Compsci 101
Stable Sorting, Review

Susan Rodger
April 6, 2023
U is for ...

- **URL**
  - https://duke.edu

- **Usenet**
  - Original source of FAQ, Flame, Spam, more

- **UI and UX**
  - User is front and center
Dr. danah boyd is a Principal Researcher at Microsoft Research, ... and a Visiting Professor at New York University. Her research is focused on addressing social and cultural inequities by understanding the relationship between technology and society.

“If I have learned one thing from my research, it’s this: social media services like Facebook and Twitter are providing teens with new opportunities to participate in public life, and this, more than anything else, is what concerns many anxious adults.”
Announcements

• APT-6 due Thursday, April 13
• Assignment 5 Clever GuessWord due tonight
• Assignment 6 Recommender out – due in two weeks
  • Discuss next time
  • Read through assignment before then

• Lab 9 Friday
  • There is a prelab!

• Exam 3 is Tuesday!
Exam 3– Tues, April 11

• Exam is in class on paper – 10:15am
  • Need pen or pencil
• See materials under 4/11 date
  • Exam 3 Reference sheet - part of exam
• Covers
  • topics
  • APTs through APT6
  • Labs through Lab 9
  • Assignments through Assignment 5
Exam 3 topics include ...

- List, tuples, list comprehensions
- Loops – for loop, while loop, indexing with a loop
- Reading from a file
  - Converting data into a list of things
- Parallel lists
- Sets – solving problems
- Dictionaries – solving problems
- Sorting – lists, tuples
- No turtles, no images - but note we are practicing other concepts with images
Exam 3

• Exam 3 is your own work!
• No looking at other people’s exam
• You cannot use any notes, books, computing devices, calculators, or any extra paper
• Bring only a pen or pencil
• The exam has extra white space and has the Exam 3 reference sheet as part of the exam.

• Do not discuss any problems on the exam with others until it is handed back
Exam 3 – How to Study

• Practice writing code on paper!
• Rewrite an APT
• Try to write code from lecture from scratch
• Try to write code from lab from scratch
• Practice from old exams
• Put up old Sakai quizzes, but better to practice writing code
• Look at Exam 3 reference sheet when writing code!
PFTD

• Review Sorting
• An APT MedalTable
• Review for Exam 3
### APT: Medal Table


#### Leaderboard

<table>
<thead>
<tr>
<th>Rank</th>
<th>Country</th>
<th>Gold</th>
<th>Silver</th>
<th>Bronze</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Norway</td>
<td>14</td>
<td>14</td>
<td>11</td>
<td>39</td>
</tr>
<tr>
<td>2</td>
<td>Germany</td>
<td>14</td>
<td>10</td>
<td>7</td>
<td>31</td>
</tr>
<tr>
<td>3</td>
<td>Canada</td>
<td>8</td>
<td>10</td>
<td>11</td>
<td>29</td>
</tr>
<tr>
<td>4</td>
<td>United States</td>
<td>6</td>
<td>6</td>
<td>9</td>
<td>21</td>
</tr>
<tr>
<td>5</td>
<td>Netherlands</td>
<td>6</td>
<td>6</td>
<td>8</td>
<td>20</td>
</tr>
<tr>
<td>6</td>
<td>Sweden</td>
<td>1</td>
<td>6</td>
<td>7</td>
<td>14</td>
</tr>
</tbody>
</table>
APT: MedalTable

**Problem Statement**

The Olympic Games will be held, and have been held (and might be being held). Given the results of the olympic disciplines, generate and return the medal table.

The results of the disciplines are given as a string list of results, where each element is in the format "GGG SSS BBB". GGG, SSS and BBB are the 3-letter country codes (three capital letters from 'A' to 'Z') of the countries winning the gold, silver and bronze medal, respectively.

The medal table is a string list with an element for each country appearing in results. Each element has to be in the format "CCO G S B" (quotes for clarity), where G, S and B are the number of gold, silver and bronze medals won by country CCO, e.g. "AUT 1 4 1". The numbers should not have any extra leading zeros.

Sort the elements by the number of gold medals won in decreasing order. If several countries are tied, sort the tied countries by the number of silver medals won in decreasing order. If some countries are still tied, sort the tied countries by the number of bronze medals won in decreasing order. If a tie still remains, sort the tied countries by their 3-letter code in ascending alphabetical order.

1. "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" ]
Tracking the Data

• What do we need to obtain for each country?
  • What's the data, how do we store it?
  • What's the data, how do we calculate it?

• Method and code to transform input
  • What will we store, how do we initialize/update
  • Verifying we've done this properly
Use a dictionary?

• 3 dictionaries
  • Country to Gold count
  • Country to Silver count
  • Country to Bronze count
Use a dictionary?

• 3 dictionaries
  • Country to Gold count
  • Country to Silver count
  • Country to Bronze count

• 1 dictionary
  • Map country to [gld cnt, sil cnt, bro cnt]
Example

• How would we create a dictionary for:

[“KOR TPE UKR”, “KOR KOR TPE”, “KOR JPN JPN”]
Example:

- Process first string: "KOR TPE UKR"

- Process second string: "KOR KOR TPE"
Example: dictionary $d$

- **Process first string: "KOR TPE UKR"**
  - $d["KOR"] = [1, 0, 0]$  # gold
  - $d["TPE"] = [0, 1, 0]$  # silver
  - $d["UKR"] = [0, 0, 1]$  # bronze

- **Process second string: "KOR KOR TPE"**
  - $d["KOR"] = [2, 0, 0]$  # gold
  - $d["KOR"] = [2, 1, 0]$  # silver
  - $d["TPE"] = [0, 1, 1]$  # bronze
Example: (2)

- What we have so far:

- Process third string: "KOR JPN JPN"
Example: dictionary $d$ (2)

- What we have so far:
  - $d[“KOR”] = [2, 1, 0]$
  - $d[“TPE”] = [0, 1, 1]$
  - $d[“UKR”] = [0, 0, 1]$

- Process third string: "KOR JPN JPN"
  - $d[“KOR”] = [3, 1, 0] \# gold$
  - $d[“JPN”] = [0, 1, 0] \# silver$
  - $d[“JPN”] = [0, 1, 1] \# bronze$
Sorting the Data

• Write a helper function to build the dictionary
  • d = build_dict(results)
  • Where results is string of countries for each event
• Use dictionary to get list of tuples

[('JPN', [0, 1, 1]), ('KOR', [3, 1, 0]),
 ('TPE', [0, 1, 1]), ('UKR', [0, 0, 1])]
Some of the code

```python
def build_dict(data):
    d = {}
    for item in data:
        [gld, sil, bro] = item.split()
```

Use three variables for the three countries
def builddict(data):
    d = {}
    for item in data:
        [gld, sil, bro] = item.split()
        if gld not in d:
            d[gld] = [0, 0, 0]

If country not in, set to [0, 0, 0] since has no medals counted yet
Some of the code

def build_dict(data):
    d = {}
    for item in data:
        [gld, sil, bro] = item.split()
        if gld not in d:
            d[gld] = [0, 0, 0]
        d[gld][0] += 1

Update appropriate index, [0] is the gold count
Review for Exam 3
Problem 4 Fall 2014 Old Tests

• A programming contest between colleges
• There are several problems to solve:
  • Problem A through Problem J
• Submit a program for a problem – it is correct or not
• Submit it again if it is not correct.
• Score is total time for problems solved with 20 minute penalty for each wrong submission that was solved eventually!
• Winner solves most problems – Tie breaker (lowest score

4/6/23
Compsci 101, Spring 2023
26
Review for Exam 3
Problem 4 Fall 2014 Old Tests

• Each entry is: 1) school, 2) name of problem, 3) time to solve in minutes, 4) correct or not

• Examples:
  ['UNC', 'A', '20', 'reject']

  ['Duke', 'A', '26', 'correct']
Review for Exam 3
Problem 4 Fall 2014 Old Tests

• Each entry is: 1) school, 2) name of problem, 3) time to solve in minutes, 4) correct or not

• Examples:
  ['UNC', 'A', '20', 'reject']
  UNC submitted problem A in 20 minutes - rejected

  ['Duke', 'A', '26', 'correct']
  Duke submitted Problem A in 26 minutes - correct
Problem 4 Fall 2014 Old tests
Just look at Duke’s submissions

[...]
['Duke', 'A', '26', 'correct'],
['Duke', 'E', '82', 'reject'],
['Duke', 'D', '200', 'correct'],

['Duke', 'E', '210', 'correct'],

\textbf{Duke score:}

26\text{not correct, no points} + 200 = 226

plus 210 + 20 (penalty) = 456

\textbf{Duke has 456 points}
Problem 4 Fall 2014 Old tests
Just look at Duke’s submissions

Duke score:

- 'Duke', 'A', '26', 'correct': 26
- 'Duke', 'E', '82', 'reject': not correct, no points
- 'Duke', 'D', '200', 'correct': plus 200 = 226
- 'Duke', 'E', '210', 'correct': plus 210 + 20 (penalty) = 456

Duke has 456 points
Problem 4 Fall 2014 Old tests

data is list of lists of submissions

```python
data = [
    ['UNC', 'A', '20', 'reject'],
    ['Duke', 'A', '26', 'correct'],
    ['UNC', 'A', '33', 'reject'],
    ['ECU', 'A', '34', 'correct'],
    ['Elon', 'A', '34', 'correct'],
    ['USC', 'G', '44', 'reject'],
    ['UNC', 'A', '45', 'correct'],
    ['NCSU', 'B', '60', 'reject'],
    ['USC', 'C', '72', 'reject'],
    ['Duke', 'E', '82', 'reject'],
    ['USC', 'C', '90', 'correct'],
    ['UNC', 'B', '98', 'reject'],
    ['NCSU', 'B', '103', 'correct'],
    ['NCSU', 'A', '115', 'correct'],
    ['USC', 'A', '116', 'correct'],
    ['ECU', 'F', '202', 'reject'],
    ['Duke', 'D', '200', 'correct'],
    ['Duke', 'E', '210', 'correct'],
    ['UNC', 'B', '212', 'reject'],
    ['USC', 'G', '220', 'reject'],
    ['NCSU', 'D', '222', 'correct'],
    ['Elon', 'H', '225', 'correct'],
    ['NCSU', 'H', '230', 'reject']
]```

Write function `listOfSchools(data)`

- returns sorted unique list of schools that submitted a program whether correct or not
- From data should return:

  ['Duke', 'ECU', 'Elon', 'NCSU', 'UNC', 'USC'].

4/6/23 Compsci 101, Spring 2023 32
Write function listOfSchools(data)

- returns sorted unique list of schools that submitted a program whether correct or not
- From data should return:

  ['Duke', 'ECU', 'Elon', 'NCSU', 'UNC', 'USC'].

Note: sorted
Unique schools (use sets)
Returns list (must convert set back to list)
Write function listOfSchools(data)

def listOfSchools(data):
    setSchools = set()
    for item in data:
        setSchools.add(item[0])
    alist = list(setSchools)
    return sorted(alist)
Write function `listOfSchools(data)`

def listOfSchools(data):
    schools = []
    for item in data:
        schools.append(item[0])
    aset = set(schools)
    return sorted(aset)
Problem 4 Fall 2014 Old tests

data is list of lists of submissions

data = [
    ['UNC', 'A', '20', 'reject'],
    ['Duke', 'A', '26', 'correct'],
    ['UNC', 'A', '33', 'reject'],
    ['ECU', 'A', '34', 'correct'],
    ['Elon', 'A', '34', 'correct'],
    ['USC', 'G', '44', 'reject'],
    ['UNC', 'A', '45', 'correct'],
    ['NCSU', 'B', '60', 'reject'],
    ['USC', 'C', '72', 'reject'],
    ['Duke', 'E', '82', 'reject'],
    ['USC', 'C', '90', 'correct'],
    ['UNC', 'B', '98', 'reject'],
    ['NCSU', 'B', '103', 'correct'],
    ['NCSU', 'A', '115', 'correct'],
    ['USC', 'A', '116', 'correct'],
    ['ECU', 'F', '202', 'reject'],
    ['Duke', 'D', '200', 'correct'],
    ['Duke', 'E', '210', 'correct'],
    ['UNC', 'B', '212', 'reject'],
    ['USC', 'G', '220', 'reject'],
    ['NCSU', 'D', '222', 'correct'],
    ['Elon', 'H', '225', 'correct'],
    ['NCSU', 'H', '230', 'reject']
]
Write function problemsAttempted(data)

• Returns list of problems attempted
• **Would return list:**
  • Note doesn’t say anything about the order but implies one of each.
Write function problemsAttempted(data)

• Returns list of problems attempted
• **Would return list:**
  • Note doesn’t say anything about the order but implies one of each.

• **Need to loop over the lists in data**
  • Collect the names of problems attempted
  • Get the unique ones
Write function problemsAttempted(data)

def problemsAttempted(data):

Write function `problemsAttempted(data)`

def problemsAttempted(data):
    problems = set([])
    for item in data:
        problems.add(item[1])
    return list(problems)
Problem 4 Fall 2014 Old tests
data is list of lists of submissions

data = [
    ['UNC', 'A', '20', 'reject'],
    ['Duke', 'A', '26', 'correct'],
    ['UNC', 'A', '33', 'reject'],
    ['ECU', 'A', '34', 'correct'],
    ['Elon', 'A', '34', 'correct'],
    ['USC', 'G', '44', 'reject'],
    ['UNC', 'A', '45', 'correct'],
    ['NCSU', 'B', '60', 'reject'],
    ['USC', 'C', '72', 'reject'],
    ['Duke', 'E', '82', 'reject'],
    ['USC', 'C', '90', 'correct'],
    ['UNC', 'B', '98', 'reject'],
    ['NCSU', 'B', '103', 'correct'],
    ['NCSU', 'A', '115', 'correct'],
    ['USC', 'A', '116', 'correct'],
    ['ECU', 'F', '202', 'reject'],
    ['Duke', 'D', '200', 'correct'],
    ['Duke', 'E', '210', 'correct'],
    ['UNC', 'B', '212', 'reject'],
    ['USC', 'G', '220', 'reject'],
    ['NCSU', 'D', '222', 'correct'],
    ['Elon', 'H', '225', 'correct'],
    ['NCSU', 'H', '230', 'reject']
]
WOTO-3 Solving problems
Write function

problemsNotAttempted(problems, data)

• problems is a list of all possible problems
  • ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J']

• Returns a list of the problems not attempted
Write function
problemsNotAttempted(problems, data)

• problems is a list of all possible problems
  • ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J']

• Returns a list of the problems not attempted

• We know how to get all the problems attempted!
  • Call problemsAttempted!

• Put both lists in sets and set operation!
Write function
problemsNotAttempted(problems, data)

def problemsNotAttempted(problems, data):

Write function
problemsNotAttempted(problems, data)

def problemsNotAttempted(problems, data):
    attempted = problemsAttempted(data)
    setProbs = set(problems)
    setAttempted = set(attempted)
    setNotAttempted = setProbs - setAttempted
    return sorted(setNotAttempted)
Problem 4 Fall 2014 Old tests

data is list of lists of submissions

data = [
    ['UNC', 'A', '20', 'reject'],
    ['Duke', 'A', '26', 'correct'],
    ['UNC', 'A', '33', 'reject'],
    ['ECU', 'A', '34', 'correct'],
    ['Elon', 'A', '34', 'correct'],
    ['USC', 'G', '44', 'reject'],
    ['UNC', 'A', '45', 'correct'],
    ['NCSU', 'B', '60', 'reject'],
    ['USC', 'C', '72', 'reject'],
    ['Duke', 'E', '82', 'reject'],
    ['USC', 'C', '90', 'correct'],
    ['UNC', 'B', '98', 'reject'],
    ['NCSU', 'B', '103', 'correct'],
    ['NCSU', 'A', '115', 'correct'],
    ['USC', 'A', '116', 'correct'],
    ['ECU', 'F', '202', 'reject'],
    ['Duke', 'D', '200', 'correct'],
    ['Duke', 'E', '210', 'correct'],
    ['UNC', 'B', '212', 'reject'],
    ['USC', 'G', '220', 'reject'],
    ['NCSU', 'D', '222', 'correct'],
    ['Elon', 'H', '225', 'correct'],
    ['NCSU', 'H', '230', 'reject']
]
Write function

dictProblemstoSchoolsSolved(data)

• Returns a dictionary of letters for problems mapped to list of schools that solved that problem
  • ‘B’ mapped to ['NCSU']
  • ‘A’ mapped to ['Duke', 'ECU', 'Elon', 'UNC', 'NCSU', 'USC']
  • ‘D’ mapped to ['Duke', 'NCSU']
  • Etc
Write function 

dictProblemstoSchoolsSolved(data)

• Returns a dictionary of letters for problems mapped to list of schools that solved that problem
  • ‘B’ mapped to ['NCSU']
  • ‘A’ mapped to ['Duke', 'ECU', 'Elon', 'UNC', 'NCSU', 'USC']
  • ‘D’ mapped to ['Duke', 'NCSU']
  • Etc

• Each letter - create a list and append schools to it
Write function

dictProblemstoSchoolsSolved(data)

def dictProblemstoSchoolsSolved(data):
    d = {}

def dictProblemsToSchoolsSolved(data):
    d = {}
    for item in data:
        if item[3] == 'correct':
            if item[1] in d:  # already in
                d[item[1]].append(item[0])
            else:  # not in yet, add
                d[item[1]] = [item[0]]
    return d
data is list of lists of submissions

data = [
    ['UNC', 'A', '20', 'reject'],
    ['Duke', 'A', '26', 'correct'],
    ['UNC', 'A', '33', 'reject'],
    ['ECU', 'A', '34', 'correct'],
    ['Elon', 'A', '34', 'correct'],
    ['USC', 'G', '44', 'reject'],
    ['UNC', 'A', '45', 'correct'],
    ['NCSU', 'B', '60', 'reject'],
    ['USC', 'C', '72', 'reject'],
    ['Duke', 'E', '82', 'reject'],
    ['USC', 'C', '90', 'correct'],
    ['UNC', 'B', '98', 'reject'],
    ['NCSU', 'B', '103', 'correct'],
    ['NCSU', 'A', '115', 'correct'],
    ['USC', 'A', '116', 'correct'],
    ['ECU', 'F', '202', 'reject'],
    ['Duke', 'D', '200', 'correct'],
    ['Duke', 'E', '210', 'correct'],
    ['UNC', 'B', '212', 'reject'],
    ['USC', 'G', '220', 'reject'],
    ['NCSU', 'D', '222', 'correct'],
    ['Elon', 'H', '225', 'correct'],
    ['NCSU', 'H', '230', 'reject'] ]
Write function
dictSchoolsToNumSubmissions(data)

- Returns a dictionary of schools mapped to the number of submissions they had (rejected or correct)
  - ‘Duke’ mapped to 4
  - ‘UNC’ mapped to 5
  - Etc
Write function

dictSchoolsToNumSubmissions(data)

• Returns a dictionary of schools mapped to the number of submissions they had (rejected or correct)
  • ‘Duke’ mapped to 4
  • ‘UNC’ mapped to 5
  • Etc
• Counting dictionary!
Write function
dictSchoolsToNumSubmissions(data)

def dictSchoolsToNumSubmissions(data):
    d = {}
    for item in data:
        if item[0] in d:
            d[item[0]] += 1
        else:
            d[item[0]] = 1
    return d
Write function
dictSchoolsToNumSubmissions(data)

def dictSchoolsToNumSubmissions(data):
    d = {}
    for item in data:
        if item[0] in d:
            d[item[0]] += 1
        else:
            d[item[0]] = 1
    return d
WOTO-4 Solving problems
Problem 4 Fall 2014 Old tests

data is list of lists of submissions

data = [ ['UNC', 'A', '20', 'reject'],
['Duke', 'A', '26', 'correct'],
['UNC', 'A', '33', 'reject'],
['ECU', 'A', '34', 'correct'],
['Elon', 'A', '34', 'correct'],
['USC', 'G', '44', 'reject'],
['UNC', 'A', '45', 'correct'],
['USC', 'G', '44', 'reject'],
['UNC', 'B', '98', 'reject'],
['NCSU', 'B', '103', 'correct'],
['NCSU', 'A', '115', 'correct'],
['USC', 'A', '116', 'correct'],
['ECU', 'F', '202', 'reject'],
['Duke', 'D', '200', 'correct'],
['Duke', 'E', '210', 'correct'],
['UNC', 'B', '212', 'reject'],
['USC', 'G', '220', 'reject'],
['NCSU', 'D', '222', 'correct'],
['Elon', 'H', '225', 'correct'],
['NCSU', 'H', '230', 'reject'] ]
Write function easiestProblem(data)

- Returns a tuple of two items
  - The name of the problem that was solved by the most schools
  - A sorted list of the schools that solved that problem
- If a tie, then pick any one
- Returns:
Write function easiestProblem(data)

• Need to calculate the problem that was solved the most
• Need to find that problem’s list of schools in the dictionary we already built
  • Will need to call that function
• Can do both as you walk through the dictionary!
Write function easiestProblem(data)

def easiestProblem(data):

d = dictProblemsToSchoolsSolved(data)
maxProb = ('',[])
for (key,value) in d.items():
    if len(value) > len(maxProb[1]):
        maxProb = (key, value)
return maxProb
Write function easiestProblem(data)

def easiestProblem(data):
    d = dictProblemsToSchoolsSolved(data)
    maxProb = ("", [])
    for (key, value) in d.items():
        if len(value) > len(maxProb[1]):
            maxProb = (key, value)
    return maxProb