Homework 3. Due November 8, 9:59PM.

CS181: Fall 2021

GUIDELINES:

- Upload your assignments to Gradescope by 9:59 PM.
- Follow the instructions mentioned on the course webpage for uploading to Gradescope very carefully (including starting each problem on a new page and matching the pages with the assignments); this makes it easy and smooth for everyone. As the guidelines are simple enough, bad uploads will not be graded.
- You may use results proved in class without proofs as long as you state them clearly.
- Most importantly, make sure you adhere to the policies for academic honesty set out on the course webpage. The policies will be enforced strictly. Homework is a stepping stone for exams; keep in mind that reasonable partial credit will be awarded and trying the problems will help you a lot for the exams.
- Note that we have a **modified grading scheme for this assignment**: A sincere attempt will get you 100% of the credit and a reasonable attempt will get you 50% for each problem. Nevertheless, please attempt the problems honestly and write down the solutions the best way you can - this is really the most helpful way to flex your neurons in preparation for the exam.

1. Show that for any NFA N, there exists a NFA N' which is equivalent to N (i.e., computes the same function as N) but has only **one** accept state. [.5 point]

2. Converting a regular expression to a NFA. In class, we saw a procedure to build a NFA for every regular expression r. Let us understand the 'size' of the NFA created by this procedure as a function of the length of the regular expression r. Here, length of the regular expression refers to the number of characters in the regular expression.

   Suppose the regular expression r had length m. How many states would be present in the final NFA built by the procedure in class as a function of m (use big-Oh notation and avoid computing constants)? Explain your reasoning in a few sentences [.5 points]

   [Hint: Look at each of the steps in the compund case, and see how the number of states can increase.]

3. Design a regular expression for the following language:

$$L = \{x : \ x \text{ has an equal number of 10's as 01's.}\}.$$

For instance, 1001 is in the language as there is exactly one 10 and one 01. So are $101001, 0100010$. But $01, 1000, 001001$ are not. You don't have to prove your expression works but write a few sentences describing why your expression could/does work. [.75 point]

4. Show that the following functions/languages are not regular:

   (a) $L = \{x : x = 0^m 1^n, m > n\}$. For instance, $00011, 001, 0001$ would be in L, but $0100, 0011$ would not be elements of L. [.75 points]

   (b) $F : \{0,1\}^* \to \{0,1\}$ defined by $F(x) = 1$ if and only if $x = 1^{3^i}$ for some $i > 0$. [.75 points]

   (c) $ADD = \{0^m 10^n 10^{m+n} : m, n \geq 1\}$. [.75 points]

   You need to use the pumping lemma appropriately for the above. Your proofs should be at the similar level of detail as the examples from lecture 11; that is, write down the different steps as we did in class.

Additional practice problems - Not to be graded

1. Show that if a language L is regular, then so is $Reverse(L) = \{x : Reverse(x) \in L\}$.

2. Design a regular expression for the following languages:

   - $L = \{x :$ third bit from end of x is a 1$\}$.
   - $L = \{x :$ number of 1's in x is divisble by 5$\}$.
   - $L = \{x :$ x has an odd number of 1's$\}$.
   - $L = \{x :$ x has at least three 1's$\}$.
   - L = All strings except the empty string.
   - $L = \{x :$ x has an even number of 0's or contains exactly two 1's$\}$.

3. Continuing problem (2) above:

   2a How many transitions/edges are present in the final NFA built by the procdure in class as a function of m (use big-Oh notation and avoid computing constants)?

   [Hint: Look at each of the steps in the compund case, and see how the number of edges added changes.]

   2b (Advanced) Do you see a way to modify the procedure to only add at most $O(m)$ transitions in the NFA? Such an NFA is what gets used in grep and other regex matching tools.

   [Hint: The most expensive step in terms of number of edges added is the compound case corresponding to the (*) operation. What happens if you maintain the invariant that all the NFAs you build only have one accept state? We can maintain this invariant by Problem (1)!]