Computability and Computational Complexity, A.Y. 2024–2025

Guide to the answers

Thursday, January 9, 2025

Exercise 1

1.1) Define the space complexity class **L**.

1.2) Consider the language of all strings in $\{0,1\}^*$where the number of 1's is strictly larger than the number of zeroes, e.g.:

$011 \in L$, $1101 \in L$,
$10 \notin L$ (the number of 1's must be *strictly* larger),
$1 \in L$, $11111111 \in L$,
$0 \notin L$, $101010 \notin L$.

Prove that $L \in \mathbf{L}$.

Hint — *For 1.2: start by sketching down a (pseudocode) program that decides L.*

Solution 1

1.1) See the lecture notes. Some clarity in defining what "space" is in the definition's context is appreciated.

1.2) An algorithm to check if a string $s \in \{0,1\}^*$ belongs to L just needs to scan s and maintain two counters on the working tape, one for the number of 0's and one for the number of 1's. At the end, we compare the two counters and decide.
As an alternative, the machine can just maintain one signed counter, increase it when it scans a 1 in the input string and decrease it upon scanning a 0; at the end, the machine accepts if the counter is positive:

```
on input $s \in \{0,1\}^*$
    counter_0 ← 0
    counter_1 ← 0
    for $c$ in $s$
        if $c$ = 0
            counter_0 ← counter_0 + 1
        else if $c$ = 1
            counter_1 ← counter_1 + 1
    if counter_1 > counter_0
        accept
    reject
```

```
on input $s \in \{0,1\}^*$
    counter ← 0
    for $c$ in $s$
        if $c$ = 0
            counter ← counter - 1
        else if $c$ = 1
            counter ← counter + 1
    if counter > 0
        accept
    reject
```

In both cases, if the input is n symbols long the counter values will never exceed $\pm n$, therefore they will occupy at most $O(\log n)$ symbols on the working tape.

Observations

- It's fine to say that a machine requires "additional logarithmic space," but we need to clarify that the input is read-only: if the machine were able to reuse the input cells, then it would be using *linear* space.

Exercise 2

2.1) Provide a precise definition of *semantic* and *trivial* in reference to properties of Turing Machines.

2.2) State Rice's Theorem.

2.3) Why does the proof of Rice's Theorem fail if the property is *not* semantic (e.g., "the TM has more than 100 states")?

Solution 2

2.1) / 2.2) See the lecture notes.

2.3) In short, the proof is all about language recognition: it starts with a TM and builds a new machine that either never halts or recognizes the same language as the original one, and from that we deduce whether they have the property or not.
More formally, given a property $\mathcal{P}$, the proof of Rice's Theorem relies on the existence of a TM $\mathcal{N}$ that has the property $\mathcal{P}$. Then it embeds $\mathcal{N}$ in a larger TM prefixed by a computation $\mathcal{M}(x)$ that might or might not halt, so that the full machine either recognizes the same language as $\mathcal{N}$ (if $\mathcal{M}(x)$ halts) or the empty language (if $\mathcal{M}(x)$ doesn't halt). Therefore, we are making the assumption that the ability to recognize the same language as $\mathcal{N}$ implies the property $\mathcal{P}$, which is precisely the semanticity assumption.
As an example, consider the non-semantic property $\mathcal{P} = \{\mathcal{M} : \mathcal{M} \text{ has more than 100 states}\}$. Let $\mathcal{N} \in \mathcal{P}$. When we use $\mathcal{N}$ inside a larger machine $\mathcal{N}'$, the latter will have even more states, therefore $\mathcal{N}' \in \mathcal{P}$ no matter what, therefore $\mathcal{N}$ having the property does not tell us anything about whether $\mathcal{M}(x)$ halts or not.

Observations

- Beware of quantifiers! One of the following formulas tells us that $\mathcal{P}$ is trivial, the other is just trivial itself (true for every property):

$$\forall \mathcal{M} \left(\mathcal{P}(\mathcal{M}) = \text{true} \lor \mathcal{P}(\mathcal{M}) = \text{false}\right)$$

$$\left(\forall \mathcal{M}\, \mathcal{P}(\mathcal{M}) = \text{true}\right) \lor \left(\forall \mathcal{M}\, \mathcal{P}(\mathcal{M}) = \text{false}\right)$$

Exercise 3

In the central hall of the ancient temple, having dodged an inordinate amount of boobytraps, Indiana Jones is ready to grab the golden statue from its pedestal; however, to avoid triggering even more deadly traps on his way back, he must replace the statue with something having its *exact* weight. Alas, his bag of sand was ripped by an arrow; all he has got is a large and heterogeneous set of archaeologist's tools whose individual weights he had luckily annotated in his notebook before leaving his office. Of course, his experienced look can precisely estimate the statue's weight he is so eager to match.

3.1) Show that, given the high precision of the ancient mechanism and the diversity of tools in his belt, he might need a very long[a] time before being able to determine if there is a combination of tools whose weight matches the statue's.

3.2) An oracle is quietly sitting in a corner of the hall. He was gifted with a very peculiar ability: when presented with any intricate map of rooms connected by tunnels, the oracle is immediately able to point out a round trip that visits all rooms exactly once, provided that such path exists. Prove that Indiana Jones could exploit the oracle's ability in order to solve the problem of matching the statue's weight in a reasonable[b] time.

Hint — *Assume that all weights are known with the precision of one gram. Both questions require answers in the form of reductions from/to known difficult (i.e., **NP**-complete) problems. Here is a list of languages that we already know to be **NP**-complete:* SATISFIABILITY, 3-SATISFIABILITY, CLIQUE, INDEPENDENT SET, INTEGER LINEAR PROGRAMMING, VERTEX COVER, 3-VERTEX COLORING, *SUBSET SUM*, KNAPSACK, HAMILTONIAN PATH, DIRECTED HAMILTONIAN CYCLE, *HAMILTONIAN CYCLE*, TRAVELING SALESMAN PROBLEM.

[a]I.e., exponential with respect to the number of tools in the worst case.

[b]I.e., polynomial with respect to the number of tools in the worst case.

Solution 3

3.1) The exercise defines a decision problem (let's call it "Indiana Jones' Weight Matching Problem," or IJWMP). As stated, we can assume that all weights are integer (e.g., expressed in grams). IJWMP is clearly equivalent to SUBSET SUM, therefore it is **NP**-complete. As such, it is unlikely that Indiana Jones will be able to find a statue-matching subset of tools anytime soon.

More formally, it is clear that IJWMP $\in$ **NP** because a solution certificate would consist of a list of tools, and we could check that the sum of their weights matches the statue's weight in polynomial time.

In order to show that IJWMP is complete, we can take a known **NP**-complete problem, SUBSET SUM, and show that it can be polynomially reduced to IJWMP. Given n integers $x_1, \dots, x_n$ and a target sum S, we just need to reformulate the problem as a set of n tools having weights $x_1, \dots, x_n$ and a statue with weight S to be matched. Therefore, IJWMP is **NP**-complete.

If IJWMP were solvable in polynomial time, then we could solve SUBSET SUM in polynomial time too, and hence any other problem in **NP**.

3.2) The problem that the oracle is able to solve in polynomial time (let's call it "Oracle's Dungeon Round Trip Problem," or ODRTP) is clearly a rephrasing of HAMILTONIAN CYCLE, which is **NP**-complete. Therefore, since we know that $IJWMP \in$ **NP**, we know that there is a polynomial reduction from IJWMP to ODRTP: Indiana Jones will be able to rephrase (in polynomial time) his weight-matching problem as a dungeon round-trip problem, and the oracle's answer will tell him if his original problem has a solution.

Formally, we prove that ODRTP $\in$ **NP** because if a round trip exists the oracle can show it to us and we can easily check in polynomial time (wrt the dungeon's size in terms of number of rooms and tunnels) that it solves the problem.
To show that ODRTP is complete, let us take any instance $G = (V, E)$ of HAMILTONIAN CYCLE (which we know to be **NP**-complete) and create a map with one room for every vertex in V and a tunnel for every edge in E, connecting the corresponding rooms. The map is an instance of ODRTP, and a round trip corresponds to a Hamiltonian path in G.
Since we know that IJRTP is in **NP** and that ODRTP is **NP**-complete, then we know that Indiana Jones can reduce his IJWMP instance to an instance of ODRTP in polynomial time, submit it to the oracle, and get an answer that is positive if and only if his problem has a solution.
Such reduction is probably far from trivial, but we could work out a chain of reductions as follows:

- Indiana Jones writes down a non-deterministic Turing Machine $\mathcal{N}$ that solves his IJWMP instance in non-deterministic polynomial time;
- by following Cook-Levin's steps, he polynomially reduces his machine $\mathcal{N}$ to a 3CNF formula F that is satisfiable if and only if his original problem has a solution;
- He reduces F to an instance G_1 of HAMILTONIAN PATH…
- …then G_1 to an instance G_2 of DIRECTED HAMILTONIAN CYCLE…
- …then G_2 to an instance G_3 of HAMILTONIAN CYCLE…
- …and finally he can rewrite G_3 as a map that he submits to the oracle as an instance of ODRTP.

Each step is guaranteed to take at most polynomial time wrt to the original instance's size, and to produce an instance of polynomial size.

Observations

- The original text did not clarify that the oracle gives an immediate answer, but students were told that the oracle's answer comes in polynomial time (which is equivalent for the purposes of the exercise).
- "Alas" is not the name of Indiana Jones' sandbag — it's an interjection to express sorrow.
- The direction of the reductions is fundamental. In point 3.1, we need to show that IJWMP is at least as hard as any other **NP**-complete problem; therefore, we must take a *known* complete problem and show that it can be reduced to IJWMP:

  $$\text{SUBSET SUM} \le_p \text{IJWMP}.$$

  In point 3.2, we want the opposite: we have an instance of IJWMP, and we need to reduce it to a problem for which we have an efficient solver (the oracle):

  $$\text{IJWMP} \le_p \text{ODRTP},$$

  possibly through a longer chain of reductions:

  $$\begin{aligned}\text{IJWMP} &\le_p \text{3-SAT} \le_p \text{HAMILTONIAN PATH} \le_p \\ &\le_p \text{DIRECTED HAMILTONIAN CYCLE} \le_p \text{HAMILTONIAN CYCLE} \le_p \\ &\le_p \text{ODRTP}.\end{aligned}$$