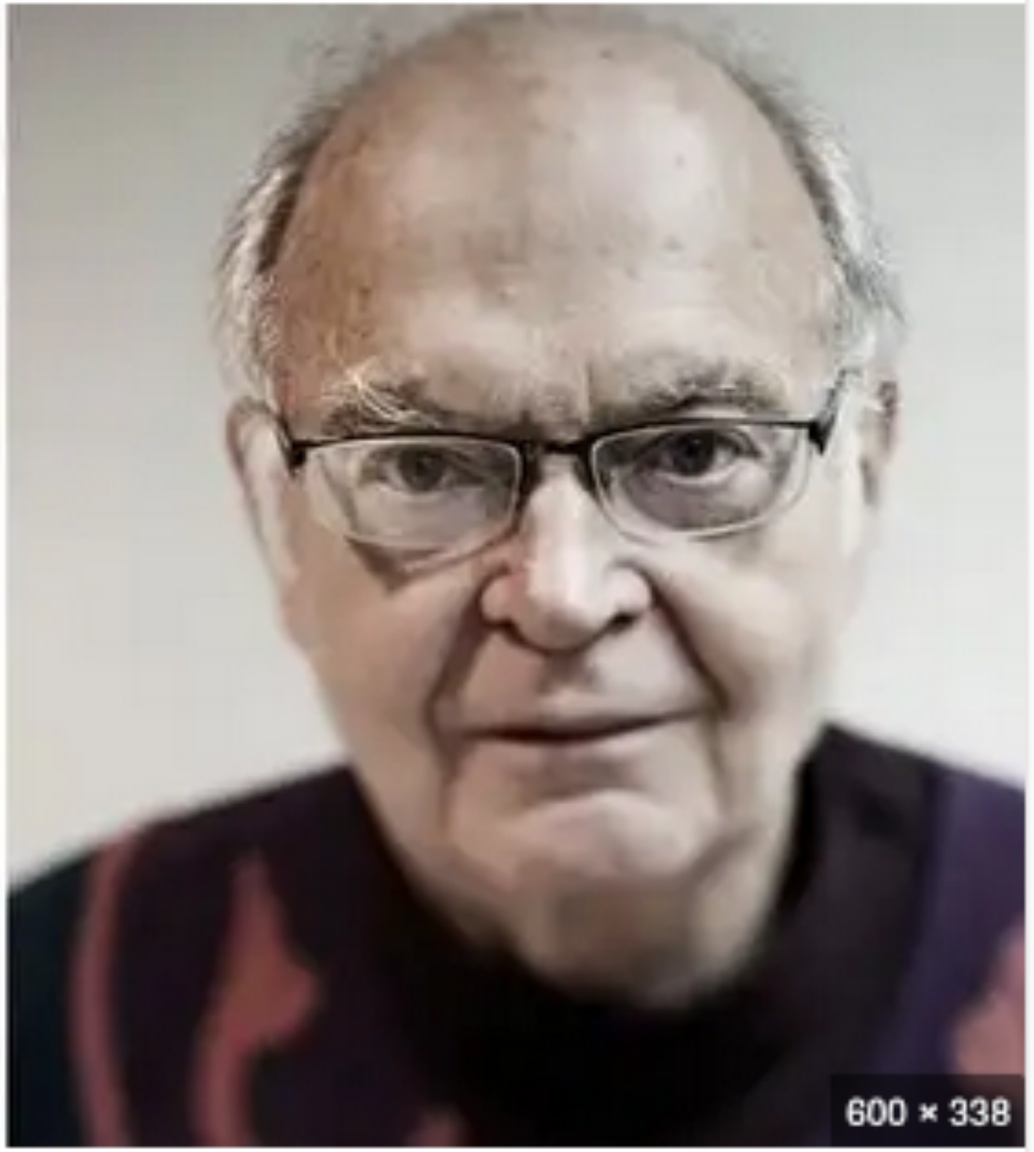
Efficiency does Matter

Part 2

Dec 4 (book ch 4.2)

> We should forget about small efficiencies, say about 97% of the time: **premature optimization is the root of all evil**

— Donald Knuth



FindMove Sort

return the sorted result, destroy the original

```
public static int[] fbSort(int[] temp) {
    int[] result = new int[temp.length];
    for (int i = 0; i < temp.length; i++) {
        int bestloc = -1;
        for (int j = 0; j < temp.length; j++) {
            if (temp[j] >= 0) {
                if (bestloc < 0) {
                    bestloc = j;
                } else {
                    if (temp[bestloc] > temp[j]) {
                        bestloc = j;
                    }
                }
            }
        }
        result[i] = temp[bestloc];
        temp[bestloc] = -1;
    }
    return result;
}
```

This implementation assumes that numbers to be sorted are all non-negative integers. It also assumes sorting in descending order

Sorting

- in fbSort you destroy the original
  - so if you care about the original you need to start by making a copy
- Need a reliable way to destroy items, or set so that same code can be used to sort in either direction
  - Objects
    - set to null
  - int
    - ???
- kind of slow

Do the sorting "in place"

Would be better to not destroy at all

Ideally do it faster

Bubble Sort

- idea, go across array
  - compare neighbors.
    - if neighbor is worse (better), SWAP

- After doing this once, what does the array look like

- So as with find, print, delete, need to repeat
  - faster than find, delete, print???

- Importantly .. NO delete

BubbleSort

- How many swaps?
  - Worst case?
- Can we improve??

```
public static void bubbleSort(int[] arr) {
    for (int i = 0; i < arr.length; i++) {
        for (int j = 1; j < (arr.length); j++) {
            if (arr[j-1] < arr[j]) {
                int temp = arr[j-1];
                arr[j-1] = arr[j];
                arr[j] = temp;
            }
        }
    }
}
```

Activity

Count operations for BubbleSort

1. Comparisons 2. Additions 3. Swaps

List 1	
0	10
1	22
2	54
3	86
4	56
5	15
6	55
7	7

List 2	
0	90
1	82
2	74
3	66
4	56
5	55
6	45
7	37

List 3	
0	10
1	12
2	14
3	16
4	26
5	35
6	45
7	47

```java
public static void bubbleSort(int[] arr) {
    for (int i = 0; i < arr.length; i++) {
        for (int j = 1; j < (arr.length); j++) {
            if (arr[j-1] < arr[j]) {
                int temp = arr[j-1];
                arr[j-1] = arr[j];
                arr[j] = temp;
            }
        }
    }
}
```

Improving Bubble

- Observation, at end of each inner loop, one additional item in place
- Moved a lot of stuff moves, but last item is done
- So, can we only move the last item?
  - Looks a lot like find, print, delete, repeat
  - BUT, we will do this "in place"
    - Algorithm: find, swap, repeat

Selection Sort

find, swap, repeat

```
public static void selectionSort(int[] arr) {
    for (int i = 0; i < arr.length; i++) {
        int best = 0;
        for (int j = 1; j < (arr.length - i); j++) {
            if (arr[best] < arr[j]) {
                best = j;
            }
        }
        int temp = arr[best];
        arr[best] = arr[arr.length - i - 1];
        arr[arr.length - i - 1] = temp;
    }
}
```

Activity

Count operations for SelectionSort

1. Comparisons 2. Additions 3. Swaps

List 1	
0	10
1	22
2	54
3	86
4	56
5	15
6	55
7	7

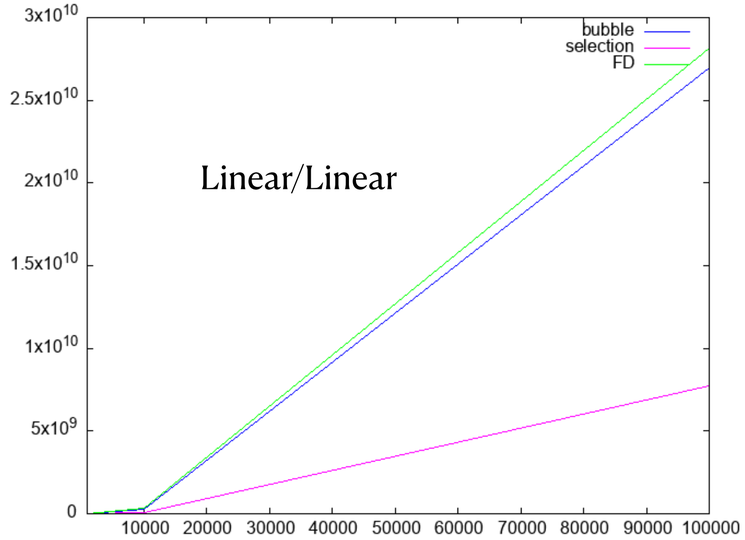
List 2	
0	90
1	82
2	74
3	66
4	56
5	55
6	45
7	37

List 3	
0	10
1	12
2	14
3	16
4	26
5	35
6	45
7	47

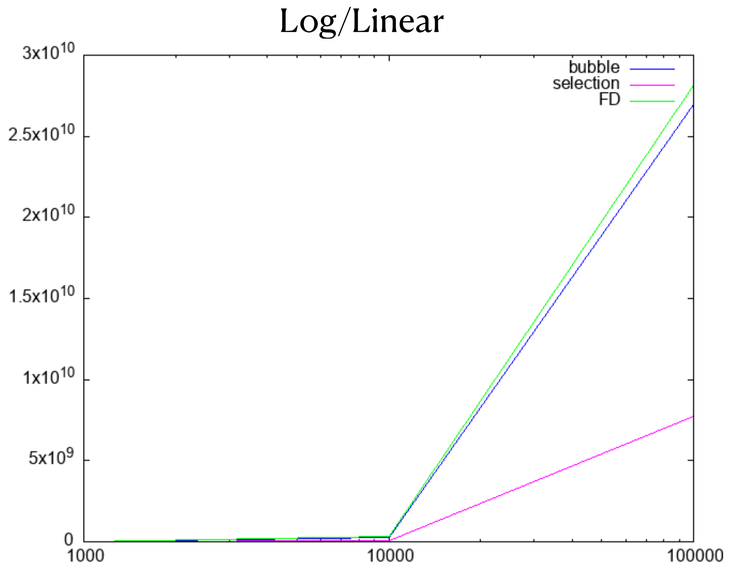
```java
public static void selectionSort(int[] arr) {
    for (int i = 0; i < arr.length; i++) {
        int best = 0;
        for (int j = 1; j < (arr.length - i); j++) {
            if (arr[best] < arr[j]) {
                best = j;
            }
        }
        int temp = arr[best];
        arr[best] = arr[arr.length - i - 1];
        arr[arr.length - i - 1] = temp;
    }
}
```

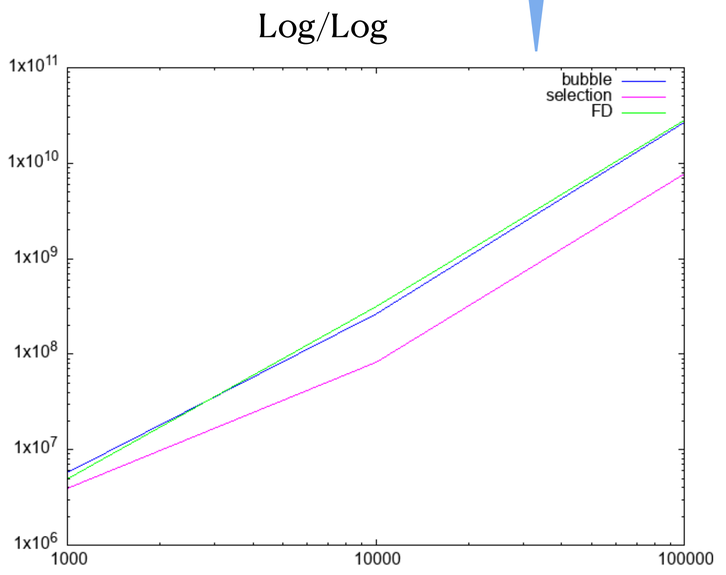
Bubble vs Selection vs Find/Delete

selection is much faster



When one variable changes as a constant power of another, a log-log graph shows the relationship as a straight line.





Make a Random String

just because

- Suppose I wanted to make a string composed of random characters of some length.
  - HOW???
  - Start with an array of chars and pick randomly from that
    - HOW?
  - Start with a string and pick randomly from that
    - HOW?
  - Use ASCII!
    - HOW?

Writing Comments

- Code should be commented in 3 ways
  - Every file/class should have a top level comment explaining what it does and why it exists

- Every instance variable should have a comment about what it does
  - maybe just one line with //

- Every method should have a comment with:
  - summary description of what it does
  - description of each param
  - description of return value
    - very simple function may not need this
    - Javadoc style

Finding things

can we do this efficiently?

- Have looked for stuff a LOT, usually the max
- Slightly different problem
  - determine if an object with a value is in a list
- Basic Algorithm

```
let arr = array of integers
let target = an integer (the thing to find)
for ii in 0..arr.length
    if arr[i]==target
        return TRUE
return FALSE
```

Finding Things

Basic Algorithm

- Need to go though whole list
- If item is in list, then on average will search 1/2 list every time

- How can I do better!!!
  - Can I re-order the list and improve?

```
let arr = array of integers
let target = an integer
for ii in 0..arr.length
    if arr[i]==target
        return TRUE
return FALSE
```

Binary Search

On a sorted list

```
let arr = array of integers
let target = an integer
let lo = 0
let hi = arr.length-1
While (lo < hi)
    let mid = (lo+hi) / 2
    if arr[mid]==target
        return TRUE
    if arr[i] < target
        lo = mid + 1
    else
        hi = mid - 1
// end while
return FALSE
```