University of Waterloo
CS240, Winter 2023
Assignment 4

Due Date: Wednesday, March 22, at 5pm

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:

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.

Problem 1 Hashing Examples [3+3+3+3+1+1=14 marks]

Consider a hash table dictionary with table of size $M = 10$. Suppose items with keys $[471, 123, 773, 499, 444, 679, 189]$ are inserted in that order using hash function $h_0(k) = k \pmod{10}$. Draw the resulting hash table if we resolve collisions using

Erratum: Corrected the hash function definition. **Clarification:** Corrected the total number of grades available for the question (no change to grades available for individual subquestions).

a) Chaining

b) Linear probing

c) Double hashing with the secondary hash function $h_1(x) = \lfloor \frac{x}{100} \rfloor$.

d) Cuckoo hashing with the second hash function $h_1(x) = \lfloor \frac{x}{100} \rfloor \pmod{5}$. (You will need to draw two tables.)

Clarification: Both tables are size 10.

e) Identify a serious problem with the choice of h_1 for part c).

f) Identify a different serious problem with the choice of h_1 for part d).

Problem 2 Chaining [1+2+1+3+4=11 marks]

Suppose we have a hash table with chaining of size M and insert n items in the hash table. Assume uniform hashing.

a) What is the probability that the first and the second items we insert end up in the first bucket? You do not need to justify your answer.

b) What is the probability that all n items end up in the same bucket?

c) What is the probability that at least one bucket contains 2 or more items if $n > M$?

d) What is the probability that at least one bucket contains 2 or more items if $n = M$, for a fixed constant M?

e) Now consider what happens as $n = M$ goes to infinity. Let c_n be the expected number of empty buckets. Prove that $\lim_{n\to\infty} c_n/n = 1/e$. You may use the fact that $\lim_{n\to\infty}(1 - 1/n)^n = 1/e$ in your solution. (Hint: Define an indicator variable I_i that indicates whether slot is i is empty and find the expected value of each I_i.)

Problem 3 Nottingham Hashing [4+2+2+2+0=10 (+3 bonus) marks]

In this question, we consider a probing variant of called Nottingham hashing. Assume uniform hashing.

Recall that in a standard probing table, the slot for key k is determined by probing with some function $h(k, i) = (h_0(k) + i \cdot f(k)) \bmod M$. (In linear probing, $f(k) = 1$, while in double hashing, $f(k) = h_1(k)$.) Define the *probe distance* of k to be the number of times k had to be probed to reach its slot, i.e., if it is stored in slot $h(k, d)$, then the probe distance is d.

In Nottingham hashing, each slot stores the probe distance of the key stored in that slot, in addition to the key and value. When inserting a new key k_A into the table, if $h(k_A, d_A)$ already contains k_B with probe distance d_B, we compare d_A and d_B to decide which key to continue probing with:

- If $d_A > d_B$, then we put k_A in the slot, kicking k_B out. Probing continues with k_B, trying slot $h(k_B, d_B + 1)$ next.
- If $d_A \leq d_B$, then we continue probing with k_A, trying slot $h(k_A, d_A + 1)$ next.

In other words, we always continue probing with the key that has the smaller probe distance, breaking ties in favour of leaving a key where it is.

In a linear probing Nottingham table, after deleting an element in slot i, we move subsequent elements back one slot each until we encounter an element that's already at probe distance 0. Doing so reduces the probe distance of all moved elements by one.

a) Consider a linear probing Nottingham hash table with size $M = 9$ and hash function $h_0(k) = k \bmod 9$. Draw the table after inserting the values $[52, 37, 17, 50, 62, 79, 21]$ (in that order) and again after deleting the values $[17, 50]$ (in that order). Be sure to include the probe distances of the keys in the table.

Erratum: A previous version of this question had an incorrect hash function.

b) Outline a proof that the expected probe distance in a Nottingham hash table is the same as the expected probe distance in a standard probing table using the same probing function $h(k, i)$. (You do not need to specify what the expected distance is, only that they are equal. Your proof outline does not need to be very detailed, but should capture the general idea of a proof.)

Clarification: You may assume that no elements have been deleted.

c) Prove that, when searching for a key k in a Nottingham table T, if we look in slot $h(k, i)$ and find a key k' with probe distance $j < i$, then we can stop searching and return "not found" immediately provided that T has never had any elements deleted from it.

d) Prove that part c) holds in a linear probing Nottingham table even when elements are deleted.

e) **Bonus:** Suppose a Nottingham hash table T contains n keys and satisfies an additional condition: For any two keys k_A, k_B stored at probe distances d_A, d_B respectively, we have that for all $i_A < d_A$ and $i_B < d_B$, $h(k_A, i_A) \neq h(k_B, i_B)$, where h is our hash function. In other words, for any 2 distinct elements in T, their probe sequences are disjoint.

Prove that the worst-case runtime to search for a key k in such a table is $O(\log n)$ in the worst case. You may assume that elements have never been deleted from the table.

Problem 4 Quadtrees [5+5+5=15 marks]

In this problem, we allow a quadtree to store arbitrary points, that is, points that are not necessarily in general position.

a) Draw the quadtree corresponding to the following set of 2D points:

$$S = \{(1, 1), (3, 1), (1, 3), (3, 3), (5, 2), (5, 5), (7, 7)\}$$

b) Let k be a positive integer and let S be the set of all integer coordinates (i, j) where $0 \leq i < 2^k$ and $0 \leq j < 2^k$, i.e.

$$S = \{(i, j) \in \mathbb{Z} \times \mathbb{Z} \mid 0 \leq i < 2^k,\ 0 \leq j < 2^k\}$$

What is the height h of the quadtree corresponding to S? Express the height as a function of n, where n is the number of points in S. Your expression should be exact, not asymptotic.

c) Consider the following modification to the quadtree data structure: each leaf is allowed to store up to four points. This means that any region containing 4 or less points is not split any further and is represented as a leaf.

i) Does the height of the quadtree representing S from part (b) change?

ii) Let S now be an arbitrary set of n points. Let h be the height of the regular quadtree on S and h' be the height of the modified quadtree on S. How large can the difference between h and h' be?

Problem 5 Range Trees [5+5+5=15 marks]

a) Draw a 2-dimensional range tree corresponding to the following set of points:

$$S = [[3, 5], [9, 1], [8, 8], [4, 6], [6, 0], [1, 3], [2, 7]]$$

Specifically, draw the primary tree T and all associate trees $T(v)$. You should have a total of 8 trees.

Erratum: As range trees are not unique, when constructing the primary and associate trees, do so by treating them as AVL-trees and inserting points in the order they appear above.

b) Assume that we have a set of n numbers (not necessarily integers) and we are interested only in the number of points that lie in the searched range R, rather than reporting all of them. Describe how a 1-dimensional range tree (i.e. a balanced binary search tree) can be modified to support such a `CountInRange` operation in $O(\log n)$ worst-case time, (independent of s, the number of points in R).

c) Now consider the 2-dimensional case: We have a set S of n 2-dimensional points. Given a query rectangle R, we want to find the number of points that lie in R. Show how you can build a data structure containing the points of S to support `CountInRectangle` operations in $O((\log n)^2)$ worst-case time.