Algorithms

CSCI 1030U - Intro to Computer Science
@IntroCS

Randy J. Fortier
@randy_fortier

Ontario Tech University

Outline

- Algorithm basics
  - Pseudo code
  - Example (insertion sort)
  - Basic algorithm analysis

What are Algorithms?

Algorithms

- Modern algorithm development began in the 1930s with:
  - Kurt Gödel
  - Alonzo Church
  - Alan Turing

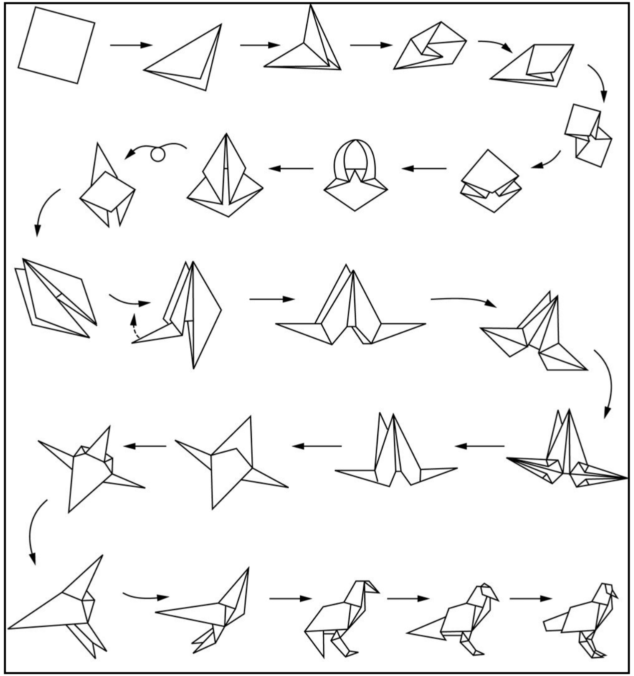
Algorithms

- To be an algorithm, a strategy must meet the following criteria:
  - The process must eventually terminate
  - The steps must be finite
  - The steps must be unambiguous (executable)

How Do We Represent Algorithms?

Ontario Tech University

An Algorithm - Origami



Algorithm Representation

- How do we represent algorithms?

Algorithm Representation

- How do we represent algorithms?
  - We could just write the algorithm directly in some programming language

Algorithm Representation

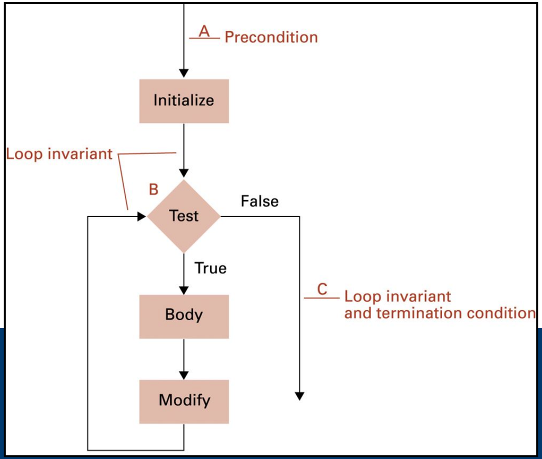
- How do we represent algorithms?
  - We could just write the algorithm directly in some programming language, however:
    - It would only be useful to someone who understands that programming language
    - Different programming languages have different syntax shortcuts (often called syntactic sugar)
    - The simplicity of implementing the algorithm in some languages (e.g. Haskell, Scheme, Python) is misleading when it comes to implementing the algorithm in an industry standard language (C++, Java, C, JavaScript)

Algorithm Representation

- How do we represent algorithms?
  - We could just write the algorithm directly in some programming language
  - We could draw a flow chart to describe the algorithm

Algorithm Representation

- How do we represent algorithms?
  - We could just write the algorithm directly in some programming language
  - We could draw a flow chart to describe the algorithm
    - This is quite cumbersome and unwieldy, however:



Algorithm Representation

- How do we represent algorithms?
  - We could just write the algorithm directly in some programming language
  - We could draw a flow chart to describe the algorithm
  - We could use a neutral, programming-language-like notation to represent the algorithm

Algorithm Representation

- How do we represent algorithms?
  - We could just write the algorithm directly in some programming language
  - We could draw a flow chart to describe the algorithm
  - We could use a neutral, programming-language-like notation to represent the algorithm
    - Pseudo-code:

```
INSERT-SORT(A)
  for j = 2 to length[A] do
     key = A[j]
     i = j-1
     while i > 0 and A[i] > key do
        A[i+1] = A[i]
        i = i-1
     A[i+1] = key
```

Case Study - Insertion Sort

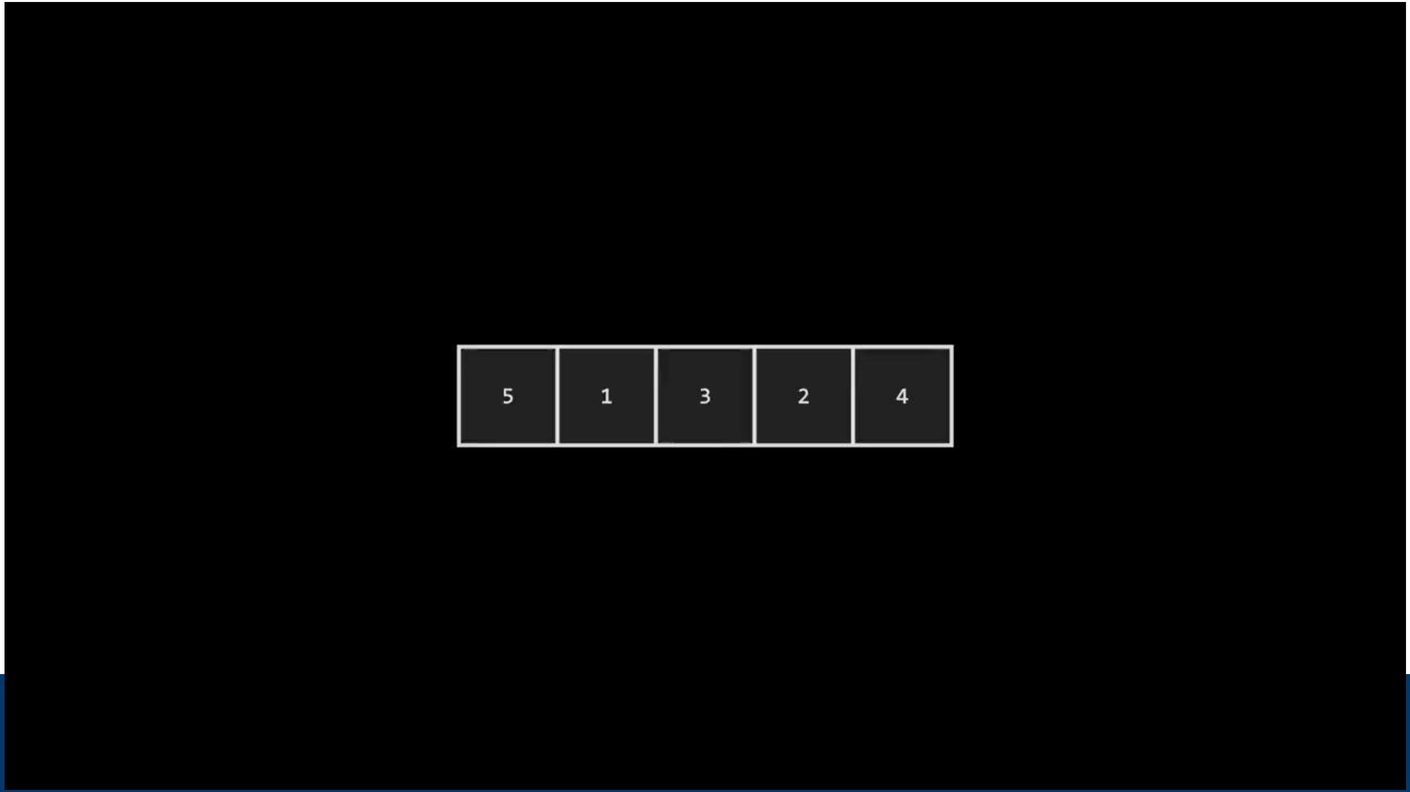
Algorithm Example - Insertion Sort

- In Computer Science, there are many sorting algorithms:
  - Insertion sort
  - Quick sort
  - Merge sort
  - Heap sort

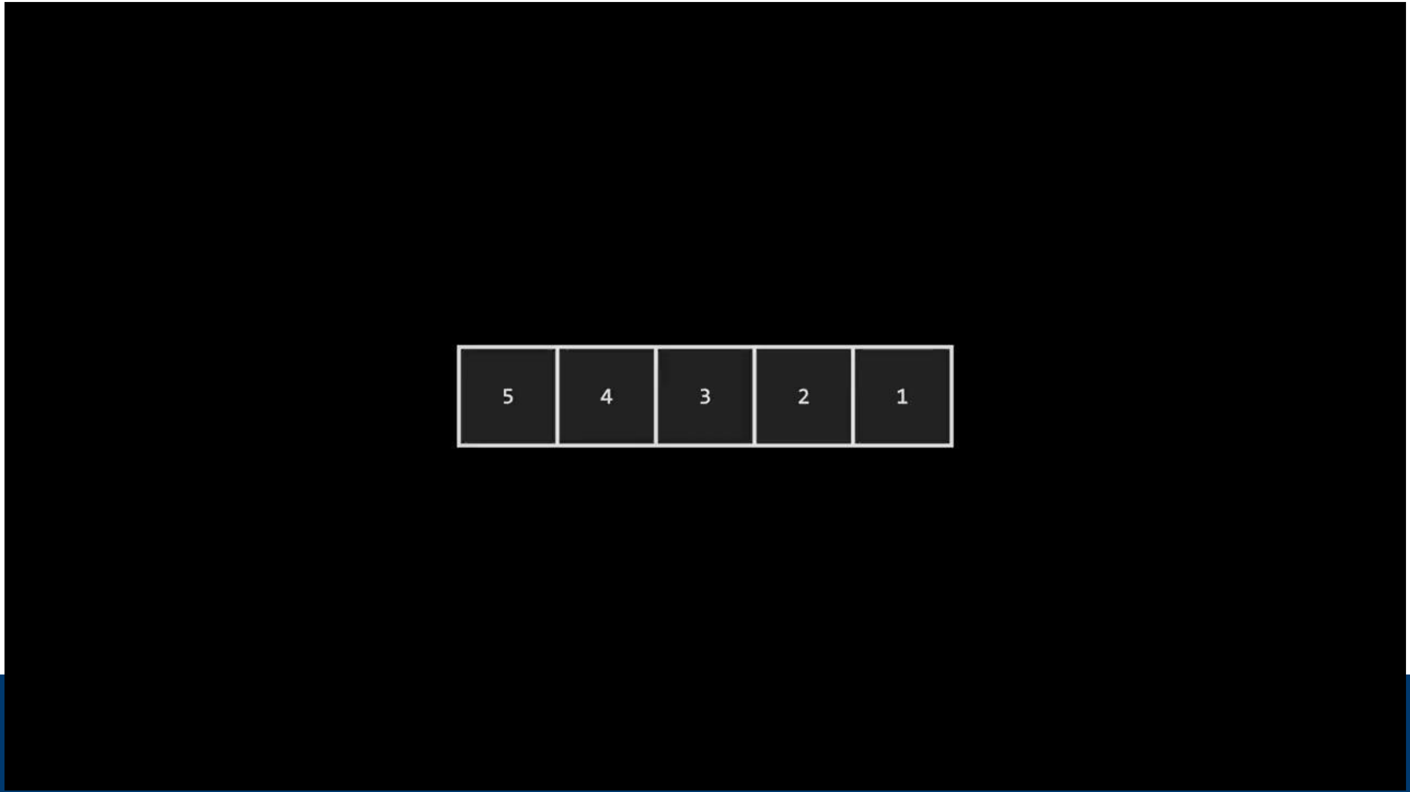
Algorithm Example - Insertion Sort

- The general idea behind insertion sort
  - Some part of the list (sorted sublist) is always sorted
  - Takes one element at a time, and inserts it into the sorted sublist (in the correct position)
  - When all elements have been added to the sorted sublist, the list has been sorted
- To start, consider this question:
  - Does a list with one element need to be sorted?

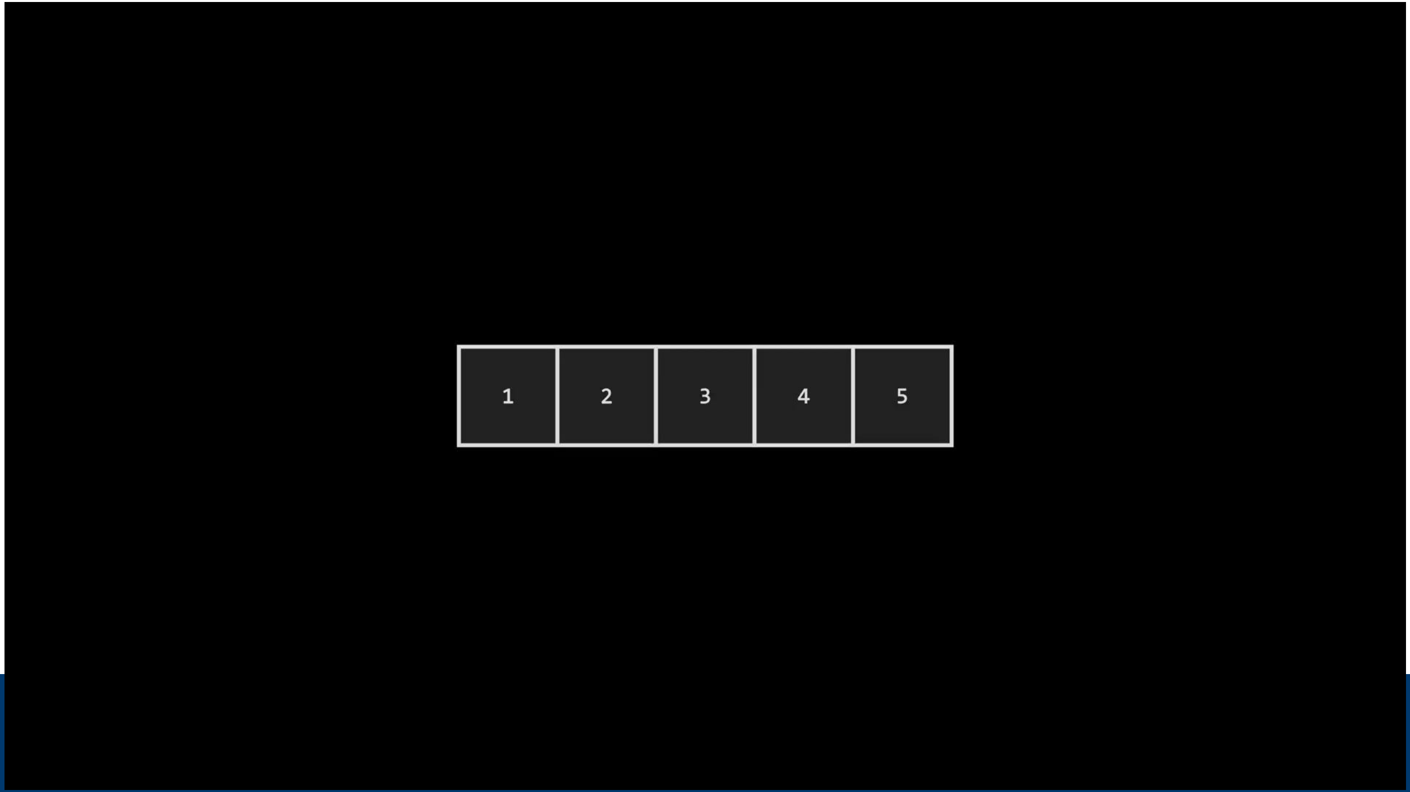
Insertion Sort - Video Example



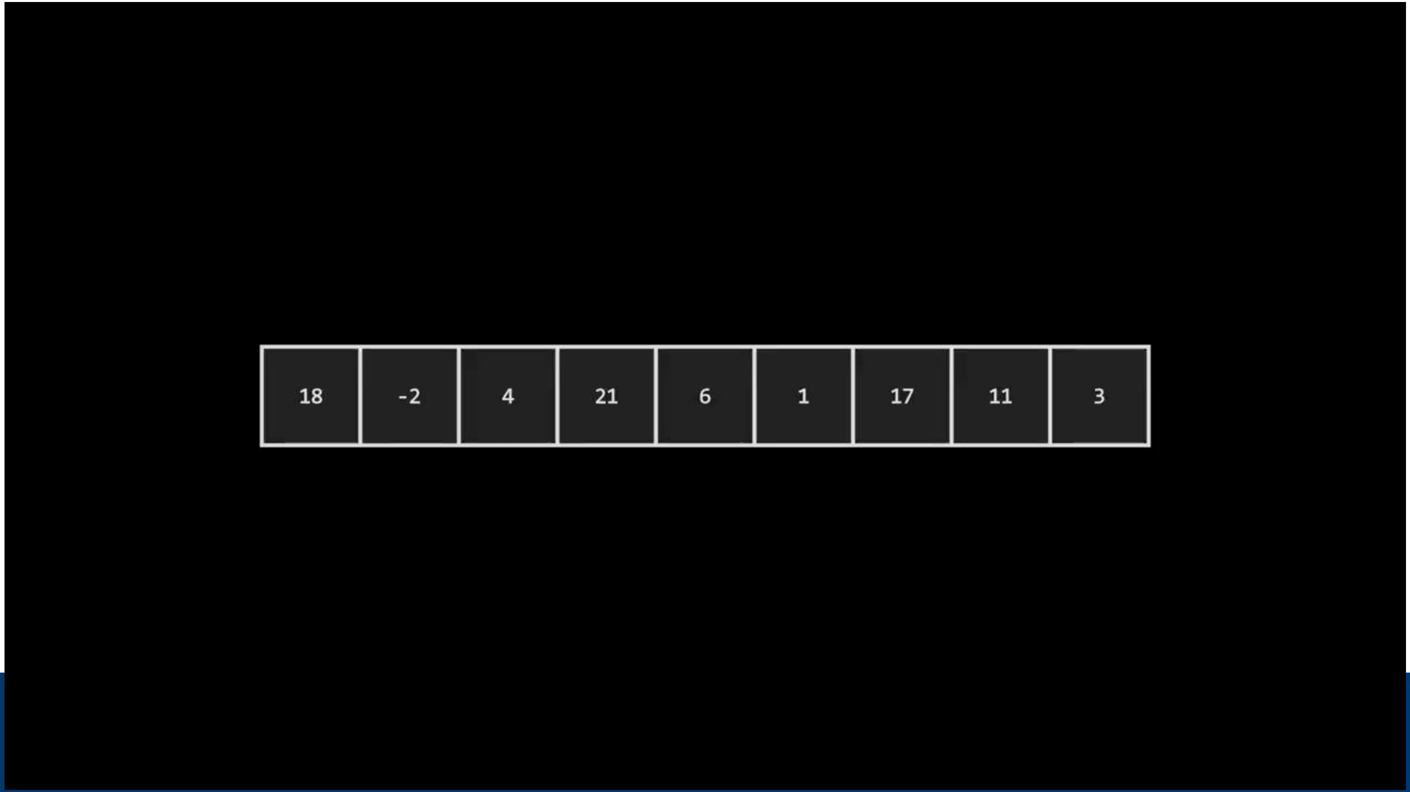
Insertion Sort - Video Example



Insertion Sort - Video Example



Insertion Sort - Video Example



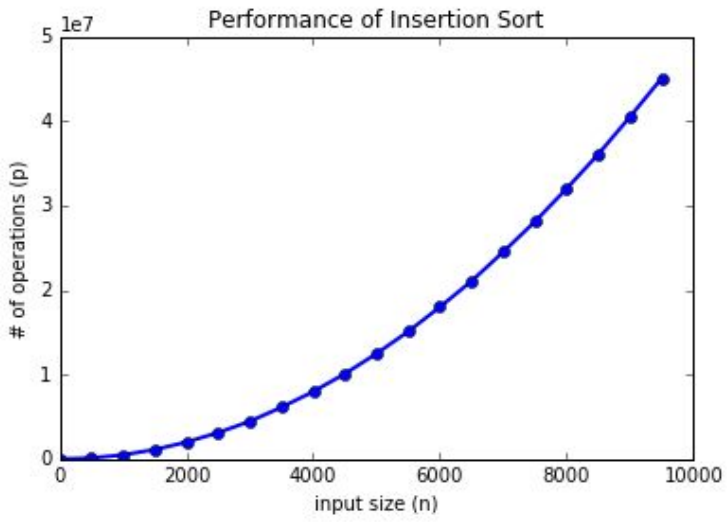
Insertion Sort - Pseudo-code

```
INSERT-SORT(A)
for j = 2 to length[A] do
    key = A[j]
    i = j-1
    while i > 0 and A[i] > key do
        A[i+1] = A[i]
        i = i-1
    A[i+1] = key
```

Coding Exercise 09a.1

- Let's write a variation of the insertion sort algorithm that builds up the sorted part of the list at the end of the list, rather than the start
  - The list items will still be sorted in ascending order

Insertion Sort - Performance



Coding Exercise 09a.2

- Let's use operation counting to estimate the performance of the following algorithm
  - The operations to be counted are the number of comparisons made
  - Try collecting data for lists of size 10, 100, 1000, and 10000

```
def sequential_search(values, to_find):
    for i in range(len(values)):
        if values[i] == to_find:
            return True
    return False
```

Wrap-up

- Algorithm basics
  - Pseudo code
  - Example (insertion sort)
  - Basic algorithm analysis

Coming Up

- Algorithm strategies
  - Divide and conquer
    - Binary search
  - Greedy Algorithms
    - Fractional knapsack problem
  - Dynamic Programming
    - Fibonacci