University of Waterloo
CS240E, Winter 2024
Assignment 5

Due Date: Thursday, April 4, 2024, at 5pm

Be sure to read the assignment guidelines (`http://www.student.cs.uwaterloo.ca/~cs240e/w24/assignments.phtml`). Submit your solutions electronically as individual PDF files named a5q1.pdf, a5q2.pdf, . . . (one per question).

Grace period: submissions made before 11:59PM on April 4, will be accepted without penalty. Please note that submissions made after 11:59PM **will not be graded** and may only be reviewed for feedback.

Question 1 [3+2+2=7 marks]

Fast Fourier Transform gives us a way to multiply two polynomials of degree $O(n)$ in $O(n \log n)$ time (assuming arithmetic operations take constant time). This question explores some of the many problems we can solve by phrasing them as a polynomial product.

a) Given two vectors $a = [a_1, \ldots, a_n]$ and $b = [b_1, \ldots, b_n]$, explain how to compute the dot product of a with every cyclic shift of b in $O(n \log n)$.

For instance, when $n = 3$, we would like to compute:

$$[a_1, a_2, a_3] \cdot [b_1, b_2, b_3],$$
$$[a_1, a_2, a_3] \cdot [b_2, b_3, b_1], \text{ and}$$
$$[a_1, a_2, a_3] \cdot [b_3, b_1, b_2].$$

b) We are given two *cyclic strips* $a = [a_1, \ldots, a_n]$ and $b = [b_1, \ldots, b_n]$ where every entry of each of a, b is either zero or one.

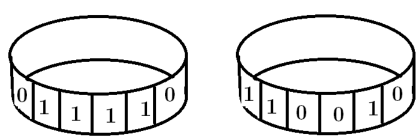

Figure 1: example cyclic strips.

Give an $O(n \log n)$ algorithm to find the number of ways to stack them on top of each other so that no two *one* (1) entries of both a and b are adjacent.

c) We are given one cyclic strip $a = [a_1, \ldots, a_n]$ where every entry of a is either zero or one. Give an algorithm to find the number of ways to stack a on top of itself so that all entries match. As before, the runtime should be in $O(n \log n)$.

Question 2 [3+2+3+3=11 marks]

Recall that we had two versions of the KMP failure function: For $j < m - 1$

- $F[j]$ is the length of the longest prefix of P that is a suffix of $P[1..j]$, and
- $F^+[j]$ is the length ℓ of the longest prefix of P that is a suffix of $P[1..j]$ and where additionally $P[\ell] \neq P[j{+}1]$, or 0 if no such ℓ exists.

This assignment asks you to explore the difference that using F^+ can make.

a) Show the Knuth-Morris-Pratt automaton for the pattern $P = aaabaac$ for $\Sigma = \{a, b, c\}$, once when using F for the failure-arcs and once when using F^+.

b) Consider the pattern $P = a^m$ for some integer m. For $1 \leq j \leq m - 2$, where does the failure-arc from state j lead to if we use F and F^+, respectively? Briefly justify your answer.

c) Show that using F^+ can cut the number of checks in half. (Recall that a *check* is testing whether $P[j] = T[i]$ for some j, i, as done in line 5 of *KMP::patternMatching*).

To do so, design (for all sufficiently large n) a text T of length n and a pattern P that does not exist in T, but detecting this with KMP takes almost twice as many checks with F than it does with F^+. (You can choose the length of P; it suffices to give one P for each n.) Justify your choice by arguing how many checks are taken with each failure-function.

["Almost twice as many" means that as n goes to infinity, the ratio between the number of checks should go to 2.

d) Show that for any text T and any pattern P not in T, using F will require at most twice as many checks as using F^+.

Question 3 [2+4+7=13 marks]

a) Consider the text $S =$`ARECEDEDDEER`. Show a Huffman-trie for this text (using $\Sigma_S = \{A, C, D, E, R\}$). Also indicate with every node (including interior nodes) the frequency that this node had when building the Huffman-trie.

b) Assume we have characters $x_1, \ldots, x_n$ where x_i has frequency $F(i)$. Here $F(i)$ is the *Fibonacci-sequence*: $F(1) = 1, F(2) = 1,\ F(i) = F(i{-}1) + F(i{-}2)$ for $i \geq 3$. Argue that any Huffman tree of these characters has height $n{-}1$.

Hint: For $i \geq 2$, what is the frequency associated with the parent p_i of x_i?

c) Assume we have characters $x_1, \ldots, x_n$ where x_i has frequency f_i and $\min_i\{f_i\} = 1$. Assume further that some Huffman-tree T for these characters has height $n{-}1$. Argue that $\max_i\{f_i\} \geq F(n{-}1)$, where $F(\cdot)$ is again the Fibonacci-sequence.

Hint: Use the structure of a binary tree of height $n-1$ to enumerate your characters suitably, and then argue a lower bound on f_i and on the frequency associated with the parent p_i of x_i.

Question 4 [2+4+2=8 marks]

This question concerns Lempel-Ziv-Welch encoding of the word A^n, which is the word consisting of n copies of the character A. In the following, use as alphabet the 128 ASCII characters, stored with code-words 0 up to 127. In particular, 'A' has code-word 65, and the first code-word you can use for strings added to the dictionary is 128.

a) Give the encoding (as list of numbers, *not* as a bit-string) of A^{16}. Show your work.

b) Recall that traditional Lempel-Ziv-Welch converts integers into 12-bit strings. This means that when we add codeword 4096 to the dictionary, this would result in an overflow-error.

When encoding A^n, what is the smallest n for which we get this overflow-error? Justify your answer theoretically (i.e., the answer "I implemented LZW and it used code 4096 at $n = X$" will *not* give you credit.)

c) Let X be the answer that you got in part (b). Prove that for *any* ASCII-string of length X or more, using Lempel-Ziv-Welch leads to a dictionary-overflow.

Question 5 [2+2+3+3=10 marks]

Recall the Elias-Gamma codes from class; we use $E_\gamma(N)$ to denote it for integer $N \geq 1$.

a) Show the trie that stores $E_\gamma(N)$ for $N \in \{1, \ldots, 7\}$.

b) Elias-Gamma codes begin with long runs of 0. For this reason, an idea to obtain shorter codes is to encode these runs recursively. Specifically the *recursive Elias-Gamma code* $E_r(N)$ is computed with Algorithm 1 given below.

Algorithm 1: *recursiveEliasGamma::encodeOneNumber*(N)

`// pre:` $N \geq 1$
$c \leftarrow$ empty word
while $N > 1$ **do**
  $w \leftarrow$ binary representation of N
  $c.prepend(w)$
  $N \leftarrow |w| - 1$
$c.prepend(0)$
return(c)

Show $E_r(N)$ and $E_\gamma(N)$ for $N = 2, 4, 8, 16$. No explanation needed.

c) You should notice that $|E_r(N)| \geq |E_\gamma(N)|$ for $i = 1, \ldots, 16$. What is the smallest value of N such that $|E_r(N)| < |E_\gamma(N)|$? Justify your answer.

d) Consider the following bitstring:

$$C = 0\,1\,1\,1\,0\,1\,0\,1\,0\,0\,1\,1\,0\,1\,0\,1\,0\,1\,0\,0\,1\,1\,0\,1\,0\,1\,0\,1$$

which has the form $C = E_r(N_1) +\!\!+ E_r(N_2) +\!\!+ \ldots +\!\!+ E_r(N_k)$ for some integer $k \geq 1$ and integers $N_1, \ldots, N_k \geq 1$. What is N_1? Explain how you obtained the answer by describing the idea for an algorithm that would convert any concatenation of recursive Elias-Gamma codes into the corresponding list of integers. Also show how this algorithm worked to obtain N_1. (You do not have to give the details of the algorithm, or analyze its correctness or run-time.)