CSSE 220 Day 28

Data-structure-palooza
Fixed-length queues
Markov chaining

Checkout *DataStructures* project from SVN

Questions

Data Structures

- Understanding the engineering trade-offs when storing data

Abstract Data Types Recap

- Boil down data types (e.g., lists) to their essential operations

- Choosing a data structure for a project then becomes:
    - Identify the operations needed
    - Identify the abstract data type that most efficient supports those operations

- Goal: that you understand several basic abstract data types and when to use them

Common ADTs

- Array List
- Linked List
- Stack
- Queue
- Set
- Map

Implementations for all of these are provided by the Java Collections Framework in the `java.util` package.

Array Lists and Linked Lists

Operations Provided	Array List Efficiency	Linked List Efficiency
Random access	O(1)	O(n)
Add/remove item	O(n)	O(1)

Stacks

- A last-in, first-out (LIFO) data structure
- Real-world stacks
  - Plate dispensers in the cafeteria
  - Pancakes!
- Some uses:
  - Tracking paths through a maze
  - Providing “unlimited undo” in an application

Operations Provided	Efficiency
Push item	O(1)
Pop item	O(1)

Implemented by **Stack**, **LinkedList**, and **ArrayDeque** in Java

Queues

- A first-in, first-out (FIFO) data structure
- Real-world queues
  - Waiting line at the BMV
  - Character on Star Trek TNG
- Some uses:
  - Scheduling access to shared resource (e.g., printer)

Operations Provided	Efficiency
Enqueue item	O(1)
Dequeue item	O(1)

Implemented by **`LinkedList`** and **`ArrayDeque`** in Java

Sets

- **Unordered** collections **without duplicates**
- Real-world sets
  - Students
  - Collectibles
- Some uses:
  - Quickly checking if an item is in a collection

Operations	HashSet	TreeSet
Add/remove item	O(1)	O(lg n)
Contains?	O(1)	O(lg n)

Can hog space

Sorts items!

Q1

Maps

- Associate **keys** with **values**
- Real–world “maps”
  - Dictionary
  - Phone book
- Some uses:
  - Associating student ID with transcript
  - Associating name with high scores

Operations	HashMap	TreeMap
Insert key–value pair	O(1)	O(lg n)
Look up value for key	O(1)	O(lg n)

Can hog space

Sorts items by key!

Q2–4

Markov Chaining

- Demonstration

Team URL (Individuals use individual repositories)
http://svn.csse.rose-hulman.edu/repos/csse220-201120-markov-teamXX

- Curt's section teams:
  - 11,bippuskw,modenejm
  - 12,bristokb,zellneaj
  - 13,czaplikg,mayhewrb
  - 14,dohertjp,tugayac
  - 15,goodca,schuenjr
  - 16,harrisse,trederdj
  - 17,maglioms,mouldema
  - 18,priceha,wagnerrj
- Curt's section individuals:
  - agnerrl, brooksma, kleinnj, petitjam , pohltm, ryanam, savrdada, veatchje, westeras

Team URL (Individuals use individual repositories)
http://svn.csse.rose-hulman.edu/repos/csse220-201120-markov-teamXX

- Delvin’s section teams:
  - 20,abdelroh,raonn
  - 21,crouchjt,handokkr
  - 22,carrila,deperarc
  - 23,drakecb,grovema
  - 24,hippstn,meyerrd
  - 25,lockeat,mccammjr
  - 26,moyessa,scolarrf
  - 27,coblebj,whiteaj
  - 28,redelmrw,sheltotj
  - 29,jacobyam,zhangr1
- Delvin’s section individuals:
  - chappljd, chenaurj, galvezdm, kaiserkp, oelschmm, schepedw, trammjn

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
- Implement FLQ in the next 25 minutes or so

- When you finish, read the (long) Markov description and start working on it
- We will **only** do **milestone 1** (so no text justification)

Check out FixedLengthQueue
from your Markov team or individual repo