# University of Waterloo
# CS240E, Winter 2022
# Assignment 4

**Due Date: Wednesday, March 16, 2022 at 5pm**

Be sure to read the assignment guidelines (`http://www.student.cs.uwaterloo.ca/~cs240e/w22/guidelines/guidelines.pdf`). Submit your solutions electronically as individual PDF files named a4q1.pdf, a4q2.pdf, ... (one per question).

## Question 0 Academic Integrity Declaration

Read, sign and submit A04-AID.txt now or as soon as possible.

## Question 1 [3 marks]

Let $X_i$ be the length of the bucket $T[i]$ in hashing with chaining. This is a random variable (we assume that the hash-function was chosen randomly and uniformly among all possible hash-functions). What is the variance of $X_i$? Give an exact bound (no asymptotic notation) that depends on $n$ and $M$. Justify your answer.

## Question 2 [1+2+2+5=10 marks]

Assume we have a hash function $h$ for some table-size $M \geq 2$, and define a probe sequence as follows:

$$\begin{aligned} h(k,0) &= h(k) \\ h(k,i) &= h(k,i-1) + i \bmod M \quad \text{for } 1 \leq i < M \end{aligned}$$

**a)** Write the probe sequence for $h(k) = 0$ and $M = 8$ starting from $i = 0$ to $i = M - 1$.

**b)** Show that this probe sequence is an instance of quadratic probing.

**c)** Show that if $h(k,i)$=$h(k,j)$ for some $0 \leq i < j < M$, then $(j-i)(j+i+1)$=0 mod $2M$.

**d)** Assume that $M$ is a power of 2, say $M = 2^m$ for some integer $m$. Prove that all entries in the probe sequence are different, therefore the probe sequence will hit an empty slot.

## Question 3 [2+4+5+2 = 13 marks]

We have seen one method of obtaining a universal family of hash-functions in class. This assignment discusses another one. Let us assume that all keys come from some universe $\{0, \ldots, U-1\}$, where $U = 2^u$. Therefore any key $k$ can be viewed as bitstring $x_k$ of length $u$ by taking its base-2 representation.

Let us assume further that the hash-table-size $M$ is $M = 2^m$ for some integer $m$, with $m < u$. To choose a hash-function, we now randomly choose each entry in a $m \times u$-matrix $H$ to be 0 or 1 (equally likely). Then compute $h_k = (Hx_k)\%2$, where $x_k$ is now viewed as a vector and '%2' is applied to each entry. The output is a $m$-dimensional vector with entries in $\{0, 1\}$; interpreting it as a length-$m$ bitstring gives a number $\{0, ..., M-1\}$ that we use as hash-value $h(k)$. For example, if $k = 18$, $u = 5$, $m = 3$ and $H$ is as shown below, then $h(k) = 1$ since

$$\underbrace{\begin{pmatrix} 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}}_{H} \quad \underbrace{\begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}}_{\text{18 as length-5 bitstring}} \quad \%2 = \underbrace{\begin{pmatrix} 0 \\ 2 \\ 1 \end{pmatrix}}_{Hx_k} \%2 = \quad \underbrace{\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}}_{\text{1 as length-3 bitstring}}$$

**a)** Let $H$ be the above matrix, $u = 5$ and $m = 3$. Consider the keys 9 and 13. What are their hash-values? Show your work.

**b)** Consider again $u = 5, m = 3$ and keys $k = 9$ and $k' = 13$. Consider the same matrix $H$, except that the bits in the third column are randomly chosen. What is the probability that $h(k) = h(k')$? Justify your answer.

**c)** Show that (for any $u, m$) this method of choosing the hash function gives a universal hash function family, or in other words, $P(h(k) = h(k')) \leq \frac{1}{M}$ for any two keys $k \neq k'$.

**d)** This method for obtaining universal hash-functions is much less popular than using the Carter-Wegman functions. Why do you think that that might be the case? (Expected length of answer is 1-3 sentences.)

## Question 4 [4+3+2+3=12 marks]

Let $P$ be a set of $n$ points in general position. A *2-dimensional partial match query* species a value $a$, and asks whether there are any points in $P$ that have either $x$-coordinate $a$ or $y$-coordinate $a$ (or both).

**a)** Assume $P$ is stored in a 2-dimensional kd-tree. Design an algorithm that can answer a partial match query in $O(\sqrt{n})$ time.

**b)** Argue that any comparison-based algorithm to do partial matches must use $\Omega(\log n)$ comparisons on some instance of size $n$.

**c)** Assume $P$ is stored in a 2-dimensional range-tree. Design an algorithm to answer a partial match query. Make it as efficient as you can. It suffices to describe the idea and analyze the run-time.

**d)** Design a data structure to store $P$ that uses $O(n)$ space and permits to insert points, delete points, and answer 2-dimensional partial match queries in $O(\log n)$ worst-case time. Briefly say how these operations are implemented.

## Question 5 [5+5+2=12 marks]

A *range-counting-query* is like a range search, except that you only need to report *how many* items fall into the range, you do not need to list which items they are.

**a)** Describe how any balanced binary search tree can be modified such that a range counting query can be performed in $O(\log n)$ time (independent of $s$, the number of points in the query-interval). Briefly state the changes needed, then describe the algorithm for the range counting query.

**b)** Now consider the 2-dimensional-case: Describe an appropriate range-tree based data structure such that you can answer range-counting-queries among 2-dimensional points in time $O((\log n)^2)$. Then describe the algorithm for the range counting query.

**c)** Assume now that the range-counting query is 3-sided. Which data structure for storing points would you use if your objective is a small run-time? Briefly (in 2-3 sentences) justify your answer.

## Question 6 Bonus [(+5) marks]

Assume you are given an array $P$ of $n$ points, where points are in general position and sorted by $x$-coordinates. Describe an algorithm that builds a priority search tree to store points $P$, and that has $O(n)$ *worst-case* time.

Clarification: The textbook suggests to use the median $x$-coordinate as split-line coordinate, but you are allowed to use any values for the split-line coordinates as long as the resulting priority search tree has height at most $\lceil \log n \rceil$.