Do Now Exercise

To prepare you for the lecture today, please do the following exercise.

Take 10 pieces of paper and write an integer value on each them.

COMP15: Data Structures

Week 5, Summer 2019

Admin

T5: **cat, head, tail, less**

Due by 6pm on Wednesday, June 26

(a quick demo)

P3: **Sorter**

Project Due by 6pm on Sunday, June 30

P3 grading:

Correctness of program functionalities

+

Written report

Questions about P3?

P2: Refinement Phase

Median XX (hiding sensitive data)

in which the highest possible score was 90.

(approximately **XX.X** with a 100-pts scale.)

Discussion:

What was a factor that made it possible to achieve such a high score?

Discussion:

How many test cases did you write when implementing your P2?

Unit Testing

Test **individual** modules

(Granularity of unit testing varis.)

Integration Testing

For example...

For example...

(We quickly discussed how we could test Event class and Scheduler class (+ Node class) in P2.)

Test-Driven Development (TDD)

Test coverage

Regression Testing

Suggestions:

- Plan how you test your modelus.
- Write tests as you implement a part of your modules.
- Make each test case tidy, so you can be absolutely sure what you are testing.
- Make each test case small, so you can test one thing at a time.
- Try coming up with corner cases. (This is not an easy task, though.)
- Keep all test cases you wrote, so you can re-run them whenever you make changes in your program.

A remaining question from last week:

How can you manage
the **first** and **last index** of a **Circular Array**?

Questions?

Asymptotic Analysis

Asymptotic Times

- Constant time: $O(1)$
- Logarithmic time: $O(log(n))$
- Linear time: $O(n)$
- Quadratic time: $O(n^2)$
- Polynomial time: $O(n^k)$ where k is constant
- Exponential time: $O(k^n)$ where k is constant

Asymptotic Times

- Constant time: $O(1)$
- Logarithmic time: $O(log(n))$
- Linear time: $O(n)$
- Quadratic time: $O(n^2)$
- Polynomial time: $O(n^k)$ where k is constant
- Exponential time: $O(k^n)$ where k is constant

Computing time complexity

```
int add1(int to){

  int r = to + 1;

  return r;
}
```

Computing time complexity

```
int sum(int* array, int size){
  int sum = 0;

  for(int i = 0; i < size; i++){
    sum += array[i];
  }

  return sum;
}
```

Computing time complexity

```
int sum(int** matrix, int rows, int columns){
  int sum = 0;

  for(int r = 0; r < rows; r++){
    for(int c = 0; c < columns; c++){
      sum += matrix[r][c];
    }
  }

  return sum;
}
```

Asymptotic Math

(constant time) + (constant time) => (constant time)

(constant time) + (linear time) => (linear time)

(constant time) + (logarithmic time) => (logarithmic time)

(constant time) + (logarithmic time) + (linear time) => (linear time)

etc.

Questions so far?

Examples

$$T(n) = 1000000 + log(n) + 0.1n$$

Examples

$$T(n) = 1.01^n + 1000n$$

Examples

Which one grows faster, 10 or $0.1n$?

Examples

Which one grows faster, $1000n$ or n^2 ?

Examples

Which one grows faster, $100n$ or 1.1^n ?

Examples

Which one grows faster, n^{100} or 1.01^n ?

Intuition:

As the **n** becomes asymptotically large,

the running time is **dominated by a faster growing term**.

A sneak peek preview (Comp 160, Algorithms)

Big-O

"asymptotically no larger than"
"asymptotic upper bound"

$f(n)$ is $O(g(n))$ if:

- There exists a positive constant c and
- There exists a positive value n_0 of n such that
- $0 \leq f(n) \leq c * g(n)$ `for all` $n \geq n_0$.

Ref: Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to algorithms.* MIT press.

A sneak peek preview (Comp 160, Algorithms)

Big-Ω

"asymptotically no smaller than"
"asymptotic lower bound"

$f(n)$ is $\Omega(g(n))$ if:

- There exists a positive constant c and
- There exists a positive value n_0 of n such that
- $0 \leq c * g(n) \leq f(n)$ `for all` $n \geq n_0$.

Ref: Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to algorithms.* MIT press.

A sneak peek preview (Comp 160, Algorithms)

Big-Θ

"asymptotically equal to"

$f(n)$ is $\Theta(g(n))$ if and only if:

- $f(n)$ is $O(g(n))$ and
- $f(n)$ is $\Omega(g(n))$.

Ref: Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to algorithms.* MIT press.

Questions?

Do Now Exercise

To prepare you for the lecture today, please do the following exercise.

Take 10 pieces of paper and write an integer value on each them.

Sorting

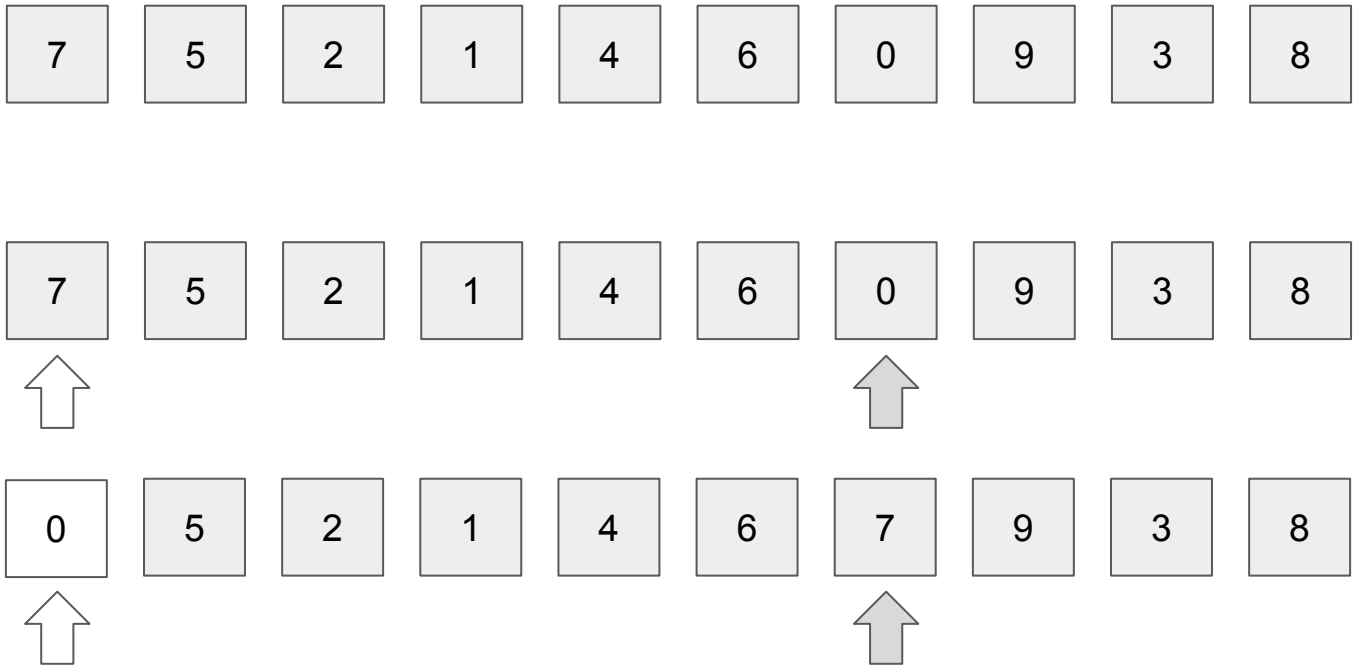
- Selection sort

- Insertion sort
- Merge sort
- Quicksort

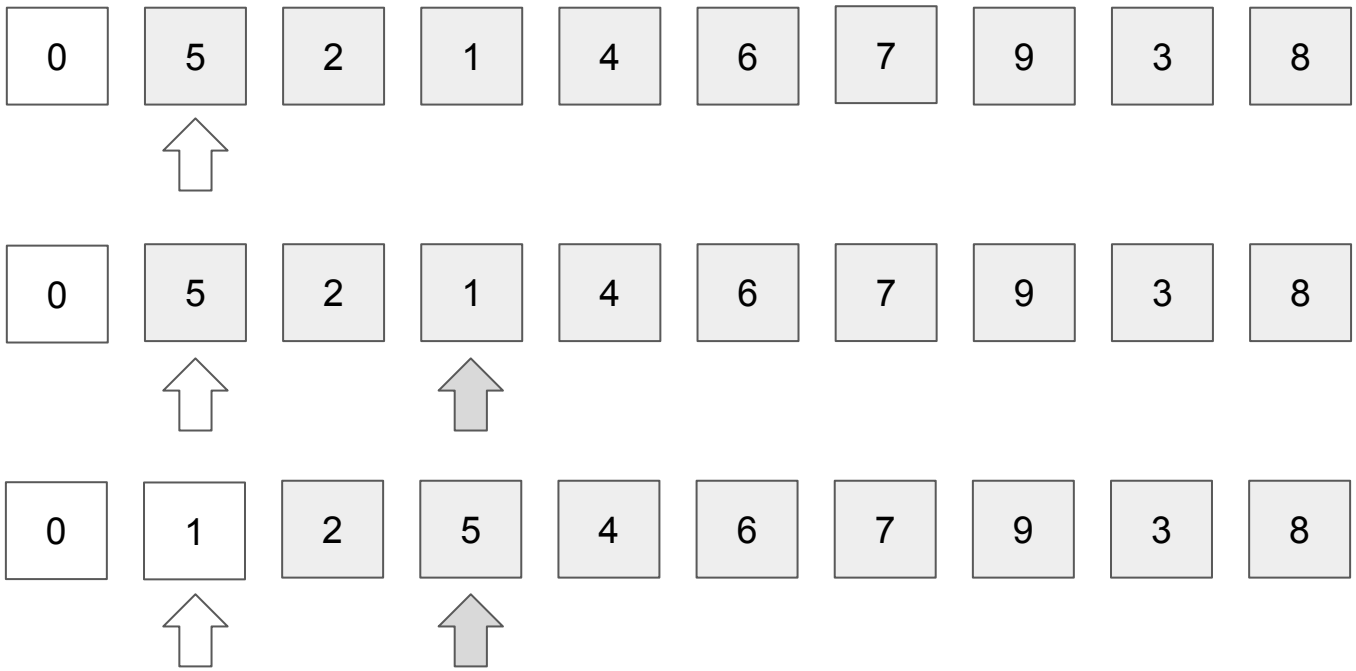
- Counting sort (if we have time)

Selection sort

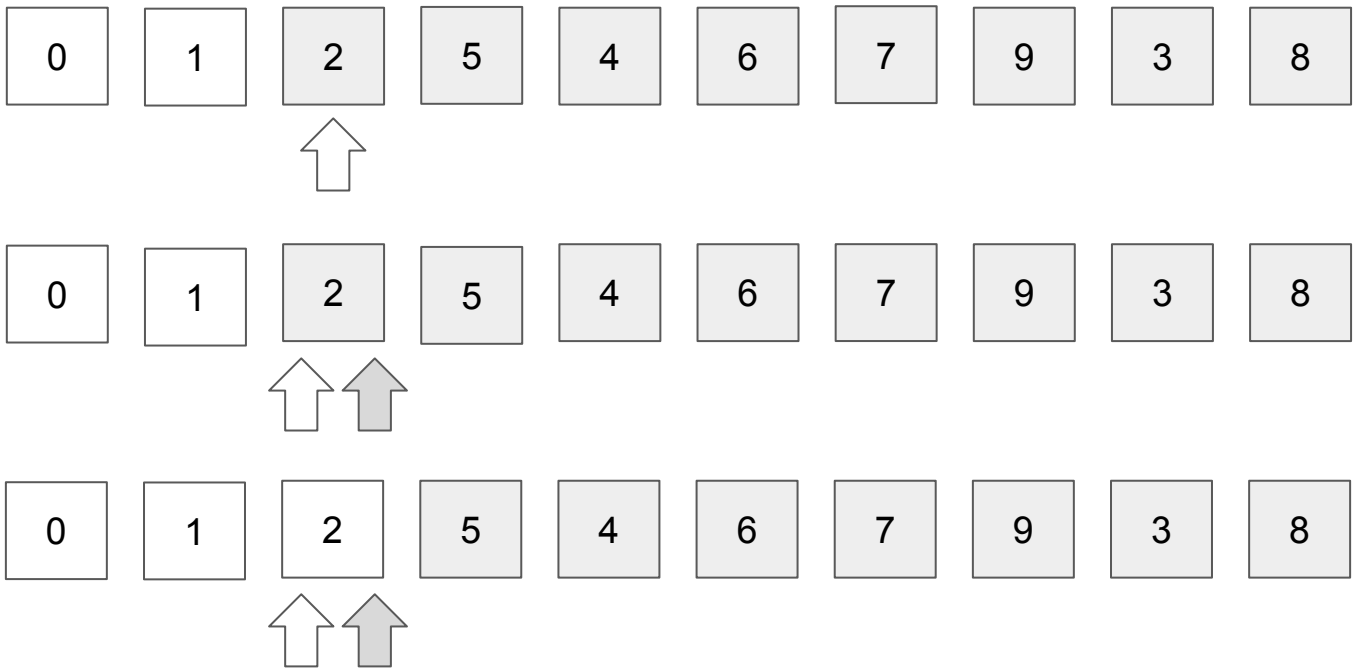
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)



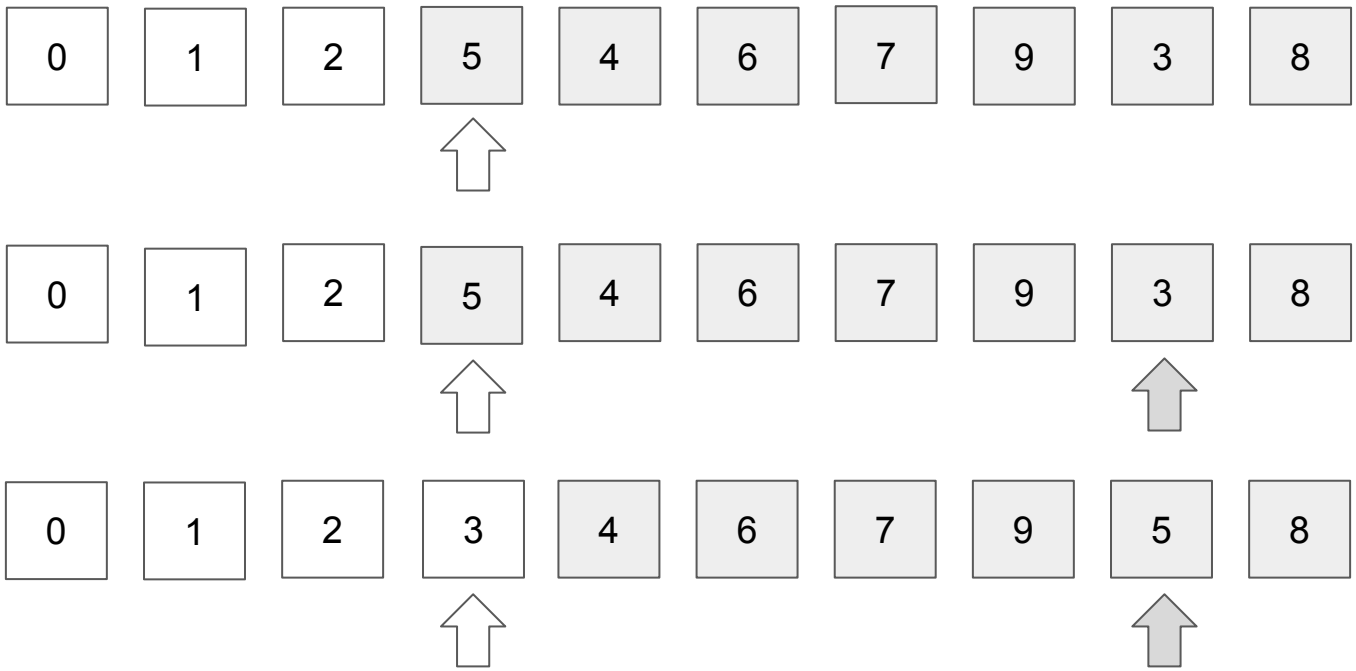
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)



(Notes from the live demo or live coding. Please do NOT assume the code is complete.)



(Notes from the live demo or live coding. Please do NOT assume the code is complete.)



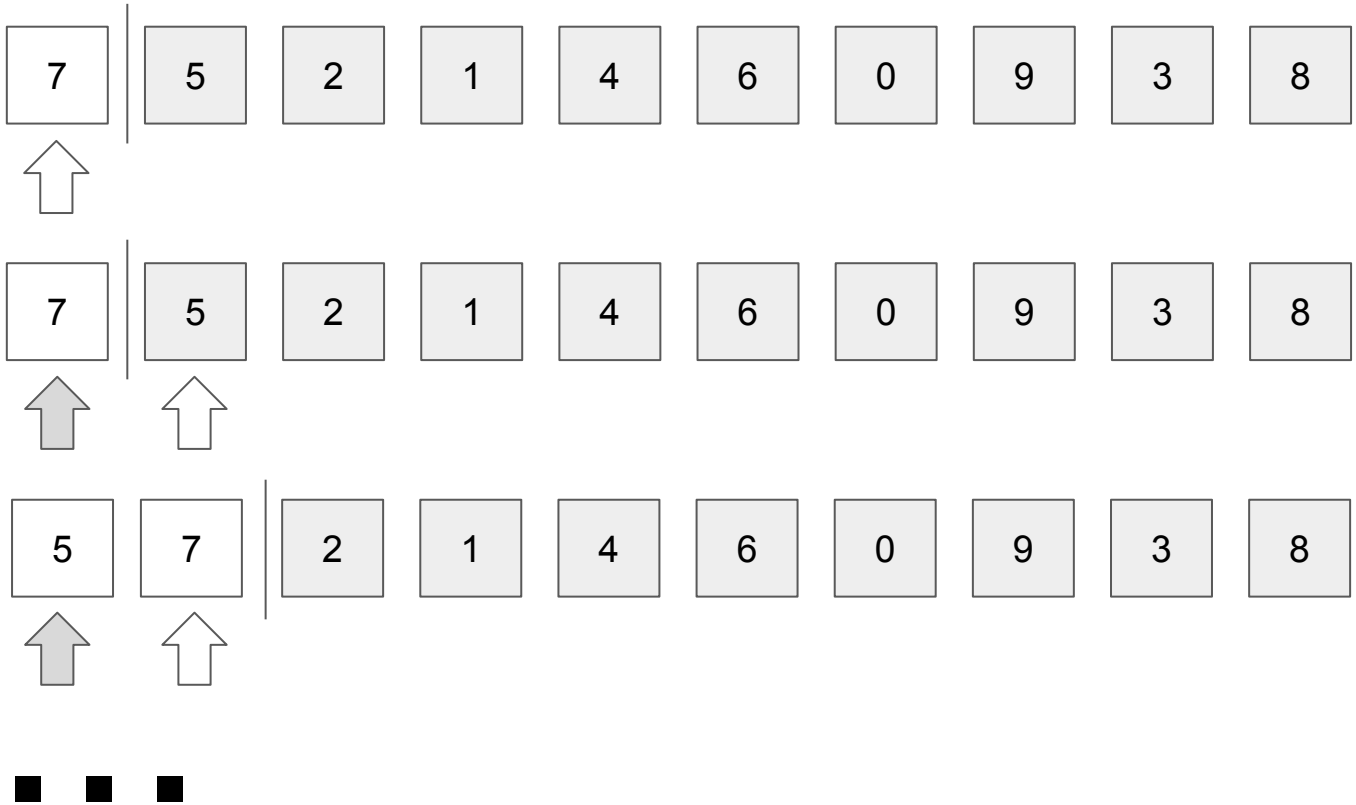
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

```
1 void selectionSort(int* const array, int size){
2   for(int target = 0; target < size - 1; target++){
3
4     int index = target;
5
6      for(int i = index + 1; i < size; i++){
7       if(array[i] < array[index]){
8         index = i;
9       }
10    }
11
12    int smallest = array[index];
13    array[index] = array[target];
14    array[target] = smallest;
15  }
16 }
```

(We also discussed how we compute the running time of this function.)

Insertion sort

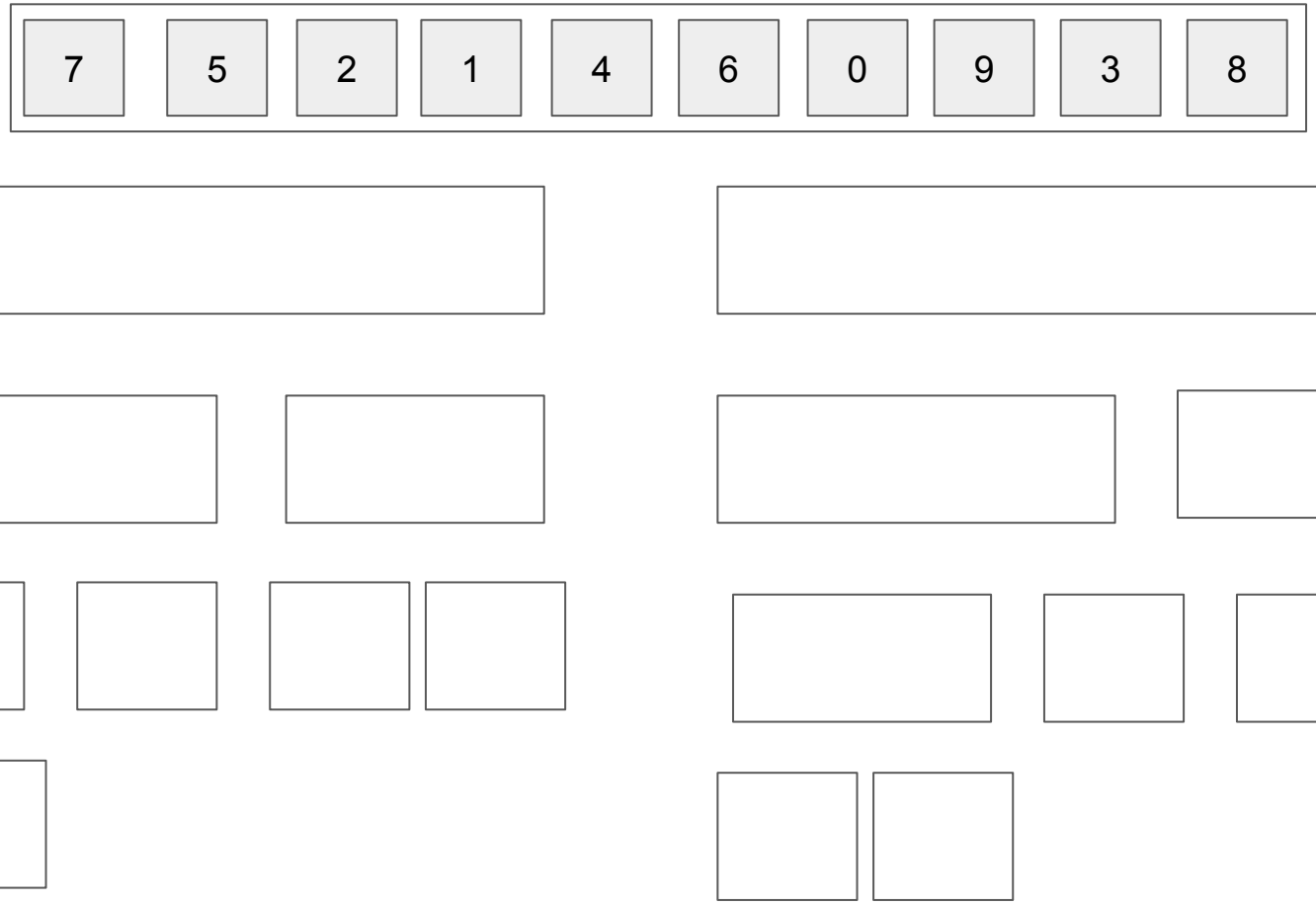
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)



In-Class Activity

Merge sort

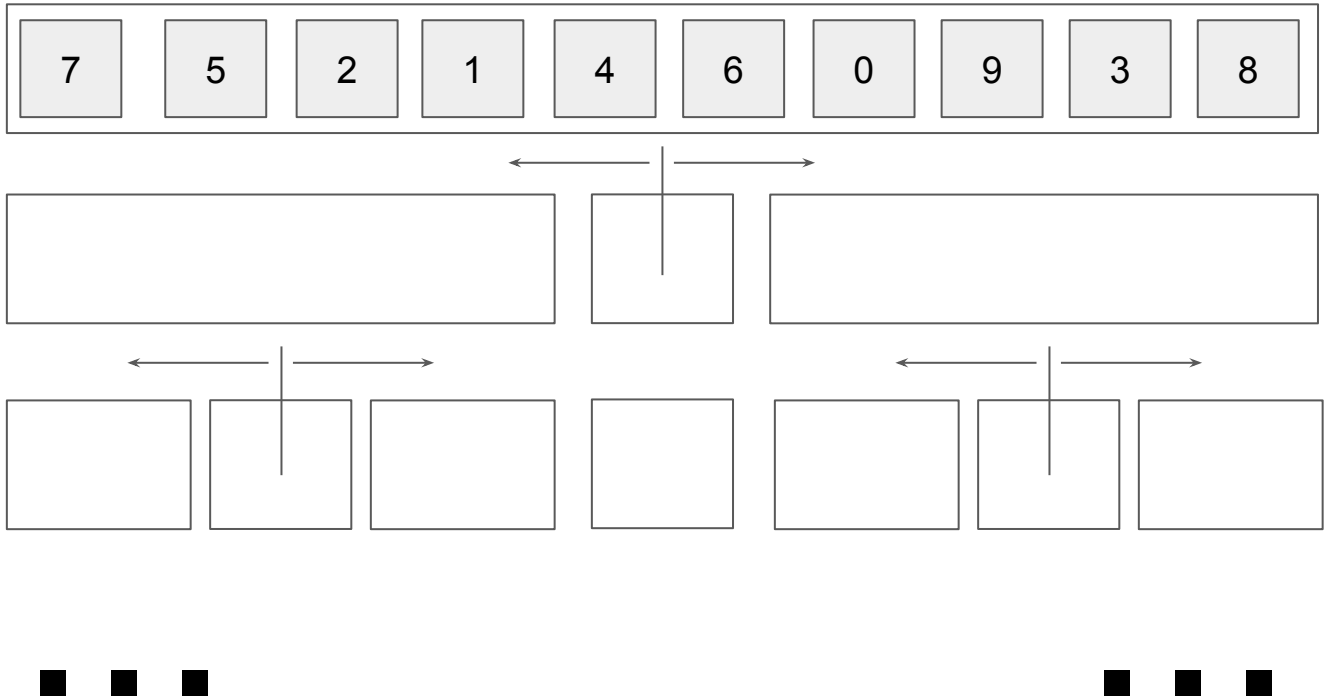
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)



merge phase

Quicksort

(Notes from the live demo or live coding. Please do NOT assume the code is complete.)



pivot selection
partition phase

Divide and conquer approach

Counting sort

(maybe next week)

In Your Pocket

arrays
linked lists
stacks
queues

man ssh exit pwd cd ls
valgrind touch mkdir cp
rm rmdir mv

Sorting Algorithms
- Selection sort

Some keywords from today's lecture:

- enumerations, enum class
- unit testing, integration testing, regression testing
- test coverage
- Test-Driven Development (TDD)
- asymptotic times
- constant, logarithmic, linear, quadratic, polynomial, exponential time
- asymptotic math
- T(n)
- Big-O, Big-Ω, Big-Θ notations
- Selection sort, Insertion sort
- Merge sort, merge phase
- Quicksort, pivot selection, partition phase
- divide and conquer approach

To the lab!