DAA PROJECT REPORT

HASHMAPS

NAME AND SRN :

PES1201701634 : P SUDHAMSHU RAO
PES1201701600 : VINAYAKA M HEGDE

OVERVIEW OF HASHMAP:

HashMap is a Map based collection class that is used for storing Key & value pairs.
The purpose of a map is to store items based on a key that can be used to retrieve the item at a later point.

PROPERTIES OF HASH MAP:

1. Contains unique keys.
2. The order of retrieval is not guaranteed. i.e., order of retrieval is not same as the order in which elements get into the hash map.
3. The value can be retrieved using the key.

IMPLEMENTATION:

Hash map implemented by a linked list and an array. The user can choose the implementation. Each node of linked list consists of four parts : hash value, key ,value and address of the next node. If the load factor of array increases above a certain value, then we rehash and continue. Keys and values will be taken strings and hashed to integers

There are two Implementations for hash map – Array and a Linked list. The user can choose his preferred implementation, but once he chooses, he cannot change the implementation.

1. Array Implementation : The user can insert, update, search or delete a key-value pair. The key-value pairs are stored in an array of structure. And the collisions are handled by Linear Probing Technique.

2. Linked List Implementation : The user can insert, update, search or delete a key-value pair. The key-value pairs are added at a particular index of the

hash table as a Linked list. And the collisions are handled by Separate Chaining Technique.

CLIENT FILE :

In the client file, we first take the input for the implementation. Then we take key,value and capacity from the user. Then we create pointers for the structures. There are three switch cases for the three operations and an if statement in each switch case to select between array and linked list.

HEADER FILE :

In the header file, there are four structure declarations.

1. structure hashtable : Contains key , value as strings , capacity as integer and p which indicates that in that index of the array, whether a valid element is present or not.

2. structure node : This is a structure for each node of the linked list which has key,value, pointer to the next node and capacity.

3. structure table_node : This is a structure for each node of the hash table. Each node of the hash table has a pointer that points to a node of type structure node.

4. structure table : This structure has data for overall hash table. It has the capacity and the current size of the hash table. The hash table is formed by an array of structure of table node.

SERVER FILE :

FUNCTIONS IN THE LINKED LIST IMPLEMENTATION :

1. hashmap_create_ll :
This function takes the capacity of the hash map as the input and creates a hash table of that size. It also sets the initial size as zero as there are no elements in the hash map initially. It also initializes every node the hash table to point to NULL. And it returns the pointer to the table.

2. hash_func_ll :
This function takes the key given and the pointer to the hash table and converts the string key to an integer key. And returns the converted key. This key is known as hash key because this is the key that we use for hashing. The ASCII values of the individual characters of the string are added and

stored in a variable called sum. If we do sum modulo capacity, we get the hash key.

3. hashmap_insert_ll :
This function takes a key and a value along with a pointer to the hash table and inserts the key-value pair to the hash map. If the key already exists, it just updates the value. First, the given key is converted to a hash key called as index. If we access the hash table at that index, we will get the chain in which the new key-value pair must be inserted. We will first iterate through this chain to check whether the key already exists, if it exists, just copy the new value and update it and return. If it doesnot exists, create a new node and initialize the key and value for that node and insert the node at the beginning of the chain and return.

4. hashmap_rehash_ll :
The collisions to the same index(hash key) in a hash map are handled by Separate Chaining. That is, a Linked List of those nodes is formed at that index. This linked list may grow hugely on one of the index which results in a necessity to rehash. The load factor of the hash table is the number of nodes currently present in the table divided by the total capacity of the table. If this load factor is greater than 0.75, then that indicates that the number of nodes present is closer to the total capacity of the hash table and we rehash. Therefore, after each insert , we check the Load Factor and rehash if it is greater than 0.75 .
Rehash function takes the pointer to the table as the input and it allocates memory equal to twice the capacity of hash table and initializes all the table nodes to point to NULL. And it copies each node from old table and inserts into the new table and after that the old hash table is freed.

5. hashmap_search_ll :
It searches for a given key and returns the value if found. If not, it returns -1. It takes the key and applies hash function to get the index. If we access the hash table at that index, we get the chain in which key is present. We iterate through the chain, if we find the key, we return the value. Else if the iteration terminates then the key is not present in the hash map and therefore we return -1.

6. hashmap_delete_ll :
It deletes the given key from the hash map or returns -1 if the key is not present. It takes the key to be deleted and the pointer to the hash table as inputs. It takes the key and applies hash function to get the index. If we access the hash table at that index, we get the chain in which key is present. We iterate through the chain, if we find the key, we delete the node from hash map and return. Else if the iteration terminates then the key is not present in the hash map and therefore we return -1. In the deletion from the hash map,

there could be two cases – If prev == NULL then then node to be deleted is the first node and therefore we have to change the head pointer. Else, the node to be deleted is not the first node and therefore we don't need to change the head pointer.

7. hashmap_loadfactor :
This function takes the pointer to the table as the input and calculates the load factor as size/capacity and returns the load factor.

FUNCTIONS IN THE ARRAY IMPLEMENTATION :

1.Hashmap_create function:
it takes the capacity as input and initializes a structure array of that size dynamically.It then returns a pointer to that location which is then used for Inserting and deleting.

2.Hashmap_insert function:
it takes pointer to the structure array,key,value as input.Key and value are given by user.Then the key is hashed.The hashed value acts as a index to the structure array.if there is some element already there in the index(Collision),then the function searches for the next free location by incrementing the index(Linear probing).If the user enters a same key,the function updates the value.if all the locations of hashtable are populated then the function will rehash the hashtable.In rehashing it doubles the capacity,rehashes all the elements in the table and inserts it into the new hashtable which has double the capacity.

3.Hashmap_search function:
It takes pointer to the structure array,key as an input to the function.The function hashes the given key.Then its searches for the key in the given location.If its not there,then it searches for it in the next location.if it does not find the key even after searching all location in the array,it displays to the user key not found ,if it was found in some location its will display to user Its key and value.

4.Hash_func_ll :
This function takes the key given and the pointer to the hash table and converts the string key to an integer key. And returns the converted key. This key is known as hash key because this is the key that we use for hashing. The ASCII values of the individual characters of the string are added and stored in a variable called sum. If we do sum modulo capacity, we get the hash key.

5.Hashmap_delete function:

It takes pointer to the structure array,key as an input. It first hashes the key and

compares the given key to the key in the index(hash value),if matched the entry is deleted by making variable p which acts as indicator to validate the presence of element in the structure array as 0.if not the it searches for it in the next index by incrementing the hash value. If the given key is not found in the array it displays user key not found.