COS210 - Theoretical Computer Science
Complexity Theory

Complexity Theory

We have looked at **what can be computed** (decidable) and **what cannot be computed** (undecidable)

But we have not considered the **efficiency of a computation**

We will now consider a classification of decidable languages based on the running time of the "**best" algorithm** that decides them

> Given a decidable language A, we are interested in the **"fastest" algorithm** that, for any given string w, decides whether or not $w \in A$

The Running Time of Machines and Algorithms

Let M be a Turing machine and let w be an input string for M. The **running time of M on input w** is defined as:

$t_M(w) :=$ the **number of computation steps** made by M on string w

Let M be a Turing machine, let $T : \mathbb{N}_0 \to \mathbb{N}_0$ be a function called the **running time function**, let A be a decidable language over an alphabet Σ

- M **decides** language A in time T, if $t_M(w) \leq T(|w|)$ for all $w \in \Sigma^*$

The running time function T is a function of the length of the input

$T(n)$ is an **upper bound on the running time** of M, on any input of length n

The Complexity Class P

Definition

An algorithm M decides a language A in **polynomial time**, if there exists an integer $k \geq 1$, such that the upper bound on the running time of M is $\mathcal{O}(n^k)$, where n is the length of the input

- A language that can be decided in one of the models (such 1-tape TM, k-tape TM, or C program) in polynomial time, can also be decided in polynomial time in all other models
- the size of the integer k may differ between the models
- The corresponding **complexity class** is

$$P := \{A : \text{ the language } A \text{ is decidable in } \textbf{polynomial time}\}$$

Example of a Language in P

The language of palindromes

$$Pal := \{w \in \{a, b\}^* : w \text{ is a palindrome}\}$$

is in the complexity class P

- A **one-tape Turing machine** can decide Pal in time $\mathcal{O}(n^2)$
- A **two-tape Turing machine** can decide Pal in time $\mathcal{O}(n)$

For any input of length n, a one-tape machine can decide in at most n^2 steps whether the input is a palindrome, whereas a two-tape machine can make the decision in at most n steps

Context-Free Languages are in P

We have proven that context-free languages are *decidable*

Moreover, it can be shown that the following theorem holds:

Theorem

Let Σ be an alphabet, and let $A \subseteq \Sigma^*$ be a context-free language. Then $A \in P$

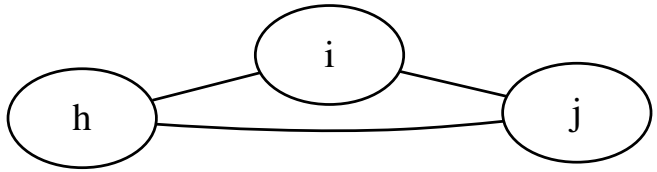
Hence, context-free languages are decidable in **polynomial time**

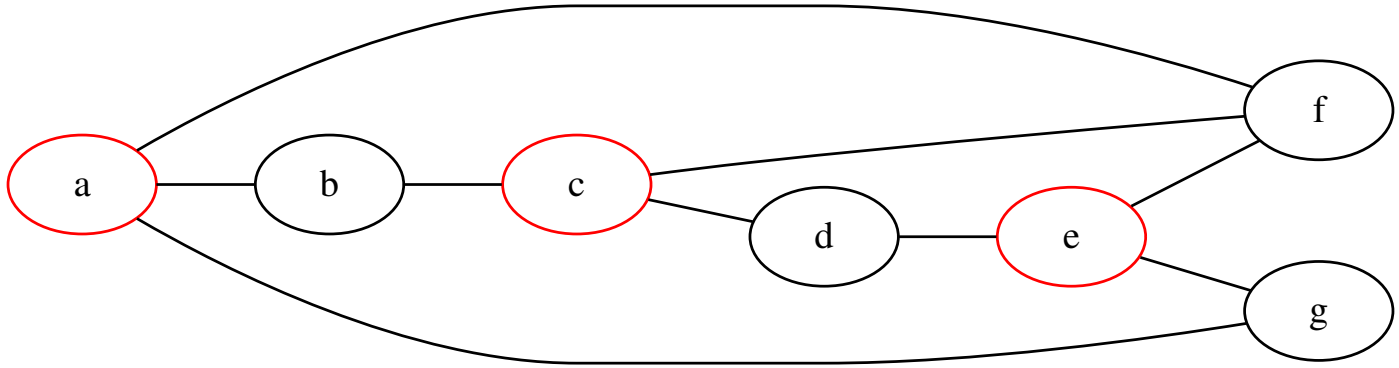
The proof is based on a technique called *dynamic programming* (not in the scope of COS210)

The 2-Colouring Problem is in P

Given a **graph** $G = (V, E)$ and a set of **two colors** $C = \{black, red\}$

Does there exist a colouring of the vertices such that **all neighboured vertices have different colors**?





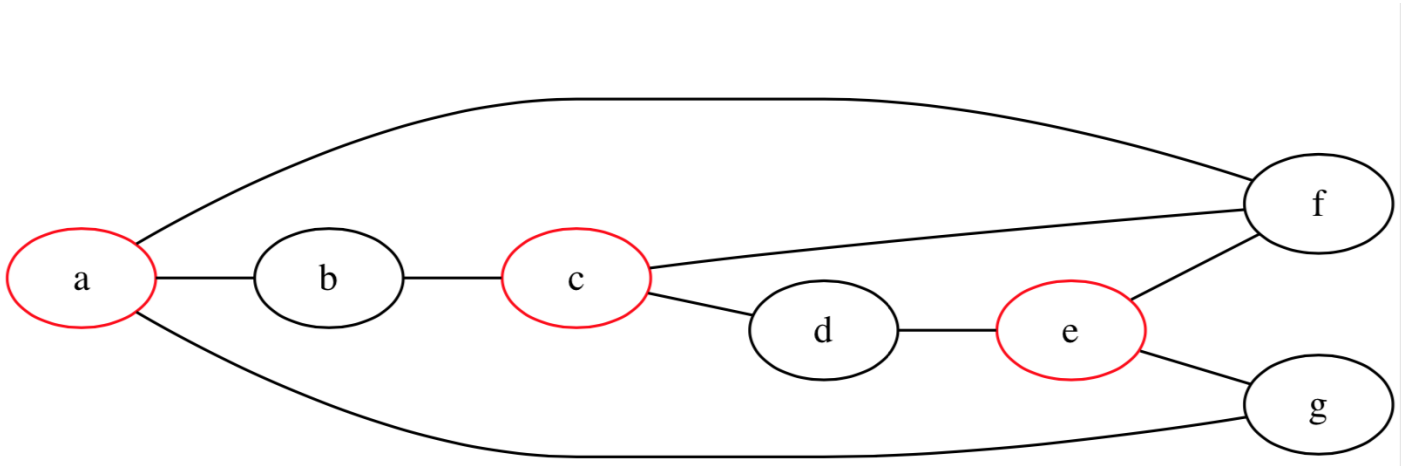
Algorithms for 2-colouring based on DFS and have running time of $\mathcal{O}(n)$

Language of the problem: $2Col = \{\langle G \rangle : G \text{ is a 2-colourable graph}\}$

The 2-Colouring Problem is in P

2-Colouring is a **decision problem**, the answer is **yes** or **no**

But what is a **solution** of this problem?



A mapping of vertices to colours:

$$c(a) = red, c(b) = black, c(c) = red,$$
$$c(d) = black, c(e) = red, c(f) = black, c(g) = black$$

For the 2-colouring problem, any solution has a length of $|V|$

The Complexity Class *NP*

While P stands for (deterministic) polynomial time,

NP stands for **non-deterministic polynomial time**

- For deterministic Turing machines, the next computation step is **uniquely determined**
- In non-deterministic machines, there are one or **more possibilities** for the next computation step, and the machine **chooses** one of them
- For certain problems it is only known that a non-deterministic machine can solve them in polynomial time
- However, it will be only polynomial time if the machine makes the **right choices** (e.g. guessing when the middle of an input string is reached)

The Complexity Class *NP*

A language A over an alphabet Σ is in complexity class NP, if for every string $w \in \Sigma^*$ the following holds:

String $w \in A$ if and only if

1. There is a solution s with length $|s| = \mathcal{O}(|w|^k)$ for some $k \geq 1$, i.e. the solution is **polynomial** in the length of the input string
2. It can be verified in **polynomial time** whether s is a **correct solution** or not

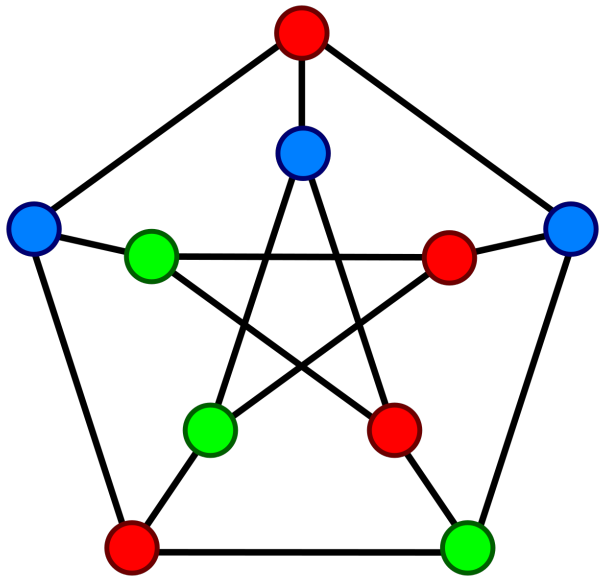
It is generally assumed that $P \neq NP$, but this could not be proven yet (million dollar problem)

It is clear that $P \subseteq NP$:
If we can **compute** a correct solution in polynomial time, then we can also **verify** whether a given solution is correct in polynomial time

The k-Colouring Problem is in NP

Given a **graph** $G = (V, E)$ and a set of $k > 2$ colours C

Does there exist a colouring of the vertices such that **all neighboured vertices have different colors**?

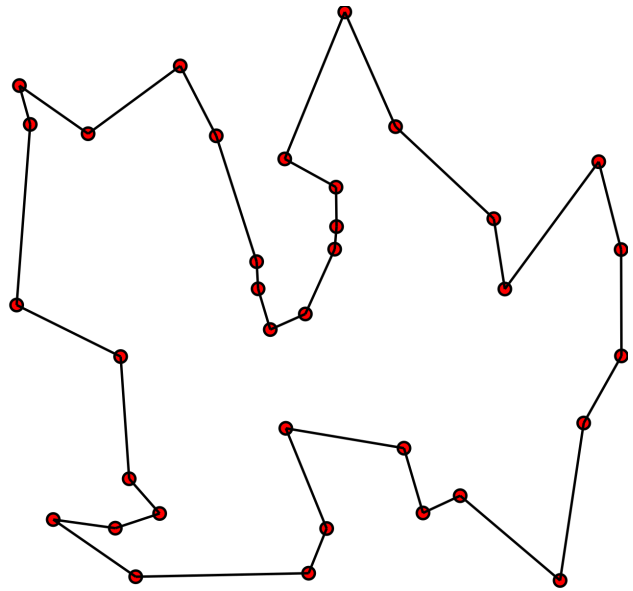


Deterministic algorithms for k-colouring have running time of $\mathcal{O}(2^n \cdot n)$

The Travelling Salesperson Problem is in *NP*

Given a list of cities and the distances between each pair of cities

Does there exist a **tour of length at most** k that visits each city and returns to the origin city?



NP-Complete and *NP*-Hard Problems

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

- $A \in NP$, and
- every problem in *NP* is reducible to A in polynomial time

A computational problem A is *NP*-**hard** if

- every problem in *NP* is reducible to A in polynomial time (here A may be a problem outside the class *NP*)

Both k-colouring and travelling salesperson are *NP*-complete

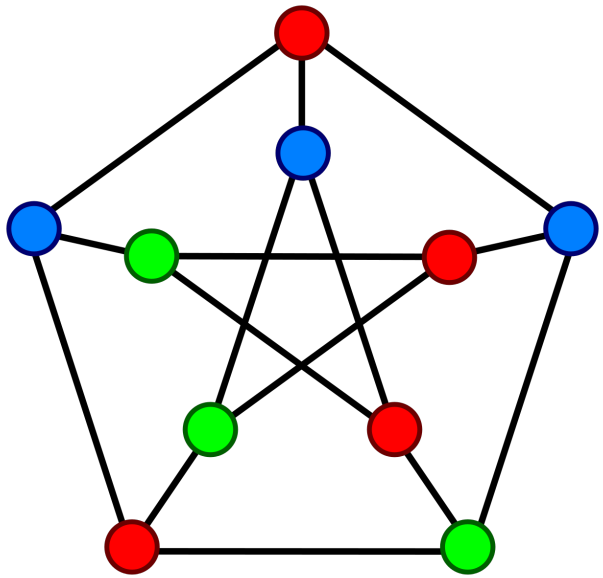
Hence, we can reduce k-colouring to travelling salesperson and vice versa

Since $P \subseteq NP$, we can also reduce 2-colouring to k-colouring or to travelling salesperson, but not vice versa

The Colouring Optimisation Problem

Given a **graph** $G = (V, E)$

What is the **minimal number of colours** that is needed for a colouring where **all neighboured vertices have different colors**?

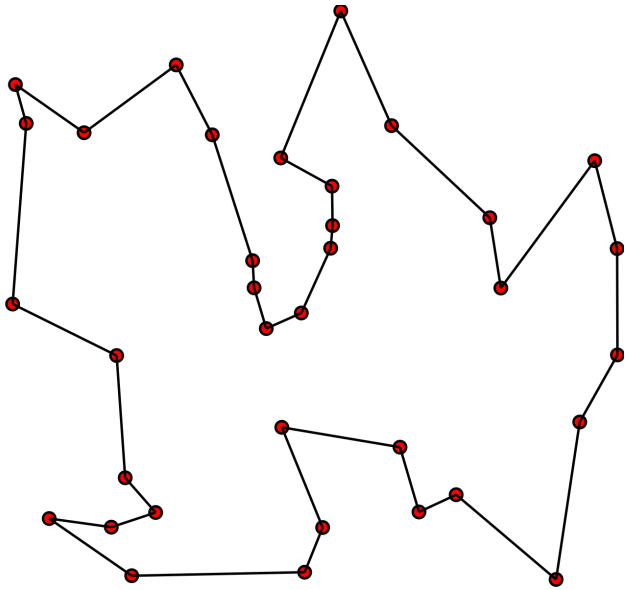


The colouring optimisation problem is *NP*-**hard**
Not solvable in non-deterministic polynomial time

The Travelling Salesperson Optimisation Problem

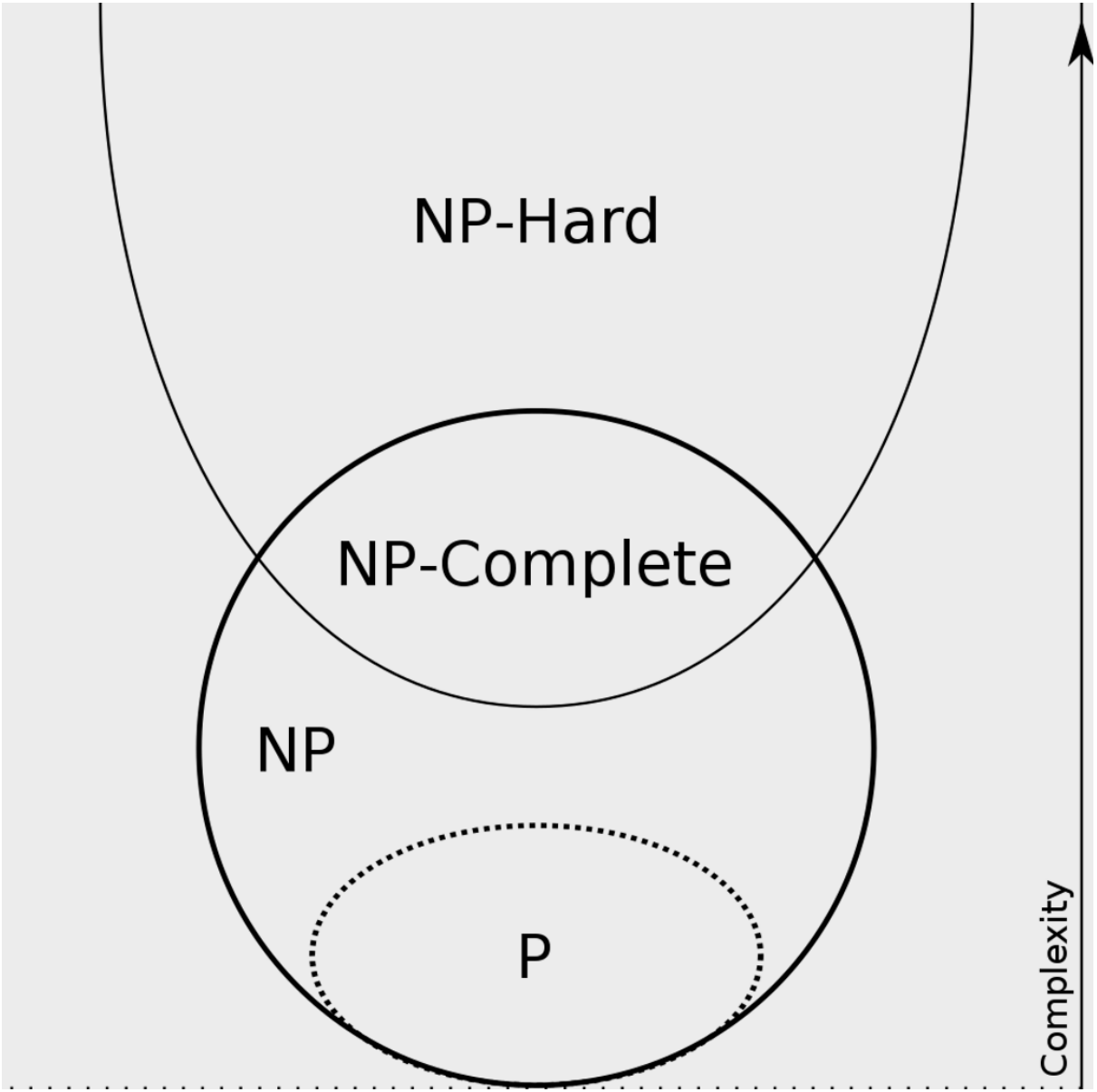
Given a list of cities and the distances between each pair of cities

What is the **shortest possible route** that visits each city and returns to the origin city?



The travelling salesperson optimisation problem is *NP*-**hard**
Not solvable in non-deterministic polynomial time

Complexity Class Relations under Assumption $P \neq NP$



The Big Picture of COS 210