# University of Waterloo
# CS240 Spring 2016
# Assignment 5

**Due Date: Wednesday, July 20, at 5:00pm**

Please read `http://www.student.cs.uwaterloo.ca/~cs240/s16/guidelines.pdf` for guidelines on submission. This assignment contains written and programming problems. Submit your written solutions electronically as a PDF with file name a05wp.pdf using MarkUs. We will also accept individual question files named a05q1w.pdf, a05q2w.pdf, a05q3w.pdf, a05q4w.pdf if you wish to submit questions as you complete them.

Problem 5 contains a programming question; submit your solution electronically as a file named `encode.cpp/encode.h`.

## Problem 1 Search [23 marks]

**a)** [4 marks] Construct the last occurrence function $L$ and suffix skip array $S$ for pattern $P = adobodoa$. Let $\Sigma = a, b, c, d, o, t$.

**b)** [4 marks] Trace the search for $P$ in $T = dotadotadotdotadobodoadot$ using the Boyer-Moore algorithm.

**c)** [3 marks] Modify the pseudocode for Boyer-Moore algorithm to find all occurrences of $P$ in $T$. Note that if $T = baboraboraboraba$, $P = borabora$ occurs at index 2 and index 6.

**d)** [4 marks] A number of heuristics can be used with Boyer-Moore to reduce the number of comparisons performed between $P$ and $T$. Suppose we use Boyer-Moore with only the Peek heuristic. The Peek heuristic states that if $P[j] \neq T[i]$ and $P[j-1] \neq T[i-1]$ then the next location to search for $P$ at is $T[i+m-1]$. Show that the Peek heuristic may fail to find $P$ in $T$, i.e., find a pattern $P$, and a text $T$ containing $P$, such that Peek fails to find $P$ in $T$.

**e)** [4 marks] Draw the suffix tree for $P = adobobodobobodaa$.

**f)** [4 marks] Let $M = 17$ be the prime number chosen by Rabin-Karp, $P = 181$ be the pattern to search for, and $h(k) = k \mod M$. Give a text $T$, with length 7, that produces the worst-case number of comparisons for Rabin-Karp fingerprinting. Explain why $T$ produces the worst-case.

## Problem 2 Compression [24 marks]

**a)** [4 marks] Draw the Huffman trie to encode words over $\Sigma = a, b, c, d, e, f, g, h$, where the frequency of each symbol is 12.5%. To break ties, choose the smallest-alphabetical letters, or trees containing the smallest-alphabetical letters to combine (i.e., a and b instead of f and h). To combine two trees of different values, place the lower-valued tree on the left.

**b)** [4 marks] Professor Quirell produced a trie that encodes: $a \to 110$, $e \to 10$, $g \to 11$, $r \to 1101$, $f \to 01$, $t \to 010$, and $y \to 010011$. Explain why this trie doesn't uniquely decode $Z = 11011010100100111101101010$. Furthermore, provide three different decodings for $Z$ (and show the partitioning of $Z$ that led to these decodings, e.g., 1101 10 10 10 010011 1101 10 10 10 $\to$ reeeyreee).

**c)** [2 marks] Encode text $T =$"betty bought a bit of better butter" using LZW.

**d)** [4 marks] Encode $T =$"betty bought a bit of better butter" with BWT. Let this encoded sequence be $Q$. Encode $Q$ using LZW.

**e)** [4 marks] Why would transforming a text $T = abacadaeafagaha$ with BWT prior to encoding with LZW yield a better compression ratio than encoding $T$ with $LZW$ directly?

**f)** [3+3=6 marks] Professor Granger has an idea for a new compression algorithm for English text. The algorithm counts the frequency of each word in a source text $T$ and produces a Huffman-like trie to encode words. This dictionary must be sent alongside the coded text to ensure proper decoding.

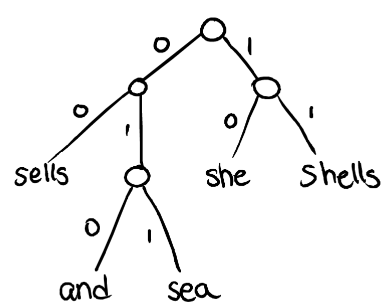


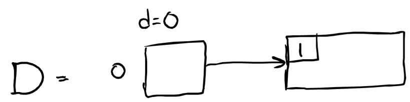

Figure 1: A trie for the text $T =$"she sells sea shells and she shells".

i Is the sum of the dictionary and coded text size guaranteed to be less than the size of the input text? Justify your answer.

ii Describe a text $T$ containing $n$ words that represents the best-case compression for the described method. Why is it the best-case?

## Problem 3 Extendible Hashing [6 marks]

Suppose we have an extendible hashing scheme with block size $S = 3$ and parameter $L = 5$. The universe of keys is non-negative integers with at most 8 bits, $U = 0, 255$, and the hash function is $h(k) = floor(k/16) + (k \mod 16)$: The dictionary $D$ is initially empty, with a single block B, pointed to by the single entry in the directory, which has initial order $d = 0$.




Insert the keys 251, 217, 27, 188, 202 and 85 in-order into $D$. Label the directory order d, and local depth $k_b$ for each block. Its not necessary to redraw $D$ for every insertion, but indicate and label every block split and directory grow (i.e., draw $D$ just before and just after).

## Problem 4 B-Trees [6 marks]

**a)** [2 marks] $T$ is a 2-3 tree with height 4. What is the smallest possible number of keys in $T$? Justify your answer.

**b)** [4 marks] Consider the sequence of keys $\{9, 2, 4, 1, 0, 81, 12, 17, 394, 172, 9412, 3, 4\}$. Insert these keys, in-order, into an empty 2-3 tree.

## Problem 5 Programming [25 + (3) + (3) marks]

Implement any compression algorithm of your choice or invention. The implementation must be able to encode a text $T$ and decode a text $T$ losslessly.

Use the skeleton provided in `encode.h/encode.cpp` Assume that text $T$ contains only ASCII values in the range $[32, 126]$.

**Rules:**

1. If encode($T$) $= T$; no marks will be given for that test
2. If decode( encode($T$) ) $\neq T$; no marks will be given for that test
3. If the size of encode($T$) is greater than 85% of the size of $T$; no marks will be given for that test
4. You may not store source text $T$ in your implementation
5. You may not use any pre-existing encoding or compression library
6. Your code must run within the specified time limit (30 seconds)

**Bonus [3 marks]** Produce a higher compression ratio than LZW on our super-secret texts.

**Bonus Competition [up 3 marks]** How does your compression algorithm compare to those of your classmates? The best-scoring algorithms will receive up to 3 bonus marks (e.g., 3 for first place, 2 for second, and 1 for third).