Bubble Sort

Advanced Higher Computing Science

Learning Objectives

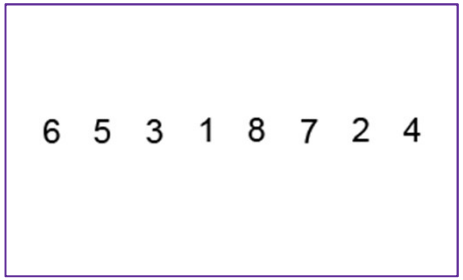
By the end of this lesson you will be able to:

- ❏ Describe, exemplify, and implement a bubble sort algorithm

Bubble Sort

A **bubble sort** compares adjacent items e.g. 1 and 2, 2 and 3 etc.

1. The largest values will ‘bubble’ to the top
2. If the first item is larger than the second then items are swapped
3. Repeats the process from the beginning of the list with the unsorted part of the list
4. Stops when no further swaps take place



Algorithm - Fixed Loop Bubble Sort

```
FOR i FROM length-2 TO 0 STEP -1 DO
        FOR counter FROM 0 TO i DO
                IF list[counter ]>list[counter +1] THEN
                        SET temp TO list[counter +1]
                        SET list[counter +1] TO list[counter ]
                        SET list[counter] TO temp
                END IF
        END FOR
END FOR
```

Python Implementation

```
#start from the right
for outer in range (len(listA)-1,0,-1):
    for inner in range(outer):
        #compare two adjacent values
        if listA[inner]>listA[inner+1]:
                #assign one of the values to a temp variable
            temp = listA[inner]
            #overwrite one of the values
            listA[inner] = listA[inner+1]
            #replace with the temp value
            listA[inner+1] = temp
```

What if we are using records/classes

Assuming we are just using public properties and using the following class structure:

```
class BMI():
    def __init__(self):
        self.height = 0.0
        self.weight = 0.0
        self.BMI = 0.0
```

What if we wanted to sort in order of BMI?

We will assume we have an array of these BMI objects held in an array called **myBMI**.

Python Implementation (records/classes)

```
for outer in range (len(myBMI)-1,0,-1):
    passes += 1
    for inner in range(outer):
        comparisons += 1

        if myBMI[inner].BMI>myBMI[inner+1].BMI:
            swaps += 1
            temp = myBMI[inner]
            myBMI[inner] = myBMI[inner+1]
            myBMI[inner+1] = temp

    print("Pass ",passes)
```

You will notice that we are swapping the whole record and not individual properties

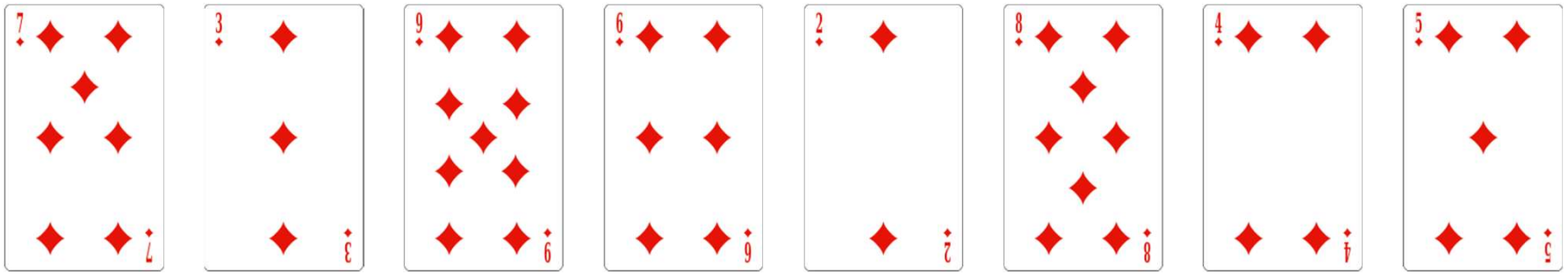
Problem with the previous algorithm

The previous algorithm uses two **fixed** loops

We can improve the efficiency of the algorithm by using a Boolean flag to indicate whether there has been any swaps, if there hasn't been any then there is no need to traverse the rest of the list

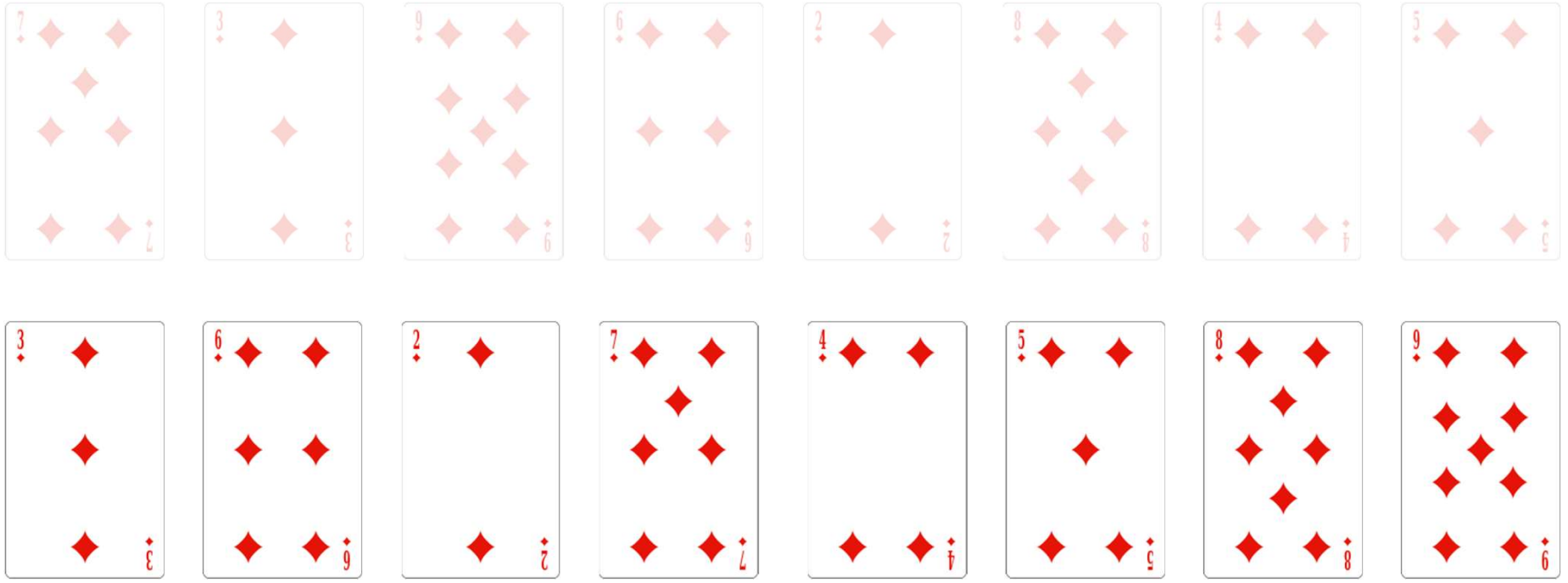
If there are **no swaps** then the outer loop should be terminated, so it has to be a **conditional loop**

Bubble Sort - Cards

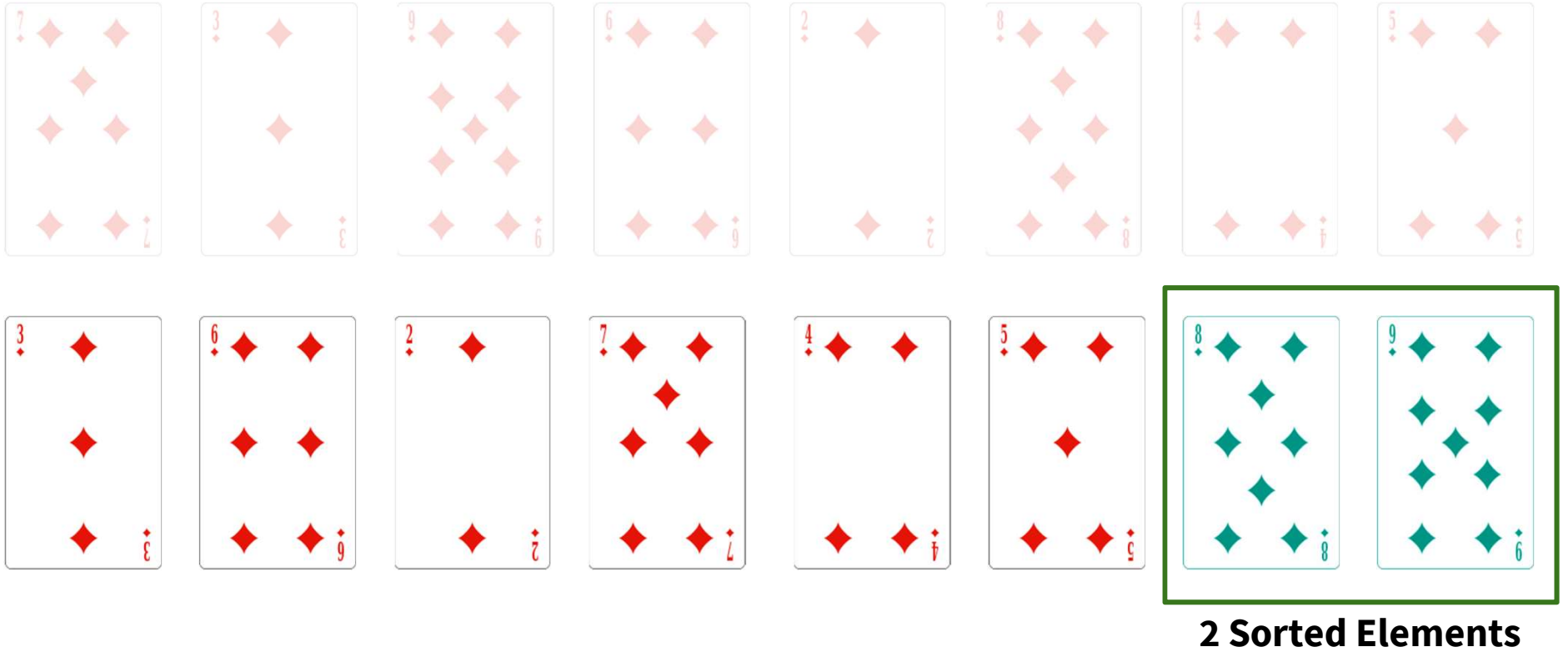


What will the list look like after 2 passes?

Bubble Sort – Cards (After 2 Passes)



Bubble Sort – Cards (After 2 Passes)



2 Sorted Elements

Bubble Sort using Swap Flag

```
DECLARE swaps INITIALLY TRUE
DECLARE outer INITIALLY length-2
WHILE swaps AND outer>=0 DO
        SET swaps TO FALSE
        FOR inner FROM 0 TO outer DO
                IF list[inner]> list[inner+1] THEN
                        SET temp = list[inner+1]
                        list[inner+1] = list[inner]
                        list[inner] = temp
                        SET swaps TO TRUE
                END IF
        END FOR
        SET outer TO outer-1
END WHILE
```

- In Python length would return the number of elements in an array but we need to remember that the largest element index will be length -1
- We use length -2 as we are swapping an element and the element immediately to the right of it

Bubble Sort using Swap Flag

```
swaps = True
outer = len(listA)-1
while (swaps == True):
    swaps = False
    for inner in range(outer):
        if listA[inner]>listA[inner+1]:
            swaps = True
            temp = listA[inner]
            listA[inner] = listA[inner+1]
            listA[inner+1] = temp
    outer -= 1
```

- The swaps flag will be set to False at the start of every pass through the list
- If a swap is made it is set and this will mean that another pass has to happen through the list