Java?

Write down all the questions you have about Java.
Sort detective

vote here: tinyurl.com/cs42sortdetective

Measured performance of implementation in nanoseconds

Theoretical performance of algorithm in comparisons

random order

sorted order

anti-sorted order

vote here: tinyurl.com/cs42sortdetective
Sort detective: results

- merge sort
- selection sort
- bubble sort
- insertion sort
Sorting algorithms

Things to consider

Theory vs Practice — Implementations vs Algorithms
Theoretical best-case performance on worst-case input: $n \log n$

What are we measuring / modeling / optimizing for?
comparisons vs swaps • time vs space vs energy vs codability

Is the algorithm stable?
Does it preserve the relative order of “equivalent” items?

Is the algorithm adaptive?
Does it perform well when the data is already sorted?
Theoretical tools: code $\rightarrow$ math

How many times does the platypus quack?

```java
for (int i=0; i<N; i++) {
  for (int j=i; j<N; j++) {
    platypus.quack();
  }
}
```

Work from the inside out.

Remember: upper bounds for loops are exclusive; but upper bounds for summations are inclusive.

Remember: a loop index *can* change by means other than increment (e.g., the index might double each time through the loop, etc); but a summation index only increments
Theoretical tools: code $\rightarrow$ math

How many times does the platypus quack?

```
for (int j=0; j<N; j++) {
    platypus.quack();
}
```

- **code**
- **math**

\[
\sum_{j=0}^{N-1} 1
\]

- **Wolfram Alpha**

\[
\text{sum } 1, j=0 \text{ to } N-1
\]

- **closed form**

\[
N
\]

- **asymptotic notation**

\[
O(N)
\]

You should be able to:
- translate code (e.g., loops) to math (e.g., summations)
- determine the asymptotic complexity, if the closed form happens to be a polynomial

It’s nice, but not necessary for 42 if you can:
- convert math to Wolfram Alpha
- come up with a closed-form, given a summation
Theoretical tools: code $\rightarrow$ math

How many times does the platypus quack?

```
for (int i=0; i<N; i++) {
    for (int j=0; j<N; j++) {
        platypus.quack();
    }
}
```

$$\sum_{i=0}^{N-1} \sum_{j=0}^{N-1} 1$$

Wolfram Alpha

```
sum (sum 1, j=0 to N-1), i=0 to N-1
```

Closed form

$$N^2$$

Asymptotic notation

$$O(N^2)$$
### Theoretical tools: code $\rightarrow$ math

How many times does the platypus quack?

<table>
<thead>
<tr>
<th>Code</th>
<th>Math</th>
<th>Wolfram Alpha</th>
<th>Closed Form</th>
<th>Asymptotic Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>for (int i=0; i&lt;N; i++) {</code></td>
<td>$\sum_{i=0}^{N-1} \sum_{j=i}^{N-1} 1$</td>
<td><code>sum (sum 1, j=i to N-1), i=0 to N-1</code></td>
<td>$\frac{N(N+1)}{2}$</td>
<td>$O(N^2)$</td>
</tr>
<tr>
<td><code>  for (int j=i; j&lt;N; j++) {</code></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>    platypus.quack();</code></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>`}</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Bubble sort


```python
func bubblesort( var a as array )
    for i from 1 to N
        for j from 0 to N - 2  // error in original
            if a[j] > a[j + 1]
                swap( a[j], a[j + 1] )
    end func
```
Selection sort


func selSortI(lst)
    max = length(lst) - 1

    for i from 0 to max
        key = lst[i]
        keyj = i

        for j from i+1 to max
            if lst[j] < key
                key = lst[j]
                keyj = j

        lst[keyj] = lst[i]
        lst[i] = key

    return lst
end func
Insertion sort

For i from 1 to N
    key = a[i]
    j = i - 1
    while j >= 0 and a[j] > key
        a[j+1] = a[j]
        j = j - 1
    a[j+1] = key
Merge sort

from: http://www.algorithmist.com/index.php/Merge_sort#Pseudo-code

```python
func mergesort(
  var a as array
)
  if ( n == 1 ) return a

  var l1 as array = a[0] ... a[n/2]
  var l2 as array = a[n/2+1] ... a[n]

  l1 = mergesort( l1 )
  l2 = mergesort( l2 )

  return merge( l1, l2 )
end func

func merge( var a as array, var b as array )
  var c as array

  while ( a and b have elements )
    if ( a[0] > b[0] )
      add b[0] to the end of c
      remove b[0] from b
    else
      add a[0] to the end of c
      remove a[0] from a
  while ( a has elements )
    add a[0] to the end of c
    remove a[0] from a
  while ( b has elements )
    add b[0] to the end of c
    remove b[0] from b

  return c
end func
```

1. How many calls to mergesort?
2. How many comparisons for merge?
3. How many comparisons for mergesort?
<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Input Sorted</th>
<th>Closed Form</th>
<th>Asymptotic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bubble</td>
<td>$\sum_{i=0}^{N-1}\sum_{j=0}^{N-2} 1$</td>
<td>$N(N-1)$</td>
<td>$O(N^2)$</td>
</tr>
<tr>
<td>Selection</td>
<td>$\sum_{i=0}^{N-1}\sum_{j=i+1}^{N-1} 1$</td>
<td>$\frac{N(N-1)}{2}$</td>
<td>$O(N^2)$</td>
</tr>
<tr>
<td>Insertion</td>
<td>$\sum_{i=1}^{N-1} 1$</td>
<td>$N-1$</td>
<td>$O(N)$</td>
</tr>
<tr>
<td>Merge</td>
<td>$T(1) = 0$, $T(N) = N - 1 + 2T(\frac{N}{2})$</td>
<td>$N\log_2 N + 1 - N$</td>
<td>$O(N\log N)$</td>
</tr>
</tbody>
</table>

Insertion sort is the only algorithm we saw that has theoretically different performance on sorted and anti-sorted lists.

Note that an anti-sorted list isn’t the worst-case input for merge sort: an “interleaved” list is (e.g., 15372648).