Computability and Computational Complexity, A.Y. 2018–2019

Written exam

Mauro Brunato

Friday, May 3, 2019

Exercise 1

1.1) Put the following complexity classes in the correct order of inclusion:

P **L** **ZPP** **NP**

1.2) Provide (possibly succint) proofs of the inclusions.

Exercise 2

An examiner must plan an oral exam for N students, where every student is asked one, and one only, question.
The examiner has the following information:

- a list of pairs of students who know each other (suppose that the relation is symmetric, but not transitive), and
- a number, k, of questions that she can ask.

We must determine whether the number of questions, k, is sufficient to avoid that two students knowing each other are asked the same question.

2.1) Describe a polynomial-time algorithm that decides the decision problem defined above when $k = 2$.

2.2) Prove that the problem is **NP**-complete in the general case (you can assume the **NP**-completeness of a language if it has been discussed in class).

2.3) Prove that if there were a polynomial-time algorithm for the decision problem in the general case, then the function problem of determining the minimum number k^* of questions could be answered in polynomial time too.

Exercise 3

For each of the following properties of a Turing machine $\mathcal{M}$, prove whether it is computable or not. If applicable, invoke Rice's theorem (and in particular clarify why it is applicable).

3.1) $\mathcal{M}$ accepts all inputs x.

3.2) $\mathcal{M}$ has an even number of states.

3.3) $\mathcal{M}$ never writes a blank on the tape for any input x.

3.4) $\mathcal{M}$ never halts in less than 100 steps for any input x.

Exercise 4

Prove that a 2-symbol Turing Machine has the same computational power of any other finite-alphabet TM.

Solution traces

Exercise 1

1.1 L $\subseteq$ P $\subseteq$ ZPP $\subseteq$ NP.

1.2 See the lecture notes.

Observations

- **L** is the set of languages that are decidable in logarithmic *space*, not time: logarithmic time makes little sense in TMs.

Exercise 2

The problem is equivalent to k-VERTEX COLORING, where students are vertices, edges are pairs of students who know each other, colors are questions.

2.1 Any polynomial solution for 2-coloring (or, equivalently, to verify if a graph is bipartite) is fine. For every connected component, start by assigning the first color to an arbitrary node; pick any node that has already been colored, and give the opposite color to its neighbors; if this is impossible (a neighbor already has the same color), halt and reject. Whenever all nodes are colored, accept.

2.2 A polynomially verifiable certificate could be, for instance, a question assignment to students. Reduction from k-VERTEX COLORING: for every vertex, let there be a student; for every edge, let the two correspoonding students know each other. Let there be k questions. There is a color assignment if and only if there is a question assignment.

2.3 If an answer for a given value of k could be obtained in polynomial time, then an iterated application of the algorithm to increasing values of k would find the minimum admissible value of k in at most N steps (a number which is linear to the problem size, since the student acquaintance list has an $O(N)$ number of entries).

Observations

- Note that the first point asked for an algorithm (in any form, even a verbal description). Therefore, simply answering “2-coloring is **P**” wouldn’t earn full marks.
- Other reductions are possible, of course, provided that the answer is motivated.
- About the third point: since VERTEX COLORING is **NP**-complete, having a polynomial algorithm for it would cause **NP** $=$ **P**. However, the problem of finding the minimum k is *not* a decision problem.

Exercise 3

3.1 The property is semantic (it is defined wrt the accepted language alone) and not trivial (some machines accept all inputs, others don't), therefore it satisfies the hypotheses of Rice's theorem. The property is uncomputable.

3.2 Given a TM representation, it is always possible to count its states and decide if they are even or odd.

3.3 We can reduce a version of the halting problem to that property. Let $\mathcal{M}$ be any TM. Transform it into a machine $\mathcal{M}'$ in the following way:

- $\mathcal{M}'$ has a second blank symbol, e.g., $\hat{␣}$;
- $\mathcal{M}'$ has the same states and transition rules as $\mathcal{M}$, with the following additions:
  - the rules that apply to ␣ also apply to $\hat{␣}$;
  - whenever $\mathcal{M}$ writes ␣, cM' writes $\hat{␣}$;
  - all rules that lead to a halting state in $\mathcal{M}$ are replaced to rules that lead to a new state $s_{␣}$ in $\mathcal{M}'$;
  - whenever $\mathcal{M}'$ reaches state $s_{␣}$, it writes a true blank ␣, then halts.

By construction, it is clear that $\mathcal{M}(x)$ eventually halts if and only if $\mathcal{M}'(x)$ eventually writes a blank, therefore $\mathcal{M}$ halts for all inputs if and only if $\mathcal{M}'$ has the stated property.

3.4 In order to verify if a machine $\mathcal{M}$ has the property, we need to emulate the first 100 steps of its execution on all possible inputs. Note that, while the set of possible inputs is unbounded, only symbols within 100 cells from the starting position are relevant. Therefore, we just need to simulate $\mathcal{M}$ for a limited number of steps (100) on a finite number of inputs (all 200-symbol strings), and we always terminate with an answer, hence the property is computable.

Observations

- The first property can be directly proved to be uncomputable by observing that in order to accept x a machine must first terminate.
- The "for any input" part has been often overlooked, particularly in reference to the fourth property.
- Note that the only semantic property is the first one.

Exercise 4

See the lecture notes.

Observations

- After we decide to represent n symbols with $\lceil \log_2 n \rceil$ bit strings, the main problem — which has been overlooked in most answers— is for the TM to "remember" partially read encodings by storing them in its states. This observation was key to obtaining full marks.