Homework 3

CS 181, Fall 2022

Out: Oct. 14
Due: Oct. 20, 11:59 PM

Please upload your solutions on Gradescope. You can use LaTeX or a word document to write up your answers, but we prefer you use LaTeX. You can use the **tikz package** in LaTeX to draw the DFAs in this homework. You may scan hand-written work or images for parts of solutions **only if** they are extremely clean and legible. Please ensure that your name does not appear anywhere in your handin.

Problem 1: Correctness of Knuth-Morris-Pratt

In class you have learned about the Knuth-Morris-Pratt algorithm for finding a pattern P in a larger text T. Recall that KMP's improvement over the naïve "sliding window" approach lies in the fact that in KMP we use the knowledge gained from earlier comparisons between P and T to avoid many unnecessary comparisons later on. To formalize this idea, we'll make the following definition.

Definition. *For each position k in the pattern P, let $s_k(P)$ denote the length of the longest proper suffix of $P_{1:k}$ that matches a prefix of P. (The notation $P_{1:k}$ is used to represent the first k characters of P.) If the pattern P is clear from context, we will simply write s_k.*

As an example, if $P = \texttt{abcxabcde}$, then $s_2 = s_3 = s_4 = 0$, $s_5 = 1$, $s_6 = 2$, $s_7 = 3$, and $s_8 = 0$.

If a mismatch between the pattern and the text is found at position $k + 1$ of P, then KMP responds by shifting the pattern $k - s_k$ places to the right. To see this rule in action, consider $P = \texttt{abcxabcde}$ and $T = \texttt{xyabcxabcxadcdqfeg}$. Suppose the left end of P is aligned with the third character of T. Then P and T match for 7 characters, but mismatch on the 8th character of P. So P is shifted to the right by $7 - s_7 = 7 - 3 = 4$ characters:

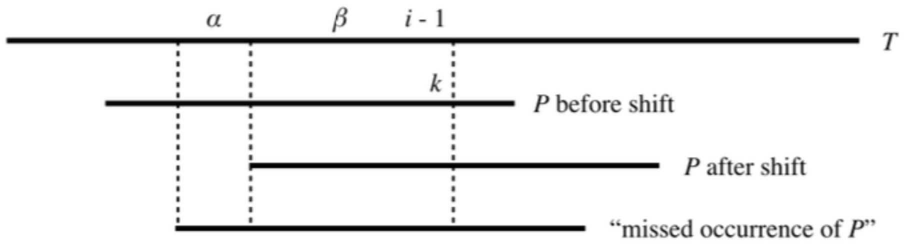
```
xyabcxabcxadcdqfeg
  abcxabcde
      abcxabcde
```

This shifting rule provides two advantages. First, we often shift the pattern by more than a single character, which is an improvement over the naïve algorithm. Second, after the shift is completed, we already know that the first s_k characters in P match their counterparts in T (see the example above). So we can start comparing P to T at position $s_k + 1$ of P, further saving ourselves from doing unnecessary work.

Everything sounds good so far. But hold on! How do we know that the KMP shift rule doesn't move the pattern too far to the right? In other words, how can we be sure that we don't inadvertently skip over the pattern we're looking for in the text? The purpose of this problem is to walk you through a proof of the following theorem, which should put your mind at ease.

Theorem. *For any alignment of P with T, if characters 1 through k of P match the opposing characters of T but character $k+1$ mismatches with T_i, then P can be shifted by $k - s_k$ places to the right without passing any occurrence of P in T.*

Our proof will proceed by contradiction. In other words, we shall assume that there *is* in fact an occurrence of P in T starting strictly to the left of the shifted P and show that this assumption leads to a contradiction. Our proof will be guided by the following picture.



In this diagram,

- α and β are the indicated substrings of T,
- the unshifted pattern P matches T up through position k of P and position $i-1$ and of T,
- $P_{k+1} \neq T_i$.

a. What is the relationship between β and P? What is the length of β? Give justification for both answers.

b. Which portion of the missed occurrence of P matches T? Which portion of the unshifted P matches T? What can you say about which portion matches between the unshifted P and the missed P as a result? Call this matching substring γ.

c. Is a γ a prefix of $P_{1:k}$? a proper prefix? a suffix? a proper suffix? Explain.

d. What can we say about the length of α?

e. Combine your results from (a), (c), and (d) to derive a contradiction and complete the proof.

Problem 2: Failure Function

For each of the following strings, draw the failure function table:

a. `AQUAQUARIUM`

b. `ABRACADABRA`

Problem 3: Finite Automata for Pattern Matching

Draw a deterministic finite automaton that accepts all strings containing the pattern `TTATAG`. Then, illustrate the computational history (i.e. the sequence of states that are visited) when the machine is run on the text `TTTTTATATTATAGTTA`. Assume that the finite automaton reads the whole string (i.e. there is no early stopping).

Problem 4: More DFAs!

Our aquarium rescue team rescued a juvenile grey seal on a distant seashore. Unfortunately, during their way back, the rescue ship gets stranded on an island, a CG island to be exact. A CG island is a short stretch of DNA in which the frequency of the dinucleotide CG sequence is higher than in other regions. CG islands are also referred to as CpG islands, where the ”p” simply indicates that ”C” and ”G” are connected by a phosphodiester bond. Biologically speaking, CpG islands are often located around the promoters of housekeeping genes (which are essential for general cell functions) or other genes frequently expressed in a cell. As luck would have it, Sorin visits the team and offers to take them back to the aquarium on the condition that he constructs a DFA that recognizes sequences that could come from a CpG island. Can you help the team back and get the grey seal get treatment it needed?

Consider the following nucleotide sequences:

```
CGCGAT
CTCGCGT
CTCGCGA
```

Construct a DFA that will accept any input that containing at least one of the patterns above. Once any of the patterns is found, the DFA should reach an absorbing acceptance state.

Note that there are many valid DFAs for this question. You should construct the simplest and most specific one possible; in other words, your DFA should include the smallest number of states possible, and it should not accept any strings that do not contain at least one of the patterns above. Remember that beauty and bioinformatics are inseparable!

Problem 5: Languages

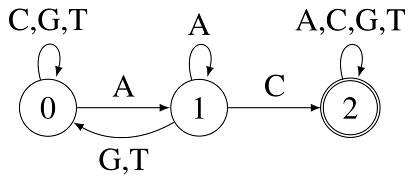
In the previous two problems, we have focused on building DFAs to recognize whether a pattern p appears anywhere in a text t. As a result, the DFAs constructed in these problems will accept languages of the form $L = \{\text{strings containing pattern } p\}$. However, what if we wanted to build a DFA that recognized a different language, such as $L = \{\text{strings ending with pattern } p\}$?

Our focus on DFAs accepting languages of the form $L = \{\text{strings containing pattern } p\}$ has enabled us to make two important assumptions to simplify our drawings of DFAs:

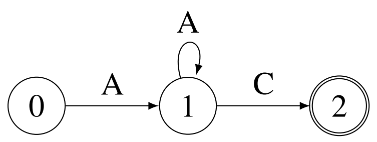
1. Once the entirety of p is found in t, the DFA should accept t regardless of what comes after p in t. Consequently, the acceptance state at the end of the DFA is also an *absorption state*, meaning that the DFA will never exit this state once it is entered. The assumption that all acceptance states are absorption states allows us to exclude arrows from acceptance states to themselves in DFA diagrams.

2. The DFA should accept t if p is found in t, regardless of what comes before p in t. Consequently, whenever the next letter in t is not consistent with p, we can move several states backwards in the DFA and continue searching for p in t (this is the failure function). Because many incorrect next letters in t will send us back to state 0 (the beginning of p), we typically exclude these arrows from DFA diagrams and assume that all missing arrows from non-acceptance states go to state 0.

As an example of these simplifications, consider a DFA accepting the language
$L = \{\text{strings containing the pattern } AC\}$ over the alphabet $\Sigma = \{A, C, G, T\}$. The complete version of this DFA is:



However, the two assumptions above allow us to simplify this DFA to:



For each of the following languages L over the alphabet $\Sigma = \{A, C, G, T\}$, draw a DFA that accepts L. List any assumptions you make that allow you to exclude arrows from your DFA diagram.

a. $L = \{\text{strings beginning with pattern } AC\}$

b. $L = \{\text{strings ending with pattern } AC\}$

c. $L = \{\text{strings exactly matching pattern } AC\}$

d. $L = \{\text{strings containing } AC\text{, but not as the first or last two characters}\}$

Problem 6: More Languages!

For each of the following languages L over the alphabet $\Sigma = \{A, C, G, T\}$, either (i) construct a DFA that accepts L, or (ii) explain why it is not possible to construct a DFA that accepts L. Again be sure to explain any assumptions you make that allow you to exclude arrows from your DFA diagrams.

a. $L = \{$DNA strings of length 4$\}$

b. $L = \{$strings of one or more TTAGGG repeats$\}$

c. $L = \{$strings of an odd number of TTAGGG repeats$\}$

d. $L = \{$palindromic DNA strings$\}$

*Note: Strings of TTAGGG repeats refers to strings that are comprised of repeated sequences of **only** TTAGGG. These repeats, called telomeres, are present at the ends of all human chromosomes. Telomeres play important roles in protecting the integrity of the genome by preventing the loss of genetic information at the ends of chromosomes during DNA replication.*

e. **Bonus:** Recall from class that DFAs, NFAs, and regular expressions are equivalent in that they all represent the same set of languages—namely, the set of regular languages. In this problem, you will begin to prove the validity of this claim. Prove that every regular expression can be converted into a NFA that accepts the same language.

   Hint: Consider each of the cases in the recursive definition of a regular expression to prove this claim recursively.

Problem 7: Genetic Testing

Genetic testing can be used to determine whether someone possesses a genetic trait. Please read pages 1-3 of this paper about genetic testing (stop when you get to the Case Presentation header).

Part I: Ethical Perspectives

a. Compare and contrast the utilitarian and libertarian perspectives, including their core values *(1)* and their views of genetic testing *(2)*. Which perspective do you think is the better framework for genetic testing *(3)*? Explain your reasoning *(4)*. Make sure to discuss all four points in your response (5-8 sentences).

b. In what ways does the justice principle align with the utilitarian view on genetic testing? In what ways does it align with the libertarian view? (3-5 sentences).

Part II: Case Study

Next, read the hypothetical case study below adapted from Kielstein and Sass (2002).

> *CASE STUDY: A 55-year-old female is a regular dialysis patient with autosomal dominant polycystic kidney disease (ADPKD). ADPKD is a dominant genetic disorder that causes cysts to grow in the kidney. Typically, kidney function slowly deteriorates in ADPKD patients to the point of kidney failure. There is no cure, although there are treatments to manage symptoms. If affected, patients usually begin to present symptoms between the ages of 30 and 40.*
>
> *The 55-year-old patient had 2 sons. The first son decided to undergo genetic screening while the second son decided not to. The genetic testing revealed that the oldest son possessed one of the genetic mutations known to cause the disease. The first son, who was 30 years old, divorced his wife and sold their home to keep from burdening her or planning a family. He did not have problems until he was 54. The second son was 21 at the time, and several years later married and had two children. At the time of the case study's publication, he was 42 and had never experienced any symptoms.*

c. Do you think the individuals in the case study handled the situation appropriately *(1)*? Did any of them act in accordance with the utilitarian or libertarian perspectives *(2)*? Does this case study change how much you agree with the perspectives in the previous question *(3)*? Make sure your response touches on all three points (4-6 sentences).

Part III: Algorithmic Implications

One of the potential computational solutions to detect genetic markers is to use a pattern matching algorithm such as Knuth-Morris-Pratt. In the example case study, ADPKD is caused by mutations of many types (splice site, in-frame, and out-of-frame deletions and insertions, nonsense mutations, and missense mutations) in two different genes, *PKD1* and *PKD2*.

d. In one to two sentences, explain why the KMP algorithm might not be able to successfully detect ADPKD.