Supermarket Management System Description:

Project DESCRIPTION

Creating a management system for supermarkets to manage activities. Supermarket owners can easily control activities at a supermarket so that They can gain much more profit from their supermarkets.

ADDITIONAL IMPROVEMENTS

-Added test methods in the Main class(testIndexingTime, testSearchingTime, testRemove).

-Created a flexible HashTable thanks to HashingMode and ProbingMode enum classes.

-Used Iterator interface to iterate a hashTable and display whole data.

-Coded my date class implements comparable interface and isValid method to determine whether a date is valid.

Project Structure

Management System Class: It is a useful class to implement operations at supermarkets.

Attribute:

idCustomerHashTable: it has a non-static HashTable type attribute named idCustomerHashTable because every supermarket has customers. Key equals ID and Value equals Customer class in this idCustomerHashTable. My ManagementSystem object takes a file path when initializing, and fills this attribute using that csv file.

Functions:

add: To add or update customer info with the given ID.

get: To give the customer object with the given ID

remove: To remove customer info from idCustomerHashTable with the given id.

display: To display data on idCustomerHashTable with the following format.

HashTable Class:

Attributes:

table: this attribute is an array that consists of HashEntry-type objects.

Functions:

add: In the add method there are 2 cases add and update. If a value for the key is already present the value was set to update the value.

get: locate function was used to determine the index of the given key on the table.

enlargeHashTable:

It was codded for a dynamic growable table. A new array was created with the length double of the old one. Key and value pairs copied new one. I used a temp and old table variable extra.

getHashIndex:

There are 2 types of hashing methods ssf and paf. Hashing converts a key to an index value.

Locate:

The locate method was used on the get method. In this project, there are 2 types of probes LinearLocate and DoubleHashingLocate.

In the linear locate, a while loop ends when found == True or the value at the produced index is null. Secondly, there is an if statement that the correct condition is the key at produced index equals to given argument key else was looked for next index. The founded index was returned by this method otherwise returned -1.

In the double hashing locate, a while loop ends when the value at the produced index is null. Secondly, there is an if statement that the correct condition is the key at produced index equals to the given argument key else was looked for next double hashing probe index. But there is a specific situation that causes an infinite loop so I determined whether the index equals to original index to avoid an infinite loop.

Probe:

The probe method was used on the add method. An open addressing technique was used instead of separate chaining when getHashIndex generates the same index. In this project, there are 2 types of probe LinearProbe and DoubleHashingProbe.

In the linear probe, a while loop ends when found == True or the value at the produced index is null. Secondly, there is an if statement that the correct condition is the key at produced index equals to given argument key else was looked for next index.

In the double hashing probe, a while loop ends when the value at the produced index is null. Secondly, there is an if statement that the correct condition is the key at produced index equals to the given argument key else was looked for next double hashing probe index. But there is a situation that causes an infinite loop so I decided to determine whether the index equals to original index to avoid an infinite loop.

HashEntry Class: it consists of a key, value, and state that shows removed or not.

Customer Class: it consists of customer ID, name, and priority queue transactions consisting of Transaction types objects.

Transaction Class: It represents an entity that consists of date and product name info.

Main Class: There are some test methods to test my project.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Load Factor** | **Hash Function** | **Collision Handling** | **Collision Count** | **Indexing Time** | **Avg. Search Time** | **Min. Search Time** | **Max. Search Time** |
| α=50% | SSF | LP | 2\_134\_144\_659 | 169\_499ms | 451micro | 8micro | 34740micro |
| DH | 6\_142\_437 | 3\_259 ms | 118micro | 3micro | 34896micro |
| PAF | LP | 50\_509 | 2\_765ms | 117micro | 2micro | 33525micro |
| DH | 45\_263 | 2\_848ms | 120micro | 3micro | 38811micro |
| α=80% | SSF | LP | 1\_804\_285\_473 | 168\_510ms | 444micro | 9micro | 34579micro |
| DH | 6\_093\_375 | 2\_956ms | 121micro | 4micro | 37921micro |
| PAF | LP | 164\_628 | 2\_770ms | 115micro | 2micro | 35785micro |
| DH | 105\_489 | 2\_920ms | 124micro | 2micro | 36029micro |

Ms = mili second micro = micro second

TAHA ABDÜLKADİR YILMAZ