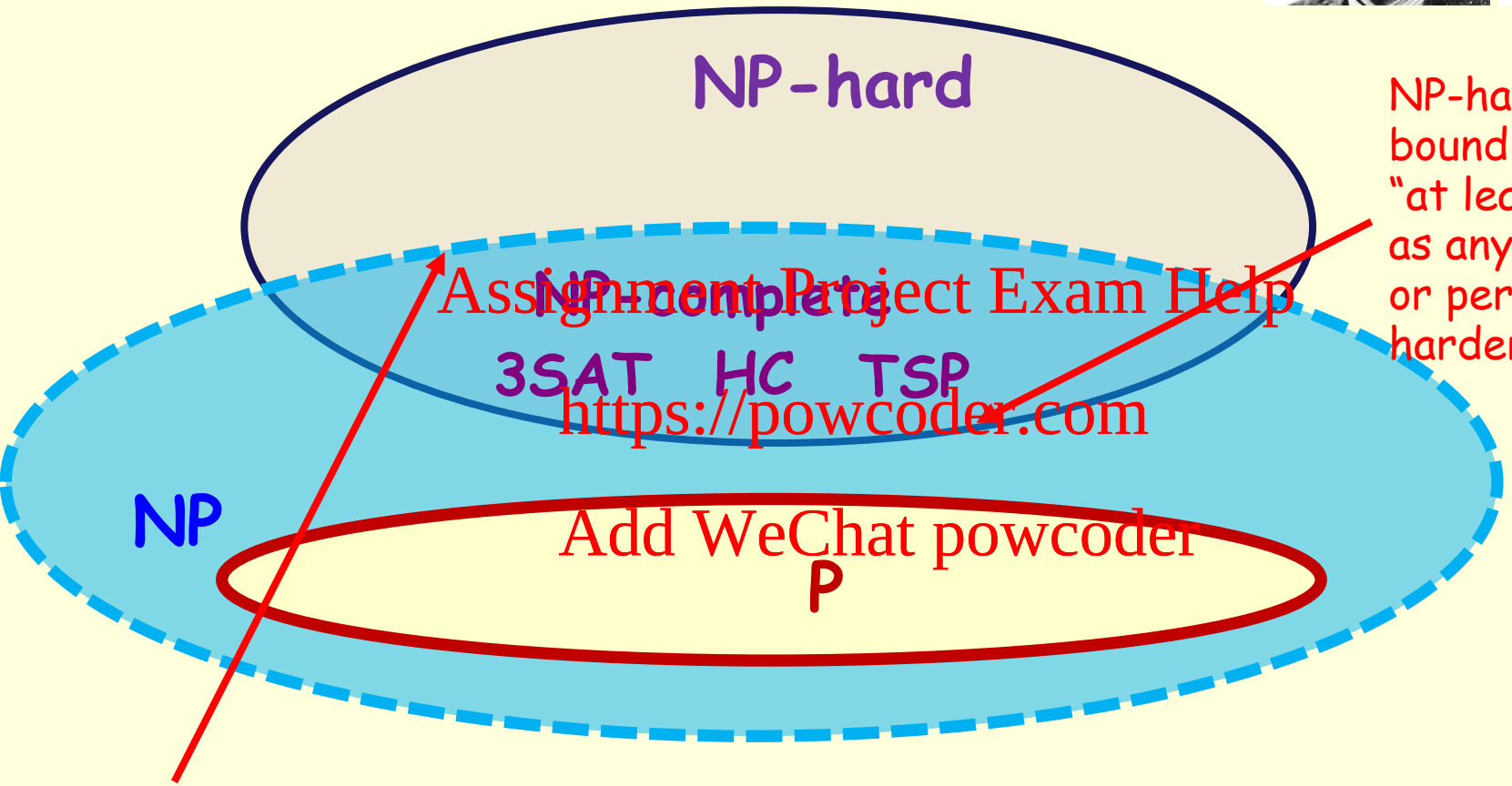
A problem R is **NP-complete** if it is both **NP-hard** and in **NP**

A problem can be **NP-hard, but not NP-complete** (so it is not in NP, as it is "too hard" for NP)

Complexity



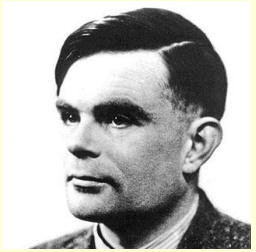


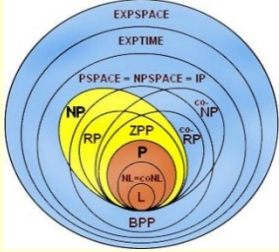


NP-hard: lower bound
"at least as hard as anything in NP, or perhaps harder"

In NP: upper bound
"No harder than checking solutions in polynomial time"

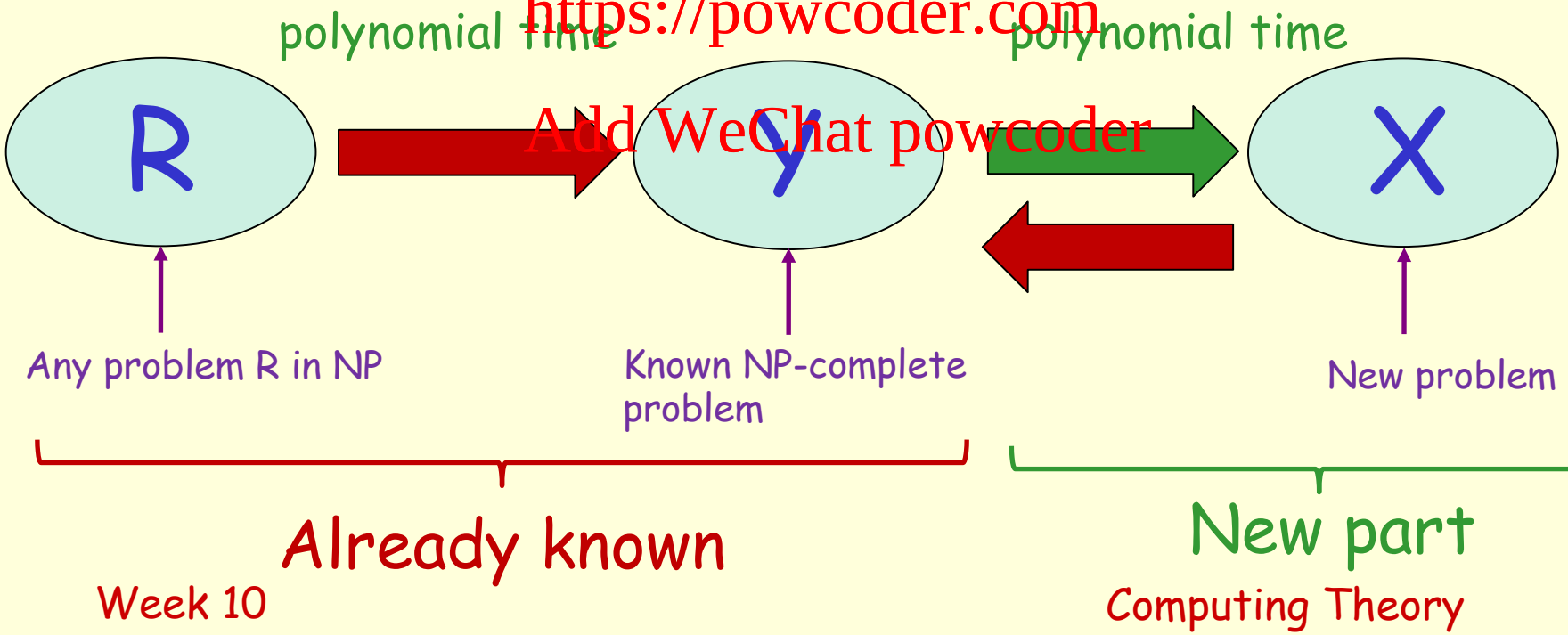
NP-completeness



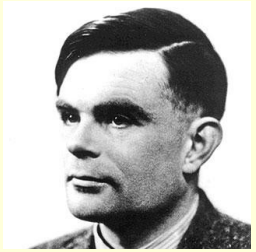


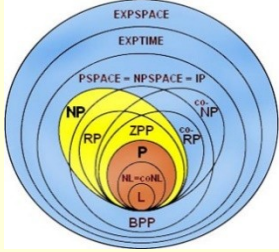
Given new problem X, how do I analyse it?

1. Show that X is in NP (usually easy)
2. Find some NP-complete problem Y
3. Find a polynomial time reduction from Y to X



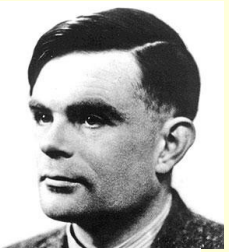
NP-completeness

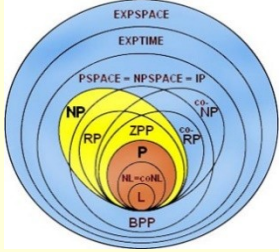


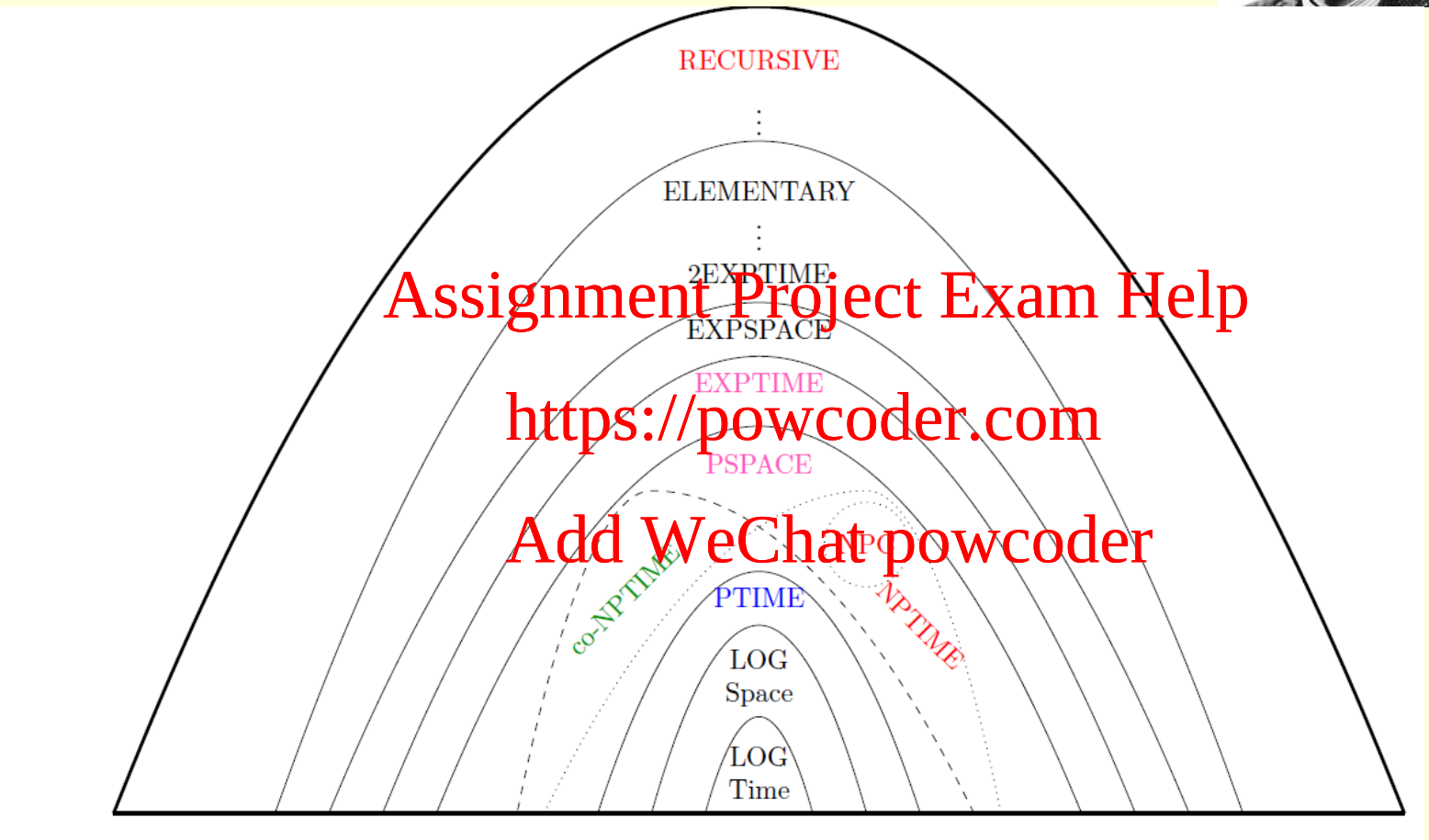


Aerospace engineering Optimal mesh partitioning for finite elements.
Biology Phylogeny reconstruction.
Chemical engineering Heat exchanger network synthesis.
Chemistry Protein folding.
Civil engineering Equilibrium of urban traffic flow.
Economics Computation of arbitrage in financial markets with friction.
Electrical engineering VLSI layout.
Environmental engineering Optimal placement of contaminant sensors.
Financial engineering Minimum risk portfolio of given return.
Game theory Nash equilibrium that maximizes social welfare.
Mechanical engineering Structure of turbulence in sheared flows.
Medicine Reconstructing 3D shape from biplane angiocardiogram.
Operations research Traveling salesperson problem, integer programming.
Physics Partition function of 3D Ising model.
Politics Shapley-Shubik voting power.
...

NP-completeness

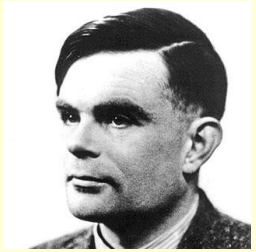


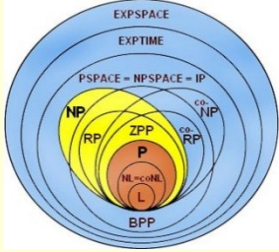


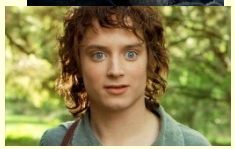


https://complexityzoo.uwaterloo.ca/Complexity_Zoo

NP-completeness







"What is the state of the art for NP-complete problems?"

"Exponential-time algorithms" (!!)

"Surely we can do better than that?"

"Er … well, I wish …"

Even Gandalf doesn't know

- whether P = NP
- whether there is a polynomial-time algorithm for some NP-complete problem

Questions?

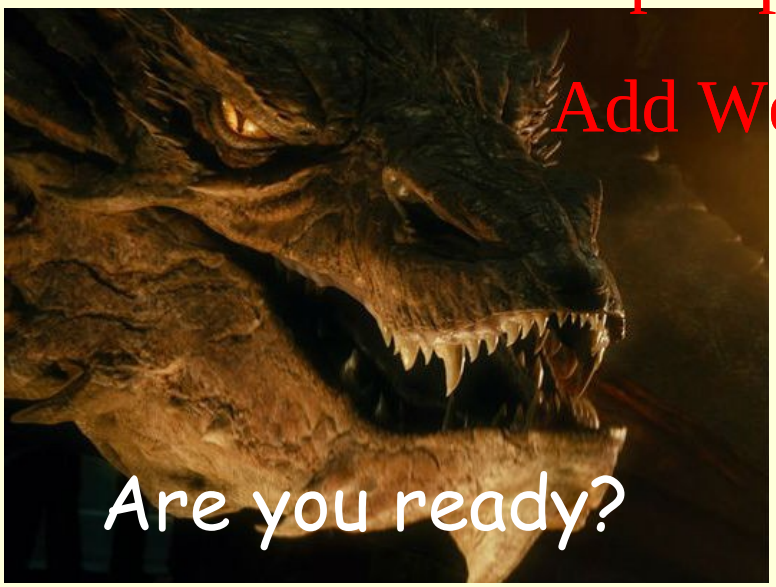
Questions?

Questions?

Quiz time!

Go to Canvas and find the quiz Lectorial 10 Question set

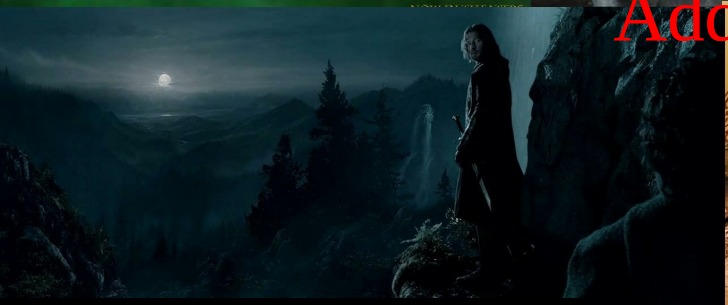
- Not worth any marks
- You can consult other students if you wish
- Time limit will be **5** minutes

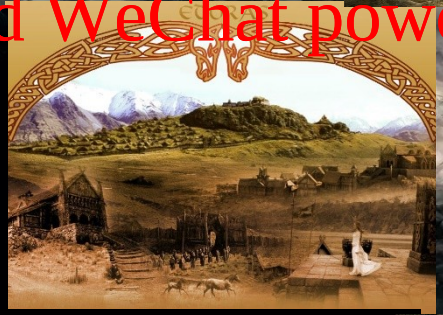


Go!

The pictures will take 5 minutes to disappear!

Thomas music means 1 minute left!



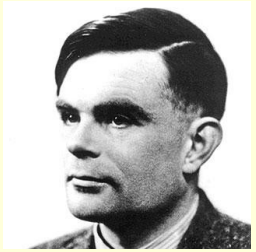


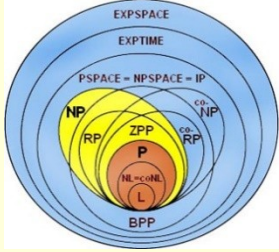
Questions?

Questions?

Questions?

NP-completeness

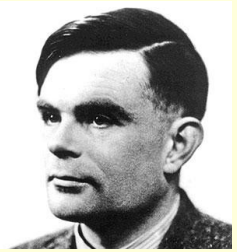


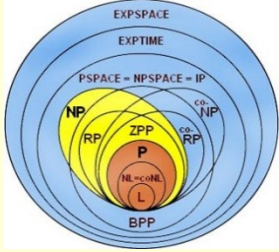


NP-complete problem

- Find an efficient algorithm (and prove P = NP) (good luck!)
- Approximation Find a sub-optimal solution efficiently
- Heuristic Algorithm which "generally" works well, but doesn't always work efficiently
- Special case Use particular information to improve performance
- Probabilistic Return an answer which is only probably correct
- Randomised Use randomised search to find something quickly

NP-completeness





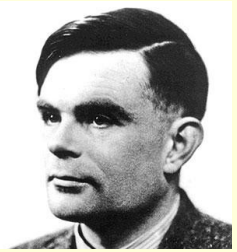
We cannot find an algorithm which

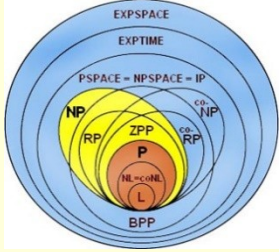
- Runs in polynomial time for all inputs
- Finds an optimal solution for all inputs

So any algorithm must

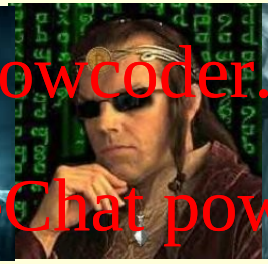
- Runs in exponential time for some inputs
- Finds a sub-optimal solution for some inputs
- Both (!!)

Approximation





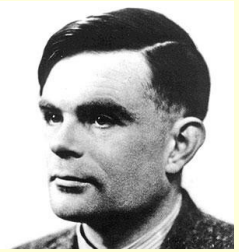
- Runs in polynomial time for all inputs
- Finds a sub-optimal solution
- Guaranteed to be efficient
- Not guaranteed to be optimal

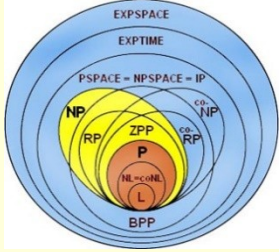


TSP

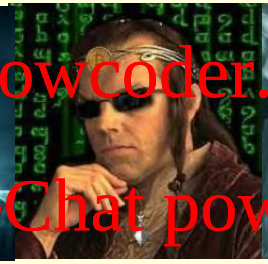
- One approximation $O(n^2)$ with solution ≤ 2 x optimum
- One approximation $O(n^3)$ with solution ≤ 1.5 x optimum

Heuristic





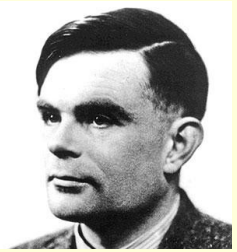
- Some inputs take exponential time
- "Common" or "typical" inputs take polynomial time
- Often use local improvements
- Few guarantees

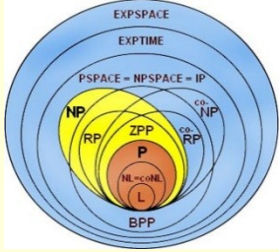


TSP

- Lin-Kernighan heuristic: swaps pairs of sub-tours
- Greedy: choose shortest next
- Inserting sub-tours
- …

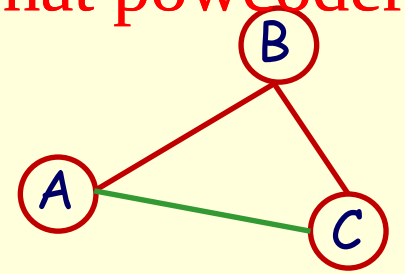
Special case





- Extra information used to improve performance
- Polynomial time for some special cases
- Approximations for some special cases

TSP with Euclidean distance:
$|AC| \leq |AB| + |BC|$



Improved approximation

HORN-SAT: **SAT** with at most one positive literal per clause
(p q r) (r q w) (p p r)

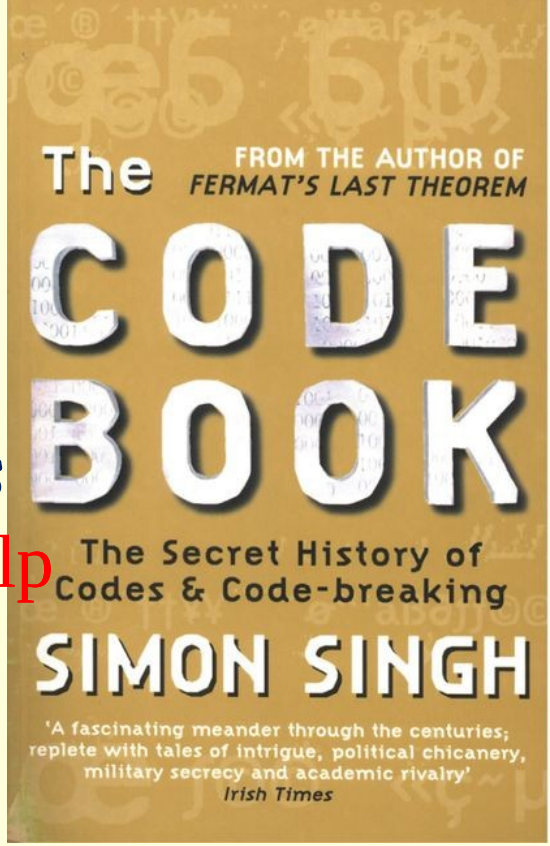
Polynomial time

Encryption

Intractability can be your friend!

Encryption historically based on secret keys

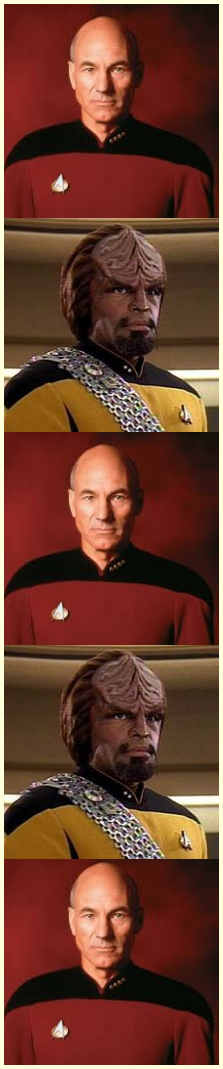
- Caesar cipher, Playfair cipher, Enigma
- Substitution and transposition
- Advanced Encryption Standard (AES)



Secret key systems have the **key distribution problem**

Question: How do you communicate a secret key securely?
Answer: With great difficulty!

Encryption



Mr Worf! Open a **secure channel** to Starfleet!

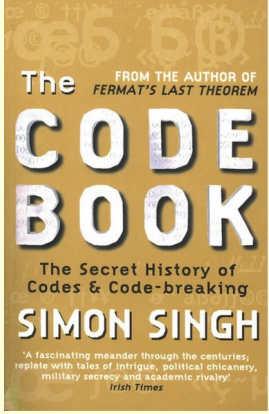
Yes Captain! How do I do that?

Send them our encryption key and get them to use that!

On an open channel? That sounds like a security risk …

Of course! Open a **secure channel**, Mr Worf!

Encryption



Asymmetric approaches were a major breakthrough!

Diffie-Hellman key exchange (1976)

- First scheme to have separate encryption and decryption keys
- Proposed by Whitfield Diffie and Martin Hellman, and also Ralph Merkle

RSA Public key cryptosystem (1977)

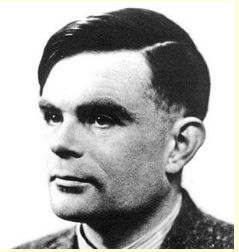
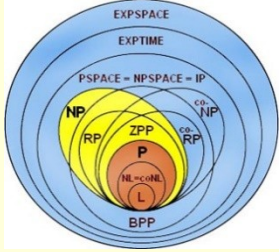
- Encryption key is public, decryption key is private
- Based on property of prime numbers (!!)

- Security assumes factorisation is intractable
- Proposed by Ron **R**ivest, Adi **S**hamir & Leonard **A**dleman

Other public key systems use more advanced mathematics

(discrete logarithms, elliptic curves, finite groups, …)

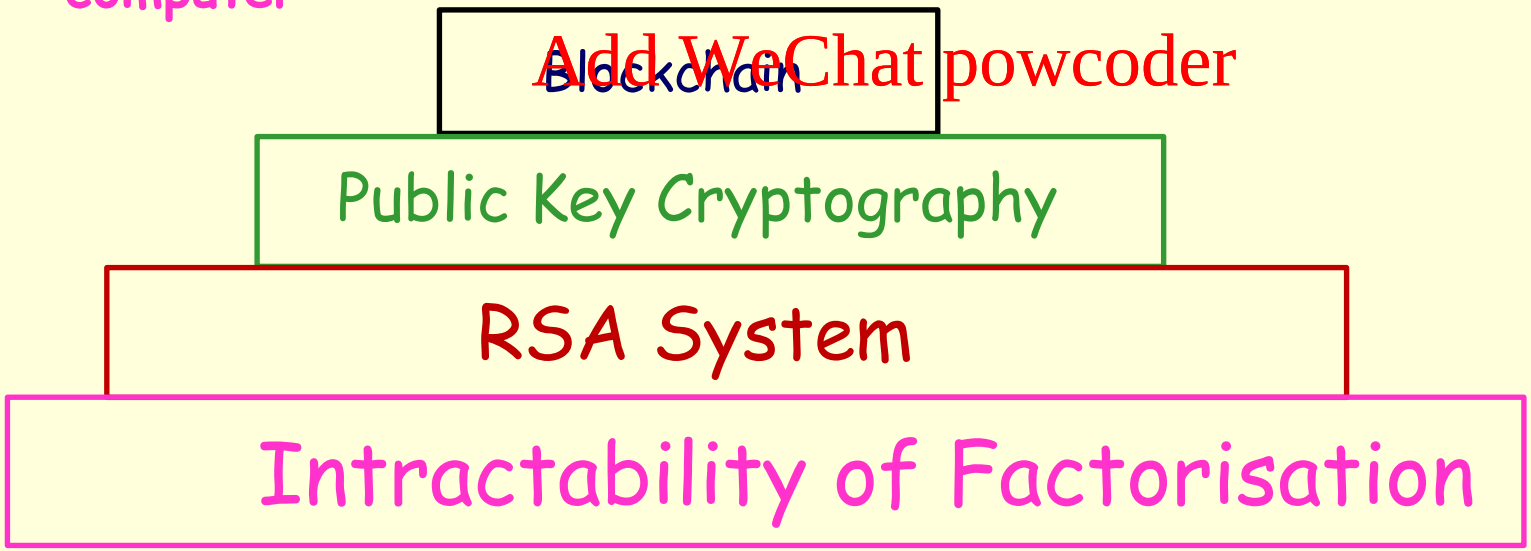
British GCHQ announced in 1990s that they knew this in 1969 … ☹

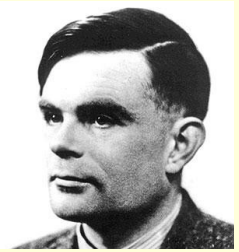
Factorisation

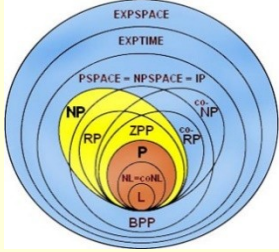
Factorisation

- 'Find factors of n'
- Intuitively harder than primality testing
- Can use factorisation for primality testing but not recommended
- **NOT NP-complete**
- Almost certainly intractable …
- Shor's algorithm is tractable for factorisation **on a quantum computer**



Intractability





Can be your friend!

INTRACTABILITY

Blockchain, SSH, ecommerce, encryption, security ...

Public Key Cryptography

Quantum Computing | RSA | Discrete Logarithms | Elliptic Curves

Intractability

Encryption

"new password is bumblebee"

Encrypt

xsgfhas fgedg

Public (so anyone can encrypt)

Private (so only Bob can decrypt)

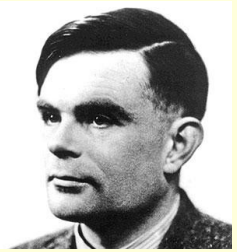
Send

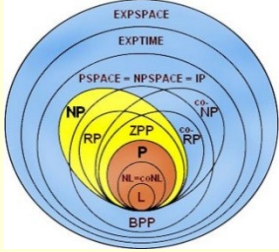
"new password is bumblebee"

Decrypt

xsgfhas fgedg

RSA System





Public Key cryptography

- Two keys, E for encryption and D for decryption
- Publish E
- Keep D secret
- Knowledge of E must not imply knowledge of D
- Hence make computing D from E intractable
- This means we need to make n large enough

E published
D kept secret

Secure if computing D from E is intractable

RSA scheme (Rivest, Shamir, Adleman 1977)

- Find two primes p and q
- Compute n = p x q and r = (p-1) x (q-1)
- Find e such that e and (p-1) x (q-1) are co-prime
- Compute d such that d x e 1 mod r
- Encrypt M by E(m) = m^e mod n
- Decrypt M by D(m) = m^d mod n
- D(E(m)) = $m^{e \times d}$ m mod n

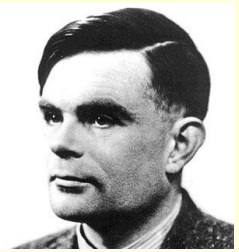
To find d from e and n, need to know p & q (and hence r)

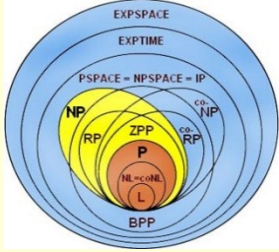
e,n public

Uses (extended) Euclid's algorithm (!!)

"Find 3 primes p, q, e" will work

Cracking RSA





RSA scheme

- Find two primes p and q
- Compute n = p x q and r = (p-1) x (q-1)
- Find e such that e and (p-1) x (q-1) are co-prime
- Compute d such that d x e 1 mod r

Public: e, n
Private: d, p, q, r

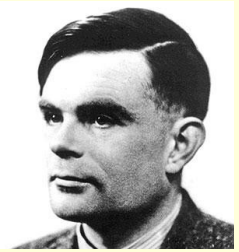
(d x e) + (z x r) = 1 = gcd(e,r)

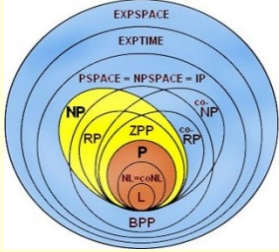
Process	Method	Complexity
Calculate d from e and r	Euclid's algorithm	Quadratic
Calculate r from p and q	Multiplication	Close to constant
Calculate p and q from n	Number field sieve	INTRACTABLE

Factorisation

- Almost certainly intractable
- Not obvious that it is harder than primality testing
- Input n has representation size log n
- Polynomial time in the size of the input means O(log n) (!!)

Setting up RSA





RSA scheme

- Find two primes p and q
- Compute n = p x q and r = (p-1) x (q-1)
- Find e such that e and (p-1) x (q-1) are co-prime
- Compute d such that d x e 1 mod r

Public: e, n
Private: d, p, q, r

Need to find primes quickly!

Primality testing

- Decision problem
- Long unknown whether polynomial or not
- Miller in 1976 showed there is a polynomial algorithm assuming the Extended Riemann Hypothesis is true

- Agrawal, Kayal, Saxena found polynomial-time algorithm in 2002 (!!)
- Kayal, Saxena were **undergraduate students at the time** (!!!)
- Little pragmatic impact because …

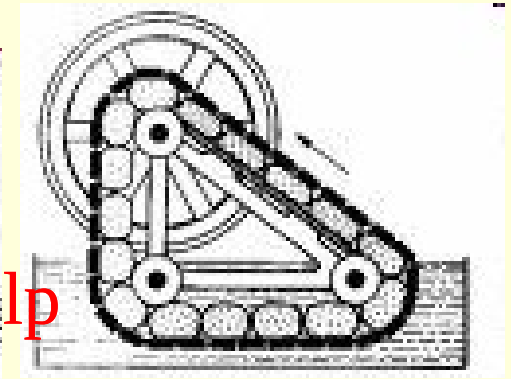
Probabilistic methods are much faster (!!)

Questions?

Questions?

Questions?

'Marvellous Machine'



`Marvellous Machine'


"Counts grains of sand **exactly** within one second"

"Yeah? Well count this!

"4,292,303,203,201,204 grains of sand"

??????

`Marvellous Machine'

"4,292,303,203,201,204"

+5

"4,292,303,203,201,209"

-12

"4,292,303,203,201,192"

+14

"4,292,303,203,201,218"

-7

"4,292,303,203,201,197"

`Marvellous Machine'

- Machine has to pass numerous trials
- Failure at any time means machine fails acceptance test

Trial 1 successful: Lucky guess

Trial 2 successful: Still just luck

Trial 3 successful: Hmm…

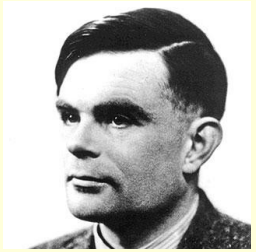
Trial 4 successful: ???

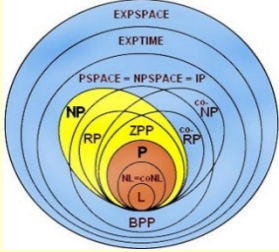
Trial 5 fails: !

`Marvellous Machine'

- Trial 1 successful: Lucky guess!
- Trial 2 successful: Still just luck
- Trial 3 successful: Hmm...
- Trial 4 successful: ???
- Trial 5 successful: So what's the trick?
- Trial 6 successful: Really ? – what is it?
- Trial 7 successful: Now this is just getting boring …
- …
- Trial 47 successful: Alright! You win! It works!

Probabilistic Algorithms

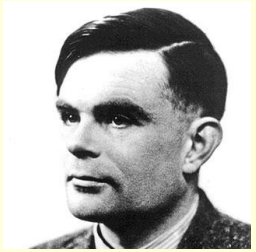


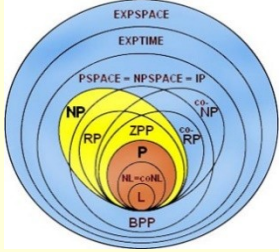


- "Approximation method" for decision problems …
- Series of tests is applied
- If **any** test fails, the answer is 'no'
- Each successful test is 'evidence' for 'yes'
- More successful tests means more likelihood of 'yes'
- Can either return
  - **No** (with certainty)
  - **Probably yes** (usually with a specific probability)

Less precision but much more efficient

Primality Testing





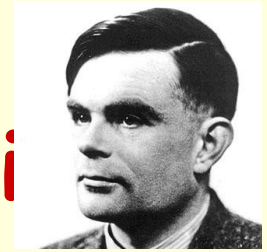
Given a number n

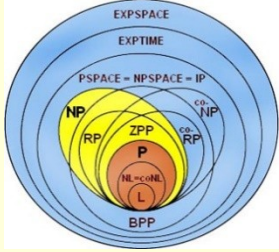
1. Choose k such that 1 < k < n
2. If n mod k = 0, halt with output 'no'
3. If enough trials done then halt 'probably yes'
4. Go to step 1.

How to choose k?

- k = 2, 3, 5, 7, … n/2(or n) gives **sieve of Eratosthenes** (exponential)
- Smarter choice means less cases to test

Probabilistic Primality Testi



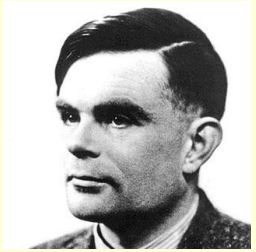


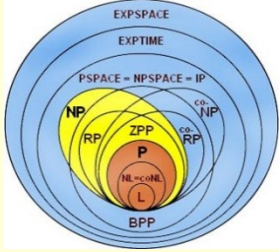
- Testing for a strict subset of {2,3,5,7,…, n} can only ever give the result `probably prime'
- Trick is to make `good' choices of k
- **Solovay-Strassen** test:

  correctness probability after **m** trials is **1 – 1/2^m** (!)
- **Rabin-Miller** test:

  correctness probability after **m** trials is **1 – 1/4^m** (!!)
- Can have arbitrarily high correctness if we perform enough trials ….

RSA Pragmatics





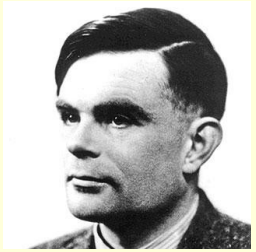
Primality testing

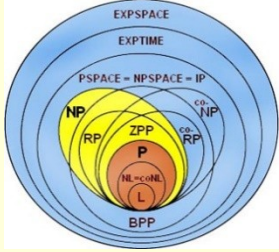
- AKS algorithm says **yes** or **no** in polynomial time
- Solovay-Strassen test says **probably yes** or **definitely no** in much shorter time!
- Rabin-Miller test says **probably yes** or **definitely no** in much shorter time still!
- The trick is to increase the probability of correctness to almost 1 after a smallish number of steps …

n	SS(n)	RM(n)
1	50%	75%
2	75%	94%
5	97%	99.9%
10	99.9%	99.9999%
25	99.999997%	99.99999999999%

Any size number can be tested for primality in about 25 trials … (!!)

RSA Properties

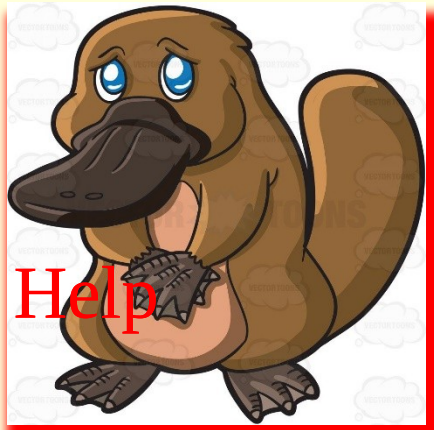




- Can be used for any encryption task
- Encryption and decryption speeds slow compared to secret key methods (eg AES)
- Often used to distribute secret keys
- Used in SSH and similar tools
- Security depends on the size of the primes used
- 1024 to 4096 bits recommended, 2048 considered 'safe'

- Threats
  - Factorisation being tractable
  - Quantum computing (using Shor's algorithm)
  - Other public-key systems with shorter keys and more efficient encryption
  - …

The Platypus Game



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

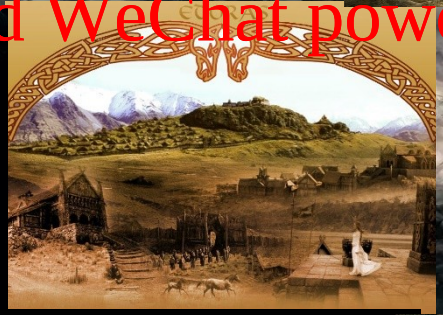
Assignment 2

- Platypus tournament for **2,500** machines
- Earlier distribution had 25,500 (in error)
- If 2,500 machines takes more than 4 hours, **reduce the number** (to say 2,000)
- Three tournaments to be run:
  - All 2,500 machines
  - Only machines classified as 'reachable'
  - Only machines classified as 'none' or 'unreachable'
- Pointers on intractability question will be posted soon

That's it!

I am out of here!

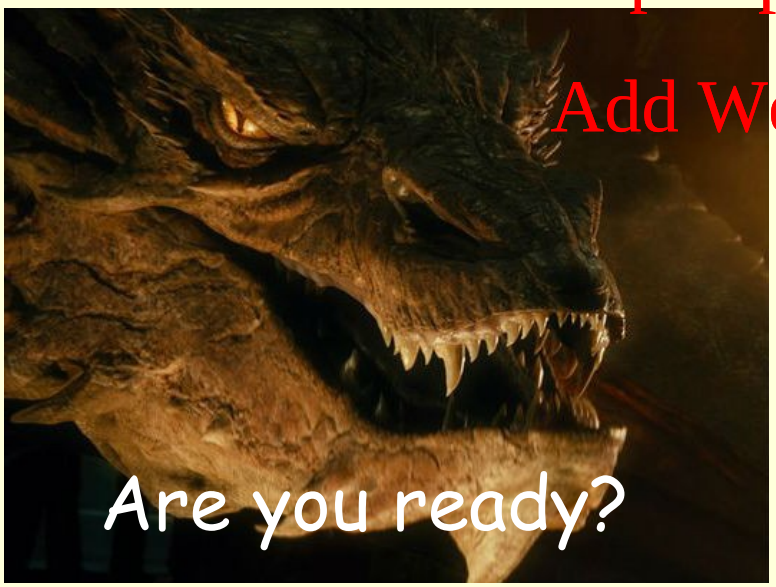
Break time! (We resume when all the pictures are gone! This will take 3 minutes!)



Quiz time!

Go to Canvas and find the quiz Lectorial 10 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!