University of Waterloo
CS240 Winter 2023
Assignment 5

Due Date: Wednesday, April 5 at 5:00pm

If you are having difficulties with the assignment, course staff are there to help (provided it isn't last minute).

Please read `https://student.cs.uwaterloo.ca/~cs240/w23/assignments.phtml#guidelines` for guidelines on submission. **Each question must be submitted individually to MarkUs as a PDF** with the corresponding file names: a5q1.pdf, a5q2.pdf, ... , a5q6.pdf. It is a good idea to submit questions as you go so you aren't trying to create several PDF files at the last minute.

Late Policy: Assignments are due at 5:00pm, with the grace period until 11:59pm. Assignments submitted after 11:59 on the due date will not be accepted but may be reviewed (by request) for feedback purposes only.

Question 1 [3+4 = 7 marks]

a)

```
KarpRabinModified(T, P)
1.  M ← suitable prime number
2.  h_P ← h(P[0..m−1]))
3.  h_T ← h(T[0..m−1]))
4.  s ← 10^{m−1} mod M
5.  NEW LINE
6.  i ← 0
7.  while i < n − m − 1
8.      if h_T = h_P
9.          if strcmp(T[i..i+m−1], P) = 0
10.             return "found at guess i"
11.     if i < n − m − 1 // compute hash-value for next guess
12.         NEW LINE
13.         i ← i + 2
14. return "FAIL"
```

Suppose we are searching for a pattern P of length m in text T of length n using KarpRabin algorithm, but we are interested in a match only at even positions in the text T. Above we provide most of this modified algorithm. Insert a single assignment on line 5 and a single assignment on line 12 to complete the algorithm. Explain your answer.

b) Suppose we have a text T and string S, both of length n, and the alphabet for both text and string are digits $\{0, 1, ...9\}$. Let h be the hash function we use for Rabin-Karp, and assume that for any substring x in text T and any substring y in string S, $x = y$ if and only if $h(x) = h(y)$. Design an algorithm, in pseudocode or in English, that returns $\sum_{i=0...k} h(S[i...n-1])$ where k is the smallest integer s.t. $S[k...n-1]$ is present in T. The running time of your algorithm must be $O(n \log n)$. Briefly justify running time and correctness.

Question 2 [4+4 = 8 marks]

a) Compute the KMP failure array for the pattern $P = bbbabb$. If you wish, you can show the intermediate steps, but providing the final answer is sufficient.

b) Show how to search for pattern $P = bbbabb$ in the text $T = bbbcbbacbbbbbabb$ using the KMP algorithm. Indicate in a table such as Table 1 which characters of P were compared with which characters of T, like the example in Module 9. Place each character of P in the column of the compared-to character of T. Put round brackets around characters if an actual comparison was not performed. You may need to add extra rows to the table.

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

Table 1: Table for KMP problem.

Question 3 [3+6=9 marks]

In this question, we will develop a version of Boyer-Moore algorithm which expands the bad character heuristic by using 'next-to-last' occurrence array, in addition to the last occurrence array.

a) Define $N(c)$ as the next-to-last occurrence of character in pattern P. For example, if $P = aabra$, then $N('a') = 1$. Design an algorithm to compute N in $O(m + \Sigma)$ running time, where m is the length of pattern P. You can describe your algorithm either in pseudo-code or in English. Briefly justify the running time and correctness of your algorithm (one sentence for correctness and one sentence for the running time).

Erratum: A previous version of this assignment required time efficiency $O(m)$, which was not intended.

b) Develop a modified version of Boyer-Moore algorithm that makes use of both the last occurrence array L and the next-to-last occurrence array N. If text and pattern characters match, just as in the standard Boyer-Moore, your algorithm must decrease both i and j. Whenever a mismatch occurs, your algorithm should make the largest possible valid[1] shift of the pattern, based on the information in L and N. You must provide pseudo-code for your algorithm as well as a justification for your modifications. Your pseudocode should be named as $BoyerMooreModified(T, P, L, N)$, where the input parameters, respectively, are the text, pattern, the last and next-to-last occurrence arrays.

Question 4 [2+4 = 6 marks]

a) Construct the suffix array for $S = balbes$. You can show the intermediate steps, but showing the final answer is sufficient.

b) Design an algorithm that given a string S and its suffix array A, checks whether a string S has a substring of length k that appears at least k times in worst case running time $O(kn)$. Here n is the length of S. For example, if $S = blahblah$, and $k = 2$, your algorithm should return true as there is a substring bl repeated 2 times. If $k = 3$, your algorithm should return false as there is no substring of length 3 of S that appears at least 3 times. You can assume $k < n$.

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

a) Build the Huffman's tree for the string `AAABBCCCCCD` and give the coded text. When building the tree, put the smaller weight subtree to the left. If subtrees have the same weight, put the tree which has the character earlier in the alphabet to the left. Show all the intermediate steps.

b) Prove that in the Huffman's tree, there are at least two leaves at the deepest level and the two leftmost deepest leaves must have the same parent.

c) Suppose we perform an in-order traversal of an arbitrary Huffman's tree (the left subtree visited before the right subtree), and as we reach a leaf, we output to a list L

[1]Here valid means that while you make the largest shift, you do not skip any shifts where a match is possible.

the character stored at this leaf, together with the depth of the leaf. Give an algorithm of runtime cost $O(n \log n)$ that reconstructs the Huffman's tree from L, where n is the number of leaves in the tree. Justify your answer.
Hint: use techniques similar to the construction of the Huffman's tree from frequencies.

Question 6 [3+3= 6 marks]

a) Use the LZW method to compress the source text `S = ABCABCACAB` where the source alphabet is $\Sigma = \{A, B, C\}$ and the corresponding codenumbers are $A = 65, B = 66, C = 67$. The next new codenumber your algorithm creates should be 68. Show the coded text using codenumbers and also give the dictionary D you constructed during encoding. You do not need to show the intermediate steps.

b) Suppose the alphabet is $\Sigma = \{a, b, c\}$. Give the string of length 10 for which LZW compression is the worst possible. If there are many strings leading to the worst compression possible, your string should be the smallest according to the lexicographic order. Explain why this string leads to the worst possible compression.