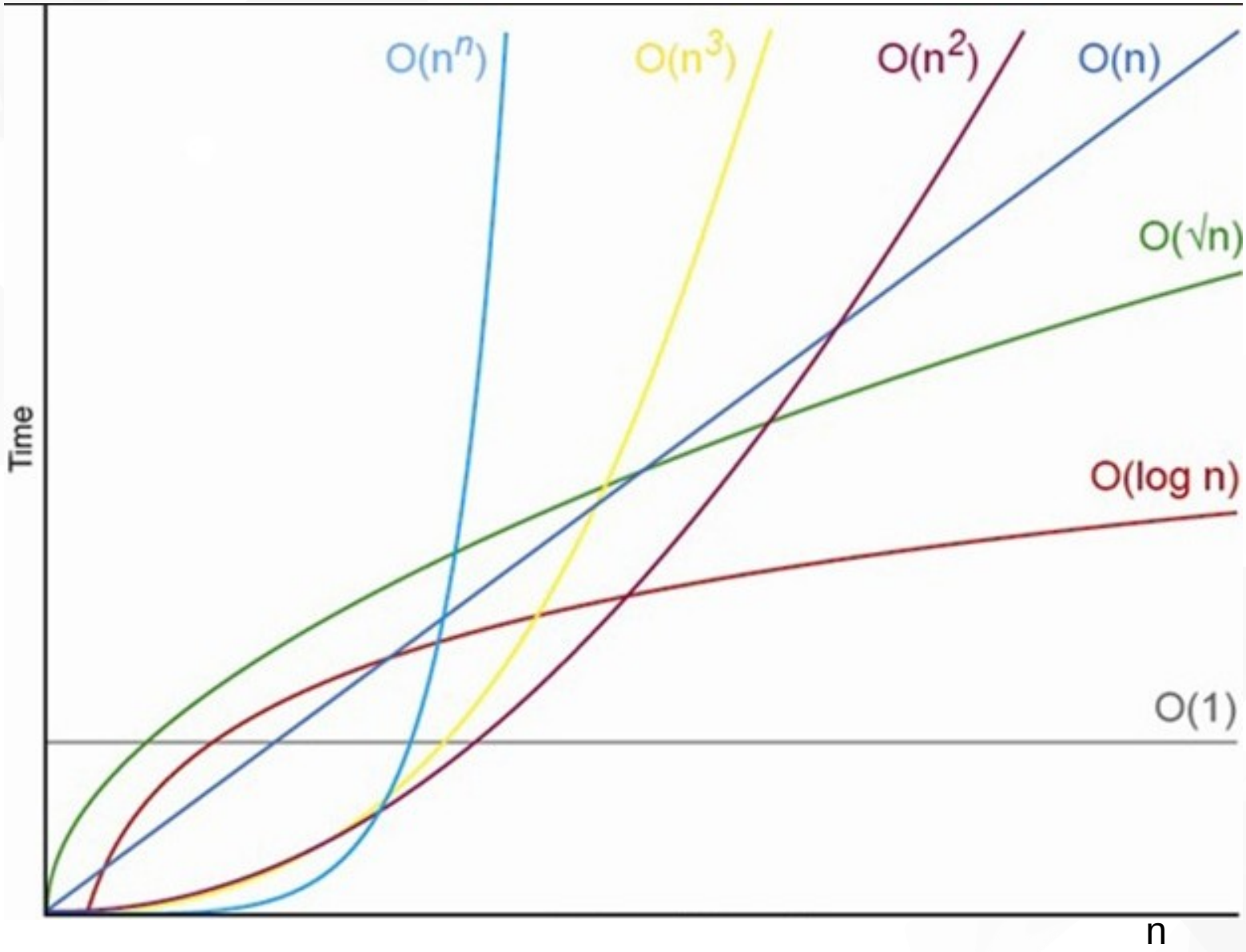
sit with your clan
if you can

CMPS 12B/M
Introduction to Data Structures

- Instructor: Nathan Whitehead

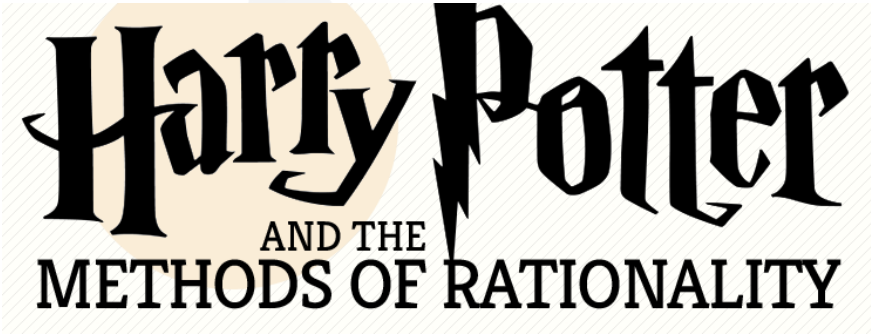
Complexities Graphed



Simple Sorting

- Well if you're sure, better be... GRYFFINDOR!

  - *Sorting Hat*



http://hpmor.com

What if Harry Potter was raised by scientists and used his brains to understand magic?

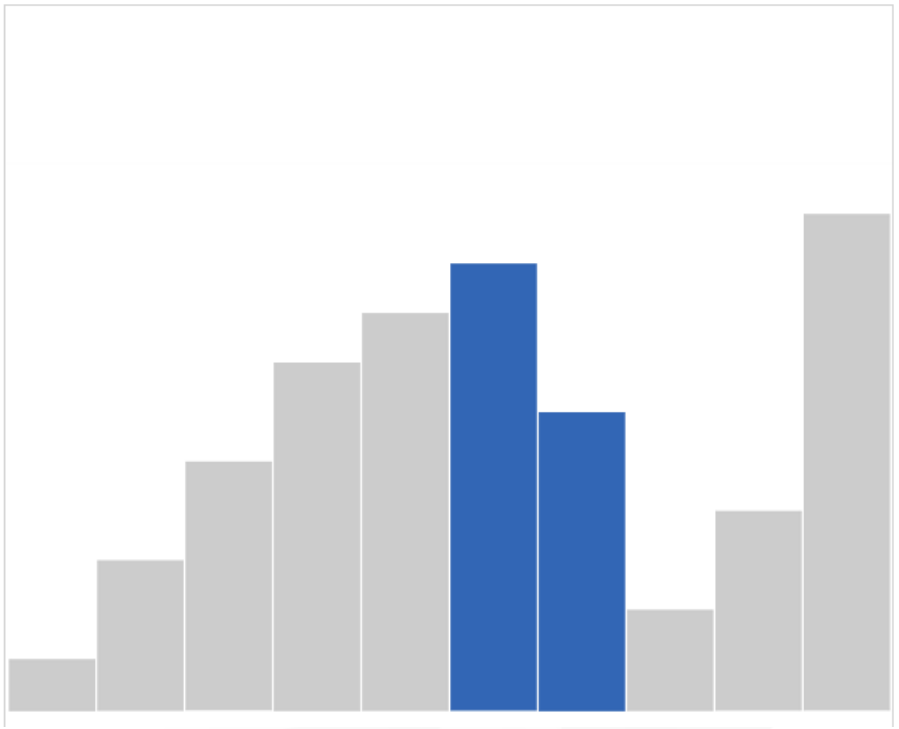
By Eliezer Yudkowsky

Sort Problem

- Suppose you had an unordered array, and wanted an ordered array
- Operations allowed:
  - Compare two items
  - Swap two items, or copy one item to a temp location
- All sorting algorithms do the same basic things
  - Produce ordered output using compares/swaps
  - Difference in the details

Bubble Sort

- Scan through left to right
  - Compare adjacent entries
  - If they are wrong way around, swap them
- If any swaps were made, loop back again
- No swaps made means you're done



http://www.algomation.com/player?algorithm=54162b10f3166302000a91f2

Bubble Sort in Java

- Examples/Chap03/BubbleSort/bubbleSort.java

Bubble Sort in Java

- Observation:
  - Each run through, we know we have the biggest element to the right
  - So stop scanning once we hit the finished section on the right

Running Time of BubbleSort

- How many comparisons does optimized bubble sort make in the worst case?
  - For n=10:
    - Our variation: 9+9+9+9+9+9+9+9+9
    - With observation: 9+8+7+6+5+4+3+2+1

what's the formula for summation?

How many comparisons does bubble sort take in the worst case for arbitrary n? (Come up with a formula involving n)

Bubble Sort Comparisons

- 9+9+9+9+9+9+9+9+9 pattern is

$$(n-1)\cdot(n-1) = n^2 - 2n + 1$$

- 9+8+7+6+5+4+3+2+1 pattern is

$$\sum_{i=1}^{n-1} i = \frac{n(n-1)}{2} = \frac{1}{2}n^2 - \frac{1}{2}n$$

what's that in big O notation?

Big O - Quadratic

- Definition:
  - O(*something*) is the set of functions that grow as n gets bigger like *something*
- How does n^2-2n+1 grow as n increases?
  - Think about taking n to infinity
  - Which term dominates? n^2
- Then ignore constant factors in front of dominant term
  - So answer is that it is in $O(n^2)$

- Conclusion: both versions of bubble sort are *quadratic*

Selection Sort

- Why bother “bubbling” values to bottom and top
  - Instead just scan and find smallest value, swap it to first position
  - Repeat down the array
- This is called selection sort
  - Examples/Chap03/SelectSort/selectSort.java

http://www.algomation.com/algorithm/selection-sort-animated

Selection Sort Code

- Examples/Chap03/SelectSort/selectSort.java

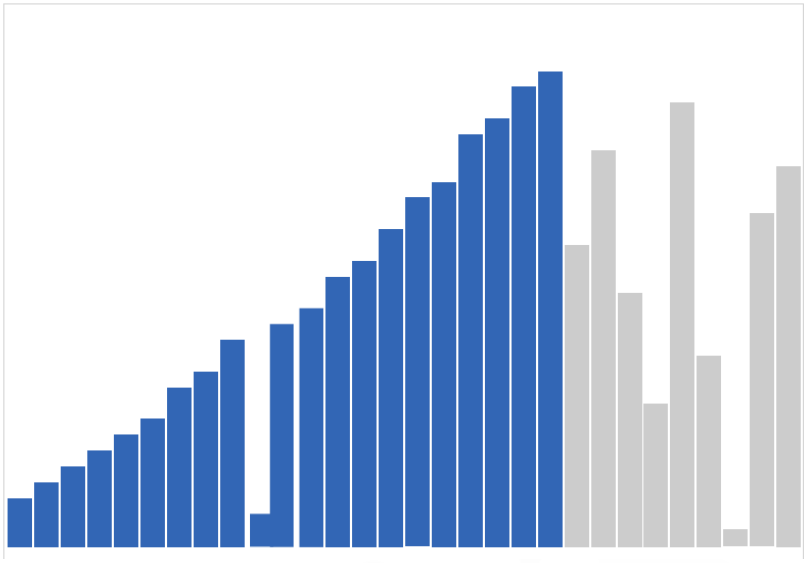
Analysis of Selection Sort

- Worst case: n swaps (might get lucky and need fewer)
- How about comparisons?
  - To find smallest, need n-1 comparisons
  - To find next smallest, need n-2 comparisons
  - ...
  - To find second biggest, need 1 comparison
  - Then biggest is automatically in last position
  - So (n-1) + ... + 3 + 2 + 1 = 1 + 2 + 3 + ... + (n-1)
- Still *quadratic* in comparisons like bubble sort
- But actually few swaps (*linear*)
  - Good choice if swapping is expensive

$$\sum_{i=1}^{n-1} i = \frac{n(n-1)}{2}$$

Insertion Sort

- Build up sorted array on left
  - Repeatedly pick next unsorted element from right
  - Keep bubbling it left, until it is correct position



http://www.algomation.com/player?algorithm=54145844e230980200679f92

Insertion Sort Code

- Examples/Chap03/InsertSort/insertSort.java

How many comparisons?

- Assume n elements in array
  - On first pass, just one comparison
  - On second pass, two comparisons
  - Last pass takes n-1 comparisons (worst case)
  - So total is 1+2+...+(n-1)
- In big O notation
  - $O(n^2)$
  - *Quadratic*

$$\sum_{i=1}^{n-1} i = \frac{n(n-1)}{2}$$

Insertion Sort – Advantages

- One big advantage of insertion sort

  What if the input is already sorted?
  - Inserting new element takes just one comparison
  - Just verifies it is in right position
  - Need to insert n-1 elements, so n-1 comparisons
- What is that in big O?
  - O(n-1) = O(n)
  - *Linear*

linear is much better than quadratic

Stability

- What happens when elements are equal?
  - ***Stable sort*** – guarantee they are not swapped from their original order
  - ***Unstable sort*** – may or may not be swapped, no guarantees
- All the sorts we've seen are stable
- ***Q***: If the elements are the same, why care?
  - ***A***: We often sort objects by keys, want to keep order when keys are the same

Stable Sort Example

- *Example*
  - List of students sorted by name
  - Sort with stable sort by test score

Adam	98
Alice	98
Bob	76
Matthew	91
Nathan	82
Olivia	98
Paula	85

original

sorted by name

Bob	76
Nathan	82
Paula	85
Matthew	91
Adam	98
Alice	98
Olivia	98

new

stable sorted by score

still sorted by name

Quadratic Sorts

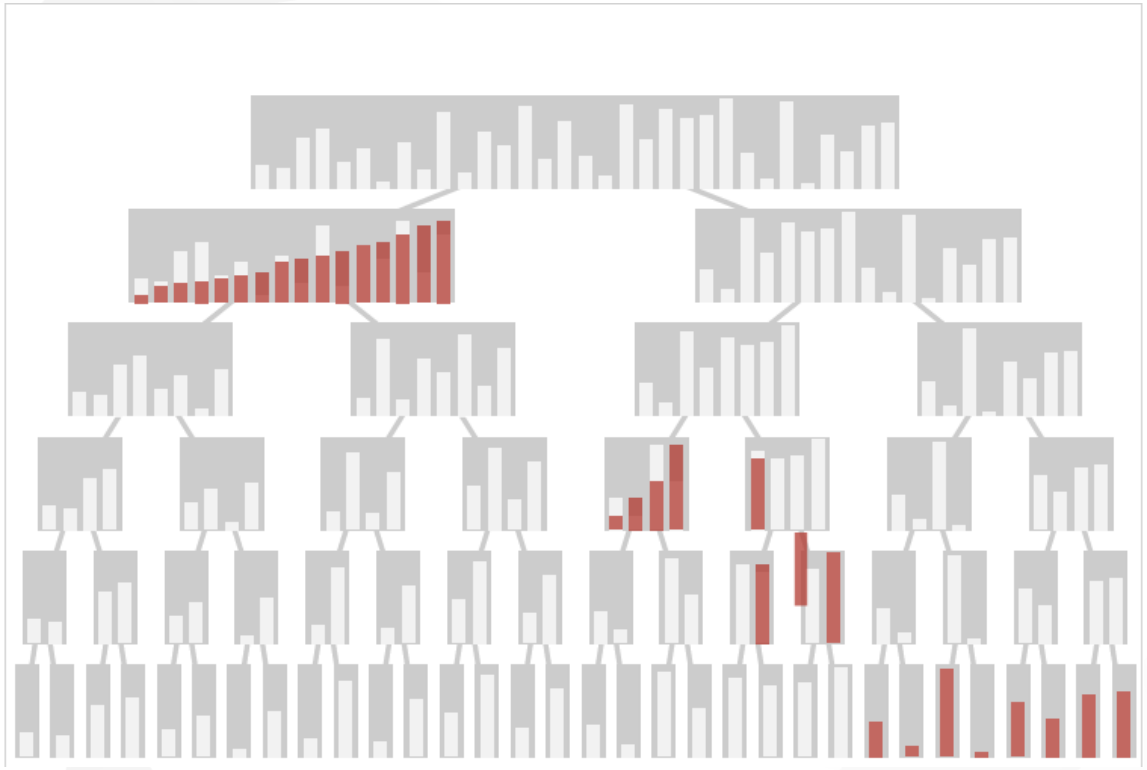
- All our sorts seem be ending up quadratic in worst case
  - Could there possibly be a better way?

Merge Sort!

Recursive Sorting

- Key idea
  - To sort a pile
    - Divide into two piles
    - Sort both piles
    - Merge sorted piles together
- Recursion – functions that call themselves
  - Seems like cheating...

Merge Sort



http://www.algomation.com/player?algorithm=543071e0b5d751020031d353

Two Functions

- merge(arrayA, arrayB, arrayC)
  - Merge A and B into array C
  - Assumes A and B are each ordered
  - Output in array C is sorted, contains all of A and B
- mergeSort(arrayA)
  - Sort the array in place recursively
- Actual code uses one array with bounds to indicate subarrays
- Need extra working space for merging

How could this possibly be faster?

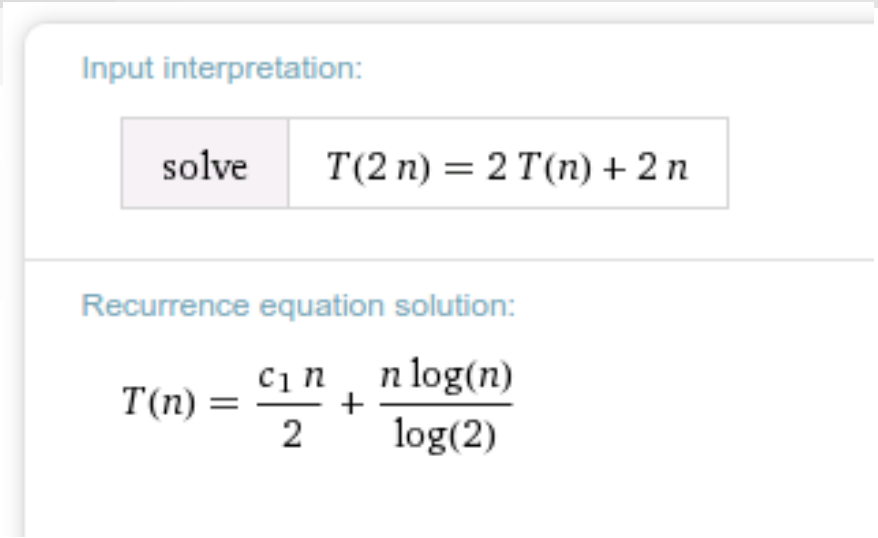
- Let T(n) be number of comparisons/copies of mergesort
- Merging arrays of size n and m
  - Linear scan through both input arrays
  - Total of n+m comparisons, n+m copies
  - Merging is O(n+m)
- Mergesort
  - What happens when we double the input?
  - T(2n) = T(n) + T(n) + n + n

sort left side

sort right side

merge

Let Wolfram Alpha do the work...



- n log(n) dominates linear term
  - Because logarithmic dominates constants
- So solution is ***O(n log(n))***

Merge Sort Code

- Examples/Chap06/mergeSort/mergeSort.java

```
nwhitehead@nwhitehe-DX4710-UB801A ~/p/c/c/d/E/C/mergeSort> java MergeSortApp
64 21 33 70 12 85 44 3 99 0 108 36
0 3 12 21 33 36 44 64 70 85 99 108
```

More Visualizations

http://www.sorting-algorithms.com/

The End