University of Waterloo
CS240 Spring 2021
Assignment 5

Written Questions Due: Wednesday, July 28 at 5:00pm
Programming Question Due: Wednesday, August 4 at 5:00pm

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 July 28th** 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 read `http://www.student.cs.uwaterloo.ca/~cs240/s21/guidelines.pdf` 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, encode.cpp, decode.cpp.

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.
Remember, late assignments will not be marked.
Late assignments, however, can be reviewed for **feedback only** upon request to the ISAs at cs240@uwaterloo.ca.
Note: you may assume all logarithms are base 2 logarithms: $\log = \log_2$.

Problem 1 [4+2+2+4=12 marks]

a) Draw the kd-tree corresponding to the set of points:
$S = \{p_1, ..., p_8\} = \{(1,1), (2,2), (3,3), (4,4), (5,5), (6,6), (7,7), (8,8)\}$.

b) Construct a kd-tree storing $n = 4$ two dimensional points in general position s.t. there exists a range search returning an empty result and yet all nodes of the kd-tree are explored. Draw the picture with the points, the splits of the kd-tree and the rectangle R representing the range search.

c) Can the result in (b) be generalized to arbitrary large values of n?

d) Suppose set S contains 2D points with integer coordinates, but these points are not necessarily in general position, namely two distinct points can have either the same x or the same y coordinate (but not both, i.e. all points in S are distinct). Therefore the standard kd-building algorithm on S does not guarantee a balanced tree.

Explain how to modify the points in S to build a new set of points S', where points in S' are in general position, and any range search on S can be transformed into an equivalent range search on S'. Coordinates in S' do not have to be integers. Explain how to convert a range search on S into an equivalent range search on S'. Note that when range-searching in S, the query boundaries are not necessarily defined by integers.

Problem 2 [4+6=10 marks]

a) Draw a range tree that corresponds to the following set of 2D points:

$$S = \{(3,4), (10,2), (9,9), (5,7), (6,1), (0,3), (1,8)\}.$$

Draw the primary tree, and then all the auxiliary trees. Make the primary tree and all the auxiliary trees of the smallest height possible. Make sure to indicate to which node v an auxiliary tree $T(v)$ belongs to.

b) Explain how to construct a two dimensional range tree containing n points in $O(n \log n)$ time. Briefly justify your time complexity.

Problem 3 [4 marks]

Let $M = 7$ be the prime number chosen by Rabin-Karp, $P = 118$ be the pattern to search for, and $h(k) = k \bmod M$. Give a text T, with length 9 that produces the worst-case number of comparisons for Rabin-Karp fingerprinting. T should not contain pattern P. Explain why T produces the worst-case.

Problem 4 [4+4+4 = 12 marks]

a) Compute the KMP failure array for the pattern $P = bbbabb$.

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.

c) Given two words w and v, each of length n, explain how to use the KMP algorithm to find if one string is a cyclic shift of another in $O(n)$ time. For example, *alloy* is a cyclic shift of *loyal* but *alloy* is not a cyclic shift of *aloyl*.

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

Table 1: Table for KMP problem.

Problem 5 [2+4 = 6 marks]

a) Construct suffix array for $S = balbes$.

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 $k < n$.

Problem 6 [3+3 = 6 marks]

a) Suppose we have 4 letters to encode with Huffman algorithm: A, B, C, and D. Let f stand for the frequency, and suppose $f(A) < f(B) < f(C) < f(D)$. Write down a single condition (equation or inequality) that is both necessary and sufficient to guarantee that each symbol is encoded with exactly two bits. Explain your answer.

b) Suppose string S has n characters with the following frequencies: $f(1) = 3$ and $f(i) = 2^i$ for $2 \leq i \leq n$. Draw the corresponding Huffman tree. When constructing Huffman tree, place the lower-weight trie on the left. Explain how you came up with the tree structure.

Problem 7 [12 marks]

In this programming question you will implement a variant of LZW compression, both encoding and decoding. The source alphabet consists of lower case characters: 'a', 'b', ..., 'z'. The output alphabet is binary. The encoding works just as in LZW, with two modifications. The initial dictionary consists of all possible single and double letter strings. For single letter strings, the codewords are as follows: 'a': 0, 'b': 1, 'c': 2,..., 'z': 25. For double letter strings, the codewords are as follows:

'aa': 26, 'ab': 27, ..., 'az': 51,
'ba': 52, 'bb': 53, ..., 'bz': 77,
.....
'za': 676 , 'zb': 677,..., 'zz': 701

The second modification is as follows. At every iteration, the encoder adds two new dictionary entries instead of one, when possible; one entry corresponds to the current string being encoded + the next character (like the usual LZW), while the other entry corresponds to the current string being encoded + the next two characters. The motivation for adding two entries is that our dictionary expands faster, thus possibly leading to better compression. Note that decoding algorithm must be appropriately modified from the original LZW to reflect the changes in the encoding algorithm.

The encoding algorithm should be in file `encode.cpp`, and it is worth 6 marks. Decoding algorithm should be in file `decode.cpp` and it is worth 6 marks. We give you starter files for both.

In `encode.cpp`, the main function reads one line from `cin`, this is the text you have to encode. You can assume the text contains only lower case characters from the range 'a'....'z'. Use 12 bits per code for binary representations, i.e. the largest dictionary has 2^{12} codewords. The output (to `cout`) should be the coded string, as a sequence of zeros and ones. For simplicity of automatic testing, your binary output code should be printed as a string that contains only characters '0' and '1'. For encoding, you must use a trie data structure, however, you can implement only the methods that you need for encoding. For example, it is not necessary to implement trie delete.

In `decode.cpp`, the main function reads one line from `cin`, this is the coded string you have to decode. You can assume the encoded string contains only zeros and ones. The output (to `cout`) should be the decoded string, printed out as a normal string, i.e. "abra".

You may not use any pre-existing code that would trivialize the implementation (e.g. built in data structures from STL, etc). Smart pointers are ok. You are allowed to use `std:string` and `std::vector`.

We provide sample input.txt and output.txt for `encode.cpp`. To test `decode.cpp`, you can reverse the order of input.txt and output.txt.