CS0048 DATA STRUCTURES AND FILES

Assignment 8 – Hash Tables

Idea

To test the efficiency of four versions of open addressing to resolve collisions in a hash table. The hash table should have a size of any prime less than 1000.

Information

Classes to be created: Create a class called **`NBAPlayer`** with 6 attributes. Then create a class called **`HashComparison`** to do the test.

Hash Table Contents: The hash table will store instances of the class **`NBAPlayer`**. If necessary, the program can create up to 1000 **`NBAPlayer`** objects using the data from the text file **NBAPlayers.txt**.

NBAPlayers.txt Format: Each record is on one line and is made up of:

Full Name[Tab]College[Tab]Country[Tab]Draft Year[Tab]Draft Round[Tab]Draft Number

The last three entries are all -1 if a player was not drafted. If the player did not go to college then the college is "None". There are over 950 records in the file.

Hash Table Size: Your program should prompt for (and check) for a prime number less than 1000 that will be used as the size of the hash table.

Collision Resolution Methods: Your program will test four different collision resolution methods:

1. Linear probing – continually check for an empty space in the next consecutive cell (cycle back to the beginning if necessary)
2. Linear probing with an interval of 5 instead of 1
3. Quadratic probing – check for an empty cell using the indexes k, k+1, k+4, k+9, k+16 etc where k is the original index hashed to (cycle back to the beginning if necessary)
4. Double Hashing using the second hash function that maps key to (prime – key mod prime) where prime is some prime that you find between 0.5* hashTableSize and hashTableSize.

Calculating the Hash Code: The key to be used is the **full name** of the player which is one of the attributes of an **`NBAPlayer`**. You calculate the hash code as follows

(1 * int equivalent of first character in name) + (2 * int equivalent of second character in name) +
(3 * int equivalent of third character in name) + (4 * int equivalent of fourth character in name) +
(5 * int equivalent of fifth character in name) and so on

Example: The first player in the file is Aaron Brooks and his hash code should be calculated as 7827.

Calculating the Hash Function: The hash function will take the hash code and mod it with the size of the hash table

Results: Your program will successively place 50, 100, 150, 200…….. players into the table (do not exceed the size of the table). For each of the numbers your program will calculate the total number of probes used to insert that number of records and calculate the average number of probes per insertion. Present your results in a systematic, understandable way.

General:

Use methods.

Although the 4 methods are all fairly simple, it is very easy to make mistakes by confusing hash codes, hash functions, keys, indexes, increments. Be very careful when coding.

IMPORTANT: You cannot use any global variables. All variables in HashComparison must be declared inside methods and be passed when necessary. Methods will be static.

Turning in the Assignment

Run your program by entering 1001, 999, 997. The first two should be rejected and then 997 should be accepted. Your program should also print some other information:

- what was the prime that was used in the double hashing
- the hash code for Lebron James
- for each method, print the position in the hash table that Wesley Matthews is located at
- for each method, print the size of the longest cluster (consecutive filled entries in the table) after all 950 records have been hashed. A cluster may continue from the end of the table to the start.
- for each method list all the empty cell indexes (there should be 47 in each case)

Take screenshots of your results (there will be a lot). Upload the screenshots, java files and class files by the start of class on April 7. You have two weeks to work on this assignment and it is worth 30 points.