Vaishak Menon

CS303 Lab 9 – HashMaps

Problem Specification:

The problem we are given to solve in this lab is to create the get, put, linear probe, and quadratic probe functions for the HashMap class. The get function, using a specific hashing function, provides the string/value of the key that we input in. The put function will take in a key and value and place it into the HashTable. The linear probe essentially does what the put function does, but it hashes differently to decrease collisions. The quadratic probe has the same concept as the linear probe, but it once again has a different hashing function. Using the UPC data file and the input.dat file, I tested the time it took to put them and get them to and from the table.

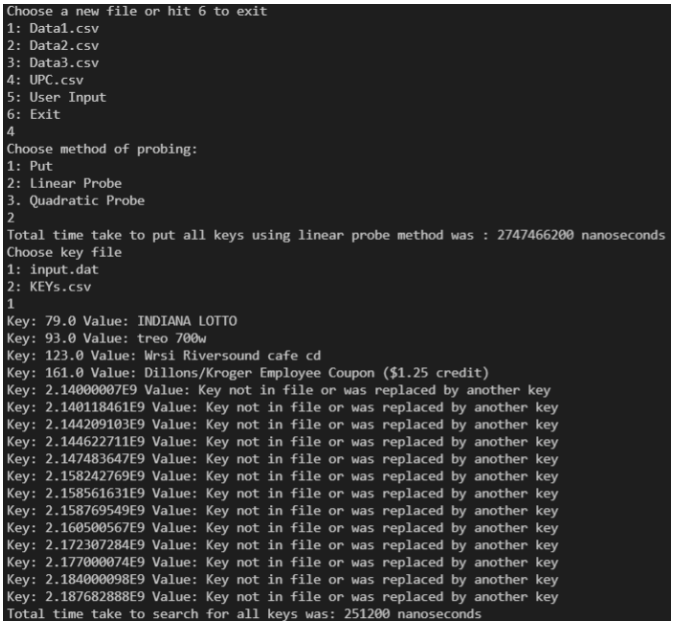
Program Design:

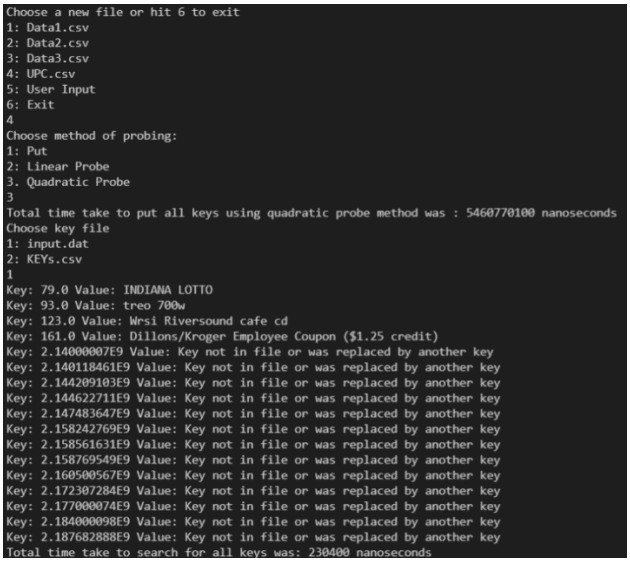
The put function takes in a key, which I made into a double due to how large they would get and use a specific hashing function that multiplies the original index by 7 adds 1 and then uses the mod function with the size of the table. The get function uses the same hashing equation as the put function to find and bring the user the value given the key. The linear probe does the same thing but hashes differently. Rather than doing the multiplication and adding, the linear probe just adds one and then uses the mod function, so you essentially get the next index. The quadratic probe is similar to the linear, the only difference being that the sequences increase by perfect squares starting at 1.

Testing Plan:

To test this, I read in the UPC file and the Input.dat file. Then I split it to two different arrays, one held all of the keys while the other held all of the values. Then depending on the chosen putting method, it would take the keys and values and put it into the HashTable accordingly.

Test Cases:





Analysis/Conclusion:

We can see from the above images that it took longer and longer to put the key and values into the table. Something surprising that occurred is that quadratic ended up being faster than the linear probing for this set of keys and values. It is expected that at some point the quadratic would be faster than the linear and at times the linear would be faster than the quadratic. It is just surprising that on this first attempt, the quadratic ended up being faster. The fastest time was for the original put function and that was the fastest because none of the keys and values stored in the array were being searched for, so all the algorithm did was go through the entire table enough times to match however many keys there were.

References:

The only references I must make are to the pseudocode and my previous lab in which I used a similar format for writing this report. Below I have added images of all my code for this lab. I apologize but the images are not in exact order of the code for this report.

```
public class HashMap
{
   private final static int TABLE_SIZE = 100;

   HashEntry[] table;

   HashMap()
   {
      table = new HashEntry[TABLE_SIZE];
   }

   public String get(double key)
   {
      String value = "";
      int index = (int) (key % TABLE_SIZE);
      if(table[index].getKey() == key)
      {
         value = table[index].getValue();
      }

      int originalIndex = index;
      while(table[index].getKey() != key)
      {
         index = (7 * index + 1) % TABLE_SIZE;
         if(index == originalIndex)
         {
            return "Key not in file or was replaced by another key";
         }
      }

      value = table[index].getValue();

      return value;
   }
```

```
public void put(double key, String value)
{
   int index = (int) (key % TABLE_SIZE);
   //System.out.println("Index: " + index);

   if(table[index] == null)
   {
      table[index] = new HashEntry(key, value);
   }
   else
   {
      int originalIndex = index;
      while(table[index] != null)
      {
         index = (7 * index + 1) % TABLE_SIZE;
         if(index == originalIndex)
         {
            break;
         }
      }
      table[index] = new HashEntry(key, value);
   }
}
```

```
   public void linearProbe(double key, String value)
   {
      int index = (int) ((key) % TABLE_SIZE);

      for(int i = 1; i < table.length; i++)
      {
         if(table[index] != null)
         {
            index = (int) ((key + i) % TABLE_SIZE);
         }
         else
         {
            table[index] = new HashEntry(key, value);
            break;
         }
      }
   }

   public void quadraticProbe(double key, String value)
   {
      int index = (int) ((key) % TABLE_SIZE);

      for(int i = 1; i < table.length; i++)
      {
         if(table[index] != null)
         {
            index = (int) ((key + Math.pow(i,2)) % TABLE_SIZE);
         }
         else
         {
            table[index] = new HashEntry(key, value);
            break;
         }
      }
   }

}
```

```
HashMap H = new HashMap();

System.out.println("Choose method of probing:");
System.out.println("1: Put\n2: Linear Probe\n3. Quadratic Probe");
int pChoice = userInput.nextInt();

switch(pChoice)
{
    case 1:
        long totalTime = 0;
        for (int j = 0; j < A.size(); j++)
        {
            Double key = A.get(j);
            String value = S.get(j);
            // System.out.println("Put Key: " + key + " Put Value: " + value);
            long bTime = System.nanoTime();
            H.put(key, value);
            long ebTime = System.nanoTime() - bTime;
            totalTime += ebTime;
        }
        System.out.println("Total time take to put all keys using put method was : " + totalTime + " nanoseconds");
        break;

    case 2:
        totalTime = 0;
        for (int j = 0; j < A.size(); j++)
        {
            Double key = A.get(j);
            String value = S.get(j);
            // System.out.println("Put Key: " + key + " Put Value: " + value);
            long bTime = System.nanoTime();
            H.linearProbe(key, value);
            long ebTime = System.nanoTime() - bTime;
            totalTime += ebTime;
        }
        System.out.println("Total time take to put all keys using linear probe method was : " + totalTime + " nanoseconds");
        break;
```

```java
public class Driver
{
    Run | Debug
    public static void main(String[] args)
    {
        ArrayList<Double> A = new ArrayList<Double>();
        ArrayList<String> S = new ArrayList<String>();

        File f = new File("");
        Scanner userInput = new Scanner(System.in);

        System.out.println("Choose which file you would like to use: ");
        String[] choices = new String[]{"Data1.csv", "Data2.csv", "Data3.csv", "UPC.csv", "User Input"};
        for(int t = 0; t < choices.length; t++)
        {
            System.out.println((t + 1) + ": " + choices[t]);
        }

        int choice = userInput.nextInt();
        do
        {
            switch (choice)
            {
            case 1:
                f = new File("Data1.csv");
                break;

            case 2:
                f = new File("Data2.csv");
                break;

            case 3:
                f = new File("Data3.csv");
                break;

            case 4:
                f = new File("UPC.csv");
                break;

            case 5:
                System.out.println("You have chosen User Input!");
                break;

            default:
                f = new File("Data2.csv");
                break;
            }
```

```java
if(choice != 5)
{
    try
    {
        Scanner sc = new Scanner(f);
        String[] s = new String[2];
        int count = 0;
        while (sc.hasNextLine())
        {
            s = sc.nextLine().trim().split(",");
            A.add(Double.parseDouble(s[0]));
            if (s.length != 1)
            {
                S.add(s[1]);
            }
            else
            {
                s = new String[2];
                s[0] = String.valueOf(A.get(count));
                s[1] = "";
                S.add(s[1]);
            }
            count++;
        }

        sc.close();
    }
    catch (FileNotFoundException e)
    {
        e.printStackTrace();
    }
}
else
{
    System.out.println("Enter in the number of keys/values to enter into the HashMap");
    int inputNum = userInput.nextInt();

    double userKey = 0;
    String userValue = "";
    for(int i = 0; i < inputNum; i++)
    {
        System.out.println("Enter in the key: ");
        userKey = userInput.nextDouble();
        System.out.println("Enter in the value: ");
        userValue = userInput.next();
        A.add(userKey);
        S.add(userValue);
    }
```

```java
                f2 = new File("KEYs.csv");
                try
                {
                    String[] s = new String[3];
                    Scanner sc2 = new Scanner(f2);
                    while (sc2.hasNextLine())
                    {
                        s = sc2.nextLine().trim().split(",");
                        testInputNum.add(Double.parseDouble(s[0]));
                    }

                    sc2.close();
                }
                catch (FileNotFoundException e)
                {
                    e.printStackTrace();
                }
                break;

            default:
                f2 = new File("input.dat");
                break;
        }

        long totalTime = 0;
        for (Double d : testInputNum)
        {
            long bTime = System.nanoTime();
            String s = H.get(d);
            long ebTime = System.nanoTime() - bTime;
            totalTime += ebTime;
            System.out.println("Key: " + d + " Value: " + s);
        }
        System.out.println("Total time take to search for all keys was: " + totalTime + " nanoseconds");


        System.out.println("Choose next action");
        System.out.println("Choose a new file or hit 6 to exit");
        for(int t = 0; t < choices.length; t++)
        {
            System.out.println((t + 1) + ": " + choices[t]);
        }
        System.out.println("6: Exit");
        choice = userInput.nextInt();
    }
    while(choice != 6);
    userInput.close();
}
```

```java
ArrayList<Double> testInputNum = new ArrayList<Double>();
ArrayList<String> testInputS = new ArrayList<String>();
File f2 = new File("");

System.out.println("Choose key file");
System.out.println("1: input.dat\n2: KEYs.csv");
int choice2 = userInput.nextInt();

switch(choice2)
{
    case 1:
        f2 = new File("input.dat");
        try
        {
            String[] s = new String[3];
            Scanner sc2 = new Scanner(f2);
            while (sc2.hasNextLine())
            {
                s = sc2.nextLine().trim().split(",");
                testInputNum.add(Double.parseDouble(s[0]));
                testInputS.add(s[2]);
            }

            sc2.close();
        }
        catch (FileNotFoundException e)
        {
            e.printStackTrace();
        }
        break;
```

```java
    case 3:
        totalTime = 0;
        for (int j = 0; j < A.size(); j++)
        {
            Double key = A.get(j);
            String value = S.get(j);
            // System.out.println("Put Key: " + key + " Put Value: " + value);
            long bTime = System.nanoTime();
            H.quadraticProbe(key, value);
            long ebTime = System.nanoTime() - bTime;
            totalTime += ebTime;
        }
        System.out.println("Total time take to put all keys using quadratic probe method was : " + totalTime + " nanoseconds");
        break;
}

/*
for(int i = 0; i < H.table.length; i++)
{
    HashEntry a = H.table[i];
    if(a == null)
    {
        System.out.println("Index: " + i + ", Associated Key: null" + ", Associated Value: null");
    }
    else
    {
        System.out.println("Index: " + i + ", Associated Key: " + a.getKey() + ", Associated Value: " + a.getValue());
    }
}
*/
/*
System.out.println("Enter key for get function");
Scanner sc2 = new Scanner(System.in);
double inputKey = sc2.nextDouble();
String returnGet = H.get(inputKey);
System.out.println("Return from get is: " + returnGet);
sc2.close();
*/

ArrayList<Double> testInputNum = new ArrayList<Double>();
ArrayList<String> testInputS = new ArrayList<String>();
File f2 = new File("");

System.out.println("Choose key file");
System.out.println("1: input.dat\n2: KEYs.csv");
int choice2 = userInput.nextInt();
```