13 Jan 2026

Small Groups

① Introduce yourselves, classes, fun fact

② Inventory of Algos we know

Algorithms Inventory

Breadth-First Search
Depth-First Search
Balance (for B-trees)
Genetic Algorithms
BOGO/Monkey Sort
Bubble Sort
Quicksort ← Most Implementations!
Selection Sort
Merge Sort
Sorkan's "Sort"
Z-algorithm (matching)
Dijkstra's Algo (SP in weighted graph)
*Cox-Zucker (is a set a basis?)

Data Structure Inventory

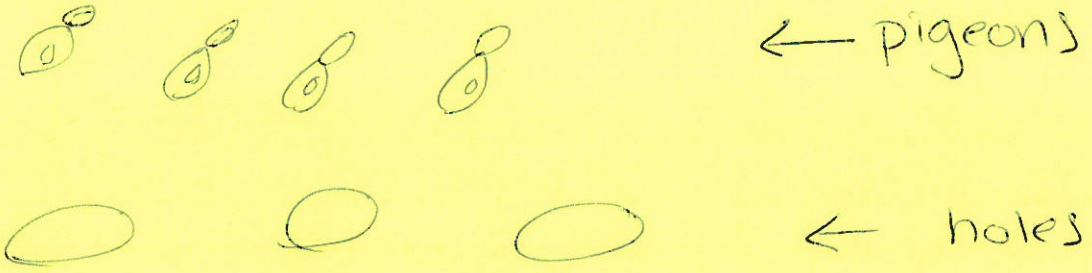
B-trees	set
stack	LL
queue	red-black t.
heap	bin. tree
graph	array
dictionary	nd-array
	tree

Problems

traveling salesman
search
sort
path finding (class of probs)
satisfiability (SAT)
shortest path
matching

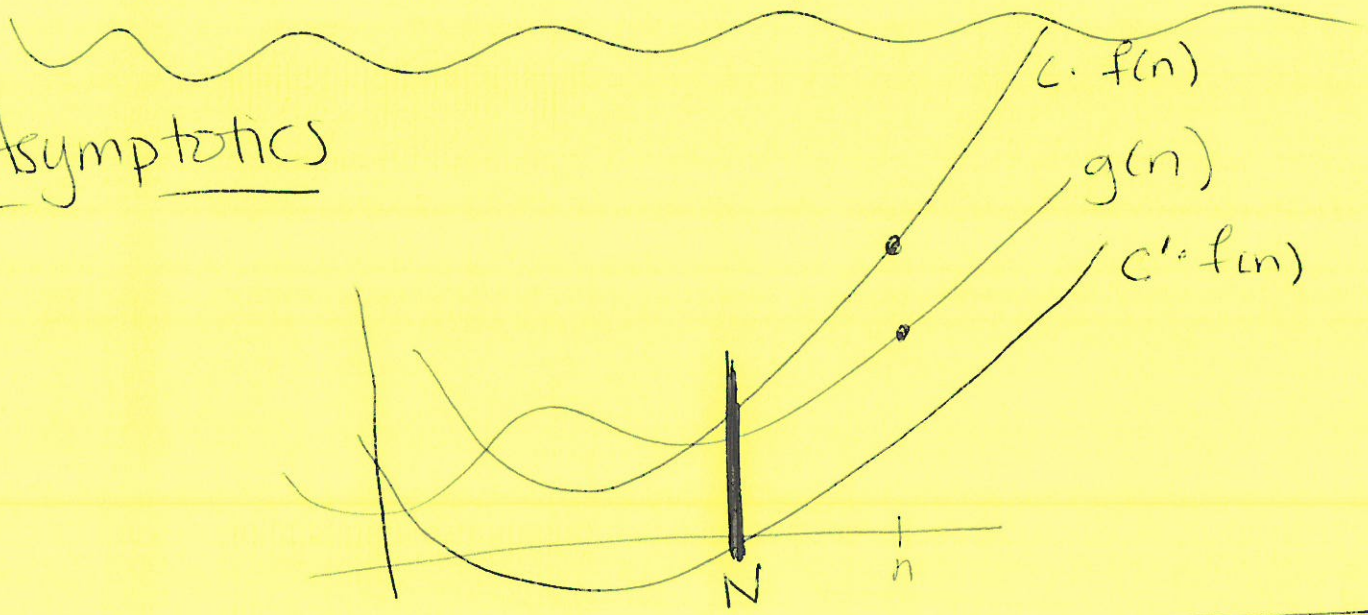
Pigeonhole principal:

ex:



↳ gives the existence of a hole w/ at least X pigeons (more formal later)

Asymptotics



$g(n)$ is $O(f(n))$ $\exists c \in \mathbb{R}, c>0$ $\forall n \geq N$ such that $\exists N \in \mathbb{R}$

$$g(n) \leq c \cdot f(n)$$

Upper Bound

$f(n)$ is $\Omega(f(n))$

$\exists N' \in \mathbb{R}, \exists c' \in \mathbb{R}, c>0$ such that

$\forall n \geq N'$, $g(n) \geq c' f(n)$

$g(n)$ is $\Theta(f(n)) \iff g(n) \in O(f(n))$ and $g(n) \in \Omega(f(n))$

What is an Algorithm?

- stmts / expressions / steps
- may or may not change the state of program
- specific way to solve a given problem
- sequence of steps that returns a result
  (↑ sometimes)
- formula / recipe
- finite!! (It must end ... otw, it is a procedure)
- well-defined
- repeatable / deterministic

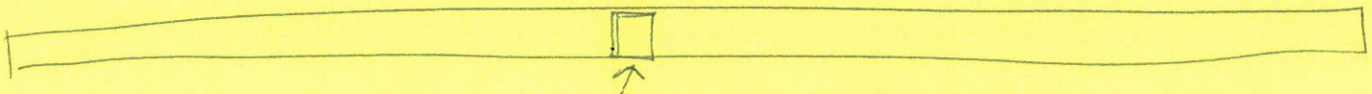
> well-defined
> a ✓ sequence of steps that solves a given problem in finite time

Def'n

Analysis Includes
- (0) Defining the algo { input, output, steps
- (1) Runtime
- (2) Correctness = does the algorithm do what it claims to do?

real-ram model of computation

- real #s: this is what our data is
- RAM = random access memory
  = data is stored in a very very large array



one unit of memory, stores a real #
+ can be accessed in $\Theta(1)$ time

Homework

1) Sign up for gradescope
2) review prerequisites (see email)
3) review today's handout

Two Linear Search Algorithms

The following algorithm takes as input a array L of real numbers and a query element q (a real number), and returns true if the element q is in the list L and false otherwise.

Algorithm 3 Linear Search II

Input: L, a list of real numbers
Output: q, a query element (a real number)

```
1: found ← FALSE
2: for i = 1 to |L| do
3:    if q=L(i) then
4:       found ← TRUE
5:    end if
6: end for
7: return found
```

Algorithm 4 Linear Search II

Input: L, a list of real numbers
Output: q, a query element (a real number)

```
1: for i = 1 to |L| do
2:    if q=L(i) then
3:       return TRUE
4:    end if
5: end for
6: return FALSE
```

Discuss the following questions:

1. Are these two pseudo-code examples of the same algorithm? Why or why not?
2. Do the two have the same runtimes?

worst-case analysis: ← what we typically do
Alg 3: $\Theta(n)$
Alg 4: $\Theta(n)$

best-case analysis
Alg 3: $\Theta(n)$ "linear"
Alg 4: $\Theta(1)$ "constant"

Search Algorithms

CSCI 432

January 13, 2026

Name:
Who did you work with today?

Two Algorithms

The following two algorithms takes as input a array L of real numbers and a query q (a real number).

Linear Search

Algorithm 1 Search I

Input: L, a list of real numbers
Output: q, a query element (a real number)

```
1: found ← FALSE
2: for i = 1 to |L| do
3:    if q=L(i) then
4:       found ← TRUE
5:    end if
6: end for
7: return found
```

$\Theta(n)$

Binary Search

Algorithm 2 Search II

Input: L, a list of n real numbers ← must be sorted!
Output: q, a query element (a real number)

```
1: found ← FALSE
2: left ← 1
3: right ← n
4: while left < right do
5:    mid ← ⌈(left+right)/2⌉        q = L(mid)
6:    if q=L(i) then
7:       found ← TRUE
8:    end if
9:    if L[mid]=q then
10:      found ← TRUE ; Left = right - 1
11:   else if L[mid] < q then
12:      left ← mid + 1
13:   else
14:      right ← mid − 1
15:   end if
16: end while
17: return found
```

$\Theta(\log n)$

Discuss the following questions:

1. What do these algorithms do? Search
2. Are these two pseudo-code examples of the same algorithm? Why or why not?
3. Do the two have the same runtimes?
4. Is one algorithm better than the other?
5. What are the common names of these two algorithms?