CSC207.01 2014S, Class 44: Hash Tables, Continued

Overview

- Preliminaries.
  - Upcoming work.
  - Admin.
  - Questions on the project.
- Questions on hash tables.
- Lab.

Preliminaries

Upcoming Work

- Reading for Monday: No reading. We're doing another "develop in class" exercise.
- Today's writeup: Exercises 5c and 6d.
- Part 2 of the project is due Wednesday.

Admin

- Welcome to our prospective students!
- I distributed a partial solution to part one of the project. (Thanks to GB for corrections.)
- We are continuing the lab on hash tables.

Extra Credit

- CS table today: Big Data (Stone leads).
- Iowater project April 19 - Tag drains. Mail iowater@grinnell.edu for details.
- http://www.strikingly.com/pioneerweekend
- Math extra Thursday: ???
- Iowater project April 26 - Tag drains. Mail iowater@grinnell.edu for details.
- Simpson meet next Friday.

Questions on Project

Can "this is wickedly fast and efficient" be my distinguishing characteristic?

Yes.

Will you be upset if my parser is not O(n)?

Yes. Well, I won't be upset if it's O(logn), but that seems impossible.

Will you be upset if my unparser is not O(n)?

No.

Issues from the Writeup

```
int find(K key)
{
  int index = Math.abs(key.hashCode()) % this.pairs.length;
  KVPair pair;
  while ((pair = (KVPair) this.pairs[index]) != null)
    {
      System.out.println(index);
      if (pair.key == key)
        {
          return index;
        }// if the keys match
      else
        {
          // two distinct keys hashed to the same place
          // increment index by the offset to avoid collision
          index = (index + PROBE_OFFSET) % this.pairs.length;
        } // if the keys don't match
    }// while the cell is not empty
  return index;
} // find(K)
```

- Document!
- Don't compare objects with `==` and `!=`.
- Make sure that your loops terminate.
- If you have to put in print statements, use `System.err.println` and not `System.out.println`.
- Generally try to write loops that exit in one place. (But there is clearly a balancing act between good structure and readability.)

Rewritten

```
int find(K key)
{
  int index = Math.abs(key.hashCode()) % this.pairs.length;
  KVPair pair;
  while (((pair = (KVPair) this.pairs[index]) != null)
     && (!key.equals(pair.key)))
    {
      System.err.println(index);
      // two distinct keys hashed to the same place
      // increment index by the offset to avoid collision
```

```
        index = (index + PROBE_OFFSET) % this.pairs.length;
      }// while the cell is not empty
    return index;
  } // find(K)
```

How do we ensure that this terminates?

- Step one: Make sure it visits every position by making sure that PROBE_OFFSET and the table capacity are relatively prime.
- Step two: Keep a counter. When that counter reaches ...
- Detour: When you repeat an index, stop.

Whoops. Rebelsky can't design functions. This should have the option of throwing an exception.

Questions on Hash Tables

Lab