University of Waterloo
CS240 - Spring 2020
Assignment 5

Due Date: Wednesday August 5, 5pm

The integrity of the grade you receive in this course is very important to you and the University of Waterloo. As part of every assessment in this course you must read and sign an Academic Integrity Declaration before you start working on the assessment and submit it **before the deadline of August 5** along with your answers to the assignment – i.e. **read, sign and submit A05-AcInDe.txt now or as soon as possible**. The agreement will indicate what you must do to ensure the integrity of your grade. If you are having difficulties with the assignment, course staff are there to help (provided it isn't last minute).

The Academic Integrity Declaration must be signed and submitted on time or the assessment will not be marked.

Please refer to the course webpage for guidelines on submission. Submit your written solutions electronically as a PDF with file name a5Submit.pdf using MarkUs. We will also accept individual question files named a5q1.pdf, a5q2.pdf, . . . if you wish to submit questions as you complete them. There is a programming question in this assignment; don't forget to submit your completed code, named `DFA.cpp`.

Problem 1 One-sided range search in a heap [5 marks]

Heaps are suitable for *one-sided range queries*. Specifically, assume you are given a max-heap H and an integer x, and you are asked to return all keys in H that fall in the range $[x, \infty)$, i.e., all keys that are at least x.

Give an algorithm that answers such a one-sided range query in time that depends only on the number k of keys in the output. Describe your algorithm, justify why it works, and analyze the runtime.

Problem 2 A variant of 2d range search [7 marks]

Consider the following variant of two-dimensional range reporting queries. We keep a set of two-dimensional points P in a data structure. All points have positive coordinates. For a query range $Q = [a, b] \times [0, c]$, we must find some point p in $P \cap Q$ or report NULL if the intersection is empty. In other words, we must find one point p such that $p.y \leq c$ and $a \leq p.x \leq b$ (we do not need all of them; only one!)

Describe a data structure that answers queries described above in $O(\log(n))$ time and uses space $O(n)$, give the search algorithm and analyze its runtime. Justify your answers.

Problem 3 KMP [6+6+2+4 marks]

1. Compute the failure array for the pattern $P =$`bacbac`.

b	a	c	b	a	b	a	a	b	a	b	a	c	b	a	c	b	a	c	a

Table 1: Table for KMP problem.

2. Show how to search for pattern $P =$`bacbac` in the text $T =$`bacbabaababacbacbaca` using the KMP algorithm. Indicate in a table such as Table 1 which characters of P were compared with which characters of T. Follow the example given in class: place each character of P in the column of the character of T we compare it to and put brackets around the character if an actual comparison was not performed. You may not need all space in the table.

3. How would you modify the KMP algorithm given in class to output all matches instead of only the first one? You should be able to change or add just a few lines.

4. We want to find the longest prefix of a string S which reads the same forward and backward. Explain how to do this, with a justification (hint: use the KMP failure array of the string $S + X + \text{reverse}(S)$, where the + sign indicates concatenation, and X is a character not in S).

Problem 4 Huffman encoding [4 marks]

Define frequencies $(f_i)_{1\leq i\leq n}$ by $f_1 = f_2 = 4$ and $f_i = 2^i + 2^{i-3}$ for $3 \leq i \leq n$. Draw the corresponding Huffman tree and give a brief justification

Problem 5 String matching with a DFA [3+2+2+6+12 marks]

In this problem, you will implement the algorithm briefly described in class for string matching with a Deterministic Finite Automaton (DFA) (p. 16/31): we construct an automaton using a pattern P, then we match in a text T. The algorithm for matching is simple and efficient, but the construction of the automaton takes more work.

Let P the pattern to search, of length m. Then the *states* of the automaton are $0, \dots, m$ and the *transition function* δ of the automaton is defined as follows, for a state q and a character C in Σ:

$$\delta(q, C) = \ell(P[0..q-1]C),$$

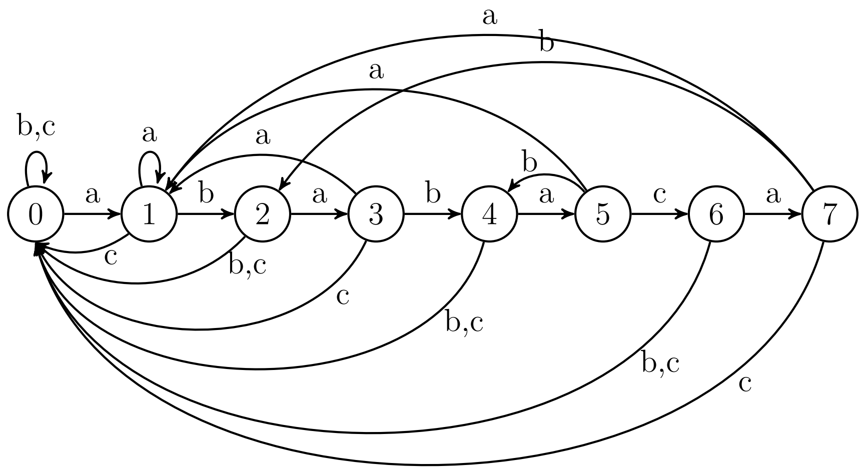
where

- $P[0..q-1]C$ is the concatenation of $P[0..q-1]$ and C
- for a string s, $\ell(s) \in \{0, \dots, m\}$ is *the length of the longest prefix of P that is also a suffix of s.*

This means that if we are in state q, and we read character C, our new state is $\delta(q, C)$ as defined above. Graphically, this corresponds to the transition

$$\begin{tikzcd} q \arrow[r, "\text{C}"] & \delta(q, C) \end{tikzcd}$$

The automaton given in class is not complete; here is the entirety of it, for the pattern $P =$ `ababaca`:



For instance, suppose we take the state $q = 5$ and the character $C =$ `b`. Then, $P[0..4]C =$ `ababab` and the longest prefix of $P =$ `ababaca` that is a suffix of $P[0..4]C =$ `ababab` is `abab`, which has length 4. So we set $\delta(5, \texttt{b}) = 4$.

If we have the automaton, the string matching algorithm is simple. We start in state $q = 0$. For $i = 0, \dots, n-1$, we replace q by $\delta(q, T[i])$; every time we hit $q = m$, we report $i - (m-1)$. The proof of correctness relies on the following fact, that we do not ask you to prove: *for all i, the state after reading $T[i]$ is* $\ell(T[0..i])$ (that is, the length of the longest prefix of P that is also a suffix of the string $T[0..i]$ we have read so far). In other words, the state tells us the best match we can have on the right-end side of $T[0..i]$.

1. Taking the fact above for granted, prove that the algorithm correctly reports the location of all matches (and only reports matches, no false positives).

Instead of tabulating all transitions (q, C), for q in $\{0, \ldots, m\}$ and C in Σ, we only store $\delta(q, C)$ for q in $\{0, \ldots, m\}$ and C a character appearing in P. We will thus write S for the set of characters in P (there is no repetition in S) and we will let s be its size.

2. If C is a character in $\Sigma - S$, what should be the value of $\delta(q, C)$? (Give a justification)

3. Explain what data structure you would use to store the values $\delta(q, C)$ for C in S, and why. Several solutions are possible; for full credit, your justification should briefly discuss the time for inserting or finding $\delta(q, C)$ for given q, C, in terms of s. (For this question, the alphabet Σ is arbitrary; do not assume it has fixed size, like ASCII for instance).

4. Give an algorithm that takes P as input and computes and stores the values of the transition function $\delta(q, C)$ for all q in $\{0, \ldots, m\}$ and C in S; state and justify its runtime in terms of m and s. For full credit, the runtime for computing the values $\delta(q, C)$ should be $O(m^3 s)$, although faster answers are possible.

   We suggest that you first give an algorithm for computing the function $\ell(\;)$ above, and study its runtime.

5. Implement the construction of the automaton as explained above and the corresponding pattern matching algorithm in C++, in a file `DFA.cpp`. Here, the alphabet is the set of `char`'s. You may use one of the containers of the Standard Template Library; refer for example to https://en.wikipedia.org/wiki/Standard_Template_Library.

   You should read the pattern from file `a5pattern.txt` and the text from file `a5text.txt`. File `a5pattern.txt` will contain a single line. The program should concatenate all lines of the text `a5text.txt` into a single string and should output two things:

   - the transition function, as a sequence of triples

     $$q \quad C \quad \delta(q, C)$$

     for q in $\{0, \ldots, m\}$ and C in S, one triple per line, with values separated by a single space (do not leave a trailing space after the last value on a line). The order is not important.

   - all indices i at which the pattern occurs at $T[i..i + m - 1]$, one number per line, in increasing order.

   The triples should be printed in a file `a5delta.txt` and the indices of the matches in a file `a5matches.txt`. For example, for the automaton seen in class, the following lines should be part of `a5delta.txt`:

```
0 a 1
0 b 0
0 c 0
```