Introductions

Dr. | Prof | Professor
Dave | Kauchak

Pronouns: he/him/his

Algorithms

“For me, great algorithms are the poetry of computation. Just like verse, they can be terse, allusive, dense and even mysterious. But once unlocked, they cast a brilliant new light on some aspect of computing.”
– Francis Sullivan

What is an algorithm?

Example algorithms

- sort a list of numbers
- find a route from one place to another (cars, pocket routing, phone routing, ...)
- find the longest common substring between two strings
- add two numbers
- microchip wiring/design (VLSI)
- solve sudoku
- cryptography
- compression (file, audio, video)
- spell checking
- pagerank
- classify a web page
Pseudocode

A way to discuss how an algorithm works that is language agnostic and without being encumbered with actual implementation details.

Should give enough detail for a person to understand, analyze and implement the algorithm.

Pseudocode examples

Mystery 1(A)
1  \( x \leftarrow -\infty \)
2  for \( i \leftarrow 1 \) to \( \text{length}[A] \)
3      if \( A[i] > x \)
4          \( x \leftarrow A[i] \)
5  return \( x \)

Mystery 2(A)
1  for \( i \leftarrow 1 \) to \( \text{length}(A)/2 \)
2      swap \( A[i] \) and \( A[\text{length}(A) - (i - 1)] \)
### Pseudocode conventions

- Array indices start at 1 not 0
- We may use notation such as $\infty$, which, when translated to code, would be something like `Integer.MAX_VALUE`
- Use shortcuts for simple functions (e.g., swap) to make pseudocode simpler
- We'll often use ← instead of = to avoid ambiguity
- Indentation specifies scope

### Proofs

**What is a proof?**

A deductive argument showing a statement is true based on previous knowledge (axioms)

**Why are they important/useful?**

- Allows us to be sure that something is true
- In algs: allow us to prove properties of algorithms

### An example

**Prove the sum of two odd integers is even**

### Proof techniques?

- Example/counterexample
- Enumeration
- By cases
- By inference (aka direct proof)
- Trivially
- Contrapositive
- Contradiction
- Induction (strong and weak)
Proof by induction (weak)

- Proving something about a sequence of events by:
  1. first: proving that some starting case is true and
  2. then: proving that if a given event in the sequence were true then the next event would be true

13

Proof by induction example

- Prove: $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$
  1. **Base case:** prove some starting case is true
  2. **Inductive case:** Assume some event is true and prove the next event is true
     a. **Inductive hypothesis:** Assume the event is true (usually $k$ or $k-1$)
     b. **Inductive step to prove:** What you’re trying to prove assuming the inductive hypothesis is true
     c. **Proof of inductive step**

15

Proof by induction (weak)

1. **Base case:** prove some starting case is true
2. **Inductive case:** Assume some event is true and prove the next event is true
   a. **Inductive hypothesis:** Assume the event is true (usually $k$ or $k-1$)
   b. **Inductive step to prove:** What you’re trying to prove assuming the inductive hypothesis is true
   c. **Proof of inductive step**

14

Base case

- Prove: $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$

Show it is true for $n = 1$

$$\sum_{i=1}^{n} i = 1 = \frac{1 \cdot 2}{2}$$

16
Inductive case

Prove: $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$

Inductive hypothesis: assume $n = k - 1$ is true

$$\sum_{i=1}^{k-1} i = \frac{(k-1) \cdot k}{2}$$

Inductive case: proof

Prove: $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$

IH: $\sum_{i=1}^{k-1} i = \frac{(k-1) \cdot k}{2}$

$$\sum_{i=1}^{k} i = \frac{k(k+1)}{2}$$

$$\sum_{i=1}^{k} i = k + \sum_{i=1}^{k-1} i$$

by definition of sum

$$= k + \frac{(k-1) \cdot k}{2}$$

by IH

$$= \frac{k^2 + (k-1) \cdot k}{2}$$

$$= \frac{k^2 + (k-1) \cdot k}{2}$$

$$= \frac{k^2 + k - 1}{2}$$

$$= \frac{k(k+1)}{2}$$

Why does this work?
Sorting

Input: An array of numbers A
Output: The number in sorted order, i.e.,

\[ A[i] \leq A[j] \quad \forall i < j \]

What sorting algorithm?

1. for \( j \leftarrow 2 \) to \( \text{length}[A] \)
2. \( \text{current} \leftarrow A[j] \)
3. \( i \leftarrow j - 1 \)
4. while \( i > 0 \) and \( A[i] > \text{current} \)
5. \( A[i+1] \leftarrow A[i] \)
6. \( i \leftarrow i - 1 \)
7. \( A[i+1] \leftarrow \text{current} \)

Insertion-Sort(A)

1. for \( j \leftarrow 2 \) to \( \text{length}[A] \)
2. \( \text{current} \leftarrow A[j] \)
3. \( i \leftarrow j - 1 \)
4. while \( i > 0 \) and \( A[i] > \text{current} \)
5. \( A[i+1] \leftarrow A[i] \)
6. \( i \leftarrow i - 1 \)
7. \( A[i+1] \leftarrow \text{current} \)

Does it terminate?

Is it correct?

How long does it take to run?

Memory usage?