BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
CSE 208 (Data Structures and Algorithms II Sessional)
July 2023

Offline 7: Hashing
Deadline: February 12, 11:55 PM

---

1) Problem Specification

You need to implement a ***HashTable*** class for this assignment. The requirements are as follows:

- Data will be kept as (key, value) pairs in this hash table.
- You can ***insert***, ***delete***, and ***find*** data from this table.
- The minimum size *N'* of this table will be given as input, but you need to find the nearest prime *N* greater than or equal to *N'* as the actual size of this table.
- You need to use randomly generated words of lengths 5 to 10 as keys for this table. Therefore, you need to implement a random word generator. The lengths of the words will be randomly determined as well. The words do not have to be meaningful. Using only lowercase alphabetical letters will suffice. The order of the incoming words will be used as their value (please see the example in Section 2 below). If your word generator produces duplicate words, you must only keep one instance of each, discarding the others. Use your *find* function to ensure this.
- Your hash table can have different collision resolution methods (see details in Section 4). The collision resolution method of your hash table instance should be specifiable via the constructor.

2) Example

Suppose your word generator has generated the following words:

puzzle
universe
brain
universe
offline

The corresponding (key, value) pairs will be:

(puzzle, 1)
(universe, 2)
(brain, 3)
(offline, 4)

Note that the second instance of the word “universe” has been discarded, and the value for “offline” has been set to 4 instead of 5.

3) Hash Function

You have to use two standard hash functions (*Hash1(k)* and *Hash2(k)*) of your own, or from any good literature where you must try to avoid collisions as much as possible. We expect that 60% of the keys will have unique hash values (e.g., at least 60 unique hash values for 100 keys).

4) Collision Resolution

You need to implement the following three collision resolution methods:

1. **Separate chaining:** Place all the elements that hash to the same slot into a linked list. Slot *j* of the hash table contains a pointer to the head of a linked list which will store all the items that hash to the *j*-th slot. If there are no such elements, slot *j* contains *null*.

2. **Double hashing:** For double hashing, use the following hashing function:

   $$doubleHash(k, i) = (Hash(k) + i \times auxHash(k)) \% N$$

   Here, *Hash(k)* is one of the hash functions described in Section 3. Note that you have to use both the Hash functions mentioned in Section 3 for report generation (see Section 5 for more details). You can use a simple hash function as the auxiliary hash function *auxHash(k)*. The initial probe goes to position *Table[Hash(k)]*, and successive probe positions are offset from previous positions by an amount *auxHash(k)*, modulo *N*.

3. **Custom probing:** For custom probing, use the following hash function:

   $$customHash(k, i) = (Hash(k) + C_1 \times i \times auxHash(k) + C_2 \times i^2) \% N$$

Here *C1* and *C2* are two auxiliary constants of your choice. The other details are the same as the Double Hashing.

5) Rehashing

You need to implement rehashing for your chaining-based hash table. The rehashing criteria will be the **maximum chain length**.

At the start of the program, take the maximum allowed chain length, *C* as input (along with the minimum table size *N'*).

After every 100 insertions into your table, calculate the maximum chain length.
Whenever the maximum chain length of your hash table exceeds *C*, rehash to a new table with approximately twice the size.

Also, after every 100 deletions from your table, check whether the maximum chain length has fallen below *0.8×C*. If so, rehash to a new table with approximately half the size. But do not rehash if the new table size would fall below the initial size, *N*.

Every time a rehash is triggered in your program, you need to print the average probe count, load factor, and maximum chain length of your table just before and immediately after the rehash. To calculate the average probe count, you may search for 10% of the elements randomly.

6) Report Generation

Generate 10000 unique words and insert them into the Hash Table. Using both hash functions (*Hash1(k)* and *Hash2(k)*), list the number of collisions in a tabular format (see the table below). Among these 10000 generated words, randomly select 1000 words and search each of these selected words in the hash table. Report the average number of probes (i.e., the number of times you access the hash table) required to search these words. These results should be reported for both hash functions. The report should present the items in the following table. You also need to report the hash functions and the auxiliary hash functions that you have used.

To test rehashing, keep deleting the words afterward. You can stop deleting after you reach the lower size limit of your hash table.

Generate your report for *N'* = 5000, 10000, and 20000. If you cannot find any slot to insert a value into the hash table, you can simply ignore that insertion.

Hash Table Size	Collision Resolution Method	Hash1		Hash2	
		# of Collisions	Average Probes	# of Collisions	Average Probes
	Chaining				
	Double Hashing				
	Custom Probing				

Note that, you may be asked to run your program during evaluation to show that the output of your program closely resembles the reported values (use a fixed seed to be able to reproduce your results exactly).

7) Submission Guideline

1. Create a directory with your 7-digit roll number as the name
2. Put only the source files and the report in the mentioned directory
3. Zip the directory in ".zip" format, rename it using your 7-digit roll number, and submit it on Moodle within the mentioned deadline (February 12, 11:55 PM).
4. Copying will result in -100% marks for both source and destination.

Failure to follow the submission instructions may result in up to 10% penalty