University of Waterloo
CS240 Fall 2021
Assignment 4

Due Date: Wednesday, Nov 17 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 November 17th** along with your answers to the assignment; i.e. **read, sign and submit A04-AID.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/f21/guidelines.pdf` for guidelines on submission. **Each question must be submitted individually to MarkUs as a PDF** with the corresponding file names: a4q1.pdf, a4q2.pdf, ... , a4q5.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. To accommodate any small time differences with our submission server, there is a grace period of 5 minutes; i.e. we will accept assignments until 5:05pm without penalty. Assignments submitted after 5:05pm will incur a penalty of 1 mark per minute to a maximum of 10 marks.
Assignments submitted after 5:15pm will not be accepted but may be reviewed (by request) for feedback purposes only.

Problem 1 [5+5+2=12 marks]

Suppose you want to sort a set of n strings over an alphabet of size d. Furthermore, suppose that the maximum-length string in your set is of length $\ell_{\max}$, and that the average length of the strings in your set is ℓ_{avg}. We assume that our sorted set will follow the standard alphabetical ordering; for example, `a` $<$ `ab` $<$ `b`.

a) One method we saw in class for accomplishing this task is *radix sort*. First, we pad all strings of length less than $\ell_{\max}$ with a suffix of end-of-word characters so that all

strings have length $\ell_{\max}$, then we run radix-sort on our set of strings as usual. Using the parameters defined above, describe the runtime and the amount of space used by this approach.

b) Another method to accomplish this task makes use of uncompressed *multiway tries*. First, we build a multiway trie containing every string in our set. Each node contains an array of children pointers of size $d + 1$. Then, we perform a preorder traversal and read each string, giving us a sorted ordering. Using the parameters defined above, describe the runtime and the amount of space used by this approach. A big-O bound is acceptable.

Hint. Assume the trie has k nodes, and try to express the runtime and space using k along with the other parameters. Then try to bound k in terms of the other parameters.

c) For which situations would the radix sort approach be preferable? For which situations would the multiway trie approach be preferable?

Problem 2 [2+5=7 marks]

a) Assume that we have a hash table of size 6 and that our keys are selected uniformly at random from the set $A = f\{1, 2, 3, \ldots, 600\}$. Consider the following two hash functions:

$$h_0(k) = k \mod 6$$
$$h_1(k) = 2k \mod 6$$

Is one hash function preferred over the other? Justify your answer.

b) Given an array A of size n storing integers and a number m, give an algorithm to determine whether array A contains a subarray whose elements sum to m. For example, the array $A = [5, 7, 9, 11, 13, 15]$ has a subarray, at consecutive indices 1 and 2, where elements sum to $m = 16$. Your algorithm must have an expected runtime of $O(n)$.

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

Consider a hash table dictionary with table of size $M = 10$ and hash function $h_1(k) = k \bmod 10$. Given the following insertion sequence: $4371, 1323, 6173, 4199, 4344, 1679, 1989$, draw the resulting hash table if we resolve collisions using the following methods:

a) Separate chaining

b) Linear probing

c) Double hashing with a second hash function of $h_2(k) = \lfloor k/1000 \rfloor$.

d) Cuckoo hashing with a second hash function of $h_2(k) = \lfloor k/1000 \rfloor$.

Problem 4 Quadtrees [2+3+5=10 marks]

For all parts of this question, use the convention that each internal node of a quadtree has exactly four children, corresponding to regions NE, NW, SW and SE, in that order. Also, as stated in Module 8 Slide 6, the bounding box must be the smallest box containing all points where the width and height are a power of 2.

a) Give three 2-dimensional points such that the corresponding quadtree has height at least 50. Give the (x, y) coordinates of the three points and show the shape of the quadtree; i.e. you do not need to show every level in the quadtree but show enough so the missing levels can be inferred. (Do not give the plane partitions.)

b) One application of quadtrees is image compression. An image (picture) is recursively divided into quadrants until the entire quadrant is only one colour. Using this rule, draw the quadtree of the following image. There are only three colours (shades of grey). For the leaves of the quad tree, use 1 to denote the lightest shade, 2 for the middle shade and 3 for the darkest shade of grey.

c) Another application is to compare two images. Given two black and white images (i.e. each pixel of the image is either 0 or 1) each of size $2^k \times 2^k$ stored as quadtrees, give an algorithm for the Intersection operation: if corresponding pixels in both images is 1, then their intersection is 1; otherwise 0.

Runtime analysis is not required but your algorithm should be as efficient as possible; i.e. marks may be deducted for terribly inefficient implementations.

Problem 5 [5+5=10 marks]

a) Draw the kd-tree representing the set of 2D points

$S = \{p_1, ..., p_8\} = \{(1,1), (2,2), (3,3), (4,4), (5,5), (6,6), (7,7), (8,8)\}$.

b) Give an algorithm using pseudocode for finding a point with the smallest x-coordinate in a kd-tree storing 2D points. Your algorithm should be as efficient as possible. State and explain the worst case running time of your algorithm.