Markov

CSSE 221

Fundamentals of Software Development Honors

Rose-Hulman Institute of Technology

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

Announcements

- Team evals due now (or last night)
- Simulation project starts Monday
  - An educational simulation or animation of some process
  - Must include a GUI
  - Must include multithreading

This week: Markov

- Monday:
  - Stacks and Queues
  - Sets and Maps
- Tuesday:
  - Introduction to *Markov*, a cool statistical text program with lots of data structures
  - File I/O
- Thursday:
  - Recursion

Markov Chain Progam

- Input: a text file

  the skunk jumped over the stump
  the stump jumped over the skunk
  the skunk said the stump stunk
  and the stump said the skunk stunk

- Output: a randomly generated list of words that is “like” the original input in a well-defined way

Markov Chain Process

- Gather statistics on word patterns by building an appropriate data structure
- Use the data structure to generate random text that follows the discovered patterns

Markov Example, n = 1

- Input: a text file

the skunk jumped over the stump
the stump jumped over the skunk
the skunk said the stump stunk
and the stump said the skunk stunk

Prefix	Suffixes
NONWORD	the
the	skunk (4), stump (4)
skunk	jumped, said, stunk, the
jumped	over (2)
over	the (2)
stump	jumped, said, stunk, the
said	the (2)
stunk	and, NONWORD
and	the

Markov Example, n = 2

- Input: a text file

the skunk jumped over the stump
the stump jumped over the skunk
the skunk said the stump stunk
and the stump said the skunk stunk

Prefix	Suffixes
NW NW	the
NW the	skunk
the skunk	jumped, said, the, stunk
skunk jumped	over
jumped over	the
over the	stump, skunk
the stump	the, jumped, stunk, said
…	

Output

- n=1:

```
the skunk the skunk jumped over the skunk stunk
```

```
the skunk stunk
```

- n=2:

```
the skunk said the stump stunk and the stump jumped over the skunk jumped over the skunk stunk
```

- Note: it's also possible to hit the max before you hit the last nonword.

Markov Data structures

- For the prefixes?
- For the set of suffixes?
- To relate them?

Prefix	Suffixes
NW NW	the
NW the	skunk
the skunk	jumped, said, the, stunk
skunk jumped	over
jumped over	the
over the	stump, skunk
the stump	the, jumped, stunk, said
…	

Fixed-Length Queue and Markov

- FixedLengthQueue: a specialized data structure, useful for Markov problem
  - Check out FixedLengthQueue from your new Markov repo

- Work to implement it this class
  - Solution is in Markov if you get stuck
- When you finish, read the (long) Markov description
- We will only do milestone 1 (so no text justification)

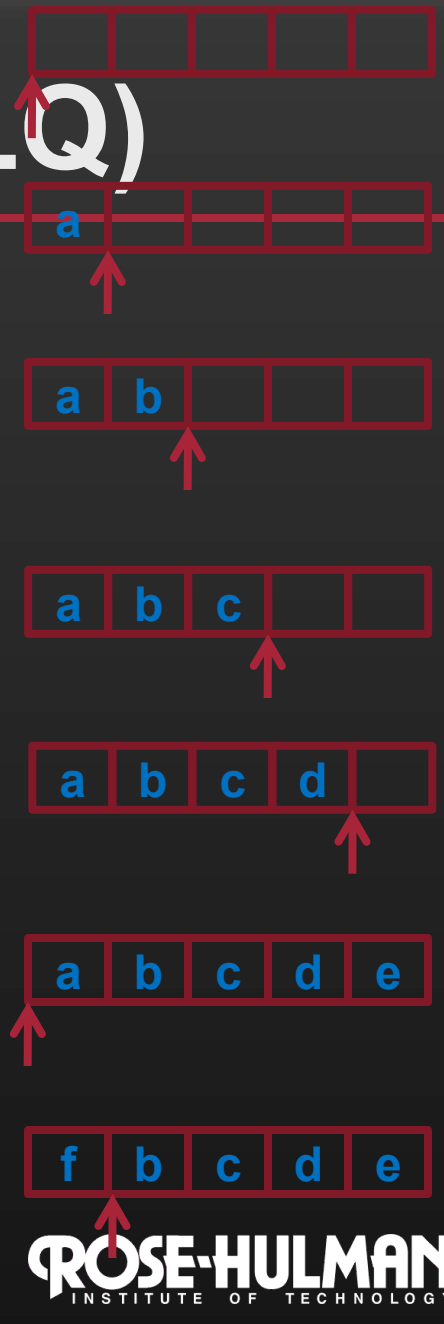
Work time, and hints

ROSE-HULMAN
INSTITUTE OF TECHNOLOGY

Arrow shows the point at which to add data next

Fixed length queue (FLQ)

- Example to the left shows the queue as elements are added
  - We'll only add, no remove
- What do you need to implement this?
  - Array whose length is the capacity of the FLQ
  - Index at which to add the next element to the FLQ
    - This index increases by 1 as you add elements, but "wraps" back to 0 when it reaches the capacity of the FLQ
  - Current size of the FLQ
    - As opposed to the capacity of the FLQ



Generating sentences by a Markov chain

Input:
Blessed are the poor for they will be Blessed are the peacemakers for they will find Blessed are meek for they will be Blessed are

Inspired by Matthew 5:3-9

To generate a new phrase, start with NONWORD NONWORD and "follow the chain", but choose *at random* from eligible suffixes

Prefix (n = 2)	Suffix
NONWORD NONWORD	Blessed
NONWORD Blessed	are
Blessed are	the the meek NONWORD
are the	poor peacemakers
the poor	for
poor for	they
for they	will will will
they will	be find
will be	Blessed Blessed
be Blessed	are are
the peacemakers	for
peacemakers for	they
will find	Blessed
find Blessed	are
are meek	for
meek for	they
are NONWORD	NONWORD

What data structures to use?

Use a **Fixed-Length Queue** whose length is n

Use a **MultiSet**

- Stores each word with its multiplicity
- Has:
  - `size()`
  - `findKth(int k)`
- To "pick at random" from a MultiSet, generate a random number, *k*, between 0 and `size()`, then call `findKth(k)` to get the random word

Prefix (n = 2)	Suffix
NONWORD NONWORD	Blessed
NONWORD Blessed	are
Blessed are	the the meek NONWORD
are the	poor peacemakers
the poor	for
poor for	they
for they	will will will
they will	be find
will be	Blessed Blessed
be Blessed	are are
the peacemakers	for
peacemakers for	they
will find	Blessed
find Blessed	are
are meek	for
meek for	they
are NONWORD	NONWORD

The Markov Map

This mapping is what we want to generate new data from the existing data, using a Markov Chain

w_{k-4} w_{k-3} w_{k-2} w_{k-1} w_k → w_{k+1}

- **When building the map: the word that follows the given prefix**
- **When generating from the map: random but according to the data distribution**

Implement as a Fixed-Length Queue whose length is *n*

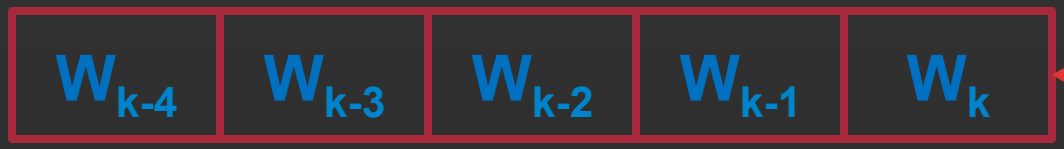
Implement the mapping as a `HashMap<String, MultiSet>` where the String is the concatenation of the words in the Fixed-Length Queue, and the MultiSet is the set of words that follow that String in the input

Implement by choosing at random from the mapped MultiSet

Do you see why these are good data structures for this problem?

Building the Markov Map

Initially, the FLQ contains NONWORD at all indices and w_{k+1} is the first word of the input



toString

String (key): W_{k-4} W_{k-3} W_{k-2} W_{k-1} w_k

get the MultiSet from the HashMap<String, MultiSet>, using this key

Previous MultiSet

If the MultiSet is null, construct the MultiSet and put it into the HashMap. In any case, add w_{k+1} to the MultiSet

Previous MultiSet plus w_{k+1}

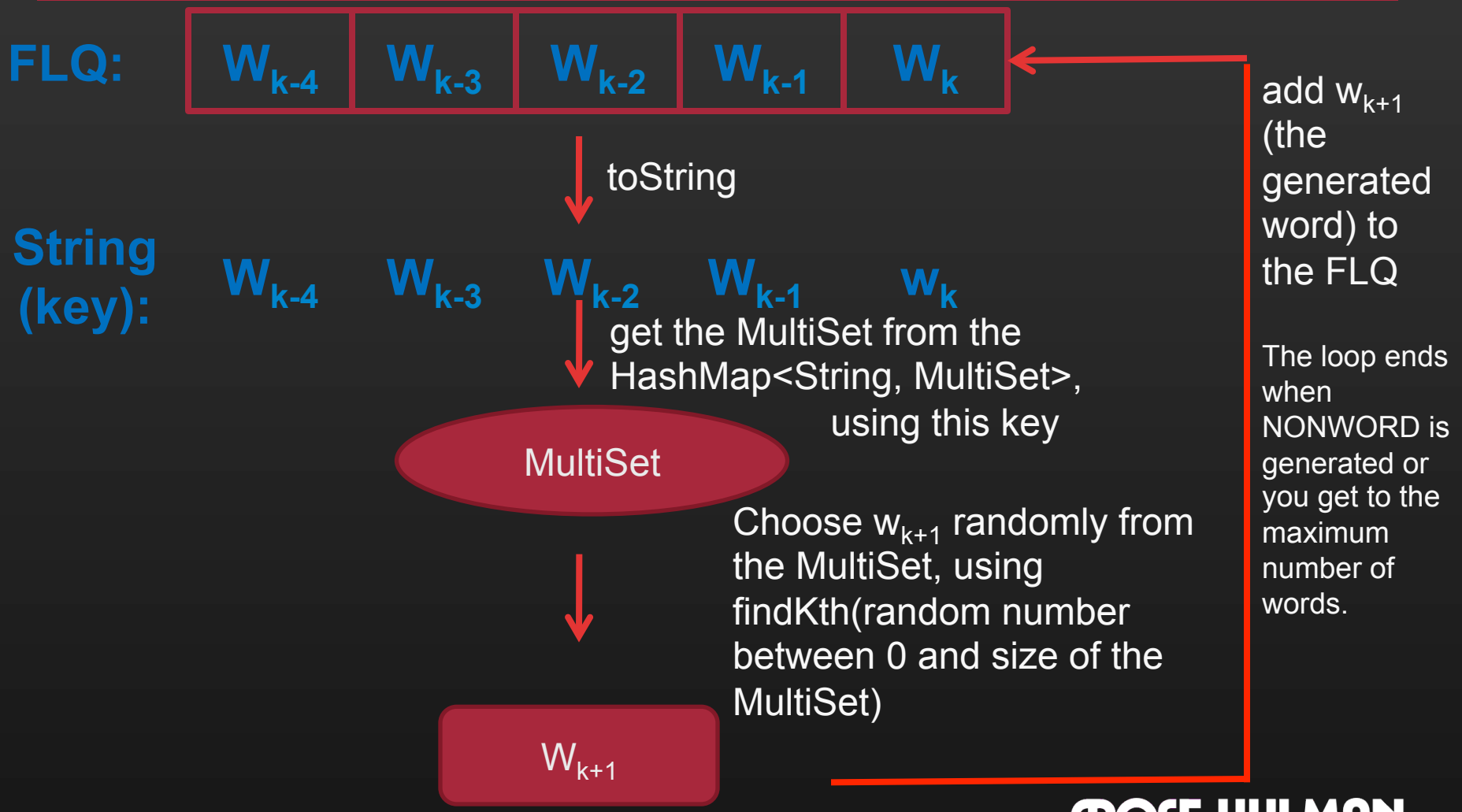
add w_{k+1} (the next word in the input file) to the FLQ

The loop ends when the input file is empty. Follow the loop by putting NONWORD as w_{k+1} n times.

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

Generating from the Markov Map

Initially, the FLQ contains NONWORD at all indices



ROSE-HULMAN INSTITUTE OF TECHNOLOGY

Reading words from a file

- ```
  Scanner scanner =
      new Scanner(
           new File (

    this.pathToInputFile)));

  while (scanner.hasNext()) {
      String word = scanner.next();
      ...
  }
  ```
  ```