Computational Complexity. Lecture 1

P vs. NP, Deterministic Hierarchy Theorem

Alexandra Kolla

Today

- Define NP, state P vs. NP problem
- Search problems/decision problems
- Diagonalization
- Time hierarchy theorem (only known theorem which allows to show that certain problems are not in P)

- http://www.win.tue.nl/~gwoegi/P-versus-NP.htm

Computational problems

- In a computational problem:
  - We are given an input encoded over the alphabet {0,1}.
  - We want to return as output a solution satisfying some property.
  - Computational problem is then defined by the property that the output has to satisfy given the input.
- 4 natural types of problems: decision, search, optimization, counting.

Decision problems

- In a decision problem:
  - We are given an input $x \in \{0,1\}^*$
  - We are required to return a YES/NO answer (verify whether input satisfies property)
- E.g is an undirected graph 3 colorable?
- Specify decision problems with set of inputs $L \subseteq \{0,1\}^*$ for which the answer is YES (language)

Search problems

- In a search problem:
  - We are given an input $x \in \{0,1\}^*$
  - We are required to compute some answer $y \in \{0,1\}^*$ that is in some relation to x, if such y exists
- Search problems specified with relations $R \subseteq \{0,1\}^* \times \{0,1\}^*$, where (x,y) $\in$R iff y is an admissible answer given x
- For graph 3 coloring, we would want the coloring as output if it exists (more demanding). Formally relation R_{3COL} contains pairs (G,c) $\in$ where G is 3-colorable and c is a valid 3-coloring

P and NP

- We study asymptotic complexity of problems
- Is there "feasible" algorithm for 3 coloring?
- "feasible algorithm" = runs in poly time
- P is class of decision problems solvable in poly time
- Easier to verify than come up with solution…

P and NP

- P is class of decision problems solvable in poly time.
- Search problem defined by a relation R is an NP search problem if there is a poly time algorithm that given x and y decides whether $(x,y) \in R$, and if there is a polynomial p s.t. if $(x,y) \in R$, then $|y| \leq p(|x|)$.
- Captures the fact that relation is efficiently computable and solutions (if they exist) are short.

P and NP

- Decision problem L is in NP if
  - (Definition 1) there is some NP relation R such that $x \in L$ iff there is y s.t. $(x,y) \in R$
  - (Definition 2) there is a poly time algorithm V(.,.) and a polynomial p s.t. $x \in L$ iff there is a y, $|y| \leq p(x)$ s.t. V(x,y) accepts.

- NP= class of NP decision problems.

P and NP

- **Theorem 1**. NP is the set of decision problems that are solvable in poly time by a non-deterministic Turing machine.

P and NP

- NP as a complexity class is defined as class of decision problems: easier to develop cleaner theory, complexity of decision problems completely characterizes complexity of search problems.

- **Theorem 2**. For every NP search problem there is an NP decision problem such that if the decision problem is solvable in time t(n) then the search problem is solvable in time $O(n^{O(1)} \cdot t(n^{O(1)}))$. In particular, P=NP iff every NP search problem is solvable in poly time.

P and NP

- For function t: $\mathrm{N} \to N$, we define
  - DTIME(t(n)) the set of decision problems that are solvable by a deterministic Turing machine within time t(n) on inputs of length n
  - NTIME(t(n)) the set of decision problems that are solvable by a non-deterministic Turing machine within time t(n) on inputs of length n
- **P**= $\bigcup_k DTIME(O(n^k))$
- **NP**= $\bigcup_k NTIME(O(n^k))$

Diagonalization

- Only known way of proving separations between complexity classes
- Similar to Cantor
- Halting Problem is undecidable

Diagonalization

- **Definition**. (Time constructible functions) A function t:$\mathbb{N}\rightarrow\mathbb{N}$ is time constructible if there is algorithm that, on input n, computes the value t(n) in time O(t(n))

- Eg. All polynomials, all combinations of exponential, polynomial and root functions are time constructible
- Not time constructible: t(n) =n^2 if the number n written in binary encodes a turing machine that halts on all inputs and t(n)=n^2+1 otherwise

Time hierarchy theorem

- **Theorem (simple version)**. For every time-constructible function t(.), there is a language L such that every algorithm that decides L must run in time >t(n) on infinitely many inputs, but L∈DTIME($t^{O(1)}(n)$).

Time hierarchy theorem

- **Theorem (alternative formulation)**. For every f time-constructible function with f(n)logf(n)=o(g(n))

$$\mathrm{DTIME}(f(n)) \subsetneq \mathrm{DTIME}(g(n))$$

E and EXP

- E=DTIME($2^{O(n)}$)
- EXP=U_kDTIME($2^{O(n^k)}$)

Corollary. P≠E and E ≠EXP

Ladner's theorem and NP-intermediate problems

- **Theorem**. If P≠NP then there exists a language L∈ NP\P that is not NP-complete.