ECE 250 - Project 2
Phone Directory as Hash Table
Design Document
Ayushi Tiwari, UW UserID: attiwari
Feb 14th, 2020

1. Overview Of Classes

Method 1: Double Hashing Hash Table

Class directory

Description: Contains the variables for phone number and caller ID, that is stored the hash table vector.

Member Variables:

1. number (data type: long) — *stores the phone number*
2. caller (data type: string) — *stores the caller ID*

Class hashTable

Description: Represents a vector of type directory to store the data and provides operations as inserting phone numbers and caller IDs, deleting the records, searching, etc.

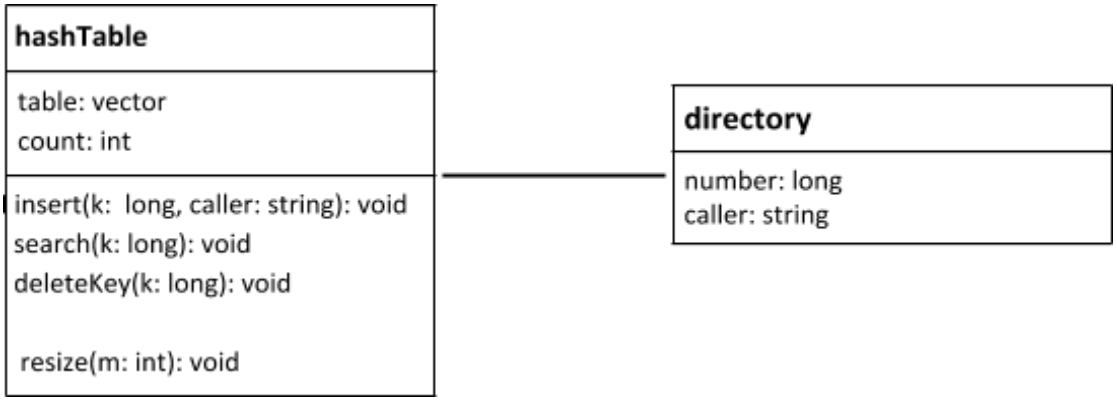
Member Variables:

1. table (data type: vector) — *stores the data*
2. count (data type: int) — *to count the number of entries*

Member Functions :

1. insert: inserts the number and caller ID provided as input at the index using double hashing.
2. search: searches for the phone number passed as parameter in the vector and prints the caller ID and index if present.
3. deleteKey: deletes the key and the information related to it if it exists in the table.
4. resize: if the n command is passed again, it resizes the table with the new capacity passed as parameter and deleted the old data.

UML Class Diagram



Constructors/Destructors

Class directory (Constructor & Destructor):
The constructor and destructor for this class assigns the number and caller with the value 0 and empty string.
Class HashTable (Constructor):
The constructor takes in the size and reserves space in memory for that size.
Class HashTable (Destructor):
The destructor cleared all the elements of the table and the memory.

Method 2: Chaining Hash Table
Class Node
*Description***:** Represents a node of a list stored in the hash function containing number, caller ID and pointer to next node.
Member Variables:

1. numbers (data type: long) *stores the phone number*
2. caller (data type: string) *stores the caller ID*
3. Next (data type: pointer) *pointer to the next node*

Member Functions:
Doesn't have any member functions because it has HashTable as its friend class so the private member variables can be directly accessed and used inside the HashTable class.
Class List
Description: This class has a pointer to add the data.
Member Variables:

1. point(data type: pointer)

Member Functions:
Doesn't have any member functions because it has HashTable as its friend class so the private member variables can be directly accessed and used inside the HashTable class.
Class hashTable
Exactly like in method 1.
Member Functions:

1. *print:* it sorts the linked list and prints all the numbers in the linked list at the position

Constructors/Destructors

Class Node (Constructor):
The constructor for this class assigns the number, caller and next pointer with null values
Class Node (Destructor):
The destructor makes the number and caller 0 and empty and also makes the next pointer null.
Class hashTable (Constructor and Destructor):
Same as method 1

Test Cases

Test 1(Double Hashing): Create a new table, insert data to check if they are added at the right place or not, delete them, search for deleted items, insert new ones.

Test 2(Chaining): Create a few nodes in the table and then delete some, print them to check if they are ordered or not, search for a number and create a new table and insert new nodes.

Example 1

n 100
i 2269786224;Ayushi
i 9999286224;Beta
p 2
d 6313718331
p 1
s 7901308491
s 7484823928
i 5623740242;D
p 4
p 2
n 53
i 123456789;Alpha
i 222222222;Beta
s 5196750336
d 8327492181
n 45

Performance

Time to compute the hash function: O(1)
Search function: O(1)
Insertion: O(1)
Deletion: O(1)