Computational Complexity

- a closer look at *decidable* problems
- emphasis on "feasibility" and "efficiency"
- measured in terms of
  1. running time
  2. memory space
  3. number of processors etc.

Running time $t_M(w)$ of a Turing machine M on input word w:
the number of steps M carries out on w from the initial configuration to a halting configuration.

Ex.: DFA can be considered as a type of Turing machine.
Upon reading a character, a DFA changes state and advances its head.
Hence, any Turing machine A simulating some DFA has running time $t_A(w) = 2 \cdot |w|$ for any input w.

Palindromes revisited

$L_{pal} = \{w \in \{a,b\}^* \mid w = w^R\}$

WHILE input nonempty DO
    Memorize current symbol
    Run to the right end
    IF current symbol does not match memorized
        THEN return FALSE
        ELSE
            Erase rightmost symbol
            Run to the left end
            Erase leftmost symbol
OD
return TRUE

Running time of this algorithm: $t(w) = O(|w|^2)$.
Exercise: Determine c_0, c_1, c_2 such that

$$t(w) = c_2|w|^2 + c_1|w| + c_0$$

What time do we measure?

The TM M for palindromes has the property that $t_M(w) = f(|w|)$ for some function f.

In general, $t(w)$ need not be expressible as a function of the length of w.

Worst-case complexity: consider the longest running time of all inputs of any given length

Average-case complexity: consider the average of all running times for inputs of a given length

In practice, both notions of complexity are important.

We will focus on worst-case (w.c.) complexity.

Then, the (w.c.) running time is a function $f : \mathbb{N} \to \mathbb{N}$.

Big-O Notation

$g(\cdot) = O(f(\cdot))$ iff, for some constant c and some argument n_0 and all $n \geq n_O$, $g(n) \leq c \cdot f(n)$.

Ex.: $f(n) = 2n^3 + 2n + 5$, $g(n) = 10n^2 + 3$.
For $n \leq 4$, $f(n) \leq g(n)$.
For larger n, $g(n) \leq f(n)$.
Hence $g(\cdot) = O(f(\cdot))$.

Models are polynomially related

We considered variants of *deterministic* TMs:

- multiple tapes
- multiple heads
- multidimensional tape
- RAM-extension

all (also in combination) do not enhance the power of (deterministic) Turing machines; the corresponding simulations "slow down" the computation "only" by a polynomial.

Ex.: There is a linear time 2-head TM deciding L_{pal}.

The class $\mathcal{P}$

A function $f : \mathbb{N} \to \mathbb{N}$ is *polynomially bounded* if there is a pol. p s.t., for all $n \geq 0$, $f(n) \leq p(n)$.

Lemma: f pol. bounded iff $\exists k : f(\cdot) = O(\cdot^k)$.

A TM is called *polynomially bounded* (p.b.) iff its (w.c.) running time is p.b.

A language L is called *polynomially decidable* iff there is a p.b. TM that decides L.

This language class is denoted by $\mathcal{P}$.

Theorem: CFL $\subset \mathcal{P}$.
CYK and $\{w\$w \mid w \in \{a, b\}^*\}$.

Theorem: $\mathcal{P}$ is closed under complement.

Codings

What is the "size" $|n|$ of a number n?

This depends on whether n is coded
in unary (then, $|n| = O(n)$) or
in binary (then, $|n| = O(\log(n))$).

Ex.: PRIMES= $\{\langle n \rangle \mid n$ is prime $\}$ is trivially in $\mathcal{P}$ if unary encodings are used, but have only recently shown to belong to $\mathcal{P}$ if binary encodings are used.

Quantative CTT

Every problem (language) that is practically feasibly computable is in $\mathcal{P}$ and vice versa.

PROS: Most "interesting" problems in $\mathcal{P}$ have running times bounded by low-degree polynomials involving small constants.

CONS:

- Seymour Robertson type cubic algorithms
- Approximation schemes
- parameterized algorithms
- "efficient" exponential algorithms

The Polynomial Halting Problem

HALT $_P = \{\langle M\rangle \,\$\, \langle w\rangle \mid$ TM M halts on w and outputs 1 in at most $2^{|w|}$ steps $\}$

Thm. 1 HALT$_P$ is decidable but not in $\mathcal{P}$.

Proof: Recursivity should be clear. we show that HALT$_P$ is not in $\mathcal{P}$. Otherwise,

$$H_0 = \{\langle M\rangle \mid \langle M\rangle \,\$\, \langle M\rangle \in \text{ HALT}_P\}$$

were in $\mathcal{P}$. Hence, $\overline{H}_0$ would be in $\mathcal{P}$.
Let M_0 be a TM deciding $\overline{H}_0$ in pol. time p with $p(x) \leq 2^x$ for all $x \geq |\langle M_0\rangle|$.

If M_0 accepts $\langle M_0\rangle$, then $\langle M_0\rangle \in L(M_0) = \overline{H}_0$, contradicting the assumption that H_0 contains all TM's that run at most $2^{|M_0|}$ steps when fed with their own description.
Hence, M_0 does not accept $\langle M_0\rangle$. Hence, $\langle M_0\rangle \notin L(M_0) = \overline{H}_0$, i.e., $\langle M_0\rangle \in H_0$, which, by definition of H_0, should mean that M_0 runs at most $2^{|M_0|}$ steps on $\langle M_0\rangle$, contradiction!

Hence, H_0 (and hence HALT$_P$) are not in $\mathcal{P}$.

Anything else beyond $\mathcal{P}$?

TSP: Travelling Salesman Problem

Given an edge-weighted undirected graph $G = (V, E)$ and a vertex v_0 in G, find a *tour* of minimum weight (cost), i.e., a path of length $n = |V|$ that visits every vertex once, returning finally to v_0.

Interpretation: vertices are cities and weights are travel costs.

Simple algorithm: consider all possible paths

$$v_0 = v_{i_0}, v_{i_1}, \ldots, v_{i_n} = v_0$$

Unfortunately, there are $(n-1)!$ many paths. Hence, this algorithm is *not* p.b.

OPEN PROBLEM: Is TSP $\in \mathcal{P}$?

This formulation is still ill-defined, since TSP is not a "language". We have first to convert the optimization problem into a decision problem:

TSP: Given an undirected graph $G = (V, E)$ and a weight-matrix $W = (w_{ij})$, as well as a bound B, is there a tour

$$v_{i_0}, v_{i_1}, \ldots, v_{i_n} = v_{i_0}$$

with total weight

$$\sum_{j=1}^{n} w_{i_{j-1}, i_j} \leq B \quad ?$$

Taking some natural coding of (the adjacency matrix of) G, of W and of B, TSP "becomes" a language:

TS $= \{\langle G \rangle \,\$\, \langle W \rangle \,\$\, \langle B \rangle \mid$ there is a tour in G with total weight not exceeding $B\}$

HCP: Hamiltonian cycle problem

is a (seemingly) simplified version of TSP:

Given a graph G, does G have a tour (which is called *Hamiltonian cycle* in this case)?

HC $= \{\langle G \rangle \mid G$ has a Hamitonian cycle $\}$

Ex.:

No essentially better algorithm that the obvious $O(n!)$ (where n is the number of vertices) is known.

The Clique Problem

A *clique* in a graph $G = (V, E)$ is a subset $V' \subseteq V$ such that

$$\forall v_1, v_2 \in V' : v_1 \neq v_2 \Rightarrow \{v_1, v_2\} \in E$$

The Clique Problem is a combinatorial optimization problem: find some clique of maximal size in a given graph G.

The corresponding language is:

$C = \{\langle G \rangle \,\$\, \langle k \rangle \mid G \text{ has a clique of size } k\}$

The obvious algorithm would try all k-subsets of V, hence having a running time of about n choose k.

The class $\mathcal{NP}$

Above, we left out one extension of TMs: NONdeterminism.

A nondeterministic TM is *polynomially bounded* if there is a polynomial p s.t., for any input string w, *at least one computation* of M on w halts in at most $p(|w|)$ steps.

$\mathcal{NP}$ is the class of languages decidable by p.b. nondeterministic TMs.

The awkward part: nonaccepting computations.

Guess and check

How to design p.b. nondeterministic algorithms?

Ex.: HC:

1. GUESS a sequence T of n vertices of graph G
   (in n nondeterministic steps)

2. CHECK that T is a Hamiltonian cycle
   (using a deterministic pol.time algorithm)

Ex.: Clique:

1. GUESS a sequence C of k vertices of graph G (k nondeterministic steps)

2. CHECK that C is a clique (in $\mathcal{P}$-time)

Hence, both HC and C are in $\mathcal{NP}$.

$\mathcal{P}$ vs. $\mathcal{NP}$

Philosphical question:
Is verifying essentially easier than finding a solution?

Common sense and experience says YES.

Most computer scientists "believe" that $\mathcal{P} \neq \mathcal{NP}$.

BUT: There is no proof for this (yet).

Basic question: Can a p.b. NONdeterministic TM be simulated by a p.b. deterministic TM?

A related (math.) OPEN question:
Is $\mathcal{NP}$ closed under complement?

Languages in $\mathcal{NP}$

We have seen that C, HC and TSP are in $\mathcal{NP}$.

Were we just too dull to find a p.b. deterministic algorithm?

What would $C \in \mathcal{P}$ mean for the basic question of the previous slide?

In what follows, we will show that $C \in \mathcal{P}$ would imply that $\mathcal{P} = \mathcal{NP}$ (similarly for the other problems).

Hence, this case is *very unlikely*.

Main tool for this sort of proofs: *reduction*.