CS3000
5/8 - Mon. (First Day!)

Laney Strange (she/her)
lstrange@northeastern.edu, ME 313
OH M, W 12-2 pm

Ask/answer questions in class, say name

Agenda

1. Algorithms Overview ✓
2. Example Algorithm
3. About CS3000

# 1. Algorithms overview

What is an algorithm? (characteristics, examples, kinds of things)

- has a run time (ex: O(n))
- sorting algorithms (ex)
- list of steps to achieve an outcome
- defined set of specific steps
- has a cost (space)

## Algos in CS3000

- solve a computational problem
- Takes in an input, produces an output
  - input → array, number, string, data structure
  - output → solution, number, string, modified array, new array
- CS3000 is pencil-and-paper, programming language agnostic
- CLRS style pseudocode

## Arrays in CS3000

$A = [-, -, -, \ldots, -]$

similar to a Python list, Java/C# array

index from 1, not 0

Every value in A has a position

$A = [\sim_1, \sim_1, \sim, \ldots, \sim]$

pos 1, 2, 3, ..., A.length (n)

Use (usually) for loop to iterate over A

```
for i = 1 to A.length        } change every
    A[i] = -1                  value to -1
```

modifying the array in a function modifies the original

```
Func(A)
    for i = 1 to A.length    } no need to return A
        A[i] = -1
```

```
Func2(A)
    Let B[1, ..., A.length] be a new array
    for i = 1 to A.length    } make and return a new array
        B[i] = -1 · A[i]
    return B
```

**What makes a good algorithm?**

- time complexity } runtime cost/space ★
- space complexity
- solves the problem it's supposed to ★
- readability
- Solves the problem in <u>all</u> cases
- extensibility (?)

- have one purpose

**Correctness**

Proving correctness:

1. Loop invariant ☆
2. Proof by induction! :)

**Efficiency**

Show time/space

- Assume each step incurs a cost of 1
- Compute cost of each step, sum them all ☆
- find complexity class

## 2. Example Algorithm - Search

```
LINEAR SEARCH (A, key)
1.  for i = 1 to A.length
2      if A[i] == key
3          return i
4.  return -1
```

→ return position of key if found, or -1 if not there

Proving correctness: Loop invariant

- something true before the loop, during, and after the loop terminates

**Invariant** At the start of the loop in lines 1-4, $A[1..i-1]$ consists of elements originally in $A[1..i-1]$ and that are not == key

**1. Initialization** → prior to first iteration

- $A[1..i-1]$ when $i=1$ $A[1..0]$, empty array
- Loop inv. is trivially true

**2. Maintenance** → If true before iteration, it's true after

- We assume key is not in $A[1..i-1]$
- Compare key with $A[i]$, if same we return, loop/function are over!
- If different, i gets incremented (only thing that happens)
- At end of iteration, $A[1...i]$ does not have the key

**3. Termination** → Loop is over, the loop invariant shows the whole algo works

- When loop is over, $i = A.length + 1$
- By loop invariant, $A[1..A.length]$ does not have the key

## Linear Search - Efficiency

- How many steps does it take on an input of size $n$?

start with worst case

```
LINEAR SEARCH (A, key)
1.  for i = 1 to A.length
2.      if A[i] == key
3.          return i
4.  return -1
```

| Line # | Cost | # times |
|---|---|---|
| 1 | 1 | $n+1$ |
| 2 | 1 | $n$ |
| 3 | 1 | 0 |
| 4 | 1 | 1 |

Worst-case runtime of linear search:

$(n+1) + n + 0 + 1 = 2n + 2$

Preview: put an algorithm's runtime in a complexity class

- drop coefficients
- drop lower-order terms

} as n grows arbitrarily large, they don't matter!

$2n \to n$

$2 \to \emptyset$

$\Theta(n)$ → next time!

## 3. CS3000

Everything on the website!

### HW

- Short Tue - Thu
- Long Thu - Tue
- Submit up to 48 hours late

### Rec

- Tue - problem set, no graded
- Thu - Fun problem solving :)

### Exams

- 5/25 in class
- 6/15 in class