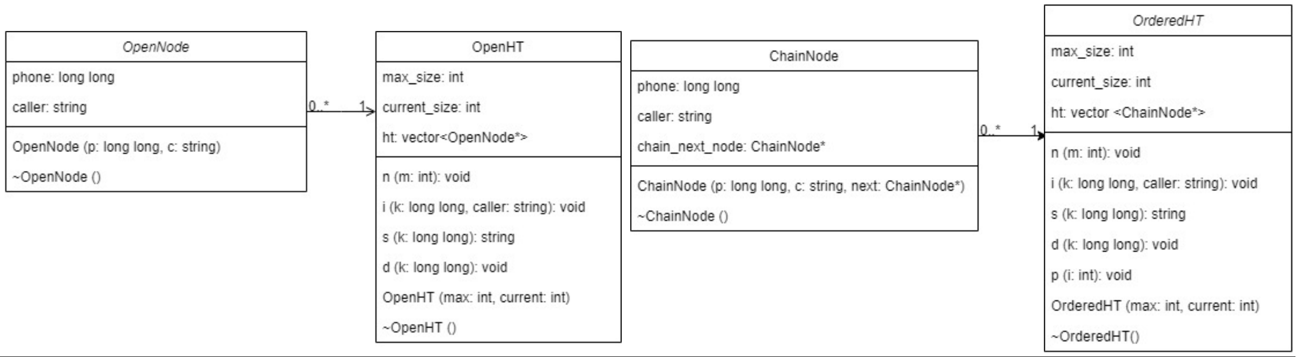
ECE250 – Project 2
Hashing
Design Document

Qinying Wu, q227wu
Feb 14th, 2020

1. <u>Overview of Classes</u>



Class Name	OpenNode	ChainNode	OpenHT	OrderedHT
Description	Objects stored in the double hashing table	Objects stored in the chaining hash table	The double hashing table	The chaining table
Member Variables	<u>phone: long long</u> stores the phone number (key) <u>caller: string</u> stores the caller name		<u>max_size: int</u> stores the maximum size of the table <u>current_size: int</u> is the current number of keys stored in the table	
		<u>chain_next_node: ChainNode*</u> is the pointer to the next node in the chain of the hash table slot	<u>ht: vector<OpenNode*></u> a vector representing the hash table	<u>ht: vector<ChainNode*></u> a vector representing the hash table
Member Functions	No member functions		<u>n (m: int): void</u> clears and reinitializes the hash table to size m, outputs "success" afterwards	
			<u>i (k: long long, caller: string): void</u> insert key "k" with caller into the hash table, outputs "success" if inserted, else output "failure"	
			<u>s (k: long long): string</u> searches for key "k" in the hash table. If found, it returns the string of key's position and the caller concatenated by a ';', otherwise it returns the string "-1"	
			<u>d (k: long long): void</u> deletes the key "k" from the table by setting the phone number to the constant "-1" and the caller to an empty string. It outputs "success" if deleted, else output "failure." If al nodes on the table are deleted, the entire hash table is cleared and resize to eliminate the "-1" erased status indicator.	
				<u>p (i: int): void</u> prints all the keys in the chain stored at position i of the hash table

Constants for the Open Addressing table:
ERASED_KEY: int =-1 is assigned to the phone number of the key being erased from a node
ERASED_CALLER: string="" is assigned to the caller of the key being erased from a node

2. <u>Constructors and Destructor Decisions</u>

Class Name	OpenNode	ChainNode	OpenHT	OrderedHT
Constructor	Initializing the phone and caller information by passing in a long long typed parameter "p" and string parameter "c"		The vector is constructed automatically using its default constructor. Therefore, the constructor is only initializing the max_size and the current_size by passing in an integer parameter "max" and "current", respectively	
		Initializing the chain_next_node by passing in a ChainNode pointer parameter "next" (usually nullptr)		

Destructor	Empty since no need to deallocate memory	deallocate the chain_next_node pointer to nullptr	Clears the vector ht that represents the hash table and resize it to 0. Finally destruct the vector.

3. **Asymptotic Upper Bounds (*Assume uniform hashing for all functions)**

Class Name	**OpenHT**	**OrderedHT**
Search Function (s)	Constant time since it is only consisted of the time to find the slot in the table that which the key should belong to using the primary function. The secondary function is only used when the key stored in that slot does not match the given key (and used repeatedly for future mismatches until it reached the same key again). Other than that, the time to search through the table does not depend on any other factors.	Constant time since it is only consisted of the time to find the slot in the table that which the key should belong to using the primary function. Other than that, the time to search through the table does not depend on any other factors.
Insert function (i)	The insertion time is constant. The time for insertion is consisted of the time to search through the table for duplicates and the time to find the slot for insertion for both methods. The time to search is constant as explained above. The time for insertion without collision is consisted of calculating the slot position to be inserted using the primary function.	
	For open addressing, if a collision occurred, the secondary function will be used to obtain a new position on the table. And it is used repeatedly until an empty slot is found. Insertion is discarded upon initial conditioning to determine whether the table is full, which is also constant time because it only needs to compare the max_size and current_size.	For separate chaining, if a collision occurred, the key will be inserted into the ascending-ordered-chain linking to that collision node. This is still considered constant time because the table is uniformly distributed, which means each node holds an average count of current_size/max_size nodes. Insertion is discarded if the table is full (by determining if max_size==current_size)
Delete function (d)	Delete time is constant. It is consisted of the time to search the table to determine whether the key exists in the table and the time to erase the key from the table. The time to search is constant as explained above. The time to erase one key is constant because there is no iteration involved.	

4. **Test Cases**

a) Insert a key
   a. Insert a unique key and after the calculation to determine the slot for insertion …
      i. the primary function returns an already taken slot on the table
         (for open addressing only)
         1. the secondary function returns an already taken slot on the table
         2. the secondary function returns an empty slot on the table
      ii.
      iii. the primary function returns an empty slot on the table
   b. Insert a duplicated key
      i. Same key, same caller
      ii. Same key, different caller
   c. Insert a key “k”, delete it, and insert it back again
   d. When the table is full (for open addressing only)

b) Search a key
   a. An existing key on the table
   b. A non-existing key on the table
      i. The key belonged to the table before
      ii. The key was never in the table before
   c. After resizing the table
   d. After deleting the last object in the table

c) Delete a key
   a. Before and/or after insertion
   b. After deleting the same key
   c. After deleting a different key
   d. Delete an existing key
   e. Delete a non-existing key
   f. After resizing the table

d) Resize the table

    a. After inserting some nodes into the table
    b. After deleting some nodes into the table
    c. When the table contains 0 objects
    d. When the table is full

e) Print (for separate chaining only)
    a. An empty slot on the table
    b. A slot with only one node
    c. A slot with a chain of nodes
        i. Before/after inserting some new nodes and/or duplicated nodes
        ii. Before/after deleting some existing nodes and/or non-existing nodes
    d. A non-existing slot on the table (a slot out of bound)
    e. After resizing the table

Examples:

n 13	n 2	n 8
i 5190000000;name1	i 5190000000;name1	i 1234;a
i 5190000001;name2	i 5190000001;name2	i 5678;b
i 5190000013;name3	i 5190000002;name3	i 1212;c
i 5190000000;name1	i 5190000003;name4	s 1234
i 5190000000;name2	p 0	s 456
d 5190000000	p 1	s 5678
i 5190000000;name2	p 2	s 1212
s 5190000000	s 5192	d 1234
n 5	s 5190000000	d 1212
s 5190000000	s 5190000001	d 5656
i 5190000000;name3	s 5190000002	d 5678
s 5190000000	s 5190000003	s 1234
p 0	d 5190000001	s 5678
p 1	p 0	
p 2	p 1	
p 3	p 2	
p 4	d 51937	
p 5	d 5190000002	
	p 0	
	p 1	
	p 2	
	p 3	
	n 11	
	s 5190000002	
	s 5190000003	