BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CSE208 (Data Structures and Algorithms Sessional II)

Offline 8: Hash Table

---

1 Problem Specification

In this assignment, you have to implement a ***HashTable*** with the following requirements.

1. You have to implement insert, search, and delete operations on the hash table. The length **N** of the hash table will be given as an input from console.

2. The data will be kept as a (key, value) pair in the hash table. You need to insert randomly generated unique words of 7 character length (these words can be meaningful or meaningless) into the hash table. Therefore, you have to implement a random word generator. The words will be used as the "keys", and the order of the incoming words will be used as the "value" (please see the example below). If your word generator generates duplicate words, you have to keep only one instance of those words.

2 Example

Suppose your word generator has generated the following 5 words.

ancient
puzzled
benefit
ancient
zigzags

The corresponding key-value pairs will be:

(ancient, 1)
(puzzled, 2)
(benefit, 3)
(zigzags, 4)

You have to discard the second instance of the word "ancient" since it has already been included in the table.

3 Hash Function

You have to use 2 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 are expecting 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. **Chaining Method**
   Place all the elements that hash to the same slot into a linked list. Slot j contains a pointer to the head of the list of all stored elements that hash to j ; if there are no such elements, slot j contains $NULL$.

2. **Double Hashing**
   In case of Double hashing, use the hash function as follows.

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

   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. Here, $auxHash(k)$ is an auxiliary hash function. To keep things simple, you can use a simple hash function as the auxiliary hash function. 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**
   In case of Custom Probing, use a hash function of the form:

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

   Here, C_1 and C_2 are two auxiliary constants of your choice. The other details are same as the Double Hashing.

5 Report Generation

Generate 10000 seven character long unique words and insert them into the Hash Table. Mention the number of collisions in a tabular format using both the hash functions (e.g., Hash1 and Hash2). 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., number of times you access the hash table) required to search the words. Please report these results for both the Hash functions (Hash1 and Hash2). The report should be generated in the following tabular format.

Collision Resolution Method	Hash1		Hash2	
	No. of Collisions	Avg. Probes	No. of Collisions	Avg. Probes
Chaining Method				
Double Hashing				
Custom Probing				

Table 1: Performance of various techniques for collision resolution with two different hash functions.

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 (it is okay if they do not match exactly).

6 Submission Guideline

1. Create a directory with your 7 digit student no. as its name.
2. Put the source files and the report into the directory created in step 1.
3. Zip the directory (compress in .zip format; .rar, .7z or any other format is not acceptable).
4. Upload the .zip file on Moodle.

For example, if your student no. is 1705xxx, create a directory named 1705xxx. Put only your source files(.c, .cpp, .java, .h, etc.) and the report document (.doc/.docx/.pdf or any other suitable format) into 1705xxx. Compress 1705xxx into 1705xxx.zip and upload the 1705xxx.zip on Moodle.

Failure to follow the above-mentioned instructions may result in upto 10% penalty.

7 Submission Deadline

10 PM, November 20, 2020 (Friday)