Week 5

Sorting & Big-O (Again!)

EXAM 1!!

6:45 - 7:45pm, Feb. 20th, Troxel 1001 (Sec. A), Hoover 2055 (Sec. B)

Topics:

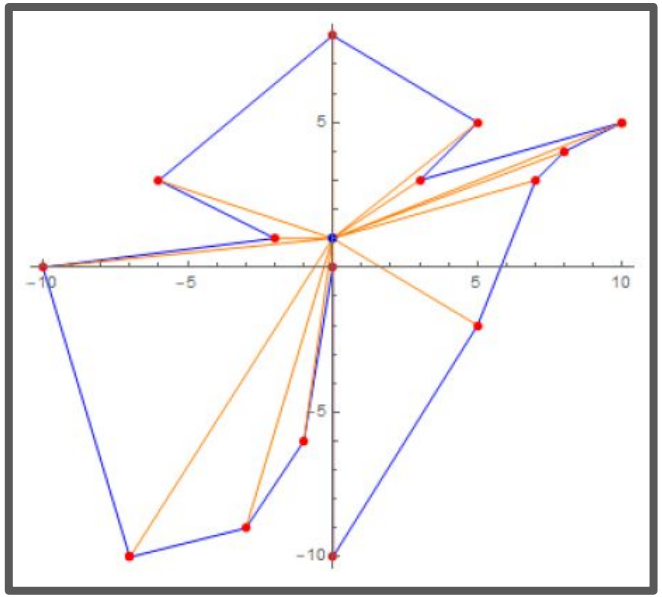
- Inheritance, Polymorphism, Interfaces, Abstract classes, Class hierarchy, Object superclass, Shallow vs. deep equals() and clone(), exception handling (try, catch, throw)
- Time complexity, Big-O notation, algorithm analysis, binary search
- Sorts: Selection, Insertion, Merge, Quick, and their runtimes
- Using the comparator/comparable interface

Project 2: Point Scanning

Due: March 3 (Sunday), midnight

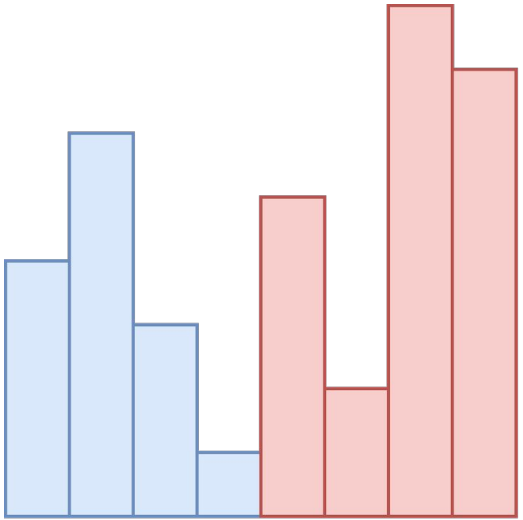
Focuses on:

- Sorting algorithms
- Comparator interface
- Drawing geometry

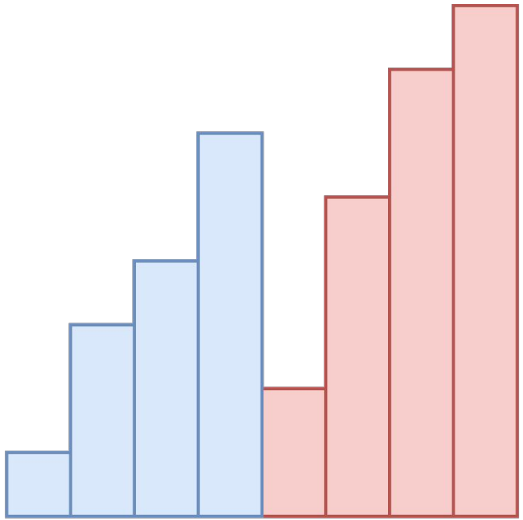


Mergesort

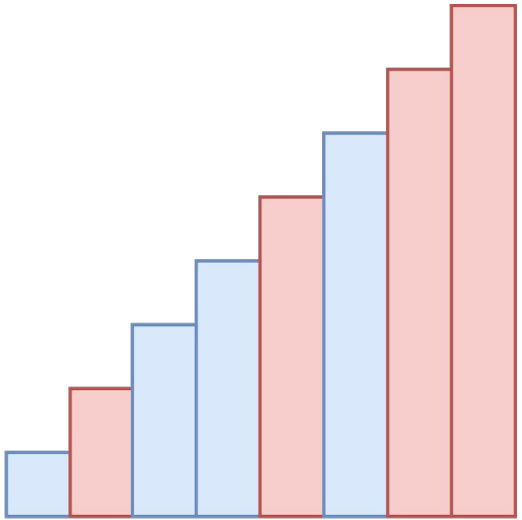
Worst case: O(nlogn)
Average case: O(nlogn)



Split array in half



Mergesort Mergesort

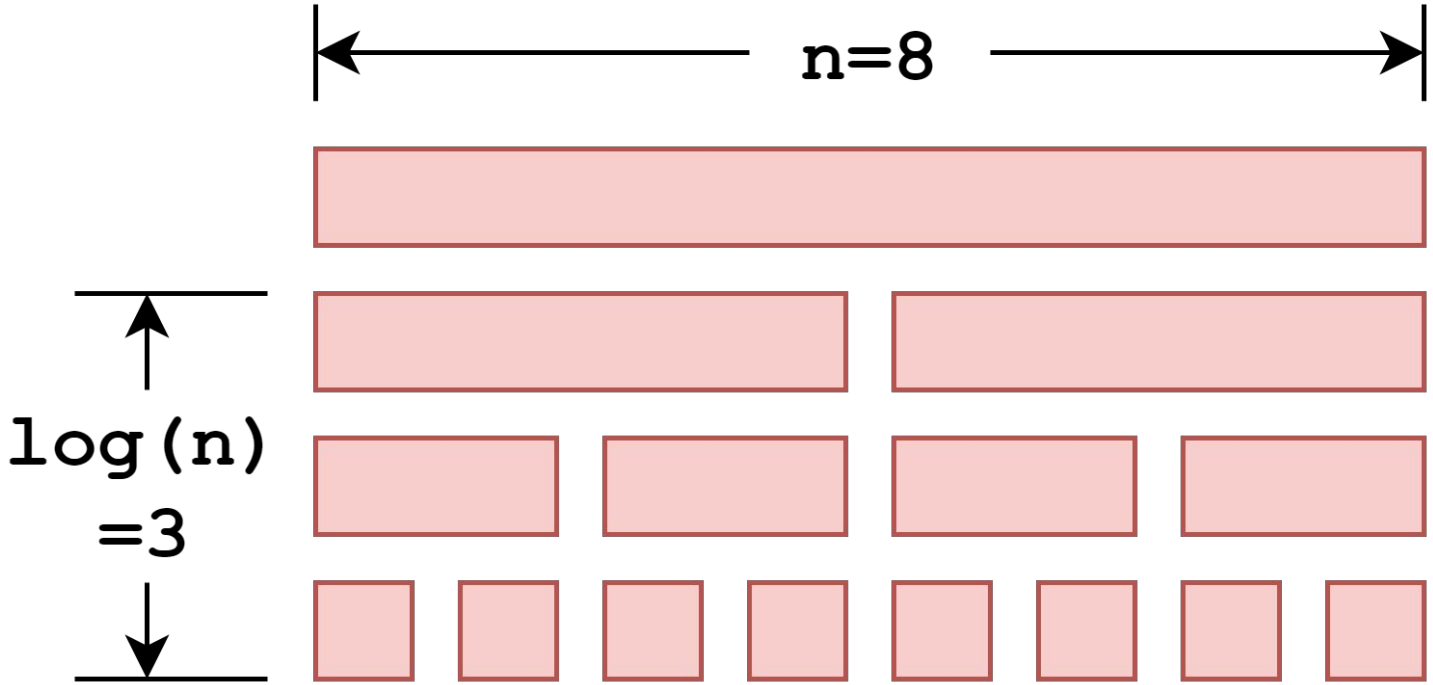


Merge

Mergesort runtime

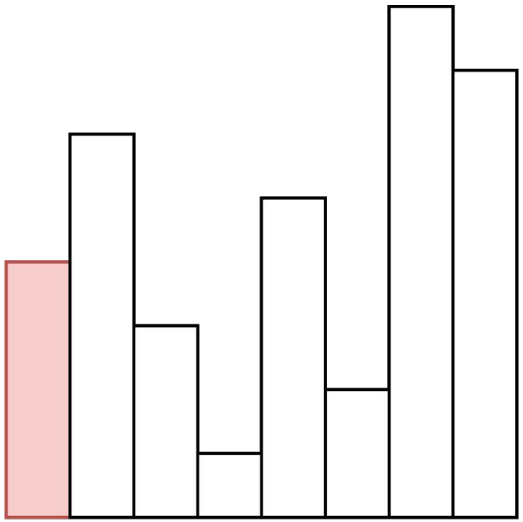
Why does it operate in O(nlogn)?

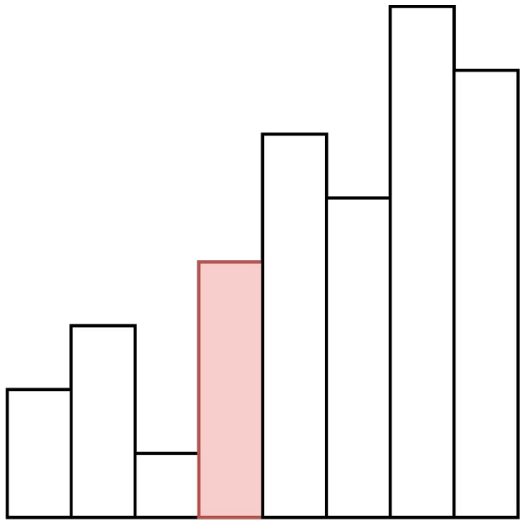
Mergesort runtime

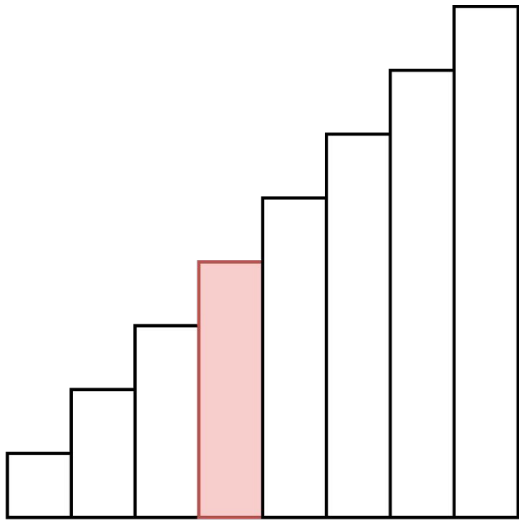


Worst case: $O(n^2)$
Average case: O(nlogn)

Quicksort

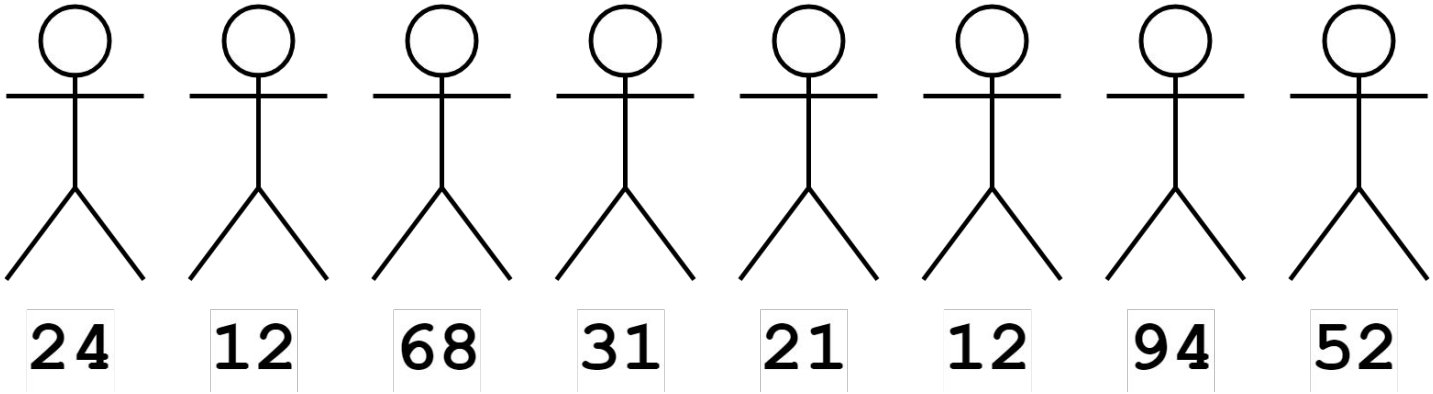




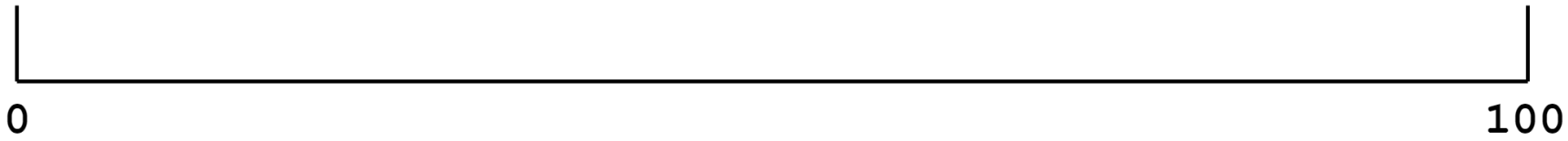


Counting sort

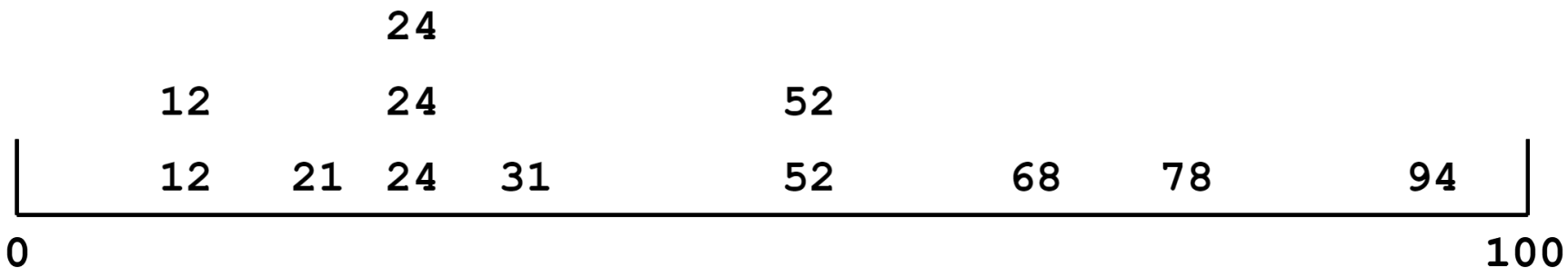
How can we sort a list of people by their age?



Counting sort

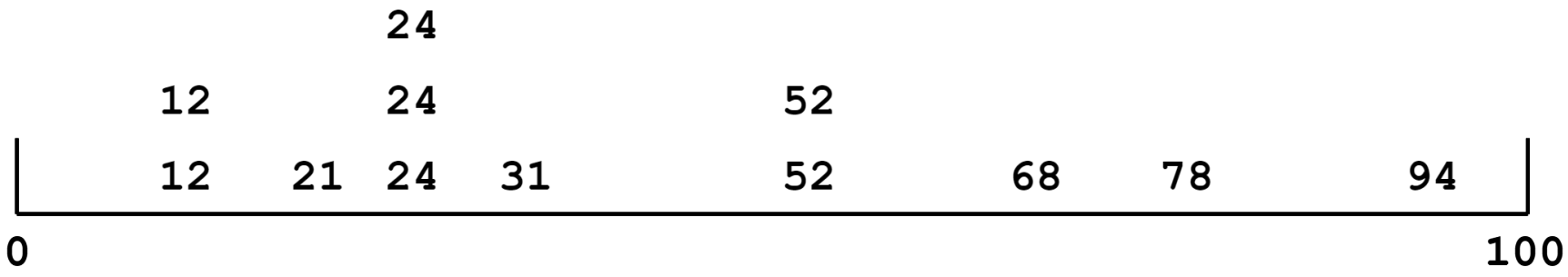


Counting sort



Worst case: O(r + n)
Average case: O(r + n)

Counting sort



Radix sort

How can we sort people in a phonebook by phone number?

```
612-749-5598
207-342-5319
805-614-6636
510-313-4327
921-405-2452
492-159-1950
211-539-0927
488-493-2009
821-667-3916
578-574-8029
103-283-7510
397-503-2189
754-710-3985
749-218-3475
903-174-5071
939-457-3908
```

Worst case: O(wn)
Average case: O(wn)

Radix sort

Radix sort process:

1. Create “buckets” to store numbers in from 0-9 (like in counting sort)
2. Place numbers in buckets according to least or most significant digit
3. Remove numbers in order from 0 to 9 and repeat with the next significant digit

Easier to understand using visualgo.net/en

```
821-667-3916
578-574-8029
103-283-7510
397-503-2189
754-710-3985
749-218-3475
903-174-5071
939-457-3908
```

Comparing Mergesort/Quicksort

Why use Quicksort instead of Mergesort when they both have an average runtime of **O(nlogn)**, but Quicksort at its worst case runs in **$O(n^2)$**?

Comparing Mergesort/Quicksort

Why use Quicksort instead of Mergesort when they both have an average runtime of **O(nlogn)**, but Quicksort at its worst case runs in **O(n²)**?

- With a **random pivot**, Quicksort practically never runs in **O(n²)**

Comparing Mergesort/Quicksort

Why use Quicksort instead of Mergesort when they both have an average runtime of **O(nlogn)**, but Quicksort at its worst case runs in **O(n²)**?

- With a **random pivot**, Quicksort practically never runs in **O(n²)**
- Quicksort plays better with hardware (Cache locality)

Comparing Mergesort/Quicksort

Why use Quicksort instead of Mergesort when they both have an average runtime of **O(nlogn)**, but Quicksort at its worst case runs in **O(n²)**?

- With a **random pivot**, Quicksort practically never runs in **O(n²)**
- Quicksort plays better with hardware (Cache locality)
- Uses less space than Mergesort

Source: https://www.quora.com/Why-is-quicksort-considered-to-be-better-than-merge-sort

Comparing runtimes

logn	n	nlogn	n^2
3	10	30	100
10	1,000	10,000	1,000,000
20	1,000,000	20,000,000	1,000,000,000,000
30	1,000,000,000	30,000,000,000	1,000,000,000,000,000,000

Comparing runtimes

Sorting countries by population

# of countries: ~200 Range of populations: 10^3-10^9

Selection sort	**(n^2)**	40,000
Mergesort	**(nlogn)**	1,500
Counting sort	**(r + n)**	10^9
Radix sort	**(wn)**	1,800

Comparing runtimes

Sorting phone numbers in the US

# of phone numbers: ~3×10^8 Range of phone numbers: 0-10^{10}

Selection sort	**(n^2)**	9×10^{16}
Mergesort	**(nlogn)**	8.4×10^9
Counting sort	**(r + n)**	10^{10}
Radix sort	**(wn)**	3×10^9

Comparing runtimes

Sorting ages of US citizens

# of US citizens: ~3×10^8 Range of ages: 0-150

Selection sort	**(n^2)**	9×10^{16}
Mergesort	**(nlogn)**	8.4×10^9
Counting sort	**(r + n)**	10^8
Radix sort	**(wn)**	9×10^8

Sorting Stability

Let us take a small array and sort it

Before: `5, 9, 7, 5`

After: `5, 5, 7, 9`

Sorting Stability

But which 5 is which?

Before: 5, 9, 7, 5

After: 5, 5, 7, 9

Sorting Stability

Stable

5, 9, 7, 5

5, 5, 7, 9

Unstable

5, 9, 7, 5

5, 5, 7, 9