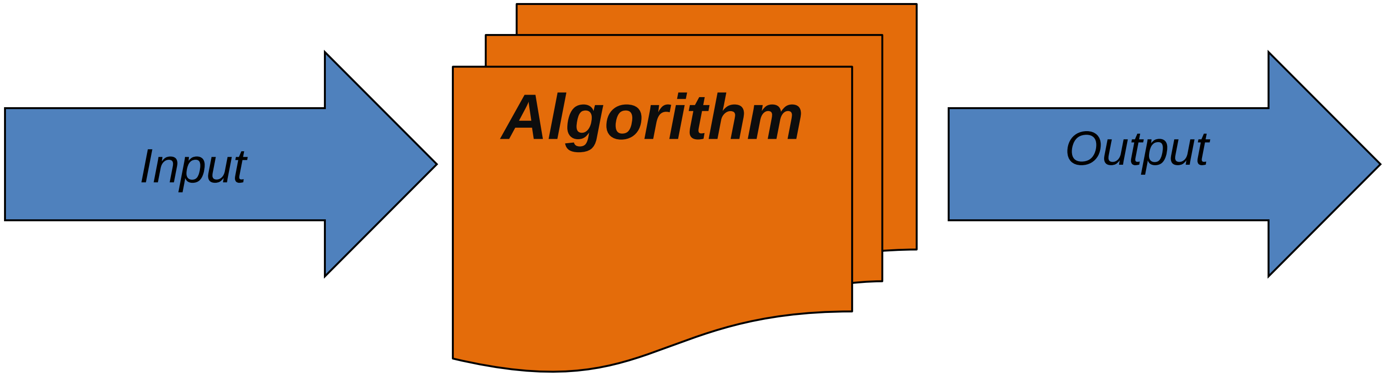
Algorithms Design and Analysis

What is an algorithm?

- A computational procedure that takes some value, or set of values, as ***input*** and produces some value, or set of values, as ***output***.
- A sequence of computational steps that transform the input into the output.



What is an algorithm?

- A computational problem is a mathematical problem, specified by an input/output relation.
- An algorithm is a computational procedure for solving a computational problem.

- Example: Sorting
  - **Input**: A sequence of N numbers $a_1 \ldots a_n$
  - **Output**: the permutation (reordering) of the input sequence such that $a_1 \leq a_2 \leq \ldots \leq a_n$

What will we study?

- Expressing algorithms
  - Define a problem precisely and abstractly
  - Presenting algorithms using pseudocode
- Algorithm validation
  - Prove that an algorithm is correct
- Algorithm analysis
  - Time and space complexity
  - What problems are so hard that efficient algorithms are unlikely to exist
- Designing algorithms
  - Algorithms for classical problems
  - Meta algorithms (classes of algorithms) and when you should use which

Pseudocode

- High-level description of an algorithm
- More structured than English prose
- Less detailed than a program
- Preferred notation for describing algorithms
- Hides program design issues

Example: find max element of an array

Algorithm *arrayMax*(*A*, *n*)
  Input array *A* of *n* integers
  Output maximum element of *A*

  currentMax $\leftarrow$ *A*[0]
  for *i* $\leftarrow$ 1 **to** *n* – 1 **do**
    if *A*[*i*] > *currentMax* **then**
      currentMax $\leftarrow$ *A*[*i*]
  return *currentMax*

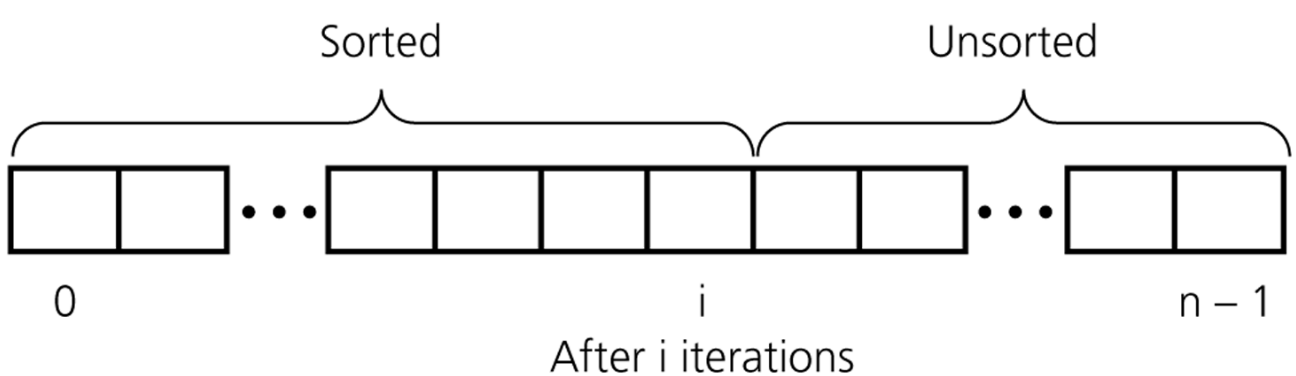
Algorithm Design

- Learn general approaches to algorithm design
  - Divide and conquer
  - Greedy method
  - Dynamic Programming
  - Basic Search and Traversal Technique
  - Graph Theory
  - Linear Programming
  - Approximation Algorithm
  - NP Problem

Sorting Algorithms

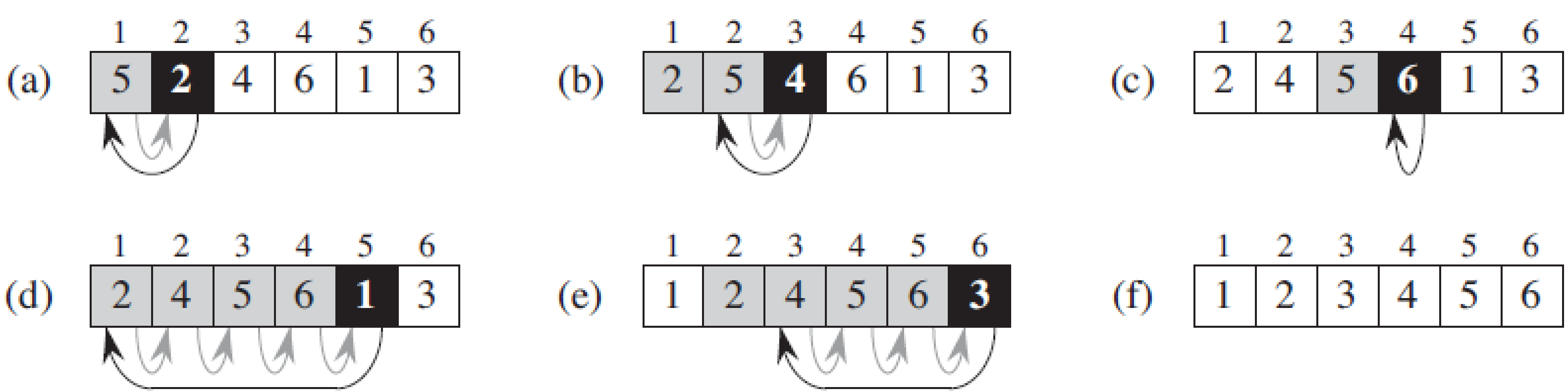
Insertion Sort

- Divide the list into two portions: sorted and unsorted.
- while some elements unsorted:
  - Using linear search, find the location in the sorted portion where the 1st element of the unsorted portion should be inserted
  - Move all the elements after the insertion location up one position to make space for the new element



Insertion Sort

```
for j = 2 to A.length
    key = A[j]
    // Insert A[j] into the sorted sequence A[1 .. j − 1].
    i = j − 1
    while i > 0 and A[i] > key
        A[i + 1] = A[i]
        i = i − 1
    A[i + 1] = key
```



Analysis of Insertion Sort

INSERTION-SORT(A)	*cost*	*times*
for $j = 2$ **to** $A.length$	c_1	n
$key = A[j]$	c_2	$n - 1$
// Insert $A[j]$ into the sorted sequence $A[1 \,..\, j-1]$.	0	$n - 1$
$i = j - 1$	c_4	$n - 1$
while $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
$A[i+1] = A[i]$	c_6	$\sum_{j=2}^{n} (t_j - 1)$
$i = i - 1$	c_7	$\sum_{j=2}^{n} (t_j - 1)$
$A[i+1] = key$	c_8	$n - 1$

The running time T(n) =

$$c_1 n + c_2(n-1) + c_4(n-1) + c_5 \sum_{j=2}^{n} t_j + c_6 \sum_{j=2}^{n} (t_j - 1) + c_7 \sum_{j=2}^{n} (t_j - 1) + c_8(n-1)$$

Analysis of Insertion Sort

- For the best case: $t_j = 1$ for $j = 2,3,\ldots\ldots,n$

$$T(n) = c_1n + c_2(n-1) + c_4(n-1) + c_5(n-1) + c_8(n-1)$$
$$= (c_1 + c_2 + c_4 + c_5 + c_8)n - (c_2 + c_4 + c_5 + c_8)$$

it is thus a ***linear function*** of n.

- For the worst case: $t_j = j$ for $j = 2,3,\ldots\ldots,n$

$\sum_{j=2}^{n} j = \frac{n(n+1)}{2} - 1$ and

$\sum_{j=2}^{n} (j-1) = \frac{n(n-1)}{2}$

Analysis of Insertion Sort

$$
\begin{aligned}
T(n) &= c_1 n + c_2(n-1) + c_4(n-1) + c_5\left(\frac{n(n+1)}{2} - 1\right) \\
&\quad + c_6\left(\frac{n(n-1)}{2}\right) + c_7\left(\frac{n(n-1)}{2}\right) + c_8(n-1) \\
&= \left(\frac{c_5}{2} + \frac{c_6}{2} + \frac{c_7}{2}\right)n^2 + \left(c_1 + c_2 + c_4 + \frac{c_5}{2} - \frac{c_6}{2} - \frac{c_7}{2} + c_8\right)n \\
&\quad - (c_2 + c_4 + c_5 + c_8)\,.
\end{aligned}
$$

It is thus a ***quadratic function*** of n