Choosing Best Hash Function

Sri Harsha Pasupuleti

SUMMARY:

This project is regarding the implementation and design concerning hash tables. Our project is about reading a large number of distinct product numbers and decide which digit among the seven gives the best-balanced storage of the pairs of glasses. Here we are tasked with organizing the pairs of glasses into cubbies such that most of them are visible. Being said that, it helps the optical to use the space efficiently as well as catch attention to customers. A hash table uses a hash function to compute an index, also known as a hash code, into an array of buckets or slots from which to find the desired value. The key is hashed during lookup, and the resulting hash shows where the corresponding value is stored. Let us go through the pseudocode and we will walk you through the steps to run the code and screenshots of the output.

PSEUDOCODE:

Pseudocode for deciding the best digit among seven is

Step 1: Reading the files from the given path

```
BufferedReader(new FileReader(file));
Initialize line;
while ((line = br.readLine()) != null)
        line.split(" ")
        addItem(str[0], str[1], str[2], Integer.parseInt(str[3]))
```

Step 2: Create a custom hashmap

```
customhashmap(capacity)
for 0 to capacity do
keylist.add(null)
```

Step 3: Put method for hashMap

```
Initilize list ls = key_ls.get(index)
Initilize Chain(itembarcode,item)
if (ls == null)
  ls = new ArrayList
  ls.add(chain)
  key_ls.set(index, ls)
 else
  ls.add(chain)
  key_ls.set(index, ls)
```

Step 4: Remove method for hashMap

```
Initilize list ls = key_ls.get(index)
```

Initilize size = ls.size
for 0 to size do
if (ls.get(count).key == barcode)
ls.remove(count)
return

Step 5: Method for removing item and switch case to navigate based on barcode

Use for loop to get index one by one
Switch (index):
Case1: Initilize digit=ht2.hashfct2(barcode);
Remove digit, barcode
Break; do for all cases

Step 6: Method for calculating low factor

For Initilize index=1 to 7do
Initilize max=0;
Initilize min=integer.maxvalue;
switch(index)

Step 7: Do for case 1 to case 7 using for loop for 9 digits and do in case if the digit is null for max and min otherwise its zero.

Step 8: Method for calculating the low factor

for 1 to 7 do
Initilize max = 0
Initilize min = Integer.MAX_VALUE
switch (index)
1 - 7 cases
List<List<CustomHashMap.Chain>> ls = ht1.key_ls to ht7.key_ls
for 0 to 9 do
if (ls.get(digit) != null)
max = max of (ls.get(digit).size(), max)
min = min of (ls.get(digit).size(), min)
else
min = 0
map[index] = max - min
break

Step 9: Method calculating best hashing

Calculatelowfactor()
Initilize ans =0
Initilize lowfactor=integer.maxvalue
For 1 to 7 do
If (map[pos]<lowfactor)
lowfactor=map[pos]
ans=pos
return ans

Step 10: Get all the custom hash map

Step 11: Main method create objects for all the Custom hashmaps from ht1 to ht7

Execute the methods
Input1File(ic1)
Input2File(ic1)
forAddItem(ic1)

DESCRIPTION ON HOW TO RUN CODE:

In Suffix Environment:

- Save the ItemCollection.java file and the in1.txt and in2.txt files into the local drive.
- Copy the path of in1.txt, in2.txt, and paste in the file reader in input1File and input2File methods.
- Run the .java file in the command prompt.
- Then take the .class file and run with javac.
- The outputs of the file will be shown in the console.

In Java IDE:

- In1.txt and in2.txt are the input files that have to be parsed and fed into the code. Just keep the file in the local disk and indicate the exact path in the code.
- Open the ItemCollection.java file using any java IDE like eclipse, sublime editor to execute the code.
- To get the output for the in1.txt file, uncomment the input1File(ic1) method in the main method and comment on the input2File(ic1) and forAddItem(ic1) methods.
- To get the output for the in2.txt file, uncomment the input2File(ic1) method in the main method and comment on the input1File(ic1) and forAddItem(ic1) methods.
- To get the output for the addItem, uncomment the forAddItem(ic1) method in the main method and comment on the input1File(ic1) and input2File(ic1) methods.
- Hit enter or run to execute the code to get the desired output.
- The outputs of the file will be shown in the console.

Steps:

- Download the .txt input files on the hard disk and copy the path and paste it into the read file in input1File(ic1), input2File(ic1) and forAddItem(ic1) methods.
- Then hit the run button so as to code execute the code to get the output of topological sorting and longest path as well.

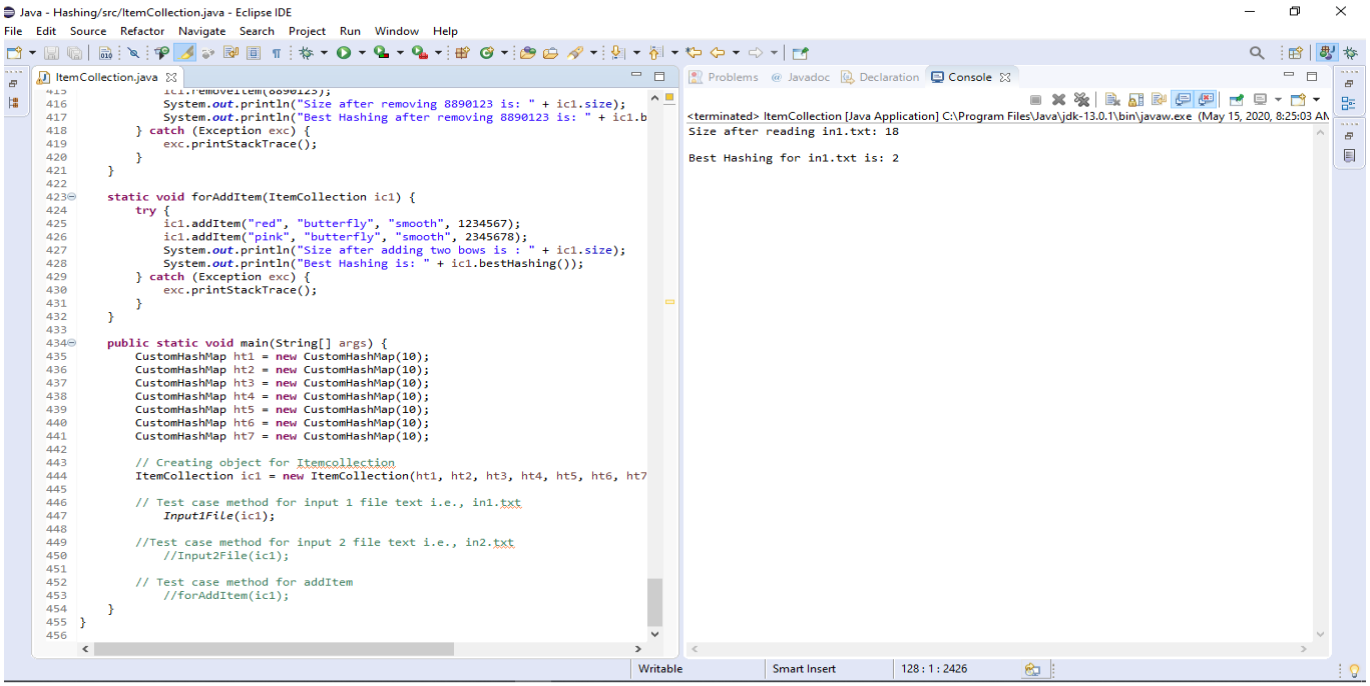
Screenshots corresponding to the input files and project members:

In1.txt output using tuffix environment and Java IDE:

Tuffix Environment:

```
student@tuffix-vm:~/Downloads$ javac ItemCollection.java
student@tuffix-vm:~/Downloads$ java ItemCollection
Size after reading in1.txt: 18

Best Hashing for in1.txt is: 2
```

Java IDE:

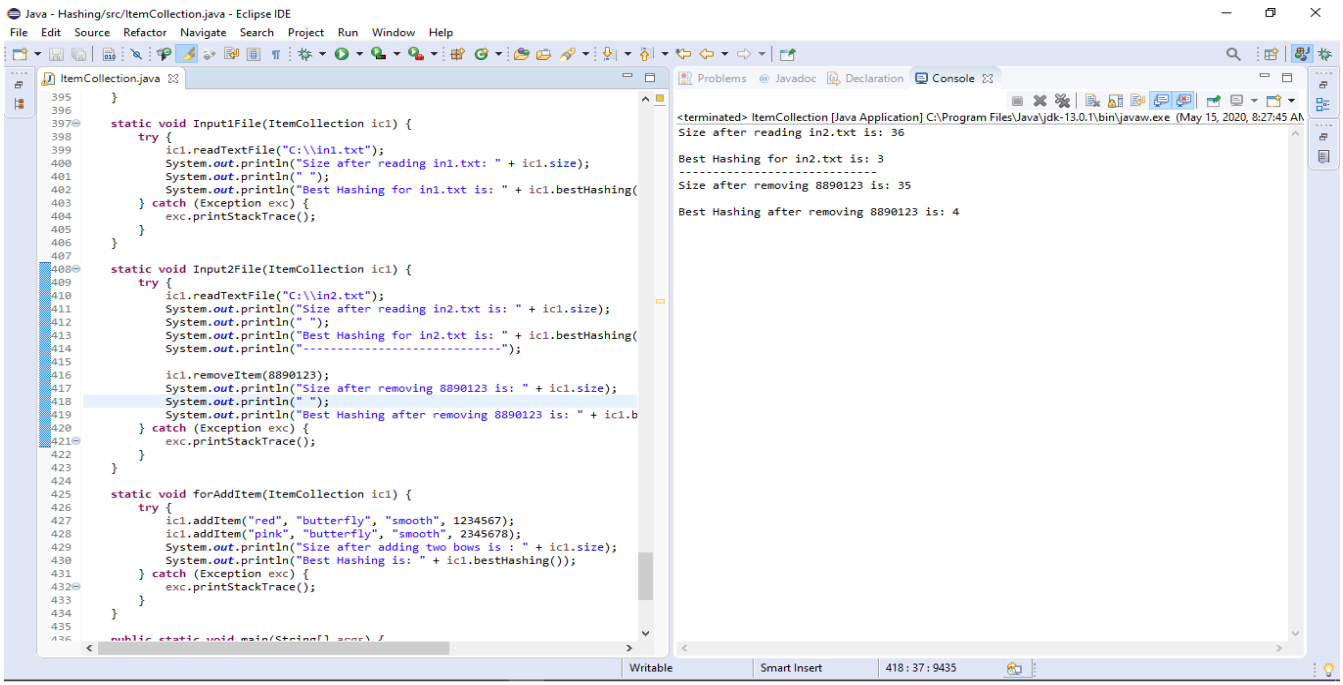


In2.txt output using tuffix environment and Java IDE:

Tuffix Environment:

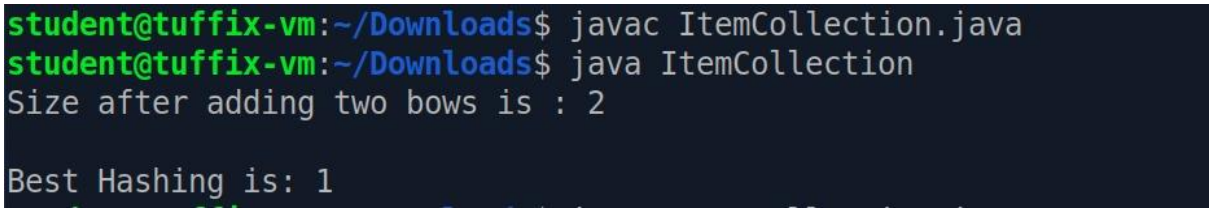
```
student@tuffix-vm:~/Downloads$ javac ItemCollection.java
student@tuffix-vm:~/Downloads$ java ItemCollection
Size after reading in2.txt is: 36

Best Hashing for in2.txt is: 3
----------------------------
Size after removing 8890123 is: 35

Best Hashing after removing 8890123 is: 4
```

Java IDE:

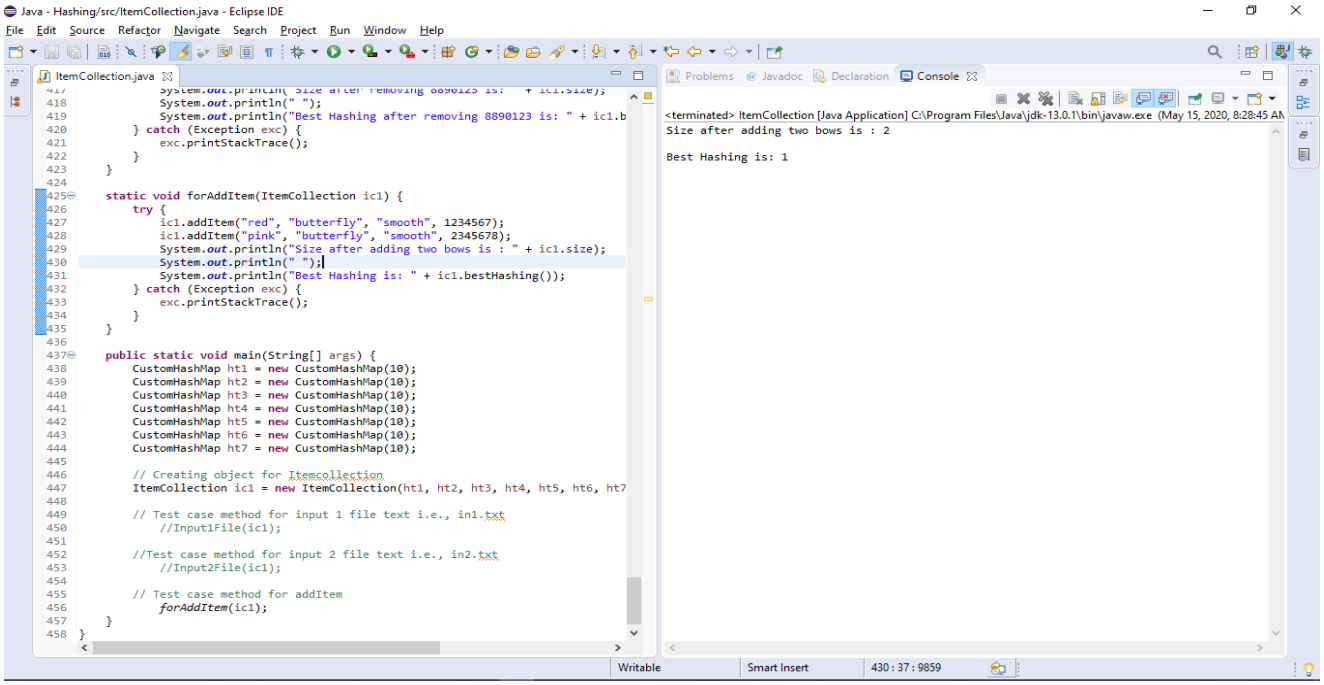


Output for adding item using tuffix environment and Java IDE:

Tuffix Environment:



Java IDE:


Java - Hashing/src/ItemCollection.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
ItemCollection.java
System.out.println("Best Hashing after removing 8890123 is: " + ic1.b
} catch (Exception exc) {
exc.printStackTrace();
}
}
static void forAddItem(ItemCollection ic1) {
try {
ic1.addItem("red", "butterfly", "smooth", 1234567);
ic1.addItem("pink", "butterfly", "smooth", 2345678);
System.out.println("Size after adding two bows is : " + ic1.size);
System.out.println(" ");
System.out.println("Best Hashing is: " + ic1.bestHashing());
} catch (Exception exc) {
exc.printStackTrace();
}
}
public static void main(String[] args) {
CustomHashMap ht1 = new CustomHashMap(10);
CustomHashMap ht2 = new CustomHashMap(10);
CustomHashMap ht3 = new CustomHashMap(10);
CustomHashMap ht4 = new CustomHashMap(10);
CustomHashMap ht5 = new CustomHashMap(10);
CustomHashMap ht6 = new CustomHashMap(10);
CustomHashMap ht7 = new CustomHashMap(10);
// Creating object for Itemcollection
ItemCollection ic1 = new ItemCollection(ht1, ht2, ht3, ht4, ht5, ht6, ht7
// Test case method for input 1 file text i.e., in1.txt
//Input1File(ic1);
//Test case method for input 2 file text i.e., in2.txt
//Input2File(ic1);
// Test case method for addItem
forAddItem(ic1);
}
}
Problems Javadoc Declaration Console
<terminated> ItemCollection [Java Application] C:\Program Files\Java\jdk-13.0.1\bin\javaw.exe (May 15, 2020, 8:28:45 AM
Size after adding two bows is : 2
Best Hashing is: 1
Writable
Smart Insert
430 : 37 : 9859