University of Waterloo
CS240E, Winter 2022
Assignment 5

Due Date: Wednesday, March 30, 2022 at 5pm

Be sure to read the assignment guidelines (`http://www.student.cs.uwaterloo.ca/~cs240e/w22/guidelines/guidelines.pdf`). Submit your solutions electronically as individual PDF files named a5q1.pdf, a5q2.pdf, ... (one per question).

Question 0 Academic Integrity Declaration

Read, sign and submit A05-AID.txt now or as soon as possible.

Question 1 [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 2 [3+3+3=9 marks]

We are searching for pattern P in text T where $|T| = n$, $|P| = m$, and $n \geq m \geq 1$.

a) Show that any pattern matching algorithm ~~must do at least $\lfloor n/m \rfloor$ checks~~ must look at at least $\lfloor n/m \rfloor$ characters of T in the worst case.

b) Consider pattern $P = 0^m$ and let text T be a string of $n \geq m$ bits that were randomly chosen to be 0 or 1 with equal probability. Let X be the number of checks done by Boyer-Moore until it mismatches for the first time or returns with success. (The check that leads to a mismatch is included in this count.) Show that $E[X] \leq 2$.

c) Consider the same setup as in the previous part. Assume you just had a mismatch. Show that the expected amount by which you shift the guess forward is at least $m - 1$.

Motivation: For the special string $P = 0^m$, the expected number of checks is hence $\approx 2\frac{n}{m-1}$ (i.e., roughly within a factor 2 of the lower bound) because you expect to do 2 checks until a mismatch and then shift forward by $m - 1$ characters.

Question 3 [3 marks]

Let T be a text of length n. Recall that the suffix tree of T has $O(n)$ nodes and height $O(n)$. Also, the trie of suffixes of T has $O(n^2)$ nodes and height $O(n)$.

Show that these bounds are tight, even if the alphabet is small. To do so, design (for all sufficiently large n) a bitstring T of length n such that its trie of suffixes has $\Omega(n^2)$ nodes and its suffix tree has height $\Omega(n)$. Justify your answer by explaining the structure of both tries. You may assume that n is divisible as needed.

Question 4 [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 5 [2+2(+5)=4(+5) marks]

Sometimes, Huffman-encoding is described in terms of the *probability* p_i (of a character $x_i \in \Sigma$), which is defined as the frequency of x_i divided by the length of the source text.

a) (Warm-up.) Consider the text $ACAGATATACACAACG$ over alphabet $\Sigma = \{A, C, G, T\}$.

What is the cost of the corresponding Huffman-encoding? Show how you obtained your answer, and also write the length of the code-word for each character.

b) Given some probabilities $p_1, \ldots, p_s$ (with $0 < p_i < 1$ and $\sum_{i=1}^{s} p_i = 1$), the *entropy* is defined to be

$$H(p_1, \ldots, p_s) = -\sum_{i=1}^{s} p_i \log_2(p_i).$$

For a text S, we define the entropy $H(S)$ to be $|S| \cdot H(p_1, \ldots, p_s)$, where $p_1, \ldots, p_s$ are the probabilities of the characters that occur in S.

Compute $H(S)$ for the text from part (a). Show how you obtained the answer (in particular, list the probabilities).

c) **(Bonus)** Let S be a text such that the length of S and the frequency of all characters in S are powers of 2. (Say $|S| = 2^{\ell_0}$, and the characters in S are $x_1, \ldots, x_k$ where x_i has frequency $f_i = 2^{\ell_i}$ for some integer $\ell_i \geq 0$.)

Show that the Huffman-encoding of S has cost $H(S)$. (Hint: What is the length of the codeword of x_i? Part-marks for this.)

Motivation: The character-probabilities are used to develop a lower bound on *any* encoding into a bit-string (regardless whether it comes from a prefix-free binary encoding or elsewhere). Namely, based on Shannon's information-theoretic lower bound, one can argue that any such encoding has length at least $H(S)$. So in the special case where the frequencies are powers of 2, Huffman-encoding gives the minimum-length encoding that is possible.

Question 6 [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.

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.

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

```
// pre:  N ≥ 1
c ← empty word
while N > 1 do
    w ← binary representation of N
    c.prepend(w)
    N ← |w| − 1
c.prepend(0)
return(c)
```

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.)