On your way in...

Homework 7 is due Wednesday.
Welcome to CS 134!

Introduction to Computer Science
Iris Howley
-Sorting-

Spring 2019
Do you want to be a Computer Science Teaching Assistant this fall?

• TA applications due April 12 (next Friday)

• [https://csci.williams.edu/tatutor-application/](https://csci.williams.edu/tatutor-application/)

• You’ll need 2 faculty references
  • Duane or Iris should be at least one of those!
  • Come talk to us!
MAKING OUR OWN DATA STRUCTURES

Building our own classes, leveraging recursion, etc.
Linked Lists

• Today, we’re only going to look at sorting our LinkedList

• See example code in shared/examples/04.08!

• Lecture notes from 4/3 and 4/5 are also useful!
Sorting LinkedLists

- **class Element:**
  - **def orderedInsert(self,v):**
    - """Inserts v in ordered list, returns new list."""
    - if v <= self.value:
      - return Element(v,self)
    - elif not self.next:
      - self.next = Element(v)
      - return self
    - else:
      - self.next = self.next.orderedInsert(v)
      - return self

- **class LinkedList:**
  - **def sort(self):**
    - """Sort a list of values."""
    - newList = None
    - for item in self:
      - if newList is None:
        - newList = Element(item)
      - else:
        - newList = newList.orderedInsert(item)
    - self._head = newList
Ordered Insert Sort

```
ll = [3, 2, 1]

if newList is None:
    newList = Element(item)
    if v <= self.value:
        return Element(v, self)

if v <= self.value:
    return Element(v, self)
```
Ordered Insert Sort

```python
ll = Ordered

1 3 2

1

3

2

elif not self.next:
  self.next = Element(v)
  return self

else:
  self.next = self.next.orderedInsert(v)
  return self

---

if v <= self.value:
  return Element(v, self)
```
Ordered Insert Sort

```python
ll = [1, 2, 3]

v = ll

if newList is None:
    newList = Element(item)

elif not self.next:
    self.next = Element(v)
    return self

else:
    self.next = Element(v)
    return self
```
Ordered Insert Sort

- How many comparisons are we making to do this sort?

- `ll = LinkedList()`
- `ll.extend(1,2,3)`
- `ll.sort()

\[ n = \text{len}(ll) \]

For each element of \( ll \), we have \( n-1 \) comparisons in the worst case.

Computer Science drops the -1
Ordered Insert Sort

• How many comparisons are we making to do this sort?

• ll = LinkedList()
• ll.extend(1,2,3)
• ll.sort()

For each element of ll → n
We have ~n comparisons in the worst case → *n
O(n^2) comparisons
SORTING TOOLS

Different ways to sort
Sorting Tools

• >>> ranks = [('Amherst', 18), ('Williams', 7), ('Middlebury', 9)]
• >>> ranks.sort()
• >>> ranks
• [('Amherst', 18), ('Middlebury', 9), ('Williams', 7)]

This isn’t what we want!
Sorting Tools

• What should we do?
  ▪ Iterate through, find highest, insert at front of new list
    ◦ Like LinkedList!
  ▪ Or maybe...use sorted() and its key parameter!

\texttt{sorted(iterable[, key][, reverse])}
Sorting Tools

- `sorted(iterable[, key][, reverse])`
- `key` should be a function that can be used for sort comparison
  - `ranks = [('Amherst', 18), ('Williams', 7), ('Middlebury', 9)]`

- `def byRank(pair):
  return pair[1]`

- `rs = sorted(ranks, key=byRank)`
Sorting Tools

- `sorted(iterable[, key][, reverse])`
- `reverse` is a boolean, 'True' if reverse-sorted
  - `ranks = [('Amherst', 18), ('Williams', 7), ('Middlebury', 9)]`
- `rs = sorted(ranks, reverse=True)`
- `[('Williams', 7), ('Middlebury', 9), ('Amherst', 18)]`
Sorting Tools

• `object.sort()`
  • Sorts object in-place
  • Only makes sense for mutable objects, like a list
    • `mystring.sort()` doesn’t make sense, because strings are immutable

• `sorted(object)`
  • Returns a copy of object, sorted
  • We need to tie it to a balloon!
    • `sList = sorted(object)`
Sorting Tools

• `def byRank` is a simple, **one-expression** function with just this one purpose!

• ...lambda functions (i.e. anonymous functions)

• `rl = sorted(ranks, key=lambda pair:pair[1])`
  
  Compare to:
  
• `rs = sorted(ranks, key=byRank)`

• `def byRank(pair):
  ▪ return pair[1]`
Lambda Functions (Another Example)

- `def mult(a,b):`
  - `return a*b`

- `p = mult(5,6)`

- *Is comparable to:*
  - `m = lambda a,b: a*b`

  - `p = m(5,6)`

A poor use of lambda functions
Lambda Functions (Another Example)

• Maybe we want to always transform a function’s output in a couple different ways:

  • `def somefunc(n):`
    • `return lambda a : a*n`

  • `doubled = somefunc(2)`
  • `print(doubled(5))` ➔ 10

  • `tripled = somefunc(3)`
  • `print(tripled(5))` ➔ 15

Use lambda functions when an anonymous function is required for a short period of time
Lambda Functions

• Historical significance to the field of computer science
• Introduced by Alonzo Church in the 1930s

• Thought they were writing about mathematical logic, ended up defining computation
  • ~1960s, connected lambda to programming languages
  • Popular in linguistics, too
    • See ‘Montague Grammar’

• Ties into Turing machines (~1935)
  • Defines an abstract machine
  • Proves fundamental limitations on the power of mechanical computation
EVERYTHING IN PYTHON IS AN OBJECT

(including functions)
Functions as Objects

• dogs = ['pixel', 'tally', 'linus', 'wally']

• def justDog(d):
  return d + " dog"

• def printDog(dList, strFunction):
  for d in dList:
    print(strFunction(d))

• >>> printDog(dogs, justDog)
  pixel dog
tally dog
linus dog
wally dog

What’s happening here?

What if I wanted to use a different function
H-INDEXES

Calculating the importance of documents, based on citations (domain knowledge for lab)
H-Index

• Estimates the impact an academic has on their field
  ▪ Or, in today’s lab, the impact of a year of court rulings has on law

• If someone has an h-index of nm that means they have n papers that have been cited by at least n other papers
  ▪ H-index of 20 means you have 20 papers that have been cited at least 20 times
H-Index

• For academic scholars, see Google Scholar
In Lab

1. Given a bunch of court ruling citation counts
2. Compute h-index for each docket (i.e., year)
3. Plot it!

   • Extensions: Determine the impact of specific courts (i.e., judges)

   • This work replicates some of that in Fowler & Jeon’s “Social Networks” journal article
QUESTIONS?
Leftover Slides
Sorting

• Other ways of sorting, we used “Ordered Insert” (similar to Insertion Sort)

• Bubble Sort:
  ▪ Swaps adjacent pairs of elements from a list repeatedly

• Merge Sort:
  ▪ Split the list into 2 sublists, sort the sublists, and then merge them

We’ll talk more about these on Wednesday.
Filter() Function

- `filter(object, iterable)`
  - Object should be a lambda function which returns a boolean
  - Iterable is a sequence with the `__iter__` method, like a list or string

- `dogge = ['pixel', 'tally', 'linus', 'wally']`
- `lyFilter = filter(lambda d: 'ly' in d, dogge)`
  - `lyFilter` is a Filter Object, convert to list:
- `ly = list(lyFilter)`
  - `['tally', 'wally']`
Map() Function

- `map(object, iterable1, iterable2, ...)
  - Object should be a lambda function to transform items in iterable
  - Iterable is a sequence with the `__iter__` method, like a list or string
  - Maps every item in iterable to corresponding output iterable by lambda

- `years = [2019, 2020, 2021, 2022]`
- `sinceM = map(lambda y: y - 1793, years)`
  - `sinceM` is a Map Object, convert to list:
- `thClass = list(sinceM)`
  - `[226, 227, 228, 229]`
- `thClass = list(map(lambda y: str(y - 1793) + "th class", years))`
Tuples, Strings, other built-in types aren’t particularly special!

You can build your own!
What is a list?

```python
class list(object):
    list() -> new empty list
    list(iterable) -> new list initialized from iterable's items

    Methods defined here:

    __add__(self, value, /)
        Return self+value.

    __contains__(self, key, /)
        Return key in self.

    __delitem__(self, key, /)
        Delete self[key].

    __eq__(self, value, /)
```
What is a list?
What is a list?

What is the last elephant holding onto? None
Thought question:
How would you build a doubly-linked list?
What is a list?

class Element:

_value
3

_next

_value
15

_next

_value
2019

_next
class LinkedList:

class Element:

_value

_next

_value

_next

_value

_next

_value

_next
Linked List

• Class LinkedList is a “wrapper class” for our “container class”, Element

• This implementation, LinkedList mostly:
  1. Handles the empty case
  2. Passes the heavy-lifting (i.e., all other cases) to Element
QUESTIONS

(you’re not the only one!)