# University of Waterloo
# CS240 Winter 2020
# Final Assessment, revised April 10, 2020

**Due Date: Saturday, April 16 at 5:00pm EST**

Please read `http://www.student.cs.uwaterloo.ca/~cs240/w20/guidelines.pdf` for guidelines on submission. The preferred method is to submit your written solutions for each problem separately in files FAq1.pdf, FAq2.pdf,...,FAq14.pdf. Alternatively, you can submit written solutions in one PDF file with name FA.pdf using MarkUs. After submission, please check that you have submitted correct files on MarkUs. We highly recommend that you submit your solutions well ahead of the official deadline. No late solutions will be accepted.

Note: questions 1(a), 5(a), 7, 10(a) have been clarified. Example in question 3 has been corrected.

## Problem 1 Short answers [30 marks]

For each of the following questions, only a brief explanation is required. Each question is worth 2 marks.

**a)** If we use 2-4 trees to implement buckets in hashing with separate chaining, then what is the worst case time complexity of insertion, in terms of $\Theta()$? Assume hash table stores $n$ items.

**b)** Professor $X$ claims that he has invented an algorithm that can construct a binary search tree from an unsorted array of size $n$ storing keys by performing $O(n \log \log n)$ comparisons. Explain why the algorithm of Professor $X$ must be incorrect.

**c)** Suppose the alphabet is $\Sigma = \{c_0, c_1, ..., c_{n-1}\}$ and text is $T = c_0c_0c_1c_1...c_{n-1}c_{n-1}$. What is the height of the suffix tree?

**d)** Let $S$ be a set of two dimensional points. Can the height of the quadtree for $S$ be smaller than the height of the kd-tree for $S$?

**e)** Consider a string $S$ of length $n = rs$, consisting of a block of $r$ zeros, followed by a block of $r$ ones, followed by a block of $r$ zeros, etc (with $s$ blocks of length $r$ in total). We apply run-length encoding to $S$. What is the length of the string we obtain?

**f)** Consider a universe $U = \{0, 1, 11, 12, 22, 23, ..., 11t, 11t+1\}$ for some integer $t$, and hash function $h(k) = k \bmod 11$. Suppose we use an array of size 11 for a hash dictionary. Does $h$ satisfy the uniform hashing assumption?

**g)** What is the smallest number of KVP in a B-tree of order 6 and height 7?

**h)** A character $c$ occurs 51% of the time in a string $S$. In a Huffman encoding of $S$ built using the frequencies of characters, what is the number of bits in the encoding of $c$?

**i)** What is the maximum height of a compressed trie storing $n$ strings?

**j)** What is the shortest string that will cause 'the' to be added to the LZW dictionary?

**k)** What is the largest number of auxiliary trees of size 1 in a 2-d range tree on $n$ points?

**l)** Consider the Boyer-Moore string matching algorithm. In terms of $n$ (text length) and $m$ (pattern length), give the best-case running time if the pattern is not in the text. Ignore the preprocessing time.

**m)** Give the run-length encoding for $S = 01000$.

**n)** Give the run-length decoding for $C = 0010101100110$.

**o)** Suppose that algorithms $A$ and $B$ both solve some specified problem. If algorithm $A$ has complexity $\Theta(n^2)$ and algorithm $B$ has complexity $\Theta(n^3)$, can we conclude that algorithm $B$ requires more time than algorithm $A$ for any input instance?

## Problem 2 [4 marks]

What is the running time of the following pseudocode segment? Give a $\Theta(\ )$ expression, in terms of $n$. Justify the time complexity.

```
p ← 1
for i ← 1 to n
    $p \leftarrow p * n$
    $j \leftarrow p$
    while $j \geq n$
        $j \leftarrow \frac{j}{2}$
```

## Problem 3 [6 marks]

Design an algorithm that takes as an input array $A$ of size $n$ and an integer $0 < k < n$. $A$ stores distinct real numbers. Your algorithm should put the $k$ smallest elements of $A$ in the beginning in sorted order. For example, if $A = \{5, 1, 2, 7, 9, 3, 6\}$ and $k = 3$, then $A = \{1, 2, 3, 5, 7, 9, 6\}$ is a valid output. Note that the first three elements of $A$ must be $1, 2, 3$, but the rest of elements in $A$ can be in arbitrary order. The worst case time complexity of your algorithm must be $O(n \log k)$ and your algorithm must use $O(k)$ additional space.

## Problem 4 [4+4=8 marks]

Suppose we have a set of 2D points in general position. Consider a variant of kd-tree where we always split points by a vertical line, i.e. based on the $x$ coordinate. No splits by horizontal line are allowed. As in the standard kd-tree, we always split points as equally as possible based on the median $x$ coordinate. The algorithm for search and range search is the same.

**a)** What is worst case time complexity of search for a single point? Express your complexity as a function of $n$, the number of points in the modified kd-tree.

**b)** What is worst case time complexity of range search? Express your complexity as a function of $n$, the number of points in the modified kd-tree and $s$, the number of items returned by the range search.

## Problem 5 [2+2+2=6 marks]

**a)** Apply Rabin-Karp algorithm with hash function $h(k) = k \bmod 7$, pattern $P = 22$, and text $T = 156492936$. List all the collisions. Collision is defined as usual in hashing: two unequal strings have the same hash code.

**b)** Compute KMP failure array for $P = cacacca$.

**c)** Compute Suffix Skip array for $P = abbnabbcabb$.

## Problem 6 [8 marks]

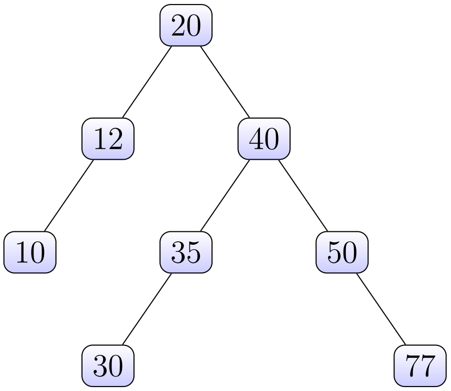
Design a data structure that stores KVP, and keys can be repeated. The data structure can perform the following operations:

- insert(KVP $x$): inserts a KVP $x$ into the data structure
- deleteMax(): removes and returns a KVP with the largest key from the data structure
- maxCount(): returns the number of KVP in the data structure that have key equal to the largest key in the data structure.

The worst case complexity of insert and deleteMax have to be $O(\log n)$, where $n$ is the number of KVP in the data structure. Worst case complexity of maxCount() has to be $O(1)$. The required space for your data structure must be $O(n)$.

## Problem 7 [2+2+2=6 marks]

Consider the following AVL-tree. Assume keys are non-repeating positive integers not larger than 100.



**a)** Give the largest key that can be inserted into the AVL tree above that results in no rotations. If no such key exists write “no key”

**b)** Give the smallest key that can be inserted into the AVL tree above that results in a double rotation (right or left). If no such key exists write “no key”

**c)** How many non-root nodes in the AVL tree above, when deleted, result in a rotation? List the keys of these nodes.

## Problem 8 [6 marks]

Consider a variation of a skip list which has fixed height $h = 3$ even though $n$ can become arbitrarily large. $S_0$ stores all the keys, $S_3$ stores only the sentinels. $S_1$ can store any subset of the keys in $S_0$, and $S_2$ can store any subset of the keys in $S_1$. The data structure is static, i.e. only searches are performed. Give the worst-case cost of searching in the skip list if the subset of keys in level $S_1$ and $S_2$ is chosen optimally. Here, ‘optimally’ means that the worst case time complexity of search is the best possible.

## Problem 9 [4 marks]

Suppose the alphabet is $\Sigma = \{a, b, c\}$. Give a string of length 10 composed for which LZW compression is the worst possible. State the entries added to the dictionary.

## Problem 10 [2+2+3=7 marks]

**a)** Computer the Burrows-Wheeler transform of the string $ALAKAZAM\$$ using the end-of-string marker $.

**b)** String $ENTT\$MRIEAXE$ is the result of a Burrows-Wheeler transform. Compute the original string.

**c)** Give an example of string of length 99 (length calculation does not include the symbol $) containing only characters ‘a’,‘b’, ‘c’, s.t. in Burrows-Wheeler transform of this sting, all letters ‘b’ are before all letters ‘c’, and letters ‘c’ are before all letters ‘a’.

## Problem 11 [2+2 = 4 marks]

**a)** The following message was compressed using single character encoding and transmitted together with its dictionary:

0010000111010101110001011010010

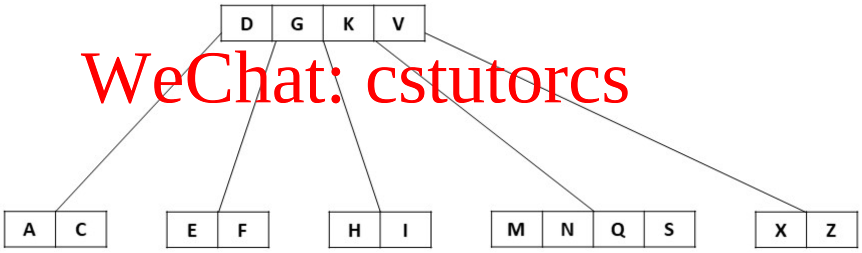
| | | |
|---|---|---|
| ’ ’ | = | 100 (blank space) |
| : | = | 1011 |
| d | = | 1010 |
| p | = | 001 |
| s | = | 000 |
| w | = | 011 |
| x | = | 010 |
| y | = | 11 |

Decode the message.

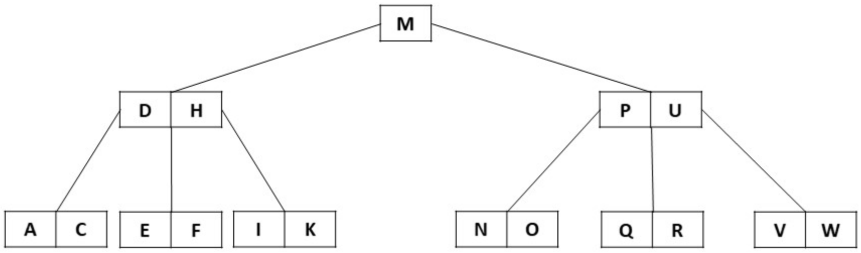
**b)** Explain why the encoding in part (a) was not obtained with the Huffman encoding.

## Problem 12 [2+3=5 marks]

**a)** Draw the result of inserting ‘R’ in the following B-tree of order 5. Show intermediate trees.



**b)** Draw the result of deleting ‘N’ from the following B-tree of order 5. Show intermediate trees.



## Problem 13 [3+4 = 7 marks]

When a mismatch occurs in the KMP algorithm at text position $i$ and pattern position $j$, neither the mismatched text character $T[i]$ nor pattern character $P[j]$ are considered when shifting the pattern. In this question, you will modify KMP to make use of $P[j]$ and $T[i]$ when shifting the pattern. You must not alter the fundamental principles of the KMP algorithm; i.e. when the pattern is shifted, the prefix of the pattern that matches with the text $T[k]$ for $k < i$ must not be rechecked.

**a)** Describe how to modify the KMP failure array and/or algorithm to make use of $P[j]$. State and briefly explain the runtime of the modified pre-processing step and algorithm if they are different from KMP.

**b)** Describe how to modify the KMP failure array and/or algorithm to make use of $T[i]$. State and briefly explain the runtime of the modified pre-processing step and algorithm if they are different from KMP. Consider the following two cases separately.

i) $T[i]$ is not a character that occurs in the pattern.

ii) $T[i]$ is a character that occurs in the pattern; i.e. when the pattern is shifted, the prefix of the pattern that matches the text $T[k]$ for $k \leq i$ must not be rechecked.

## Problem 14 [6+4 = 10 marks]

You are given a text $T$ of length $n$ and an array $P$ of patterns of size $\ell$. Each $P[i]$, for $0 \leq i < \ell$, is a pattern, i.e. an array of characters, where:

- the length of $P[k]$ is $(k+1)m$ for $k \geq 0$
- $P[k-1]$ is a prefix of $P[k]$ for all $k > 0$

You may assume $m$ is much smaller than $n$.

**a)** State how to modify the Rabin-Karp algorithm so that it returns $(i, k)$ where $k$ is maximal such that $P[k]$ is found in $T$; i.e. $P[k+1]$ is not found in $T$. The returned $i$ is the position in the text where pattern $P[k]$ is found. Your algorithm may use $O(m\ell)$ auxiliary space.

**b)** Suppose that $P[k]$ is the largest pattern found in $T$; i.e. $k$ should be left arbitrary. Analyze the algorithm from part (a) and give the best case expected runtime and worst case runtime. Explain how you derived the runtimes and show your work.
Note: The possible arguments for the runtime function include: $n$, $m$, $\ell$ and $k$. Depending on your algorithm, they may not all be required.