Theory, Practice, last two weeks

- Solving * problems requires developing algorithms
  - How do we … [at all | efficiently | at-scale | …]
- Solving * problems requires developing code
  - Implementing [algorithms | ideas | …]
- What's possible? How do we know?
  - How do I make a resizable table-like web page using CSS?
  - How do I store distributed responses in a concurrently-accessible database?
  - How do I sort a million 32-bit integers (see Obama)

* == computational

Why searching and sorting?

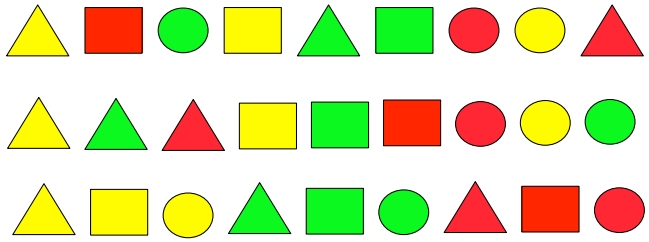
- Simple to understand, hard to do fast and at-scale
  - [do | do not] think of sorting as "arrange deck of cards"
  - Organizing data to facilitate [search | visualization | …]
  - Study named-algorithms in 201 and other courses
    - bubble sort, insertion sort, merge sort, quick sort, …
- Basics of algorithm analysis: theory and practice
  - How does algorithm scale, e.g., search and sort
  - n^2 and nlog n for sorts, n, log n, and constant for search
    - What does this mean?

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?

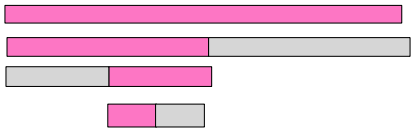
Stable, Stability

- What does the search query 'stable sort' show us?
  - Image search explained
  - Why are numeric examples so popular?



Binary Search

- Before the first guess, there are 1024 numbers



How many times can we divide list in half?

$\log_2(N)$ for N element list, why?

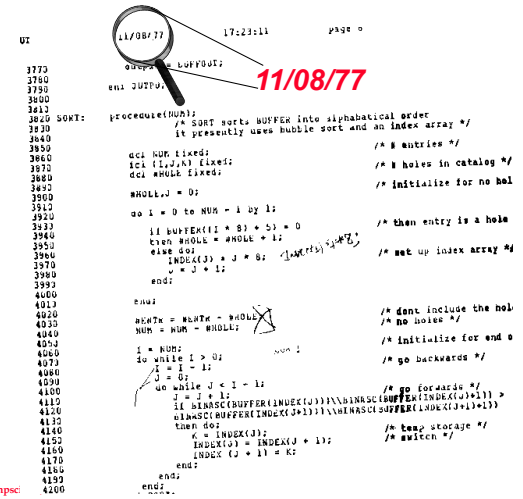
What must be true to use binary search?

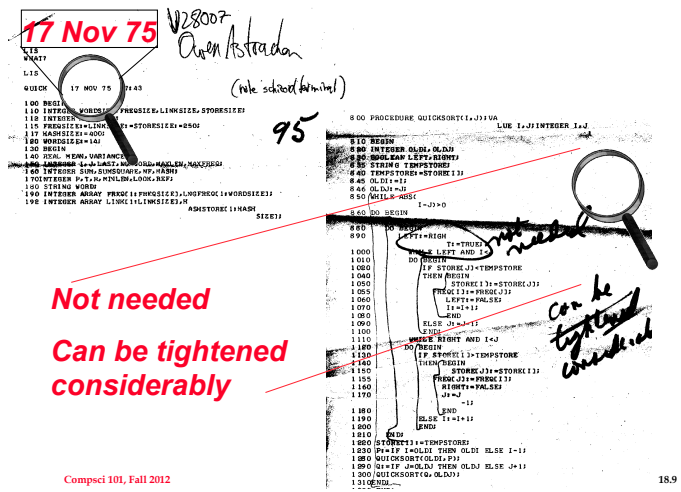
How is this done in Python?

Binary Search: raw mode

```
def binary_search(values,target):
    low = 0
    high = len(values)-1
    while low <= high:
        mid = (low+high)/2
        if values[mid] == target:
            return mid
        elif values[mid] < target:
            low = mid+1
        else:
            high = mid-1
    return -1
```

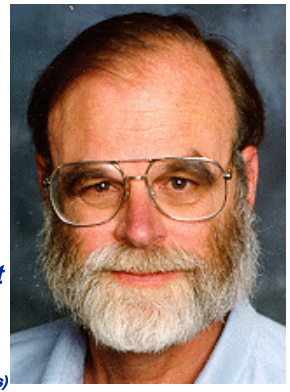

Bubble Sort, A Personal Odyssey





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 Award 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)

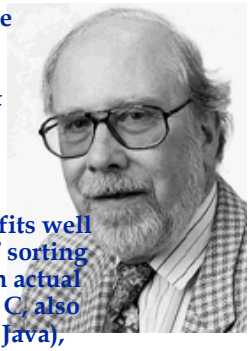
Brian Reid (Hopper 1982)

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 award 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.

Merge Sort: raw mode

```
def mergesort(values):
    def merge(low,mid,high):
        #not shown

    def dowork(low,high):
        if low < high:
            mid = (low+high)/2
            dowork(low,mid)
            dowork(mid+1,high)
            merge(low,mid,high)

    dowork(0,len(values)-1)
```


Quicksort: raw mode

```
def quicksort(values):
    def partition(low,high):
        #not shown

    def dowork(low,high):
        if low < high:
            pivot = partition(low,high)
            dowork(low,pivot-1)
            dowork(pivot+1,high)

    dowork(0,len(values)-1)
```


Shafi Goldwasser

- RCS professor of computer science at MIT
  - Twice Godel Prize winner
  - Grace Murray Hopper Award
  - National Academy
  - Co-inventor of zero-knowledge proof protocols

How do you convince someone that you know [a secret] without revealing the knowledge?

- Honesty and Privacy

Work on what you like, what feels right, I now of no other way to end up doing creative work