Announcements

- Regrades Exam 2 – submit by Friday, April 28
- Assign 8 due today
- APT 9 due Thursday, April 27
- Assign 9 – due Friday April 28
- Final Exam:
  - Sec 01 Sat May 6, 9am, LSRC B101
  - Sec 02 Tues May 2, 7pm, LSRC B101
  - Get accommodations? Fill out form for Final Exam
  - Room for some to take final with the other section
  - Must fill out form by Next FRIDAY, April 28.

Calculate Your Grade

- From “About” tab on course web page

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%

<table>
<thead>
<tr>
<th>Labs</th>
<th>5%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reading Quizzes</td>
<td>5%</td>
</tr>
<tr>
<td>Lecture Group work</td>
<td>5%</td>
</tr>
<tr>
<td>Apts</td>
<td>12%</td>
</tr>
<tr>
<td>Programming Assignments</td>
<td>12%</td>
</tr>
<tr>
<td>APT Quizzes</td>
<td>6%</td>
</tr>
<tr>
<td>Two Midterm Exams</td>
<td>30%</td>
</tr>
<tr>
<td>final exam</td>
<td>25%</td>
</tr>
</tbody>
</table>
More Announcements

• Be a UTA for CompSci 101
  – Rewarding and Learning Experience
  – Apply: http://www.cs.duke.edu/csed/uta

• Today:
  – Finish from last time
  – 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
  – Anonymous Feedback for course
  – Anonymous feedback on UTAs

Review Recursion and Regex
bit.ly/101s17-0420-1

Dictionary Comprehension

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

• Format
  d = { key:value for key in somelist if ....}
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
- A dictionary may really be a list underneath, it’s 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) = last 2 digits mod 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)

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>267-89-5431</td>
<td>31 % 11 = 9</td>
<td>John Smith</td>
<td>703-25-6141</td>
<td>41 % 11 = 8</td>
<td>Jack Adams</td>
<td>319-86-2115</td>
<td>15 % 11 = 4</td>
<td>Betty Harris</td>
<td>476-82-5120</td>
<td>20 % 11 = 9</td>
</tr>
</tbody>
</table>

Hashing, dictionaries
bit.ly/101s17-0420-1A
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

```python
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)]
a1
[('a', 2, 0), ('b', 3, 0), ('c', 2, 5),
 ('d', 2, 4), ('e', 1, 4), ('f', 2, 0)]
a2
[('e', 1, 4), ('a', 2, 0), ('f', 2, 0),
 ('d', 2, 4), ('c', 2, 5), ('b', 3, 0)]
```

Two-pass (or more) sorting

```python
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), ('c', 2, 5),
 ('d', 2, 4), ('e', 1, 4), ('f', 2, 0)]
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?

```
[triangle, square, circle, star, rectangle]
```
Stable sorting: respect re-order

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

MedalTable APT
Sort items by their frequency, then sorted in frequencies.

```python
["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"
]
```

Timingsorts.py, what sort to call?

- Simple to understand, hard to do fast and at-scale
  - Scaling is what makes computer science …
    - Efficient algorithms don't matter on lists of 100 or 1000
  - Named algorithms in 201 and other courses
    - bubble sort, selection sort, merge, quick, …
    - See next slide and TimingSorts.py

- Basics of algorithm analysis: theory and practice
  - We can look at empirical results, would also like to be able to look at code and analyze mathematically! How does algorithm scale?

New sorting algorithms happen …

- timsort is standard on…
  - Python as of version 2.3, Android, Java 7
  - According to http://en.wikipedia.org/wiki/Timsort
    - Adaptive, stable, natural mergesort with supernatural performance

- What is mergesort? Fast and Stable
  - What does this mean?
  - Which is most important?
  - Nothing is faster, what does that mean?
  - Quicksort is faster, what does that mean?
### TimingSorts.py

<table>
<thead>
<tr>
<th>size</th>
<th>create</th>
<th>bubble</th>
<th>select</th>
<th>timsort</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>0.026</td>
<td>0.127</td>
<td>0.081</td>
<td>0.002</td>
</tr>
<tr>
<td>2000</td>
<td>0.045</td>
<td>0.537</td>
<td>0.273</td>
<td>0.001</td>
</tr>
<tr>
<td>3000</td>
<td>0.058</td>
<td>1.126</td>
<td>0.646</td>
<td>0.002</td>
</tr>
<tr>
<td>4000</td>
<td>0.082</td>
<td>2.174</td>
<td>1.208</td>
<td>0.003</td>
</tr>
<tr>
<td>5000</td>
<td>0.101</td>
<td>3.521</td>
<td>1.862</td>
<td>0.003</td>
</tr>
<tr>
<td>6000</td>
<td>0.118</td>
<td>4.617</td>
<td>3.005</td>
<td>0.004</td>
</tr>
<tr>
<td>7000</td>
<td>0.168</td>
<td>7.504</td>
<td>4.237</td>
<td>0.005</td>
</tr>
<tr>
<td>8000</td>
<td>0.156</td>
<td>9.074</td>
<td>6.152</td>
<td>0.007</td>
</tr>
<tr>
<td>9000</td>
<td>0.184</td>
<td>11.611</td>
<td>8.089</td>
<td>0.007</td>
</tr>
<tr>
<td>10000</td>
<td>0.212</td>
<td>14.502</td>
<td>9.384</td>
<td>0.008</td>
</tr>
</tbody>
</table>

### TimingSorts.py Questions

bit.ly/101s17-0420-3

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

1 9 5 4 1 3 6 - find smallest, swap
1 5 4 9 3 6 - end of 1st pass
1 5 4 9 3 6 - find smallest, swap
1 3 4 9 5 6 - end of 2nd pass
1 3 4 9 5 6 - find smallest, swap
1 3 4 5 9 6 - find smallest, swap
1 3 4 5 6 9 - end of 5th pass, done

Selection Sort (cont.)


- 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
Code for Selection Sort

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