Feasible vs not feasible

2^n – Not feasible: takes more time than lifetime in the universe

Definition: An algorithm A is called *feasible* if $t_A(x) \leq P(len(x))$ for all inputs x, for some polynomial $P(n)$

$t_A(x) \equiv$ comp. time of algorithm A on input x

1) This definition is not perfect.

Example 1: $t_A(x) = 10^{100} \cdot len(x)$

From practical viewpoint: Not feasible
From the viewpoint of a definition: – it is feasible

Example 2: $t_A(x) = \exp(10^{-100} \cdot len(x))$

From the viewpoint of the definitions: Not feasible
From the practical viewpoint: feasible

Feasible ≡ polynomial-time
$t_A(x) \leq P(len(x))$

Algorithms feasible vs non-feasible

Problems: some are easy, some are hard.

- What is a problem? Well-defined.
- Let's recall what problems people solve in different disciplines

Mathematicians: Prove results

input x – formulation of the statement
What do we want: y – a proof of x
or ~x.
Checking whether y is a proof is feasible $C(x, y)$ alg. for checking
The proof has to be of feasible size:
$len(y) \leq P_C(len(x))$

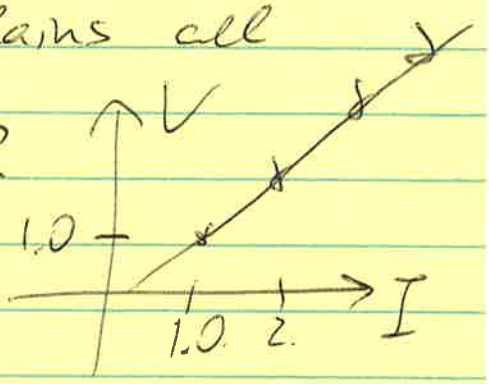
We have: – a feasible algorithm $C(x, y)$
– a polynomial $P_C(n)$
Given: x
we want: y such that $C(x, y)$ is true
and $len(y) \leq P_C(len(x))$

Physicists:

Input: x - observations & results of experiments

What do we want: We want some formula y that explains all the observations

Example: Ohm's law $V = I \cdot R$



Checking whether x fits formula y is feasible: $C(x, y)$

$len(y) < len(x)$

Engineers:

Input - specifications (specs) X

weight, stress, cost

What do we want: a design y

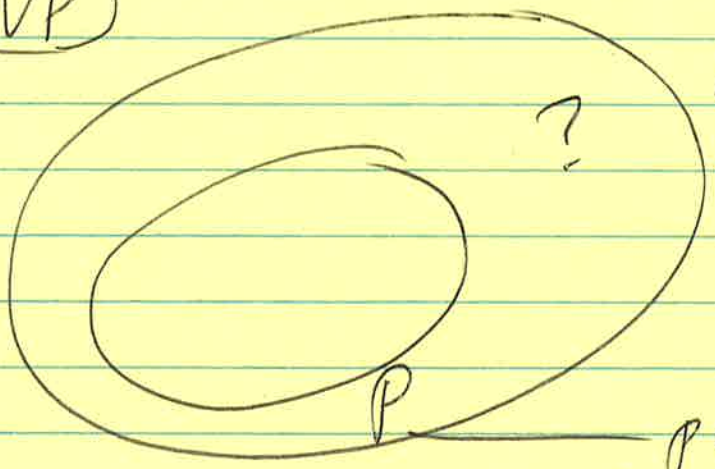
Checking is feasible: computer simulation $C(x, y)$

$len(y)$ should be feasible

A problem means if we make a guess, we can check in polynomial time whether this guess is a solution.

= (Non-deterministic (Polynomial algorithm

(NP)



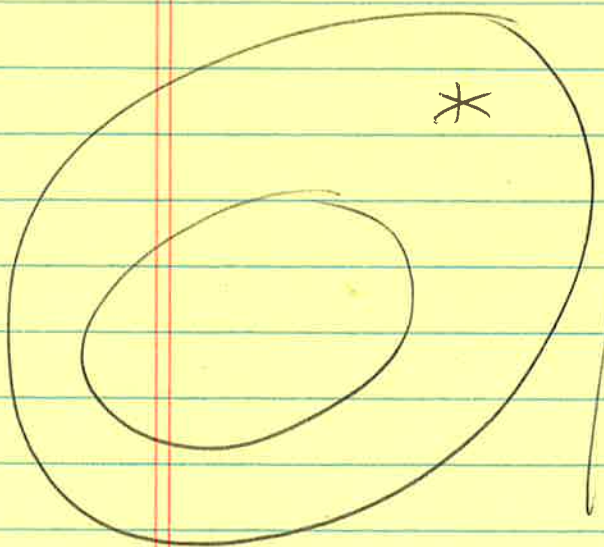
NP → all problems from the class P

P — problems that can be solved by a feasible algorithm

Open problem: Is $P = NP$?

What do we know?

Reduction:



$ax^4 + bx^2 + c = 0$ —?

we know:

$ay^2 + by + c = 0$

$y_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$

11/28/18
S—

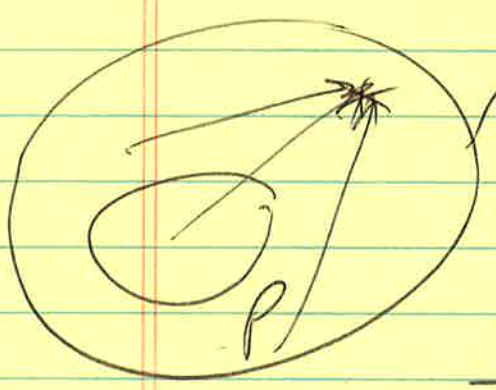
We reduce;

$y = x^2$ $\quad x = \sqrt{y}$

$a + \frac{b}{x} + cx = 0$ $\quad \frac{ax + b + cx^2}{x} = 0$

$0 \cdot x^2 + ax + b = 0$

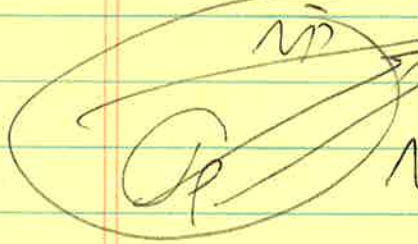
We do know: Some problems are hardest possible in NP



every problem from NP can be reduced to this problem

NP-complete.

Problems not in NP: - optimization problems
- games & conflicts



NP-hard — more general

NP-complete ≡ NP-hard and in NP

Examples of NP-complete problems:

Propositional satisfiability: SAT

Given: a prop. formula (Boolean expression)
$(a \vee b \vee \neg c) \wedge (d \vee b \vee \neg c) \wedge \ldots$

Finds values of variables that make it true

NP-complete problems:

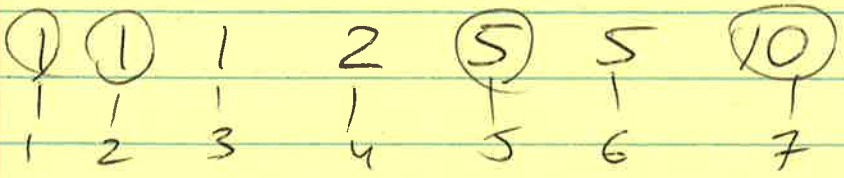
- Solving a system of quadratic equation is NP-complete
- linear equations with $x_i \in \{0,1\}$ NP-complete
- Simple example: exact change problem

$s_1, s_2 \dots s_n$ — coins in your pocket

you want S

you want $x_i \in \{0,1\}$

$$\boxed{\sum x_i s_i = S}$$



$S = 17$

$x_1 = 1 \quad x_6 = 0$

$x_2 = 1 \quad x_7 = 1$

$x_3 = 0$

$x_4 = 0$

$x_5 = 1$

11/28/18
-7-

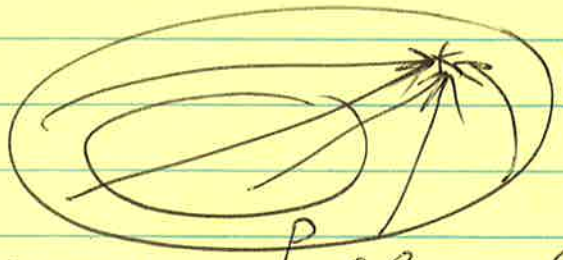
NP: class of all problems in which, once you have a ~~guess~~ candidate for a solutions, you can check feasibly whether this candidate is a solution

---

A more precise definition:
- ~~alt~~ a feasible algorithm $C(x, y)$
- a polynomial $P_C(n)$

Given: a string x
we wants to find y such that $C(x, y)$ is true and $len(y) \leq P_C(len(x))$

NP-complete: a problem is NP-complete if it is in NP and every problem from NP can be reduced to this problem



NP-hard: every problem from the class NP can be reduced to this problem

Why do we need all this? If a problem is NP-hard, we probably cannot solve it in feasible time