Introduction

Admin: no homework submission

Computability and Complexity

Computability

- abstract, fully known, explored
- what can be computed, what can not?
- which
    - functions can be computed?

      $$f : \mathbb{N}^k \to \mathbb{N}$$

    - languages can be decided?

      $$L \subseteq \Sigma^*$$

    - "problems" can be solved?
    - with a Turing machine
    - *three equivalent statements*
- Cardinalities
    - How many functions

      $$\mathbb{N} \to \mathbb{N}$$

      uncountably many
    - How many

      $$f : \mathbb{N} \to \{0, 1\}$$

        - any f can be characterized by an indicator sequence (0-0, 1-0, 2-1, 3-1, ...) → as many as natural numbers

0	**1**	**2**	**3**	**4**	**...**
0	0	1	0	1	...

        - this is how we can characterize subsets of natural numbers (bitmask idea)
        - uncountably many
    - Turing machines
        - Finite transition table

          → only countably many

→ many functions are uncomputable

Complexity

- Only concerned with computable functions / problems
- How difficult / expensive
    - time: run length
    - memory requirement
    - algorithm description complexity
    - energy consumption
    - descriptive complexity (understanding the problem)
        - how much logic to invest in order to understand the algorithm
    -
- Example functions in
$$\mathbb{N} \to \mathbb{N}$$
    - super cheap:
    $$f : n \mapsto 0$$
    - little more expensive:
    $$f : n \mapsto n$$
    - more:
    $$f : n \mapsto n^2$$
    $$f : n \mapsto e^n$$
    $$f : n \mapsto 0 \text{ if } n \text{ is prime - 1 if } n \text{ is not prime}$$
    $$f : n \mapsto \text{smallest prime factor of } n$$

Example:

- suffix tree algorithm
    - input: long string (e.g. DNA sequence)
    - output: reordering string into a tree (suffix tree) →
    $$O(n) + c \text{ (huge overhead)}$$
    - naive algorithm:
    $$O(n^2)$$
- Matrix multiplication
    - Naive algorithm
    $$O(n^3)$$
    - Strassen's algorithm

$$O(n^{2.3\dots}) + c$$

- Conjecture

$$\rightarrow O(n^{2+\epsilon})$$