CS3102 Theory of Computation
Problem Set 7
Department of Computer Science, University of Virginia

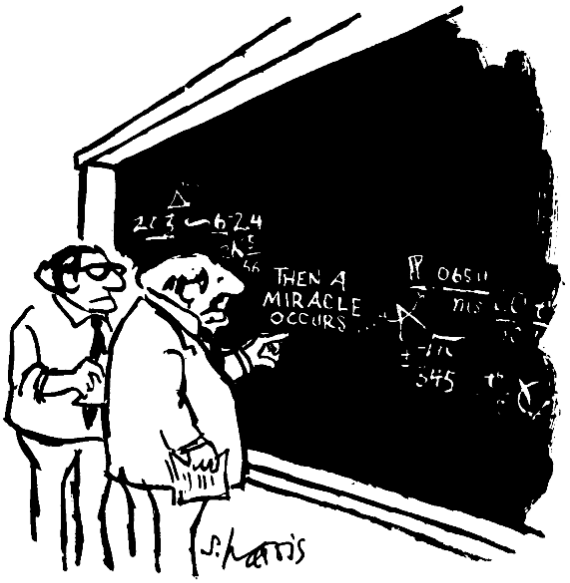
Gabriel Robins

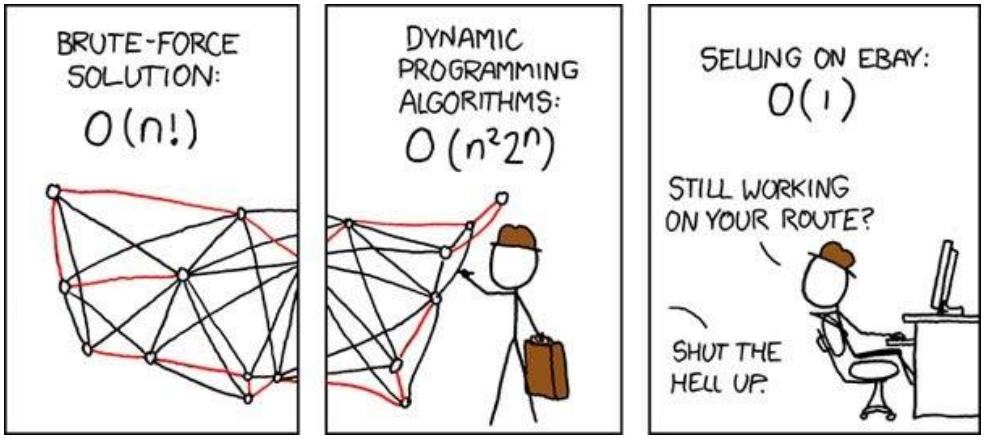
Please start solving these problems immediately, don't procrastinate, and work in study groups. Please prove all your answers; informal arguments are acceptable, but please make them precise / detailed / convincing enough so that they can be easily made rigorous if necessary. To review notation and definitions, please read the "Basic Concepts" summary posted on the class Web site, and also read the corresponding chapters from the Sipser textbook and Polya's "How to Solve It".

Please do not simply copy answers that you do not fully understand; on homeworks and on exams we reserve the right to ask you to explain any of your answers verbally in person (and we have exercised this option in the past). Please familiarize yourself with the UVa Honor Code as well as with the course Cheating Policy summarized on page 3 of the Course Syllabus. To fully understand and master the material of this course typically requires an average effort of at least six to ten hours per week, as well as regular meetings with the TAs and attendance of the weekly problem-solving sessions.

This is not a "due homework", but rather a "pool of problems" meant to calibrate the scope and depth of the knowledge / skills in CS theory that you (eventually) need to have for the course exams, becoming a better problem-solver, be able to think more abstractly, and growing into a more effective computer scientist. You don't necessarily have to completely solve every last question in this problem set (although it would be great if you did!). Rather, please solve as many of these problems as you can, and use this problem set as a resource to improve your problem-solving skills, hone your abstract thinking, and to find out what topics you need to further focus on and learn more deeply. Recall that most of the midterm and final exam questions in this course will come from these problem sets, so your best strategy of studying for the exams in this course is to solve (including in study groups) as many of these problems as possible, and the sooner the better!

Advice: Please try to solve the easier problems first (where the meta-problem here is to figure out which are the easier ones ☺). Don't spend too long on any single problem without also attempting (in parallel) to solve other problems as well. This way, solutions to the easier problems (at least easier for you) will reveal themselves much sooner (think about this as a "hedging strategy" or "dovetailing strategy").





"I think you should be more explicit here in step two."

1. The following problems are from [Sipser, Second Edition]:

   Pages 211-214: 5.2, 5.4, 5.6, 5.7, 5.9, 5.10, 5.11, 5.12, 5.13, 5.14, 5.15, 5.16, 5.20, 5.28, 5.29, 5.30, 5.31

   Pages 242-243: 6.23, 6.24

   Page 294-300: 7.6, 7.7, 7.9, 7.10, 7.11, 7.13, 7.17, 7.21, 7.26, 7.36, 7.38, 7.42, 7.44

2. Prove whether given a TM M and string w, each of the following properties is decidable, Turing-recognizable, or not Turing-recognizable:

   a) an arbitrary given string w causes M to enter state 3.

   b) there exists some string that causes M to enter state 3.

   c) an arbitrary given string w causes M to enter each and every one of its states.

   d) a given string w causes M to move its head to the left at least once when M runs on w.

   e) M accepts a finite language.

   f) M accepts a regular language.

   g) M accepts a decidable language.

   h) M accepts a Turing-recognizable language.

   i) M never writes a nonblank symbol on its tape when it runs on a given string w.

   j) M never overwrites a nonblank symbol when it runs on a given string w.

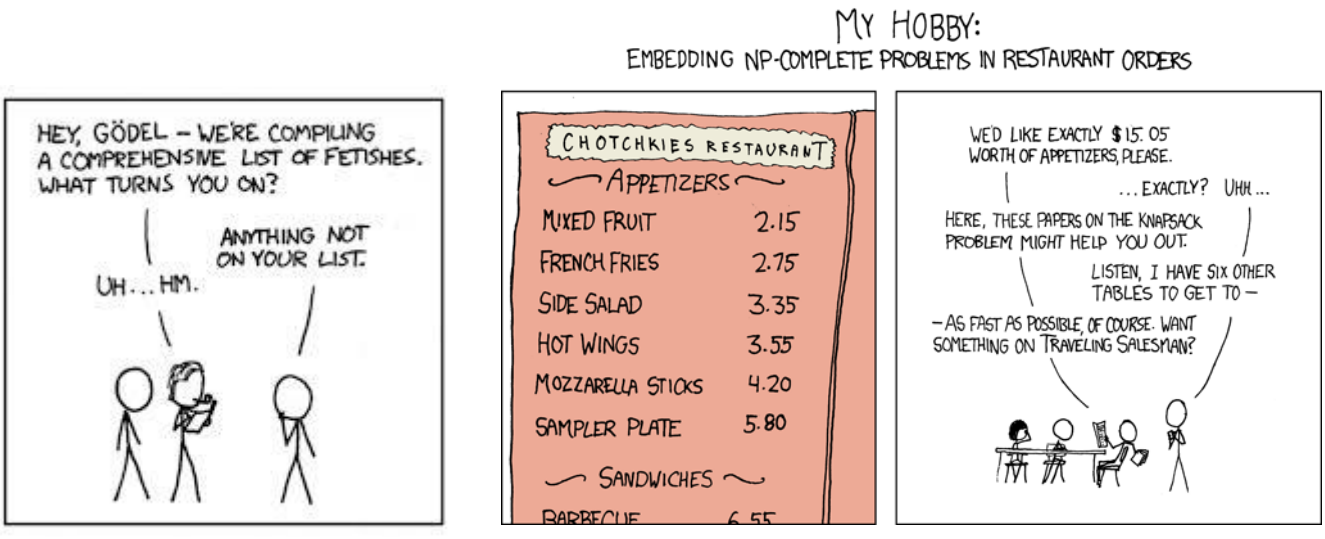
   k) M never overwrites a nonblank symbol when it runs on any string.

   l) M is a universal Turing machine.

3. Are there two non-context-free languages whose concatenation is regular? Are there a countably infinite number of such examples? Are there an uncountable number of such examples?

4. Are there two undecidable languages whose concatenation is regular? Are there a countably infinite number of such examples? Are there an uncountable number of such examples?

5. Are there two non-recognizable languages whose concatenation is regular? Are there a countably infinite number of such examples? Are there an uncountable number of such examples?



6. We define the SHUFFLE of two strings $v,w \in \Sigma^*$ as:

$$\text{SHUFFLE}(v,w) = \{v_1w_1v_2w_2...v_kw_k \mid v=v_1v_2...v_k,\ w=w_1w_2...w_k,\ \text{and for some } k \geq 1,\ v_i,w_i \in \Sigma^*,\ 1 \leq i \leq k\}$$

For example, 212**ab**1**baa**2**b**22 ∈ SHUFFLE(**abbaab**,2121222)

Extend the definition of SHUFFLE to two languages $L_1,L_2 \subseteq \Sigma^*$ as follows:

$$\text{SHUFFLE}(L_1,L_2) = \{w \mid w_1 \in L_1,\ w_2 \in L_2,\ w \in \text{SHUFFLE}(w_1,w_2)\}$$

a) Is the SHUFFLE of two decidable languages necessarily decidable?
b) Is the SHUFFLE of two recognizable languages necessarily recognizable?

7. Given an arbitrary alphabet $\Sigma = \{a_1,a_2,...,a_n\}$, we can impose a total ordering on it in the sense that we can define < so that $a_1 < a_2 < ... < a_n$. We now proceed to define a new operation called the SORT of a string $w = w_1w_2...w_k \in \Sigma^*$ (where $w_i \in \Sigma$ and $k = |w|$) as:

$$\text{SORT}(w) = w_{\sigma(1)}w_{\sigma(2)}...w_{\sigma(k)} \text{ so that } w_{\sigma(i)} < w_{\sigma(i+1)} \text{ for } 1 \leq i \leq k-1$$

and σ is a permutation (i.e., a 1-to-1 onto mapping $\sigma:[1..k] \rightarrow [1..k]$)

For example, SORT(11210010120)=00001111122. Now extend the definition of SORT to languages, so that SORT(L) = {SORT(w) | w∈L}. For each one of the following statements, state whether it is true or false and explain:

a) SORT(Σ^*) is regular.
b) SORT(L) ⊆ L
c) SORT(SORT(L))=SORT(L)
d) SHUFFLE(L_1,L_2)=SHUFFLE(L_2,L_1)
e) SORT(SHUFFLE(L_1,L_2)) = SORT(L_1L_2)
f) ∃ L such that SORT(L)=SHUFFLE(L,L)=L
g) SORT preserves regularity.
h) SORT preserves context-freeness.
i) SORT preserves decidability
j) SORT preserves non-decidability
k) SORT preserves recognizability
l) SORT preserves non-finite-describability

(The definition of SHUFFLE operator is the same as above.)

8. Is NP countable?

9. Is P ∩ NP countably infinite?

10. Is PSPACE countable?

11. Is it decidable whether given a one-state PDA accepts all input strings? How about a three-state PDA?

12. Define the "Busy Beaver" function BB:$\mathbb{N}\rightarrow\mathbb{N}$ as follows: BB(n) is the maximum number of 1's printed on the tape of any Turing machine with n states which halts when running on the blank tape (i.e., with no input). Is BB finitely describable? Is BB computable? How fast does BB grow asymptotically?

13. If we had free access to an oracle that computes the Busy Beaver function for us in constant time, prove either that all functions (mapping naturals to naturals) are computable relative to such an oracle, or else give a counter-example. (Please don't do both. ☺)

14. Two cyborgs walk into your home, both claiming to be oracles for the graph 3-colorability decision problem. They both always give a yes/no answer in constant time for any instance, and are each self-consistent (i.e. each always gives the same answer for the same instance). However, one is a true oracle and the other is a shameless impostor, and you have a large instance of 3-colorability upon which they disagree. Prove whether it is possible to expose the impostor within time polynomial in the size of that instance.

15. Two space aliens walk into your home, both claiming to be oracles for the Boolean Satisfiability (SAT) decision problem. They both always give a yes/no answer in constant time for any SAT instance, and are each self-consistent (i.e. each always gives the same answer for the same SAT instance). However, one is a true oracle and the other is a shameless impostor, and you have a large instance of SAT upon which they disagree. Prove whether it is possible or not to expose the impostor within time polynomial in the size of that SAT instance.

16. True or false: If a rooted binary tree has infinitely many nodes, then it has an infinitely long path from the root.

17. True or false: Most Boolean functions on N inputs have an exponentially long (as a function of N) minimal description (in any fixed reasonable encoding / formalism).

18. True or false: Most Boolean functions on N inputs require an exponential (as a function of N) number of 2-input Boolean gates to implement.

19. True or false: most strings are (losslessly) compressible.

20. Is the set of incompressible strings decidable?

21. Is the set of incompressible strings recognizable?

22. Is NP closed under the Kleene-star operator?

23. Is P closed under the Kleene-star operator?

24. Is integer multiplication is NP-complete? (hint: this is subtle).