ESKISEHIR TECHNICAL UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

FALL 2019 - 2020

EEM 480 ALGORTIHMS AND COMPLEXITY

CREATING SPOTIFY-LIKE ENVIRONMENT

HOMEWORK 4

ALİCAN ÖZELOĞLU

66277019558

10.01.2020

TASK

The purpose of the homework is to create a spotify-like environment with hash structure. Songs and persons will be stored in different hash. Hash table will have methods as follows, Insert, Like, Delete, Erase, Match, Recommend, Reading command from a file.

HOW IT WAS IMPLEMENTED

The Project was created in the Apache NetBeans IDE 11.1 environment. Project have 4 classes, one of which is main class. In the main, input is received from the user, the command and name sections are separated. The key generating function for names is also executed in this section. The command and key numbers generated from input are sent to the methods as parameters. These operations executed in an infinite loop. When the user wants to close the program, "X" command must be entered. The person and song classes contain a person and song pointer, a string variable for holding name. In the hash class, all necessary methods have been implemented. The implementation details of these methods as follow.

Insert Method

Received parameters: Person name, Person key

Command: I <Name>

This method was used to add a new person to our database. The index(key) sent as parameter of person hash was checked. If it is empty, the new person is saved there. If it is full, next index was checked, until an empty place was found. This method is called linear probe. Collisions in this project was aimed to solve with this method.

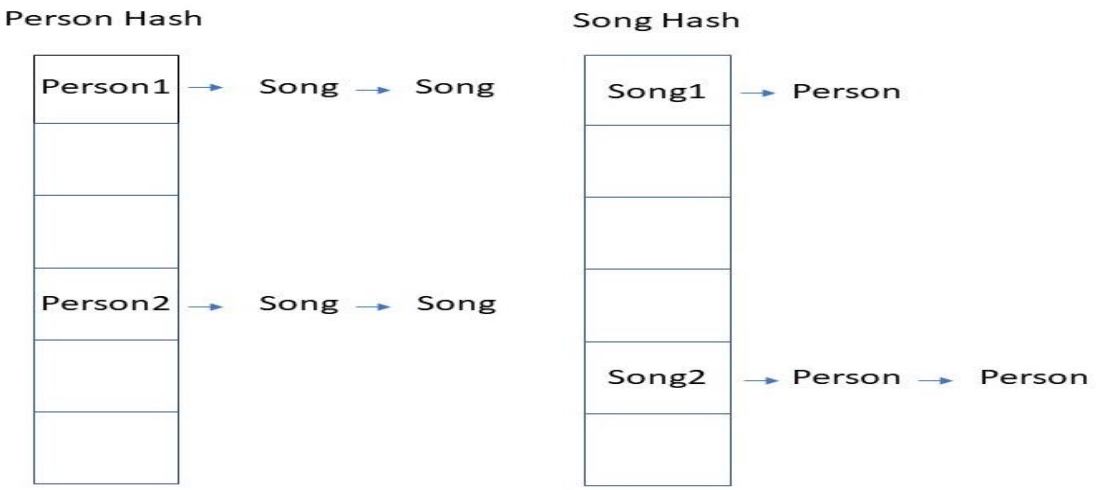

Figure 1: Hash Structure

Like Method

Received parameters: Song name, Person name, Song key, Person key

Command: L <Name> <Song>

This method was used to receive favorite songs of people. First check whether a person with this name exists. After the person was found, a structure was formed as shown in the figure 1. Two different objects from the song class were created for the same song. One has been added to song hash, and the other has been added to linked list which is created for person.

Erase Method

Received parameters: Song name, Person name, Song key, Person key

Command: E <Name> <Song>

This method was used to erase a song from the list of given person. Erase operation was performed separately for person hash and song hash. The list of songs in person hash was searched and the desired song to be deleted was removed from linked list. The list of person in song hash was also searched and the person removed from the linked list. If the songs liked by only that person, songs was also removed from song hash. Because no one like it anymore, don't need to keep there.

Delete Method

Received parameters: Person name, Person key

Command: D <Name>

This method was used to delete a person from the database. The person was deleted from the person hash. Then, all the songs he/she liked was visited and person was removed from the linked list which hold the people who like that song.

Match Method

Received parameters: Person name, Person key

Command: M <Name>

This method was used to determine how much common song person have with other people in the database. The songs that the person likes are taken in order and in the song hash other people who like that song are detected. The number of common songs of each person is kept in another array. For example, the first song was taken and other people who liked it were checked. When another person who likes the same song was detected, value of the index of the array that holds the match numbers increment by 1. After doing this for all songs, matches are gathered in the array. Results are generated from values in this array.

Recommend Method

Received parameters: Person name, Person key

Command: M <Name>

This method was used to recommend a song to the given person. People with the most common songs was detected, non-common songs of these people were recommended to given people. Starting from the person with the most common songs, 5 songs were recommended.

Print Method

Received parameters: Person name, Person key

Command: P <Name>

This method simply lists given person favorite songs.

CONCLUSION

Complexity is very important in searching algorithms. The exponential increase of searching for an element in large databases may cause delays. The hash algorithm has a constant number of complexity, no matter how large the database. Therefore, the hash algorithm is really important for data structure. The complexities of the methods used in this study in terms of theta are as follows.

Insert – (1)

Like – (1)

Erase – (1)

Delete – (1)

Print – (1)

Match – (n)

Recommend – (n)