# CSSE 473 – Day 9

DECREASE AND CONQUER

## Exam Info

Exam 1 is this Thursday, 7-8:30 pm
- Section 01 - Crapo G219
- Section 02 - Crapo G220
- Section 03 - Crapo G221
- Section 04 - Crapo G222

We still have regular class on Thursday

You may bring a single-sided 8.5 x 11 cheat sheet
- Anything you want, as long as it is human-readable

Exam covers everything up to an including Homework 04
- Chapters 1, 2, 3, and Chapter 4 through 4.4 (excluding 4.5)

# Overview

Decrease by a constant factor:
- Binary Search

Variable-size decrease:
- Lomuto partition
- Quickselect
- Interpolation Search

# Binary Search

Best case:
O(1)

```
public static int binarySearch(int[]a, int e){
    int low = 0;
    int high = a.length-1;
    int mid;
    while (low <= high){
        count++;
        mid = (low + high) / 2;
        if (a[mid] < e) low = mid + 1;
        else if (a[mid] > e) high = mid - 1;
        else return mid;
    }
    return -1;
}
```

4

# Binary Search

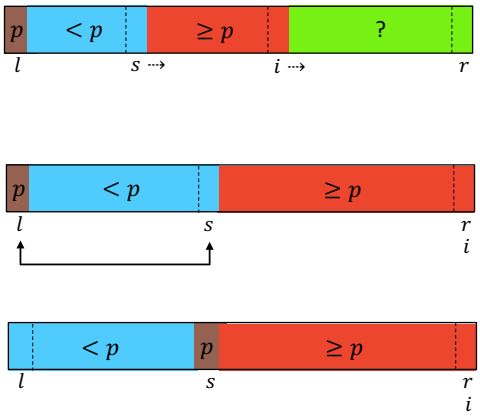
Worst case:
$C_w(n) = 1 + C_w(\lfloor n/2 \rfloor)$
$C_w(1) = 1$

$C_w(n) = \lfloor \log_2(n+1) \rfloor$

```
public static int binarySearch(int[]a, int e){
    int low = 0;
    int high = a.length-1;
    int mid;
    while (low <= high){
        count++;
        mid = (low + high) / 2;
        if (a[mid] < e) low = mid + 1;
        else if (a[mid] > e) high = mid - 1;
        else return mid;
    }
    return -1;
}
```

5

# Lomuto Partitioning




```
LomutoPartition(A[l..r])
    p ← A[l]
    s ← l
    for i=(l+1)..r
        if A[i] < p
                increment s
                swap(A[s],A[i])

    swap(A[l],A[s])

    return s
```

# Tracing Lomuto Partitioning

```
LomutoPartition(A[l..r])
    p ← A[l]
    s ← l
    for i=(l+1)..r
        if A[i] < p
                increment s
                swap(A[s],A[i])

    swap(A[l],A[s])

    return s
```

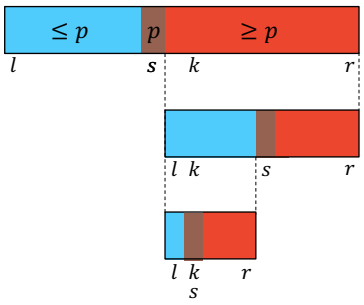
| $s$ | $i$ | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **4** | 1 | 10 | 8 | 7 | 12 | 9 | 2 | 15 |
| | $s$ | $i$ | | | | | | |
| **4** | 1 | 10 | 8 | 7 | 12 | 9 | 2 | 15 |
| | $s$ | | | | | | $i$ | |
| **4** | 1 | 10 | 8 | 7 | 12 | 9 | 2 | 15 |
| | | $s$ | | | | | | $i$ |
| **4** | 1 | 2 | 8 | 7 | 12 | 9 | 10 | 15 |
| | | $s$ | | | | | | |
| **4** | 1 | 2 | 8 | 7 | 12 | 9 | 10 | 15 |
| 2 | 1 | **4** | 8 | 7 | 12 | 9 | 10 | 15 |





# Quickselect: Finding the kth smallest element

```
Quickselect(A[l..r],k)
        s ← Partition(A[l..r])
        if k = s
                return A[s]
        else if k < s
                Quickselect(A[l..s-1],k)
        else
                Quickselect(A[s+1..r],k)
```



# Analysis of Quickselect

Average case (average split in the middle):

$C(n) = C(n/2)+(n+1)$ $\quad C(n) \in \Theta(n)$

Worst case (degenerate split): $C(n) \in \Theta(n^2)$





# Interpolation Search

Hymnal search
Phone book search

$$probe = lowEnd + \frac{(highEnd - lowEnd) \times (item - data[lowEnd])}{data[highEnd] - data[lowEnd]}$$

Image source: https://www.baeldung.com/java-interpolation-search

# Interpolation Search working well

```
int interpolation_search(int a[], int size, int key){
    int low = 0;
    int high = size - 1;
    int mid;
    while ((a[high] != a[low]) && (key >= a[low]) && (key <= a[high])) {
        mid = low + ((key - a[low]) * (high - low) / (a[high] - a[low]));
        if (a[mid] < key) low = mid + 1;
        else if (key < a[mid]) high = mid - 1;
        else return mid;
    }
    if (key == a[low]) return low;
    else  return -1;
}
```

1 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100

# Interpolation Search working not so well

```
int interpolation_search(int a[], int size, int key){
    int low = 0;
    int high = size - 1;
    int mid;
    while ((a[high] != a[low]) && (key >= a[low]) && (key <= a[high])) {
        mid = low + ((key - a[low]) * (high - low) / (a[high] - a[low]));
        if (a[mid] < key) low = mid + 1;
        else if (key < a[mid]) high = mid - 1;
        else return mid;
    }
    if (key == a[low]) return low;
    else  return -1;
}
```

1 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100000

# Analysis of Interpolation Search

Efficiency

average case: $C(n) < \log_2 \log_2 n + 1$

worst case: $C(n) = n$