Hash tables - review

Supports the basic dynamic dictionary ops: insert, find, remove

Three design decisions: tableSize, hash function, collision resolution

Table size: a prime of the form \((4k+3)\), keeping load factor constraints in mind

Hash function

- Java’s `hashCode()` method
- item goes to hash(item) \(\%\) tableSize

Collision: multiple items at the same location

Collision resolution:

- chaining
  - probing (open addressing)
    - Linear probing
    - Quadratic probing
    - Double Hashing
    - Cuckoo Hashing

Java-specific – `hashCode()` and `equals()`

```java
public class Employee {
    String name;
    int id;
    public Employee(String n, int i){name = n; id = i;}

    ...  
    ...  
}

public static void main(String[] args) {
    Employee e1=new Employee("weiss", 001);
    Employee e2 = e1;
    System.out.println(e1.hashCode() + ", " + e2.hashCode());
    System.out.println(e1 == e2);
    System.out.println(e1.equals(e2));
}
```
Rehashing

When load factor becomes too large...

(Approximately) double tableSize

Scan old table, inserting each non-deleted item into the new table

Worst-case time?
- $O(N^2)$

Average-case: $O(N)$

Amortized analysis

Average cost per insert, over a sequence of repeated re-hashings

[Not great for interactive applications...]

Array Initialization

5.14 Describe a procedure that avoids initializing a hash table (at the expense of memory).

count = 0

int[] arr1

int[] arr2

T[] arr

To assign arr[i] the value v:
Array Initialization

5.14 Describe a procedure that avoids initializing a hash table (at the expense of memory).

count = 1

<table>
<thead>
<tr>
<th>i</th>
<th>j</th>
<th>i</th>
</tr>
</thead>
</table>

int[] arr1

| i | j |

int[] arr2

| 1 | 0 |

T[] arr

| w | v |

To assign arr[i] the value v:
To assign arr[j] the value w:

Array Initialization

5.14 Describe a procedure that avoids initializing a hash table (at the expense of memory).

count = 2

<table>
<thead>
<tr>
<th>i</th>
<th>j</th>
<th>i</th>
</tr>
</thead>
</table>

int[] arr1

| i | j |

int[] arr2

| 1 | 0 |

T[] arr

| w | v |

To assign arr[i] the value v:
To assign arr[j] the value w:
**Array Initialization**

5.14 Describe a procedure that avoids initializing a hash table (at the expense of memory).

```java
boolean isValid(int i){
    return (arr2[i] >= 0) && (arr2[i]<count) && arr1[arr2[i]] == i);
}

void assign(T x, int i){
    arr[i] = x;
    if (isValid(i) == false){
        arr1[count] = i;
        arr2[i] = count;
        count += 1;
    }
}

T retrieve(int i){
    if (isValid(i)) return arr[i];
    return null;
    //Change if T is a primitive type
}
```