String Processing, Streaming Algorithms, and Modeling

The goal of this problem set is to motivate the ideas which are to be discussed in the Theory of Computation course. You may code the following in the programing language of your choice. Sample input and output are in the document attached.

1 String Processing

A string is a sequence of characters of an alphabet, concatenated together. The concept is no different from how you have seen strings in your previous programming courses. The following two problems are about string processing.

1.1 Substring Matching

Fix an alphabet $\Sigma = \{a, b, c, d\}$. Write a program that takes two strings u and v with characters from Σ and returns **True** if v is a substring of u, and **False** otherwise. Consider the following variations:

1. Let us say that u is fixed, and you are allowed to hard-code it in your program. In some sense, enumerate all the substrings of u, and check if v is a member of the set of substrings of v.
2. If v is a substring of u, then u is a superstring of v. Suppose v is fixed, and you are allowed to hard-code it in your program. In some sense, enumerate (or characterize) all the superstrings of v and check if u belongs to that set.
3. How would you deal with the streaming versions of the above mentioned variations?

1.2 Substring Matching

Fix an alphabet $\Sigma = \{a, b, c, d\}$. Write a program that takes two strings u and v with characters from Σ and returns **True** if v is a subword of u, and **False** otherwise. Consider the same variations as in **1.1**.

2 Streaming Algorithms

Streaming algorithms represent a theoretical framework for studying problems in interactive computing. They model, in particular, that the input in an interactive system does not arrive as a batch but as a sequence of input portions and that the system must react in response to each incoming portion. That is to say that the input is given sequentially, and not at once. They take into account that at any point in time future input is unknown. We wish to design streaming algorithms, which is optimal with respect to time and space complexity.

2.1 Division by Six

Consider the alphabet $\Sigma = \{0, 1\}$. Consider a string on this alphabet to represent reverse binary encoding of a number, that is, "10010" $\equiv 9$ and "011001" $\equiv 38$. Given a string on Σ, return **True** if the corresponding number is divisible by 6, and **False**, otherwise. Design a streaming version of this program where the input is given character by character, and output is given after each character, representing if or not the string provided till that point represents a number divisible by six.

2.2 Palindrome

Consider the alphabet $\Sigma = \{0, 1\}$. Write a program that returns **True** if the input string is a palindrome, and **False**, otherwise. Design a modified streaming algorithm where the input is given character by character, representing if or not the string provided till that point represents a palindrome.

3 Modeling

We would like to model the following objects using an abstraction called finite state machine, and see how it behaves with respect to a given sequence of activities/actions.

3.1 Turnstile

A Turnstile[1] is a type of a gate which allows only one person to pass at a time. In some scenarios, it attempts to restrict passage only to people who insert a coin, a ticket, a pass, or similar. One may easily find it at public transport stations, amusement parks, and at buildings restricted to authorized personnel. Here we wish to model a coin-operated turnstile.

Thus a turnstile can be used in the case of paid access (sometimes called a faregate or ticket barrier when used for this purpose), for example to access public transport, a pay toilet, or to restrict access to authorized people, for example in the lobby of an office building.

[1] https://en.wikipedia.org/wiki/Turnstile

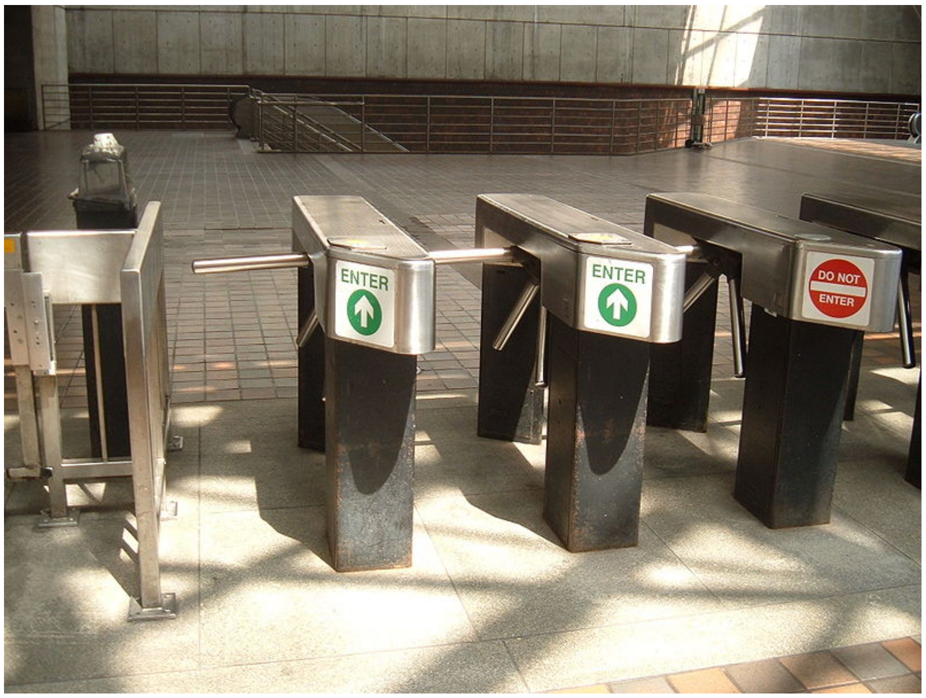

Figure 1: Turnstiles at Alewife Station, Cambridge MA

We look at a particular case of coin-operated turnstile. There are two states of the turnstile - locked and unlocked. There are two actions that one can perform - inserting a coin, and pushing the bar. We assume that the initial condition is locked. We can summarize the behavior of the turnstile using the following state diagram:

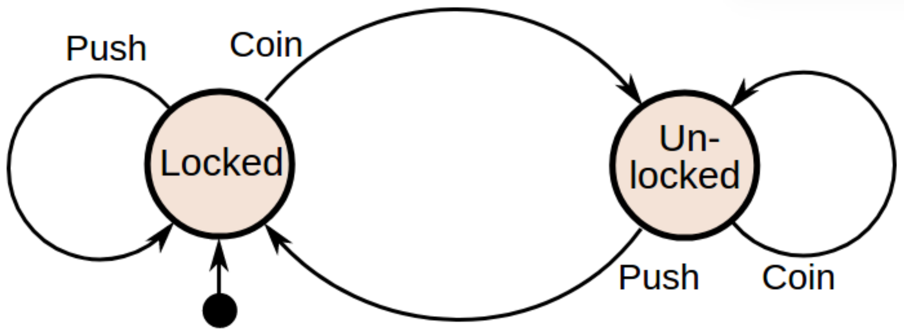

Figure 2: A state diagram for modeling turnstile

Now, say P represents push and C represents coin. Then, a string on the alphabet $\{P, C\}$ represents the sequence of actions on the turnstile. Keeping the state diagram in mind, design a program that takes a string on $\{P, C\}$ and returns **LOCKED** if the turnstile reaches locked state (after the sequence of executions of the corresponding actions) and returns **UNLOCKED** otherwise.

3.2 Coffee Dispenser

Consider a Coffee Dispenser where you have two options - milk coffee (which costs seven rupees) and black coffee (which costs five rupees). You first have to select the coffee of your choice, and then insert the corresponding amount. If excess amount is inserted, the machine returns it. If exact change is not inserted, the machine returns the change. The machine can only accept one rupee, two rupee and five rupee coins. Then, it returns the coffee.

The given description is incomplete, and does not mention what are the states and what are the actions. Attempt to define it precisely, and design a state diagram to represent the coffee dispenser. For each action, assign a capital letter of the Latin alphabet, and design a program which takes a string over that alphabet and returns the name of state the system reaches on execution of the sequence of the corresponding actions.

3.3 Algorithms

Can you do the same/similar process of identifying states and actions, and designing a state diagram for the algorithms/programs designed in the first two sections?