

CompSci 101

Introduction to Computer Science



compsci 101 fall 2017

December 5, 2017

Prof. Rodger

1

Announcements

- Regrades Exam 2 – submit by Thursday, Dec 7
- Regrades for Asg 1-5, APT 1-7 by Dec 8
 - Check your grades! RQ too!
- Assign 8 due today, last late day Dec 8!
- APT 8 due Thursday, Dec 7, last late day, Dec 10!
- Assign 9 – due Dec 11, no late after this date
- Final Exam:
 - Sec 01 Thur, Dec 14, 9am, LSRC B101
 - Sec 02 Sat, Dec 16, 2pm, LSRC B101
 - Get accommodations? Fill out for Final Exam

compsci 101 fall 2017

2

Calculate Your Grade

- From “About” tab on course web page

Labs	5%
Reading Quizzes	5%
Lecture Group work	5%
Apts	12%
Programming Assignments	12%
APT Quizzes	6%
Two Midterm Exams	30%
final exam	25%

3

More on Grades

- Lecture – ignore the first two weeks (drop/add period), plus drop 4 points
- Reading Quizzes – will drop 30 points
 - Check your grades to make sure they copied over – fill out duke oit help form if they are wrong
- Lab – drop 6 points (each lab is 4 pts)
 - 44 pts total– 38 pts is 100%
 - Lab 11 covers two new topics!

compsci 101 fall 2017

4

More Announcements

- Be a UTA for CompSci 101
 - Rewarding and Learning Experience
 - Apply: see link in Sakai announcement
- Today:
 - Finish from last time
 - Why are dictionaries so fast?
 - More on Recursion, Regex
 - More on Sorting and analyzing it

Answer Questions bit.ly/101f17-1205-1

SortByFreqs APT

Sort items by their frequency, break ties alphabetically

```
data = ["apple", "pear", "cherry", "apple", "pear", "apple", "banana"]  
Returns: ["apple", "pear", "banana", "cherry" ]
```

Review Recursion and Regex bit.ly/101f17-1205-2

Dictionary Comprehension

- List comprehension - builds a new list
- Dictionary comprehension - builds a new dictionary
- Format
 $d = \{ \text{key:value for key in somelist if} \}$
:
:

Example: From Exam 2 **Sec 01**–
dict of clubs to list of tuples

```
def dictClubsToMeetings(data):
    d = {}
    for item in data:
        club = item[0]
        person = item[1]
        meetings = int(item[3])
        if club not in d:
            d[club] = []
        d[club].append((person, meetings))
    return d
```



```
def dictClubsToMeetings(data):
    d = {item[0]:[] for item in data}
    for item in data:
        club = item[0]
        person = item[1]
        meetings = int(item[3])
        d[club].append((person, meetings))
    return d
```

9

Example: From Exam 2 **Sec 02**–
dict of names to list of tuples

```
def dictNamesToMeetings(data):
    d = {}
    for item in data:
        club = item[0]
        person = item[1]
        meetings = int(item[3])
        if person not in d:
            d[person] = []
        d[person].append((club, meetings))
    return d
```



```
def dictNamesToMeetings(data):
    d = {item[1]:[] for item in data}
    for item in data:
        club = item[0]
        person = item[1]
        meetings = int(item[3])
        d[person].append((club, meetings))
    return d
```

10

Why are dictionaries so fast?

- They use a technique called hashing
- Each key is converted to hopefully a unique storage location address.
- Then each key's value can be found quickly by indexing to that location
- A dictionary may really be a list underneath, its how you use the list....

Simple Example Hashing

Want a mapping of Soc Sec Num to Names

- Duke's ACM Chapter wants to be able to quickly find out info about its members. Also add, delete and update members. Doesn't need members sorted.

267-89-5431 John Smith

703-25-6141 Jack Adams

319-86-2115 Betty Harris

476-82-5120 Rose Black

- Hash Table size is 0 to 10
- Possible Hash Function: $H(ssn) = \text{last 2 digits} \bmod 11$

Have a list of size 11 from 0 to 10

- Insert these into the list
- Insert as (key, value) tuple
(267-89-5431, John Smith)
(in example, only showing name)

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	

compsci 101 fall 2017

13

Have a list of size 11 from 0 to 10

- Insert these into the list
- Insert as (key, value) tuple
(267-89-5431, John Smith)
(in example, only showing name)

H(267-89-5431) = 31 % 11 = 9
John Smith
H(703-25-6141) = 41 % 11 = 8
Jack Adams
H(319-86-2115) = 15 % 11 = 4
Betty Harris
H(476-82-5120) = 20 % 11 = 9
Rose Black

0	
1	
2	
3	
4	Betty Harris
5	
6	
7	
8	Jack Adams
9	Rose Black
10	

compsci 101 fall 2017

14

Have a list of size 11 from 0 to 10

- Insert these into the list
- Insert as (key, value) tuple
(267-89-5431, John Smith)
(in example, only showing name)

H(267-89-5431) = 31 % 11 = 9
John Smith
H(703-25-6141) = 41 % 11 = 8
Jack Adams
H(319-86-2115) = 15 % 11 = 4
Betty Harris
H(476-82-5120) = 20 % 11 = 9
Rose Black

0	
1	
2	
3	
4	Betty Harris
5	
6	
7	
8	Jack Adams
9	John Smith
10	Rose Black

compsci 101 fall 2017

15

Hashing, dictionaries bit.ly/101f17-1205-3

compsci 101 fall 2017

16

Review: Sorting with itemgetter

- We can write: `import operator`
– Then use `key=operator.itemgetter(...)`
- We can write: `from operator import itemgetter`
– Then use `key=itemgetter(...)`

Review Example with itemgetter

- Because sort is stable sort first on tie-breaker, then that order is fixed since stable

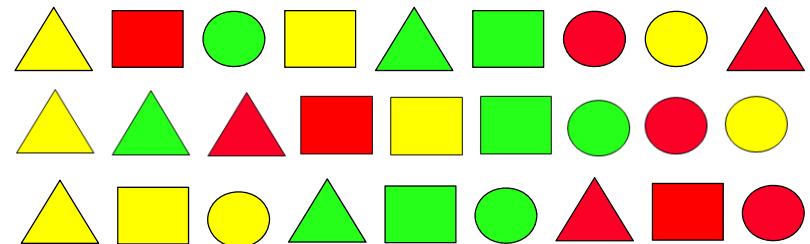
```
a0 = sorted(data, key=operator.itemgetter(0))
a1 = sorted(a0, key=operator.itemgetter(2))
a2 = sorted(a1, key=operator.itemgetter(1))
data
[('f', 2, 0), ('c', 2, 5), ('b', 3, 0),
 ('e', 1, 4), ('a', 2, 0), ('d', 2, 4)]
a0
[('a', 2, 0), ('b', 3, 0), ('c', 2, 5),
 ('d', 2, 4), ('e', 1, 4), ('f', 2, 0)]
```

Two-pass (or more) sorting

```
a0 = sorted(data, key=operator.itemgetter(0))
a1 = sorted(a0, key=operator.itemgetter(2))
a2 = sorted(a1, key=operator.itemgetter(1))
a0
[('a', 2, 0), ('b', 3, 0), ('c', 2, 5),
 ('d', 2, 4), ('e', 1, 4), ('f', 2, 0)]
a1
[('a', 2, 0), ('b', 3, 0), ('f', 2, 0),
 ('d', 2, 4), ('e', 1, 4), ('c', 2, 5)]
a2
[('e', 1, 4), ('a', 2, 0), ('f', 2, 0),
 ('d', 2, 4), ('c', 2, 5), ('b', 3, 0)]
```

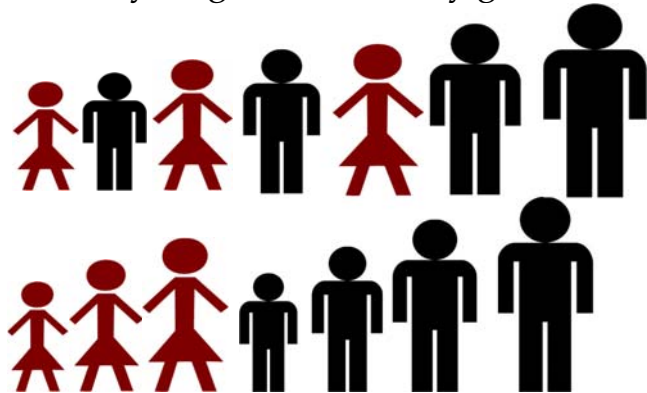
Stable, Stability

- What does the search query 'stable sort' show us?
– Image search explained
– First shape, then color: for equal colors?



Stable sorting: respect re-order

- Women before men ...
 - First sort by height, then sort by gender



compsci 101 fall 2017

21

Answer Questions bit.ly/101f17-1205-4

MedalTable APT

Sort items by their frequency, then
sorted in frequencies.

```
["ITA JPN AUS", "KOR TPE UKR", "KOR KOR GBR", "KOR CHN TPE"]  
  
Returns:  
[ "KOR 3 1 0", "ITA 1 0 0", "TPE 0 1 1", "CHN 0 1 0", "JPN 0 1 0",  
  "AUS 0 0 1", "GBR 0 0 1", "UKR 0 0 1"  
]
```

compsci 101 fall 2017

22

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?

compsci 101 fall 2017

23

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	???
--------------------------------------	-----

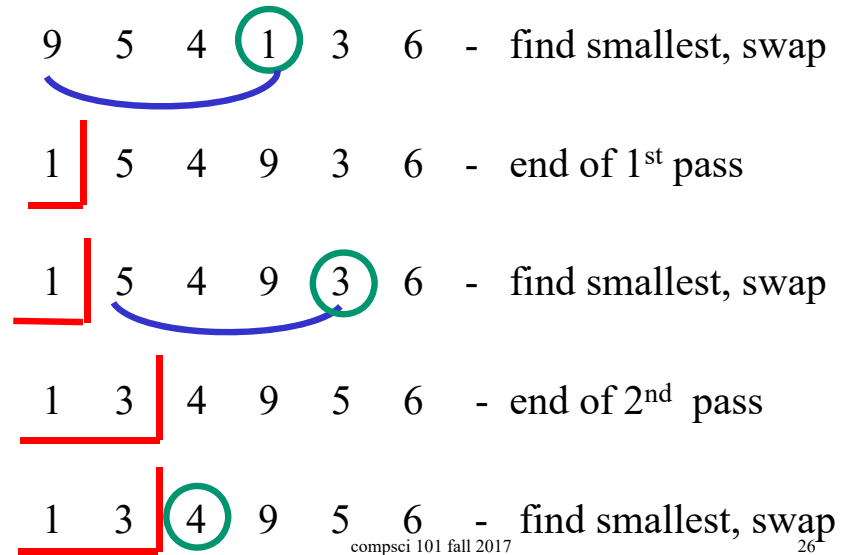
compsci 101 fall 2017

24

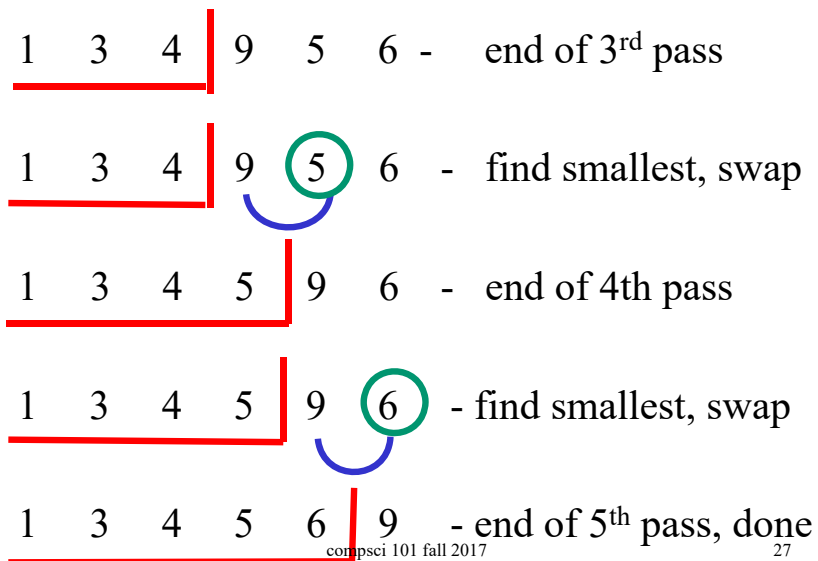
Example: Selection Sort

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

Selection Sort – red area sorted



Selection Sort (cont.)

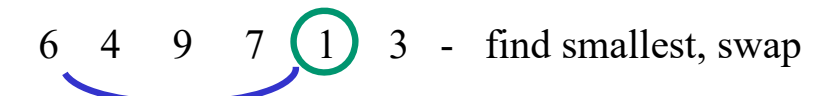
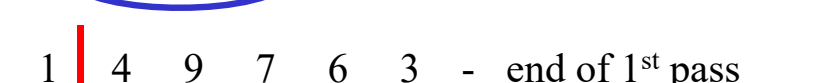


Selection Sort

<http://bit.ly/101f17-1205-5>

- Sort the list of numbers using Selection Sort.
- The body of the loop is one pass.
- Show the elements after each pass.
- 6, 4, 9, 7, 1, 3

Selection Sort – red area sorted

6 4 9 7 **1** 3 - find smallest, swap

 1 4 9 7 6 3 - end of 1st pass

 1 3 9 7 6 **3** - find smallest, swap

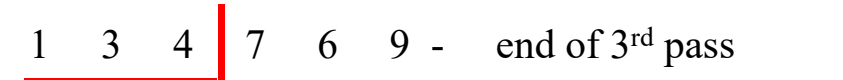
 1 3 9 7 6 4 - end of 2nd pass

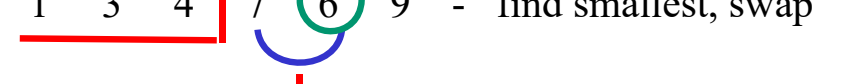
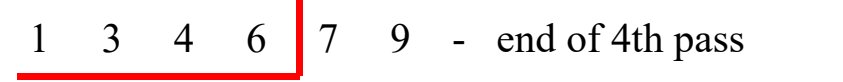
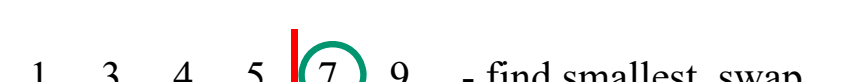
 1 3 9 7 6 **4** - find smallest, swap


compsci 101 fall 2017

29

Selection Sort (cont.)

1 3 4 7 6 9 - end of 3rd pass

 1 3 4 7 6 9 - find smallest, swap

 1 3 4 6 7 9 - end of 4th pass

 1 3 4 5 7 9 - find smallest, swap

 1 3 4 5 7 9 - end of 5th pass, done


compsci 101 fall 2017

30

Code for Selection Sort

```
def selectsort(data):
    for i in range(len(data)):
        minindex = minindex(i)
        # swap elements at indexes i and minindex
        tmp = data[i]
        data[i] = data[minindex]
        data[minindex] = tmp
```

compsci 101 fall 2017

31

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

???	<i>Sorted, won't move final position</i>
-----	--

compsci 101 fall 2017

32

Bubble Sort – red area sorted

9 5 4 1 3 6 - compare, swap
 5 9 4 1 3 6 - compare, swap
 5 4 9 1 3 6 - compare, swap
 5 4 1 9 3 6 - compare, swap
 5 4 1 3 9 6 - compare, swap
 5 4 1 3 6 9 - end of 1st pass
 5 4 1 3 6 9

Bubble Sort – red area sorted

5 4 1 3 6 9 - compare, swap
 4 5 1 3 6 9 - compare, swap
 4 1 5 3 6 9 - compare, swap
 4 1 3 5 6 9 - compare, no swap
 4 1 3 5 6 9 - end of 2nd pass
 4 1 3 5 6 9

Bubble Sort – red area sorted

4 1 3 5 6 9 - compare, swap
 1 4 3 5 6 9 - compare, swap
 1 3 4 5 6 9 - compare, no swap
 1 3 4 5 6 9 - end of 3rd pass
 1 3 4 5 6 9

Two more passes would guarantee sorted.
 Or Check if sorted and skip last two passes

Bubble Sort

bit.ly/101f17-1205-6

- Sort the list of numbers using BubbleSort.
- The body of the loop is one pass.
- Show the elements after each pass.
- [6, 4, 9, 7, 1, 3]

Bubble Sort – red area sorted

6 4 9 7 1 3 - compare, swap
 4 6 9 7 1 3 - compare, no swap
 4 6 9 7 1 3 - compare, swap
 4 6 7 9 1 3 - compare, swap
 4 6 7 1 9 3 - compare, swap
 4 6 7 1 3 9 - end of 1st pass
 4 6 7 1 3 9

Bubble Sort – red area sorted

4 6 7 1 3 9 - compare, no swap
 4 6 7 1 3 9 - compare, no swap
 4 6 7 1 3 9 - compare, swap
 4 6 1 7 3 9 - compare, swap
 4 6 1 3 7 9 - end of 2nd pass
 4 6 1 3 7 9

Bubble Sort – red area sorted

4 6 1 3 7 9 - compare, no swap
 4 6 1 3 7 9 - compare, swap
 4 1 6 3 7 9 - compare, swap
 4 1 3 6 7 9 - end of 3rd pass
 4 1 3 6 7 9

Bubble Sort – red area sorted

4 1 3 6 7 9 - compare, swap
 1 4 3 6 7 9 - compare, swap
 1 3 4 6 7 9 - end of 4th pass
 1 3 4 6 7 9

Sorted, just needed 4 passes

Code for Bubblesort

```
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
```

Enjoy a Cookie! One per person

