Complexity Theory: SV 1/3.

Algorithms and problems, Time and space, Time complexity, Non-determinism, Reductions, NP-completeness

1 Instructions

- **Please submit your work at most 48 hours before your supervision to my cam email.**
- I'd appreciate it if you could typeset your work, but I'll accept legible scans of handwriting.
- Please make it clear which question you're writing a solution to, by referring to the numbering scheme of this sheet.
- I don't expect you to spend more than 3–4 hours of focussed work on each supervision's worth of work.
- The questions are not ordered by difficulty. If you're stuck on a question, feel free to ask me for a hint or wait till the supervision to discuss.
- Please never paste in answers that you don't understand – that defeats the purpose. It's not an issue if you leave an answer empty when you couldn't solve a problem.

2 Short questions

1. What is the difference between the complexity of a problem and an algorithm?
2. Sort in order of magnitude: $\Theta(2^n), \Theta(n^n), \Theta(n^2 2^n), \Theta(n!)$ (and justify).
3. How do we make the notion of an Algorithm precise in this course and why does it seem to be ok to choose that representation in particular and not some other?
4. In words, what is a configuration of a DTM (deterministic Turing Machine) and what is a computation?
5. Is it important to differentiate between two DTMs with the same language for the purposes of complexity theory?
6. What's a recursively enumerable language and how many are there?
7. In chess it is definitely true that either white player can force a win, black player can force a win, or neither can and the game shall end in a draw if both players play optimally.
   (a) Consider the problem of deciding which of the three is the case (humans don't know yet). What time complexity would you say that problem has?
   (b) If that problem is solved, how do you expect it will influence chess as a human sport? (*This is an open-ended question, at most vaguely related to complexity theory, but I'd like to see how you think. Don't spend too much time on this.*)
8. Briefly explain the difference between the Church-Turing thesis and the strong Church-Turing thesis.
9. What's the difference in the definitions of a DTM and an NDTM (non-determinstic Turing Machine)? Answer both with mathematical notation and a one/two-sentence-long intuitive explanation.
10. Recall $\mathsf{P} := \bigcup_{k=1}^{\infty} \mathrm{TIME}(n^k)$.

(a) Would P be changed if in this definition we swapped the $k = 1$ for $k = 5$? Why or why not?

(b) The upper limit of that union $\bigcup$ is ∞ – does this mean that we're allowing undecidable problems into P?

11. If $f_i(n)$ for $i \in \mathbb{Z}$ are all functions in $\bigcup_{k=1}^{\infty} O(n^k)$, is $g(n)$ also in $\bigcup_{k=1}^{\infty} O(n^k)$, where...

(a) $g(n) = \max_{i=1}^{\infty} f_i(n)$?

(b) $g(n) = f_1(n) + f_2(n)$?

(c) $g(n) = \sum_{i=1}^{\infty} f_i(n)$?

(d) $g(n) = \sum_{i=1}^{n} f_i(n)$?

(e) $g(n) = f_1(n) \cdot f_2(n)$?

(f) $g(n) = \prod_{i=1}^{n} f_i(n)$?

(g) $g(n) = f_1(f_2(n))$?

12. Explain briefly why, in the industry, instances of some problem that's **not in** P might be commonly solved on a daily basis, *whereas* there could be a problem **in** P which they would give up on solving?

13. If a boolean expression ϕ is (a) satisfiable / (b) unsatisfiable, what can you say about $\neg\phi$?

14. State what it means that an NDTM accepts a string in polynomial time (it's ok to not be super rigorous here).

15. What are polynomially-verifiable languages?

16. Explain briefly, in your own words, why NP is the same as the set of all polynomially-verifiable languages.

17. Define rigorously what a reduction from L to L' is. Explain how this concept relates to relative difficulty of the two corresponding problems. In particular, mention why the notation $L \leq L'$ makes sense.

18. Are there NP-hard languages that are not NP-complete? (Feel free to give an intuition only for now, you should be able to prove it by the end of the course.)

19. The proof of NP-completeness of 3SAT is tedious. Explain why reductions allow us to have cleaner / higher-level proofs of NP-completeness for other problems, that do not have to go into details of Turing Machines.

20. Let L be some arbitrary problem in P. Sketch a poly-time reduction **from** L **to** 3SAT.

21. Does it make sense to talk about P-completeness in the sense of polynomial time reductions?[1]

[1] Complexity theorists have studied other types of P-completeness though, e.g. w.r.t. logarithmic-space reductions. This is outside the scope of your course.