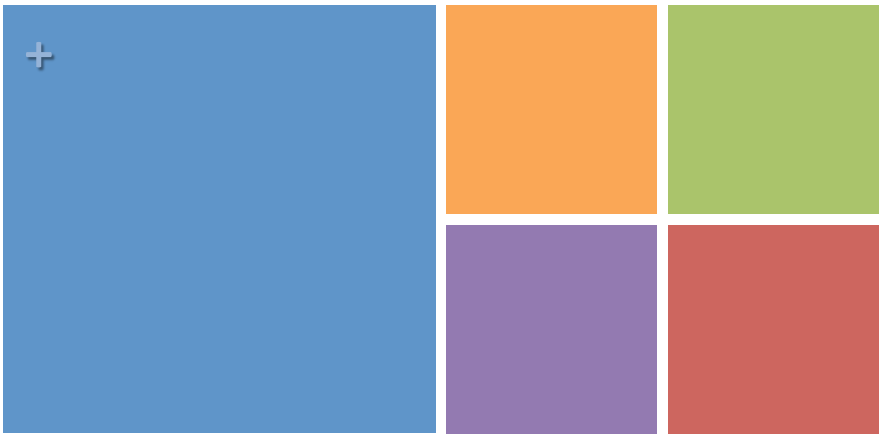


Lecture 10: Linked Lists

Today

- Reading
  - JS Ch. 9 (Linked Lists)
- Objectives
  - Helpful info. for weekly assignment
  - Recap Iterators
  - Introduce Linked Lists

Announcements

- New mentor hours!
  - Thur 8-10pm
  - Sat 2-4pm
  - Sun 2-4pm
  - Sun 8-10pm
- Mergesort proof on Piazza
- Good idea to read through the weekly programing assignment writeup on Mondays!

Weekly Programming Assignment

- File class:
  - Represents a file or directory
  - Doesn't have to exist
- Use the BufferedReader and PrintWriter classes for reading and writing to files.
- PrintWriter out =
  new PrintWriter(new FileWriter(...));
- BufferedReader in =
  new BufferedReader(new FileReader(...));

Weekly Programming Assignment

- Many methods/constructors throw exceptions
  - public String readLine() throws IOException
- Handle exceptions by try-catch construct

```
try {
    ... myFile.readLine() ...
} catch (IOException ex) {
    // code to be executed if exception raised
}
```

Iterators

- DinerMenu uses an array
- PancakeHouseMenu uses an ArrayList
- Waitress class
  - Creates a dinerMenu and a pancakeMenu
  - Wants to iterate over the contents of both menus!
  - One uses an array. The other an ArrayList
- Solution: Iterators!
  - Provide uniform iteration over a collection
  - Hides all details of how the collection is implemented

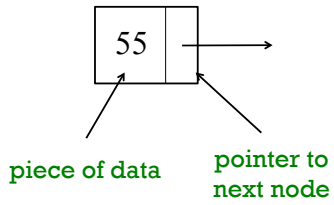
Iterators

```
class ArrayList<E> implements Iterable<E> {
      private E[] array;
      private int capacity;
      private int numElts;

      // Returns an iterator over the menu items
      public Iterator<E> iterator(){
            return new ArrayListIterator();
      }

  class ArrayListIterator implements Iterator<E> {
        private int curr = -1;

        // Is there a next item?
        boolean hasNext(){ return (curr < numElts-1); }

        // Returns the next item
        E next(){ return array[++curr]; }

        // Optional
        void remove(){...}
  }

}
```

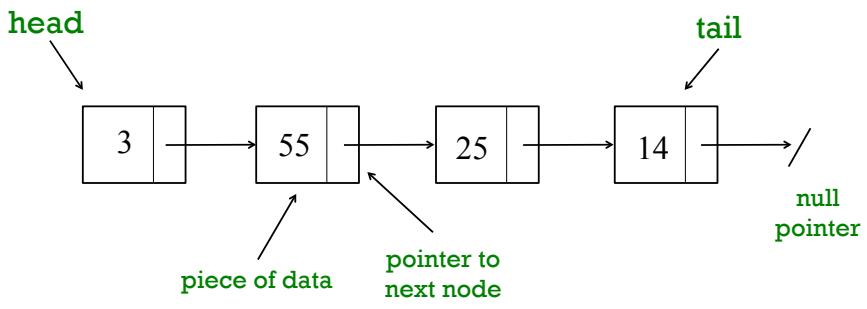
An inner class!

Linked Lists

- Our second data structure!
- A **linked list** consists of a chain of nodes
  - Add and remove nodes when necessary
  - Add/remove is fast
  - No more random access!
- Each node contains
  - A piece of data
  - A pointer to the next node



Linked List (Singly linked)



Implementing a Linked List (on board)

Linked List Operations

- Constructor
- addFirst, removeFirst
- get(i)
- indexOf(e)
- add(i,o)
- remove(e), remove(i)
- iterator

What is the complexity of each?