University of Waterloo
CS240 - Spring 2020
Assignment 4

Due Date: Wednesday July 22, 5pm

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 22** along with your answers to the assignment – i.e. **read, sign and submit A04-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 refer to the course webpage for guidelines on submission. Submit your written solutions electronically as a PDF with file name a4Submit.pdf using MarkUs. We will also accept individual question files named a4q1.pdf, a4q2.pdf, ... if you wish to submit questions as you complete them.

Problem 1 [6 marks]

Consider a set $X = \{ x_1, \ldots, x_n \}$ where every x_i is a positive integer and $x_i \le n^f$ for some constant f. We represent x_i as strings over an alphabet $0, 1, \ldots, 2^t - 1$ (i.e., representing each x_i in base 2^t) and store all x_i from X in a compressed trie T. For every node u of T we keep an array $A(u)$ of size $2^t + 1$. $A(u)[i]$ contains a pointer to the child u_i of u such that the edge from u to u_i is marked with i; $A(u)[i] =$ `NULL` if there is no such u_i in T.

Specify all of the following in big-O in terms of f, n, and t (as necessary). What is the maximum height of T? What is the total space usage of T and all $A(u)$? What time is needed to search for a key k in T ?

Problem 2 [2+2+2+2+3+4 marks]

(a) Draw the standard trie that is obtained after inserting the following keys into an initially empty trie:

$$1011\$, 001\$, 1101\$, 10010\$, 10\$, 11\$, 10100\$, 1\$, 000\$, 101\$, 00\$$$

(b) From your answer to part (a), draw the trie that is obtained after removing the following keys:

$$10100\$, 11\$, 1011\$$$

(c) Repeat part (a), except use a compressed trie.

(d) Repeat part (b), but on the compressed tree that is obtained at (c).

(e) Find the exact height of the trie with keys that are binary representations of numbers $0, 1, 2, 3, 4, \ldots, 2^k - 1$ without leading 0s, and inserted into the trie in increasing order. That is, insert $0\$, 1\$, 10\$, 11\$, 100\$$, etc. Justify your answer.

(f) Repeat part (e) using compressed tries. Also, for this part, prove your result, including any structural properties of the compressed tries, using mathematical induction.

Note that in order to obtain full marks for proofs involving mathematical induction, solutions must clearly state:

- what you are doing induction on;
- the base case(s);
- the inductive hypothesis;
- the inductive step.

Problem 3 [3+3+3+3+1 marks]

Consider a hash table dictionary with universe $U = \{0, 1, 2, \ldots, 25\}$ and size $M = 5$. If items with keys $k = 17, 10, 20, 13$ are inserted in that order, draw the resulting hash table if we resolve collisions using:

(a) Chaining with $h(k) = (k + 2) \bmod 5$.

(b) Linear probing with $h(k) = (k + 2) \bmod 5$

(c) Double hashing with $h_1(k) = (k + 2) \bmod 5$ and $h_2(k) = 1 + (k \bmod 4)$.

(d) Cuckoo hashing with $h_1(k) = (k + 2) \bmod 5$ and $h_2(k) = \lfloor k/5 \rfloor$.

(e) Identify a serious problem with the choice of hash functions in part (d).

Problem 4 [3+4 marks]

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

$$h_1(k) = k \quad \bmod 6,$$

$$h_2(k) = 2k \quad \bmod 6.$$

Which hash function is better? Justify your answer.

(b) Let S be a set of n keys mapped to a hash table also of size n using chaining. We make the uniform hashing assumption, and we call c_n the expected number of empty slots. Prove that $\lim_{n\to\infty} c_n/n = 1/e$, where $e \approx 2.71828183$.

You can use the fact that $\lim_{n\to\infty}(1-1/n)^n = 1/e$. We recommend you use indicator variables $I_0, \ldots, I_{n-1}$, that take values 0 or 1, depending on whether the corresponding entry in the table is empty or not; then, find the expected value of each I_i.

Problem 5 [4+4+4 marks]

In this question, we consider a modified version of open-addressing hashing. In this modified version, the insert operation works as follows. To insert a key k, we perform linear probing until we either reach an empty position in the hash table or probe a position that is not empty. Say the nonempty position in the table contains the key k'. Then, if $k' > k$, we replace k' with k at that position and call the insert operation on k'. If $k' \leq k$, continue trying to insert k in the position after k', as is done in normal linear probing. In this way, the value of the key we are trying to insert into the hash table is strictly increasing.

For each question that follows, we may assume that no key is ever deleted from the hash table.

(a) For a hash table of size $M = 10$ and the hash function $h(x) = x \bmod 10$, run this modified hash algorithm using the insertion sequence $\{31, 26, 16, 23, 11, 30, 20\}$. Show the hash table after each insertion is complete; that is, after no more probing is required.

(b) Argue that, regardless of the values of M and $h(x)$, the insertion operation will always terminate after a finite number of probes.

You may assume that there is still space in the hash table upon each insertion and that all probe sequences consist of some permutation of the positions of the hash table.

(c) Argue that the number of probes made during an insert could be $\Omega(n^2)$. That is, for arbitrarily large values of n, give an example of a hash table with n keys and size $M > n$ such that insertion of a key into the hash table will require at least cn^2 probes for some constant $c > 0$. Justify insertions.

The most straightforward approach would be to use linear probing with the hash function $h(x) = x \bmod M$, but you can choose to use another hash function if you want (as long as it can map to all entries of the table).