Compsci 101, Sorting and Testing

Owen Astrachan
Kristin Stephens-Martinez

March 29, 2018

S is for …

- **Software**
  - Joy, sorry, fun, changing the world
- **System and sys**
  - Connecting to the machine at different levels
- **Sorting**
  - From hat to tim to more

Plan for the Week

- Sorting out sorts
  - Why we might want to use a simple sort even when timsort is faster
  - The ola sorting sojourn

- Toward getting APTs and Clever Hangman done
  - Testing and then more testing

Two Pass WOTO

http://bit.ly/101spring18-march27-1

WOTO Second Pass

http://bit.ly/101spring18-march27-2

All time Medal Table?

Utility and Awareness

- Parallel lists: sorting related lists together

```
>>> names = ['fran', 'chris', 'sam', 'joe']
>>> grades = [80, 90, 85, 88]
>>> names
['fran', 'chris', 'sam', 'joe']
>>> grades
[80, 90, 85, 88]
>>> combo = zip(grades,names)
>>> combo
<zip object at 0x1021b9b48>
>>> lcombo = list(combo)
>>> lcombo
[(80, 'fran'), (90, 'chris'), (85, 'sam'), (88, 'joe')]
```

Finishing things up, but …

- Extracting individual lists from tuple-ized lists

```
>>> lcombo
[(80, 'fran'), (90, 'chris'), (85, 'sam'), (88, 'joe')]
>>> scombo = sorted(lcombo)
>>> scombo
[(80, 'fran'), (85, 'sam'), (88, 'joe'), (90, 'chris')]
>>> names = [t[1] for t in scombo]
>>> grades = [t[0] for t in scombo]
>>> names
['fran', 'sam', 'joe', 'chris']
>>> grades
[80, 85, 88, 90]
```

Selection sort, TimingSorts.py

- Selection sort
  - Find smallest element, swap into **`list[0]`**
  - Repeat indexes 1, 2, … smallest in **`list[i:]`**

- Nested function **`minIndex`** in **`selectsort`**
  - What is scope of minIndex?

- Notice how variables are swapped
  - This is actually tuple assignment

Bubblesort, TimingSorts.py

- Bubblesort
  - Swap adjacent elements that are out of order
  - Am I bigger than you? Swap. Largest at end
  - Repeat **`[0:-1], [0:-2], [0:-n]`**

- As with selection sort, we could easily sort parallel lists that are small. How small? Analyze!

Empirical and Theoretical Analyses

size	create	bubble	select	timsort
1000	0.026	0.127	0.081	0.002
2000	0.045	0.537	0.273	0.001
3000	0.058	1.126	0.646	0.002
4000	0.082	2.174	1.208	0.003
50s0	0.101	3.521	1.862	0.003
6000	0.118	4.617	3.005	0.004
7000	0.168	7.504	4.237	0.005
8000	0.156	9.074	6.152	0.007
9000	0.184	11.611	8.089	0.007
10000	0.212	14.502	9.384	0.008

Sir Anthony (Tony) Hoare

"*There are two ways of constructing a software design. One way is to make it so simple that there are obviously no deficiencies. And the other way is to make it so complicated that there are no obvious deficiencies.*"

Turing award, Invented Quicksort, Born in Sri Lanka, educated in England

Tim Peters

https://en.wikipedia.org/wiki/Zen_of_Python

https://www.python.org/dev/peps/pep-0020/

Timsort and Zen of Python

Beautiful is better than ugly

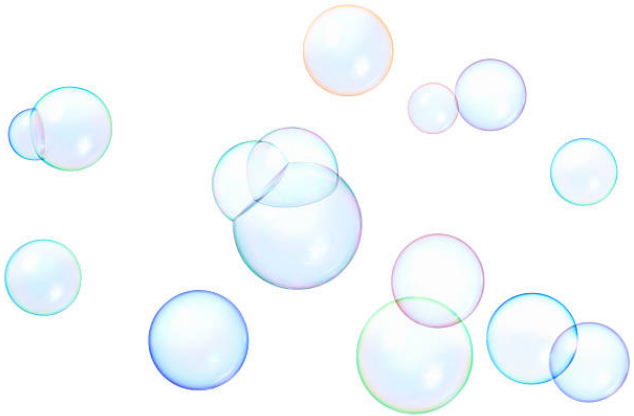
Simple is better than complex

Complex is better than complicated

Readability counts

I've even been known to get Marmite *near* my mouth -- but never actually in it yet. Vegamite is right out

Bubblesort: ola's odyssey



1975: Quicksort in class

W28007
Owen Astrachan

(note school form)

95

```
LIS
WHAT?

LIS

QUICK      17 NOV 75   17:43

100 BEGIN
110 INTEGER WORDSIZE,FREQSIZE,LINKSIZE,STORESIZE;
112 INTEGER HASHSIZE;
115 FREQSIZE:=LINKSIZE:=STORESIZE:=250;
117 HASHSIZE:=400;
120 WORDSIZE:=14;
130 BEGIN
140 REAL MEAN,VARIANCE;
150 INTEGER I,J,LAST,NUMWORD,MAXLEN,MAXFREQ;
160 INTEGER SUM,SUMSQUARE,NF,HASH;
170INTEGER P,T,R,MINLEN,LOOK,REF;
180 STRING WORD;
190 INTEGER ARRAY FREQ[1:FREQSIZE],LNGFREQ[1:WORDSIZE];
192 INTEGER ARRAY LINK[1:LINKSIZE],H
                                    ASHSTORE[1:HASH
                                                  SIZE];
```

Not needed

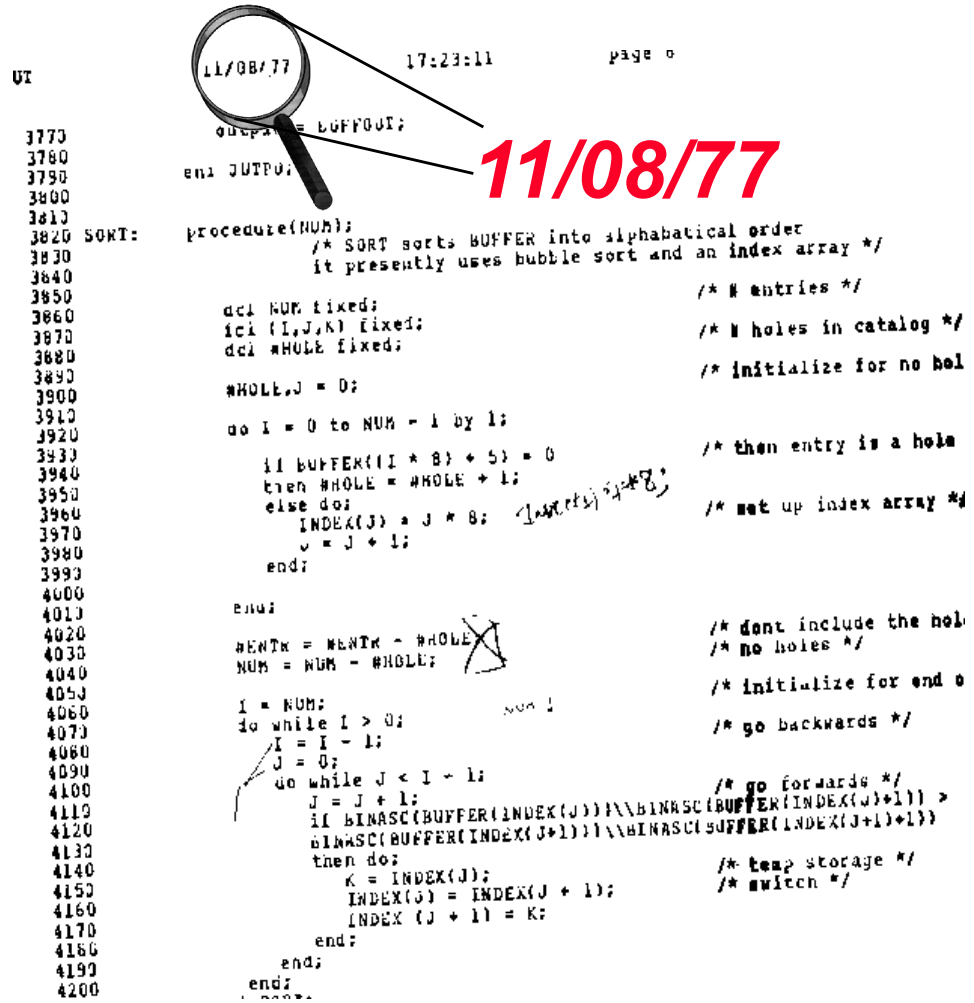
Can be tightened considerably

```
800 PROCEDURE QUICKSORT(I,J);VA
                                LUE I,J;INTEGER I,J
810 BEGIN
820 INTEGER OLDI,OLDJ;
830 BOOLEAN LEFT,RIGHT;
835 STRING TEMPSTORE;
840 TEMPSTORE:=STORE[I];
845 OLDI:=I;
846 OLDJ:=J;
850 WHILE ABS(
              I-J)>0
860 DO BEGIN
880     DO BEGIN
890       LEFT:=RIGH
                T:=TRUE;
1000        WHILE LEFT AND I<
1010        DO BEGIN
1020          IF STORE[J]<TEMPSTORE
1040          THEN BEGIN
1050                STORE[I]:=STORE[J];
1055            FREQ[I]:=FREQ[J];
1060              LEFT:=FALSE;
1070              I:=I+1;
1080            END
1090          ELSE J:=J-1;
1100          END;
1110        WHILE RIGHT AND I<J
1120        DO BEGIN
1130          IF STORE[I]>TEMPSTORE
1140          THEN BEGIN
1150                STORE[J]:=STORE[I];
1155            FREQ[J]:=FREQ[I];
1160              RIGHT:=FALSE;
1170              J:=J
                      -1;
1180            END
1190          ELSE I:=I+1;
1200          END;
1210    END;
1220 STORE[I]:=TEMPSTORE;
1230 P:=IF I=OLDI THEN OLDI ELSE I-1;
1280 QUICKSORT(OLDI,P);
1290 Q:=IF J=OLDJ THEN OLDJ ELSE J+1;
1300 QUICKSORT(Q,OLDJ);
1310END;
```

not needed

can be tightened considerably

1977, Uses Bubblesort for job



Hmmmm. What can I write?

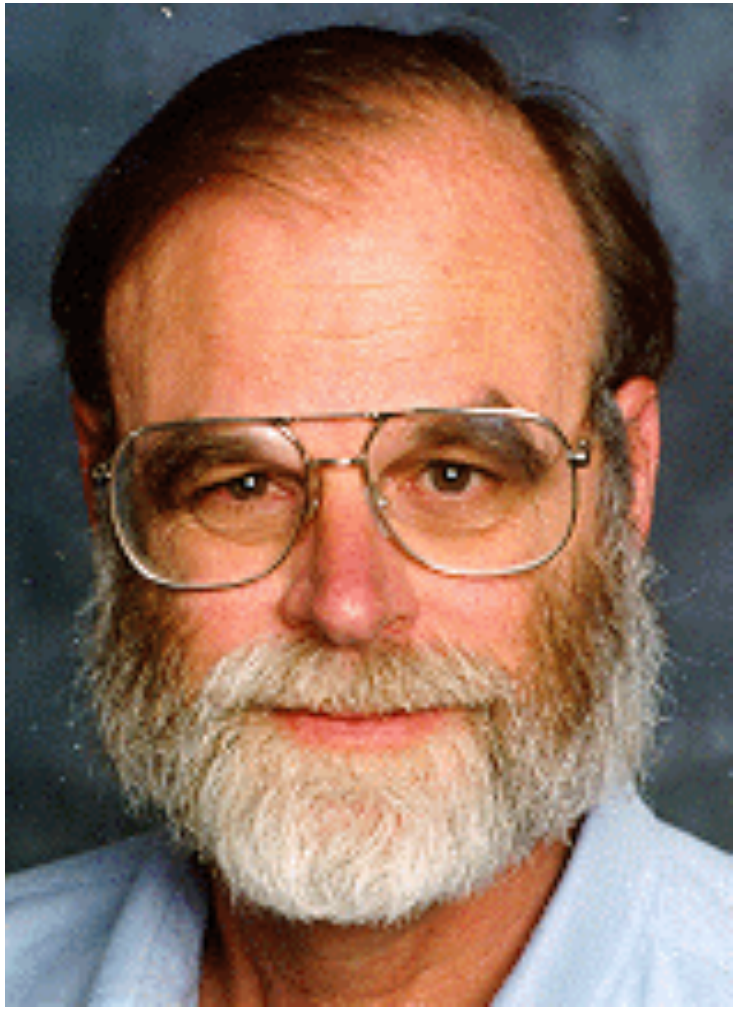
- Astrachan, O. (2003, February). Bubble sort: an archaeological algorithmic analysis. In *ACM SIGCSE Bulletin*(Vol. 35, No. 1, pp. 1-5). ACM.

- From 1955, to 1959, to 1962 to 1963 to today!

Jim Gray, Turing 1998

Bubble sort is a good argument for analyzing algorithm performance. It is a perfectly correct algorithm. But it's performance is among the worst imaginable. *So, it crisply shows the difference between correct algorithms and good algorithms.*

(italics ola's)



Brian Reid, Hopper 1982

Feah. I love bubble sort, and I grow weary of people who have nothing better to do than to preach about it. Universities are good places to keep such people, so that they don't scare the general public.

(continued)

Brian Reid

I am quite capable of squaring N with or without a calculator, and I know how long my sorts will bubble. I can type every form of bubble sort into a text editor from memory. If I am writing some quick code and I need a sort quick, as opposed to a quick sort, I just type in the bubble sort as if it were a statement. I'm done with it before I could look up the data type of the third argument to the quicksort library.

I have a dual-processor 1.2 GHz Powermac and it sneers at your N squared for most interesting values of N. And my source code is smaller than yours.

Brian Reid
who keeps all of his bubbles sorted anyhow.

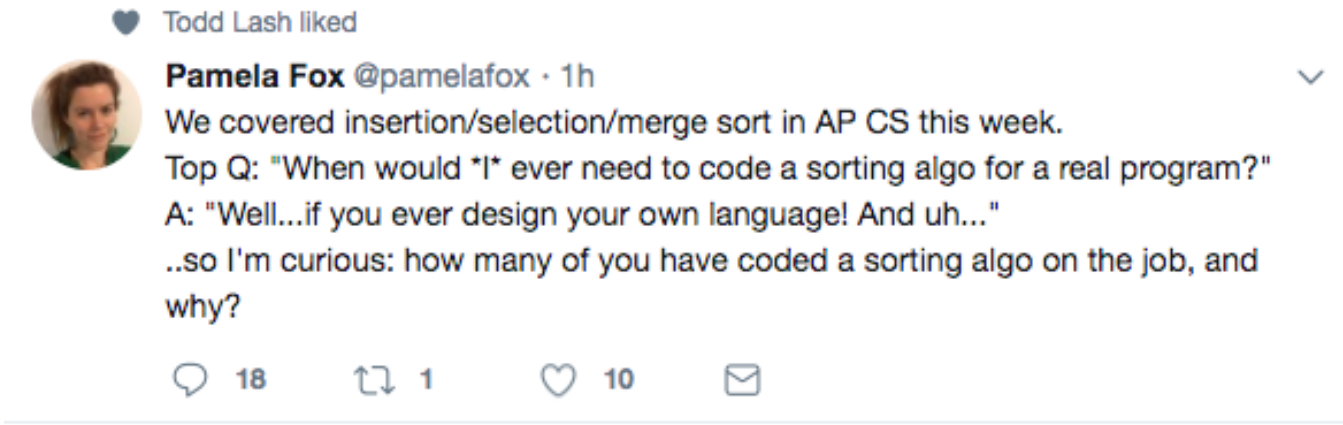
Niklaus Wirth, Turing 1984

I have read your article and share your view that Bubble Sort has hardly any merits. I think that it is so often mentioned, because it illustrates quite well the principle of sorting by exchanging.

I think BS is popular, because it fits well into a systematic development of sorting algorithms. But it plays no role in actual applications. Quite in contrast to C, also without merit (and its derivative Java), among programming codes.

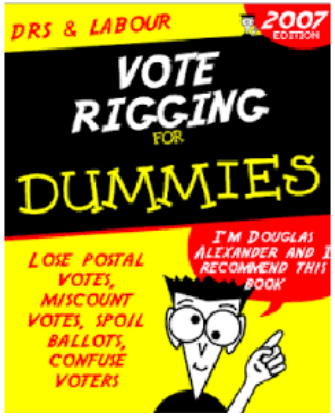
WOTO

http://bit.ly/101spring18-march29-1



VoteRigging APT

- http://www.cs.duke.edu/csed/pythonapt/voteriggi ng.html
- Example [5,10,7,3,8] answer is 4. Why?
  - From whom do you "steal" a vote? Why?
  - Similar to making change or clever hangman?



Testing Your Ideas

- How do we find maximal value in a list?
  - Can we leverage sorting?
  - Can we call max? Why will we need .index too?

- When can we stop stealing votes?
  - Mapping conceptual idea to code
  - Getting all green

What is the purpose of APTs?

- Testing one problem/idea via a single method
  - We know what inputs to method are
  - We know what expected output is
  - We use this to test and verify our code

- Unit Testing: one "unit" independing of other code
  - Often functions depend on each other
  - Unit testing is a single function

Clever v Plain Hangman

- What changes in the gameplay?
- Minor changes, though they require coding
  - Plain: show 'a', 'e', 't', 'w'
  - Clever: show _bcd_fghijklmnopqrs_uv_xyz
- Major changes
  - List of potential words changes at each turn
  - Function **`getNewWordList`** called from loop

Testing your code

- Alternative to lowerwords.txt?
  - Create your own file of words. Small file
  - Facilitates testing

- Call random.seed(123)
  - Same words/order every time you play
  - Reproduce errors more easily

Testing your methods

- Testing **`getNewWordlist(guess,letter,words)`**
  - From watching the DEBUG game play?
  - Perhaps test in isolation from game

- This calls **`createTemplate(template,word,letter)`**
  - How we test one without the other?
  - Testing each function separately: unit test!

Reveals Broken code!

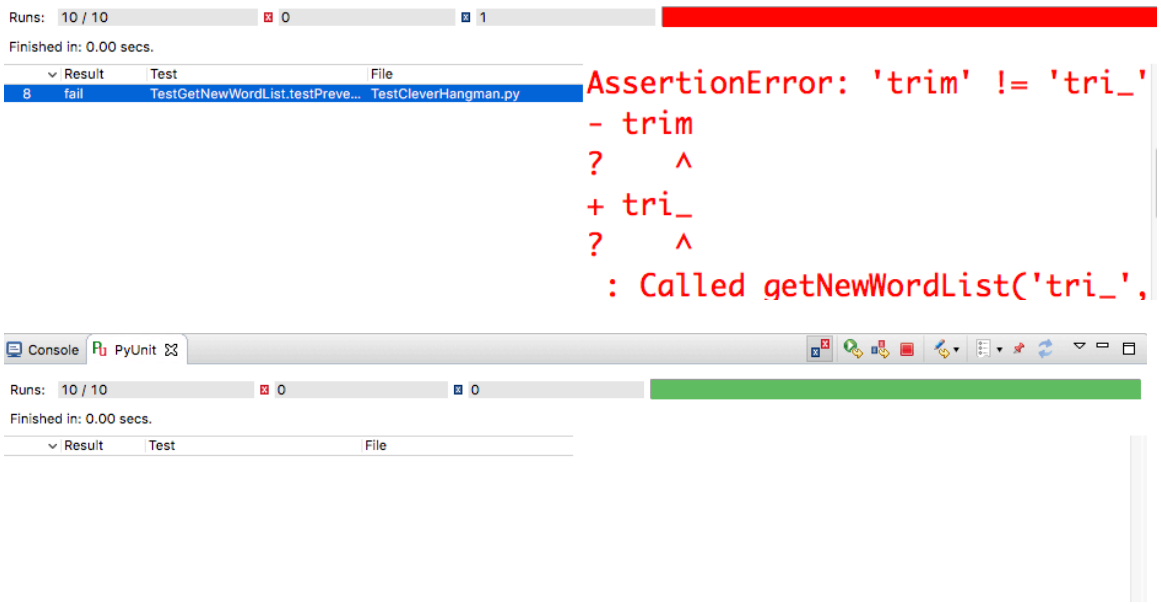
- What should this call return:

```
getNewWordList("tri_", 'm',
               ['trim', 'trio'])
```

- What are the two keys?
  - What are the lists/values these keys map to?
  - What will your code return?
    - If there's a tie? Force a miss!

Run Unit Test

- See code in TestCleverHangman.py
  - Calls functions in your code via API!



APT Green Dance

- SortByFreqs and SortedFreqs
  - Both leverage sorting? Freqs?
- MedalTable and CharityDonor
  - Dictionary and ordering/sorting/max
- VoteRigging, TrophyShelf, Badges
  - Greedy, Looping (helper function), Sets

Badges

- http://www.cs.duke.edu/csed/pythonapt/badges.html

```
def findLabel(labels,deeds,needs):
    ds = set(deeds)
    for dex in range(len(needs)):
        me = needs[dex].split()
        setme = set(me)
        if setme is subset of ds:
            return labels[dex]

    return "nobadge"
```

Questions