INTRO AND INSERTION SORT

CS340

Your experience

- You have had
  - CS 234 Database and Web Development
  - CS 286 Computer Architecture
  - CS 314 Operating Systems
  - CS 325 Software Engineering
  - CS 321 Human-Computer Interaction
  - Math 224 Discrete Math
  - Calculus and other math?
  - Artificial Intelligence
  - Networking
  - Other?

Your goals

- Professions
  - Software Engineer
  - Application Developer
  - IT / Support
  - Database Administrator
  - Web Developer
  - Information Security Analyst
  - Researcher / Educator
  - Other?

Your skills and attributes

- I know Python or Java?
- I know more than 4 programming languages.
- I am a good Object-Oriented programmer.
- I know databases / database programming.
- I am a good communicator.
- I am a good technical writer.
- I like to test software.
- I am comfortable making presentations.
- I am good at math.
- I am comfortable writing a proof.
- I have a job and work more than 20 hours per week.
- I have had professional programming experience or an internship.

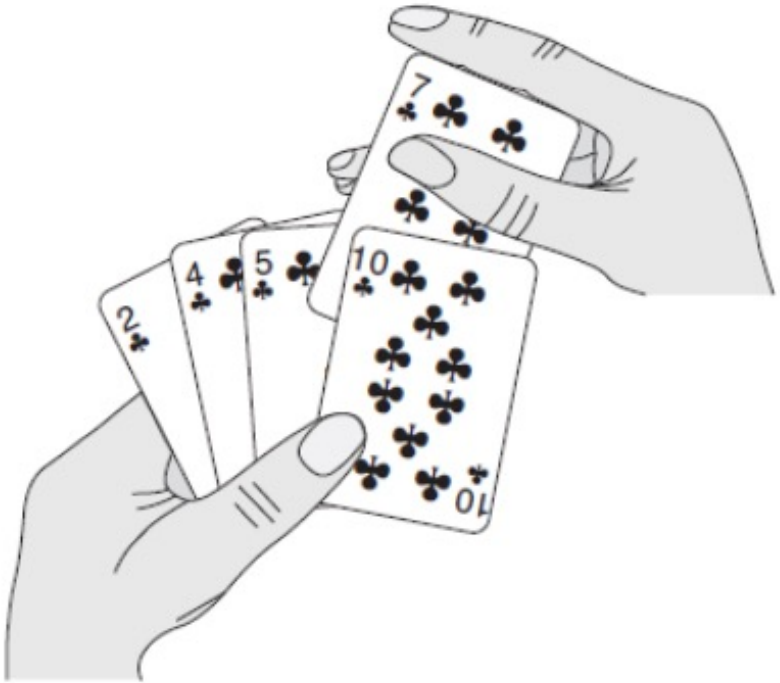
Algorithms

- An algorithm is any well-defined computational procedure that takes some value, or set of values, as **input** and produces some value, or set of values, as **output**.

Sorting: Insertion Sort

- Input
  - a sequence of n numbers $\langle a_1, a_2, a_3, \ldots a_n \rangle$
- Output
  - a reordering $\langle a_1', a_2', a_3', \ldots a_n' \rangle$ of the input sequence such that $a_1' \leq a_2' \leq a_3' \leq \ldots \leq a_n' \rangle$
  - Often the numbers are stored in an array, A

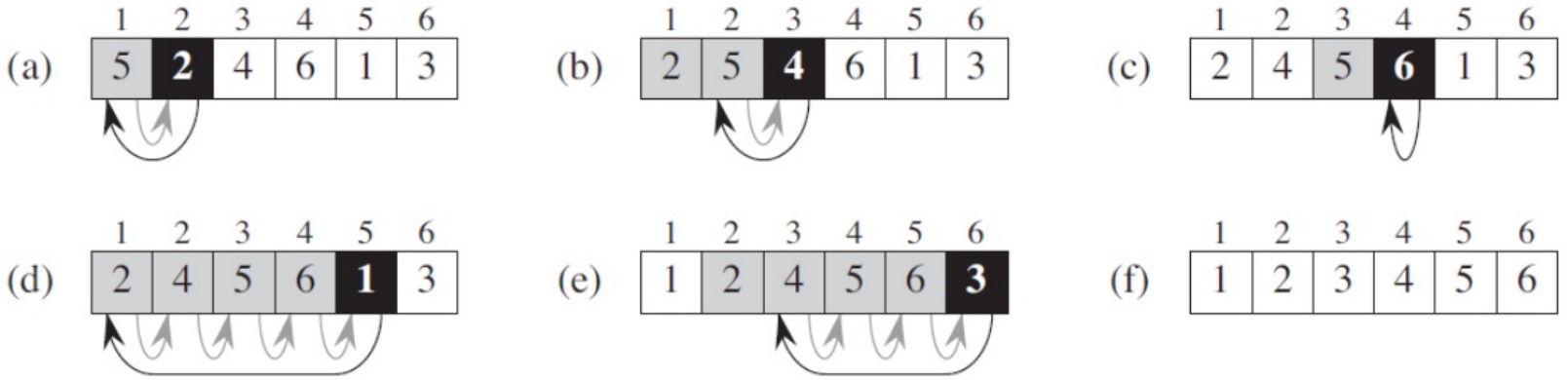
Insertion Sort



Insertion Sort

- The idea is to insert the current element into a sub-array that is assumed to be sorted.
- What we need:
  - A loop L1 going through the elements A[i]
  - Within L1 we need to perform insertion of
    - the current element A[i] within already sorted sub-array A[1], … ,A[i-1]

Insertion Sort



(c) no elements are moved
(d) all elements are moved

Insertion Sort

- If A[i] is already in the “correct position”
  - No extra “insertion” needed
  - What it means: A[i] > A[i-1]
  - Namely, when A[i] is the maximum of the subarray
- On the other hand, if A[i] is the minimum of the subarray
  - Namely, when A[i] < A[1]
  - Then, A[i] needs to be inserted into the first index.
  - **Entire subarray must be right-shifted.**

Insertion Sort

- The larger A[i] is in comparison to subarray A[1]…A[i-1]
  - The smaller the amount of right-shifting.
- Whereas L1 goes left to right through the A[i]
  - There is an inner loop L2
  - used for insertion into subarray
  - which goes right to left

Insertion Sort

```
INSERTION-SORT(A)
1  for j = 2 to A.length
2      key = A[j]
3      // Insert A[j] into the sorted sequence A[1 .. j − 1].
4      i = j − 1
5      while i > 0 and A[i] > key
6          A[i + 1] = A[i]
7          i = i − 1
8      A[i + 1] = key
```

Proving correctness with a loop invariant

3 Things must be shown

1. **Initialization**: It is true prior to the first iteration of the loop.
2. **Maintenance**: If it is true before an iteration of the loop, it remains true before the next iteration.
3. **Termination**: When the loop terminates, the invariant—usually along with the reason that the loop terminated—gives us a useful property that helps show that the algorithm is correct.

Loop invariant for insertion sort

- Loop invariant: At the start of each iteration of the “outer” for loop—the loop indexed by j—the subarray A[1.. j-1] consists of the elements originally in A[1.. j-1] but in sorted order.
- In other words, A[1..j-1] is a sorted subarray.
- Notice that this tells us about the algorithm and gives us a goal:
  - If the sorted subarray can consist of all items in the array, then the array will be sorted!

Correctness of insertion sort

- **Initialization**
  - Just before the first iteration, j = 2. The subarray A[1.. j-1] is the single element A[1], which is **trivially** sorted.
- **Maintenance:**
  - The body of the inner **while** loop puts *key* (which has the value that started out in A[j]) into the correct sorted position. The subarray A[1..j] then consists of the elements originally in A[1..j], but in sorted order.
- **Termination:**
  - The outer **for** loop ends when j > n, which occurs when j=n+1. Therefore, j-1=n. Plugging n in for j-1 in the loop invariant, the subarray A[1..n] consists of the elements originally in A[1..n] but in sorted order. In other words, the entire array is sorted.

Interview Questions

- Write a proof of correctness for LINEAR-SEARCH, which scans through a sequence, looking for value v. Use a loop invariant to prove that your algorithm is correct. (Make sure that your loop invariant fulfills the three necessary properties – initialization, maintenance, termination.)

14	3	7	2	27
A[1]	A[2]	A[3]	A[4]	A[5]

```
LINEAR-SEARCH(A, v)
for i = 1 to A.length
   if A[i] == v
      return i
return NIL
```