Sorting
Insertion and Bubble Sort

S.V. N. (vishy) Vishwanathan

University of California, Santa Cruz
vishy@ucsc.edu

January 6, 2016
Outline

1 Sorting

2 Insertion Sort

3 Bubble Sort
Sorting

Input
- An array e.g., [1, 5, 6, 7, 3, 2, 1]

Output
- Sorted array [1, 1, 2, 3, 5, 6, 7]
Formally

**Input**
- An array e.g., \([a_1, a_2, a_3, \ldots, a_n]\)

**Output**
- A permuted version of the array \([a'_1, a'_2, a'_3, \ldots, a'_n]\) such that \(a'_1 \leq a'_2 \leq a'_3 \leq \ldots \leq a'_n\)
What is an Algorithm?

- A sequence of well defined computational steps that
  - takes some input
  - transforms it into an output (with a certain property)

Alternatively

- An algorithm describes a specific computational procedure for achieving a specified input/output relationship
Outline

1 Sorting

2 Insertion Sort

3 Bubble Sort
Game of Cards
On an Array
Insertion Sort

**Insertion-Sort**

**Insertion-Sort**

1. **for** $j = 2$ **to** $A.length$
2. \( key = A[j] \)
4. \( i = j - 1 \)
5. **while** \( i > 0 \) and \( A[i] > key \)
7. \( i = i - 1 \)
8. \( A[i + 1] = key \)
Proving Correctness

- Hint: Think of the property/invariant that the algorithm maintains.
Proving Correctness

- Loop Invariant: At the start of each iteration of the \textit{for} loop of lines 1–8, the subarray $A[1..j-1]$ consists of the elements originally in $A[1..j-1]$, but in sorted order.
Proof by Induction

- Typically consists of three parts. You need to show that
  1. The property holds at the beginning (Initialization)
  2. The algorithm maintains the property in the intermediate stages (Maintenance)
  3. If the property holds when the algorithm finishes, then it solves your problem (Termination)
Proof

**Initialization**
- \( j = 2 \) which implies that \( A[1 \ldots j - 1] = A[1] \) is (trivially) sorted

**Maintenance**
- The while loop ensures that all \( A[i] > A[j] \) are shifted by one position to the left.
- This creates a “hole” in the array for inserting \( A[j] \) such that:
  - all elements to the right of \( A[j] \) are smaller than \( A[j] \)
  - all elements to the left of \( A[j] \) are larger than \( A[j] \)
- Since \( A[1 \ldots j - 1] \) was already sorted, so \( A[1 \ldots j] \) is now sorted

**Termination**
- \( j = n + 1 \) which implies that \( A[1 \ldots n] = A \) is sorted
Exploration

Question: Can we speed up insertion sort?
Outline

1. Sorting
2. Insertion Sort
3. Bubble Sort
Bubble Sort

BubbleSort(A)

1. for 1 = 1 to A.length - 1
2. for j = A.length downto i + 1
Bubble Sort

On an Array
Proving Correctness

- Hint: Think of the property/invariant that the algorithm maintains.
Proving Correctness

- Loop Invariant: At the start of each iteration of the for loop of lines 2-4, \( A[j] = \min \{ A[k] : j \leq k \leq n \} \) and the subarray \( A[j..n] \) is a permutation of the values that were in \( A[j..n] \) at the time that the loop started.
Questions?