CompSci 101
Introduction to Computer Science

April 21, 2016
Prof. Rodger

ACM Meeting

- Duke ACM is the student chapter of the professional organization for Computer Scientists
- Election and matching up with CS Buddies
- Monday, 6:15pm – LSRC D106

Grace Hopper Conference

- Conference for women in computing
- 15000 women in Computing
- Apply for Duke CS scholarship to go
- Form up soon

Announcements

- Last RQ due by 5pm tomorrow
- Assign 8 due today, Assign 9 Tues
- APT 11 (optional) due on Tuesday
- APT Quiz 3 Sun-Tue
- Final Exam: Sec 01 Tues 7pm, Sec 02 Fri 7pm
  - Get accommodations?
  - Three exams in a 24 hour period?
  - Room for some to take final with the other section
  - Form available soon to request different time
More Announcements

• Regrade for Exam 2 – get to Prof Rodger soon
• Be a UTA for CompSci 101
  – Rewarding and Learning Experience
  – Form available soon!

• Today:
  – More on Recursion, Regex
  – More on Sorting and analyzing it

Provide Comments on UTAs

• Lab UTAs
• Any other UTAs who helped you?

• See announcement in Sakai for comments
  – Feedback for course
    • Has a link for anonymous feedback on UTAs

Review Recursion and Regex
bit.ly/101sp16-0421-1a

Dictionary Comprehension

• List comprehension - builds a new list
• Dictionary comprehension - builds a new dictionary

• Format
  \[ d = \{ \text{key:} \text{value for key in somelist if ...} \} \]
Example: ReadFood: Initialize dictionary ratingsdict

- Compute number of restaurants, say $n$
- Create `alldata` – list of
  - `[[name1, ratings1], [name2, ratings2], [name3, ratings3], ...]`
  - `[['Will', ['ABP', 1, 'Loop', 1, 'Panda', 5]], ...]`
- Then create dictionary:
  - `ratingsdict = {person[0]: [0]*n for person in somelist}`
- Then update dictionary by processing `alldata`

Sorting

- In python:
  - `alist = [8, 5, 2, 3, 1, 6, 4]`
  - `alist.sort()` OR `result = sorted(alist)`
  - Now `alist` OR `result` is `[1, 2, 3, 4, 5, 6, 8]`
- How does one sort elements in order? How does “sort” work?

Selection Sort

- Sort a list of numbers.
- Idea:
  - Repeat til sorted
    - Find the smallest element in part of list not sorted
    - Put it where it belongs in sorted order.
      - Swap it with the element where it should be
- Sort example
  - `Sorted, won’t move final position | ???`
Selection Sort
• Sort the list of numbers using Selection Sort.
• The body of the loop is one pass.
• Show the elements after each pass.
• 9, 5, 1, 4, 3, 6

Code for Selection Sort

def selectsort(data):
    for i in range(len(data)):
        mindex = minindex(i)
        data[i], data[mindex] = data[mindex], data[i]

Bubble Sort

• Sort a list of numbers.
• Idea:
  – Repeat til sorted
    • Compare all adjacent pairs, one at a time. If out of order then swap them
• Sort example

<table>
<thead>
<tr>
<th>???</th>
<th>Sorted, won’t move final position</th>
</tr>
</thead>
</table>

BubbleSort
• Sort the list of numbers using BubbleSort.
• The body of the loop is one pass.
• Show the elements after each pass.
• [9, 5, 1, 4, 3, 6]
Code for Bubblesort

```python
def bubblesort(data):
    for j in range(len(data)-1, 0, -1):
        print(data)
        for k in range(0, j):
            if data[k] > data[k+1]:
                data[k], data[k+1] = data[k+1], data[k]
    return data
```

Insertion Sort

- Sort a list of numbers.
- Idea:
  - Sort by repeated inserting another element
  - Leftmost element is sorted part of list
  - Insert another element in that sublist keeping it sorted
  - Insert another element in that sublist keeping it sorted
  - Etc.
- Sort example

```
| Sorted relative to each other | ???
```

Insertion Sort

bit.ly/101sp16-0421-3

- Sort the list of numbers using InsertionSort.
- The body of the loop is one pass.
- Show the elements after each pass.
- [9, 5, 1, 4, 3, 6]

Merge Sort

- Idea: Divide and Conquer
- Divide list into two halves
- Sort both halves (smaller problem)
- Merge the two sorted halves

```
9 5 1 4 3 6 2 7
```
Merge Sort

- Idea: Divide and Conquer
- Divide list into two halves
- Sort both halves (smaller problem)
- Merge the two sorted halves

9 5 1 4 3 6 2 7
9 5 1 4 3 6 2 7 divide list into 2 halves

What does recursively sort mean?

Merge Sort

- Use the same Merge Sort algorithm
  - Divide list into two halves
  - Sort both halves (smaller problem)
  - Merge the two sorted halves

9 5 1 4
9 5 1 4 divide list into 2 halves
5 9 1 4 recursively sort each half
1 4 5 9 merge the two sorted list
MergeSort idea for code

def mergesort(data):
    n = len(data)
    if n == 1:
        return data
    else:
        d1 = mergesort(data[:n//2])
        d2 = mergesort(data[n//2:])
        merge(d1, d2)