THE OPEN UNIVERSITY OF SRI LANKA

FACULTY OF ENGINEERING TECHNOLOGY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

BACHELOR OF SOFTWARE ENGINEERING HONOURS

EEX4465 – Data Structures and Algorithms

Mini Project Deadline on 11 April 2023 2022/2023

Objective:

To design and implement a non-collision Hash table by using the Quadratic probing technique. In order to make this project success, you will need to implement the following steps:

(1) Read the specific text file to obtain the words in the English alphabet and compute the **Hash key** for each word and store them in a file.

(2) Compute the relevant parameters in the **Quadratic probing** and store them in another file.

(3) Construct the **non-collision Hash table** by using the Quadratic probing technique.

More information about **Step (1)** to **Step (3)** is examined as follows.

Step (1) – Read the specific text file to obtain ...

Choose the *input* file according to your registration number as shown in Table 1. Each text file contains multiple lines of words with English alphabet {a,... z, A,..., Z} and special characters (*numbers, commas, full-stops*, etc). Ignore all special characters and consider the words with English alphabet only (both *lower case* and *upper case*). A number is assigned for each character in English alphabet according to Table 2. You have to build two *output* files and a Hash table as indicate in Table 1 and they are described in the following sections.

Table 1: Relevant text files

First digit of your registration number	Names of the different files:			Construct the hash table for
	input file	*output* files		
n, where $n = 1, \ldots, 7$	file$n$.txt	wordsHKn.txt	wordsQHKn.txt	the file: file$n$.txt

Table 2: Assigned numbers for each character in English alphabet

a	b	c	... ...	x	y	z	A	B	C	... ...	X	Y	Z
0	1	2	... ...	23	24	25	26	27	28	... ...	49	50	51

Task 1

(a) Read each word and use Table 2 to compute the Hash key defined by Eq. (1).

$$k_j = \text{Sum of the numbers corresponding to the characters in } j\text{th word,} \tag{1}$$

where j = 1, 2, 3, ... is the index for the words in *input* file "file$n$.txt" and k_j is the Hash key for the jth word. Take only a single appearance of exact repeated words in the *input* file.

NOTE: "the" and "The" are two separate words, hence you need to consider both of them. From Table 2, the Hash keys for "the" and "The" are k = 30 and 56, respectively.

(b) Store each word (no need to sort them) in an *output* file with the name "wordsHKn.txt" as shown in Figure 1.

Word index j	Word	Hash key, k_j
⋮	⋮	⋮
12	the	30
13	A	26
⋮	⋮	⋮
63	a	0
⋮	⋮	⋮
212	The	56
213	ps	33
⋮	⋮	⋮

Figure 1: Sample format of an *output* file: wordsHKn.txt.

NOTE: The sample output shown in Figure 1 is provided only as a format example. The content of the first column is chosen arbitrarily. The table is not produced by any program and is not relevant to any of the seven *input* files. Hash keys are computed using Table 2.

Step (2) – Compute the relevant parameters in the Quadratic probing and . . .

Quadratic probing operates by taking the original hash index (k_j) and adding successive values of an arbitrary quadratic polynomial until an open slot is found. Here the probe function is according to the quadratic formula:

$$c_1\, i^2 + c_2\, i + c_3, \qquad i = 1, 2, 3, \dots$$

where $c_1 \neq 0$, c_2 and c_3 are some choice of constants. Then, the ith collision in the probe sequence for the jth word is given by the Quadratic hash function, $h(j, i)$ below:

$$h(j,i) = \underbrace{\left(k_j + c_1\, i^2 + c_2\, i + c_3\right)}_{\text{new Hash key, } Qk_j} \text{ MOD } size \tag{2}$$

where $Qk_j \equiv (k_j + c_1\, i^2 + c_2\, i + c_3)$ is the new Hash key with Quadratic probe for the jth word, $size$ is the table size and MOD is the modulo operation.

Task 2

(a) Choose a suitable table size for the above quadratic probing and explain why you select that number.

(b) Select $c_1 = c_2 = 1$ and $c_3 = 0$ in Eq. (2). Compute the new Hash key Qk_j and the Quadric hash function $h(j, i)$ for each word that you stored in the file: "wordsHKn.txt".

(c) Store the new values in two columns with the previous contents in another *output* file called “wordsQHKn.txt” as shown in Figure 2.

Word index j	Word	Hash key, k_j	new Hash key, Qk_j	Quadratic h-f, $h(j, i)$
⋮	⋮	⋮	⋮	⋮
12	the	30	⋮	⋮
13	A	26	⋮	⋮
⋮	⋮	⋮	⋮	⋮
63	a	0	⋮	⋮
⋮	⋮	⋮	⋮	⋮

Figure 2: Sample format of the second *output* file: wordsQHKn.txt.

Step (3) – Construct the non-collision hash table using Quadratic probing

By now, you have computed the relevant parameters that need to construct a non-collision hash table by using Quadratic probing for each word.

Task 3

(a) Construct the non-collision hash table from the file “wordsQHKn.txt” as shown below.

Non-collision hash table using Quadratic probing for the *input* file: “file$n$.txt”

Index	Words in “file$n$.txt”
0	⋮
1	⋮
2	The
3	frequency
⋮	⋮
⋮	Cameras
⋮	the
⋮	⋮

NOTE: The above hash table is a sample output and not given any correct result.

(b) Compute the Time complexity of your Quadratic probing algorithm.
(c) Explain briefly what the best-case and worst-case situations occur in your design.

~ End of steps.~

Work to submit *on or before the deadline:*

- **Report**

You must provide a project report that includes the following sections:

(a) A very brief overview design of the mini project.
(b) Print the results of the two *output* files and the non-collision hash table drawn in **Step (3)**.
(c) Answers to all relevant sub-parts: (a), (b) and (c) in each Task: **Task 1, Task 2** and **Task 3**.

- **Implementation**

(a) You should bring all source files with the two *output* files generated (and the executable file) when demonstrating your implementation to show the same results documented in your project report.
(b) Your implementation must be able to live running to get the two output files. Therefore, DO NOT hardcode to get the result of two *output* files.

Note: we will examine parts of your code to make sure that you are implementing the data structures as required. If you are getting the correct results but not implementing the proper data structure, your will get zero marks for that portion of implementation.

The viva dates will be scheduled after the deadline and it is a compulsory.

— End —