Problem Set 9

In this final problem set of the quarter, you will explore the limits of what can be computed efficiently. What problems do we believe are computationally intractable? What do they look like? And are they purely theoretical, or might you might bump into one some day?

As always, please feel free to drop by office hours or send us emails if you have any questions. We'd be happy to help out!

This problem set has 85 possible points. It is weighted at 5% of your total grade (note that this is slightly less than usual).

Good luck, and have fun!

Due Friday, December 6th at 2:15 PM.
No late periods may be used on this problem set.
No late submissions will be accepted.

Problem One: 4-Colorability (24 Points)

If you'll recall, an undirected graph G is called 3-colorable iff there exists a way to color each the nodes in G one of three colors such that no two nodes of the same color are connected by an edge. The following language consists of all graphs that are 3-colorable.

$$\textit{3COLOR} = \{ \langle G \rangle \mid G \text{ is an undirected, 3-colorable graph} \}$$

This language is **NP**-complete. You'll see a proof of this in Wednesday's lecture.

An undirected graph is called 4-colorable iff there is a way to color each of the nodes in G one of four colors so that no two nodes of the same color are connected by an edge. We can formalize the 4-coloring problem as a language as follows:

$$\textit{4COLOR} = \{ \langle G \rangle \mid G \text{ is an undirected, 4-colorable graph} \}$$

Note that every 3-colorable graph is also 4-colorable, but not all 4-colorable graphs are 3-colorable. In other words, 3-colorability is a stricter requirement than 4-colorability. However, it is still the case that *4COLOR* is **NP**-complete, and in this problem you will prove this result.

i. Prove that *4COLOR* $\in$ **NP** by designing a polynomial-time NTM for it. Prove that your NTM has language *4COLOR*, then justify informally why it runs in polynomial time.

ii. Prove that *4COLOR* is **NP**-hard by proving *3COLOR* $\leq_P$ *4COLOR*. That is, show how to take an arbitrary graph G and construct (in polynomial time) a graph G' such that graph G is 3-colorable iff graph G' is 4-colorable.

   For simplicity, you do not need to formally prove that your reduction is correct and runs in polynomial time. Instead, briefly answer each of the following questions about your reduction (two or three sentences apiece should be sufficient):

   1. If the original graph G is 3-colorable, why is your new graph G' 4-colorable?
   2. If your new graph G' is 4-colorable, why is the original graph G 3-colorable?
   3. Why can your reduction be computed in polynomial time?

   (Hint: As a reminder, your reduction can't color any of the nodes in G or G'. Your job is to produce a graph G' where G is 3-colorable iff G' is 4-colorable. Try thinking about introducing a new node and some new edges to G to form G'.)

Problem Two: Resolving P $\stackrel{?}{=}$ NP (32 Points)

This problem explores the question

What would it take to prove whether P = NP?

For each statement below, decide whether the statement would definitely prove **P** = **NP**, definitely prove **P** ≠ **NP**, or would do neither. Write "**P** = **NP**," "**P** ≠ **NP**," or "neither" as your answer to each question – *we will not award any credit if you write "true" or "false,"* since there are three possibilities for each statement. No justification is necessary.

1. There is a **P** language that can be decided in polynomial time.
2. There is an **NP** language that can be decided in polynomial time.
3. There is an **NP-*complete*** language that can be decided in polynomial time.
4. There is an **NP-*hard*** language that can be decided in polynomial time.
5. There is an **NP** language that ***cannot*** be decided in polynomial time.
6. There is an **NP-*complete*** language that ***cannot*** be decided in polynomial time.
7. There is an **NP-*hard*** language that ***cannot*** be decided in polynomial time.
8. There is ***some*** **NP**-complete language that can be decided in $O(2^n)$ time.
9. There are ***no*** **NP**-complete languages that can be decided in $O(2^n)$ time.
10. There is a polynomial-time ***verifier*** for every language in **NP**.
11. There is a polynomial-time ***decider*** for every language in **NP**.
12. There is a language $L \in$ **P** where $L \leq_P$ 3SAT.
13. There is a language $L \in$ **NP** where $L \leq_P$ 3SAT.
14. There is a language $L \in$ **NPC** where $L \leq_P$ 3SAT.
15. There is a language $L \in$ **P** where 3SAT $\leq_P L$.
16. There is a language $L \in$ **P** where 3SAT $\leq_M L$.
17. All languages in **P** are decidable.
18. All languages in **NP** are decidable.
19. There is a polynomial-time algorithm that correctly decides SAT for all strings of length ***at most*** 10^{100}, but that might give incorrect answers for longer strings.
20. There is a polynomial-time algorithm that correctly decides SAT for all strings of length ***at least*** 10^{100}, but that might give incorrect answers for shorter strings.

Problem Three: The Big Picture (24 Points)

We have covered a *lot* of ground in this course throughout our whirlwind tour of computability and complexity theory. This last question surveys what we have covered so far by asking you to see how everything we have covered relates.

Take a minute to review the hierarchy of languages we explored:

$$\mathbf{REG} \subset \mathbf{CFL} \subset \mathbf{P} \subseteq \mathbf{NP} \subset \mathbf{R} \subset \mathbf{RE} \subset \mathbf{ALL}$$

The following questions ask you to provide examples of languages at different spots within this hierarchy. In each case, you should provide an example of a language, but you don't need to formally prove that it has the properties required. Instead, describe a proof technique you could use to show that the language has the required properties. There are many correct answers to these problems, and we'll accept any of them.

i. Give an example of a regular language. How might you prove that it is regular?

ii. Give an example of a context-free language is not regular. How might you prove that it is context-free? How might you prove that it is not regular?

iii. Give an example of a language in **P**. How might you prove it is in **P**?

iv. Give an example of a language in **NP** suspected not to be in **P**. How might you prove that it is in **NP**? Why is it suspected not to be contained in **P**?

v. Give an example of a language in **RE** not contained in **R**. How might you prove that it is **RE**? How might you prove that it is not contained in **R**?

vi. Give an example of a language in co-**RE** not contained in **R**. How might you prove that it is co-**RE**? How might you prove that it is not contained in **R**?

vii. Give an example of a language that is neither **RE** nor co-**RE**. How might you prove it is not contained in **RE**? How might you prove it is not contained in co-**RE**?

Problem Four: Course Feedback (5 Points)

We want this course to be as good as it can be, and we'd really appreciate your feedback on how we're doing. For a free five points, please answer the feedback questions for this problem set, available online at https://docs.google.com/forms/d/1LK7s-XqAB-WEG_9NixJX9e6MafrEJX-HXTEijlX9fxeg/viewform. We'll give you full credit regardless of how you answer, as long as you answer all the provided questions.

Extra Credit Problem: $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$ (Worth an A+, $1,000,000, and a Stanford Ph.D)

Prove or disprove: $\mathbf{P} = \mathbf{NP}$.