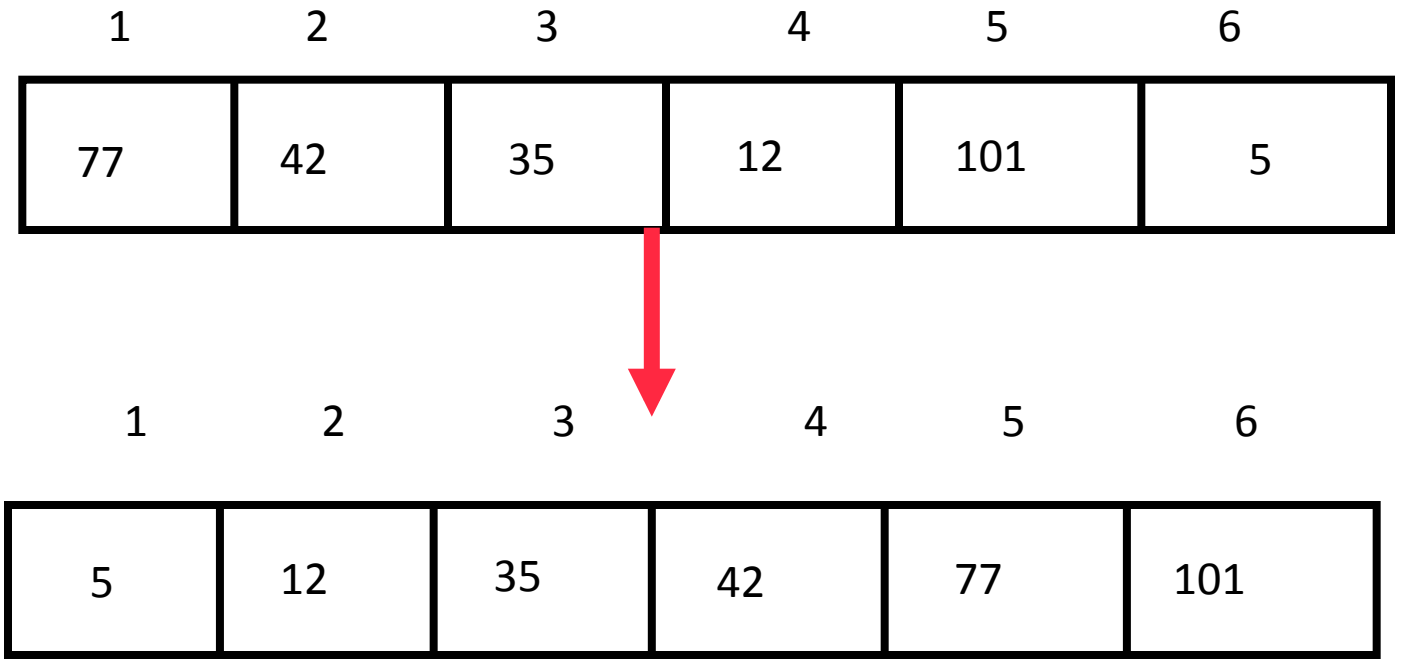
Sorting

T#12

Sorting

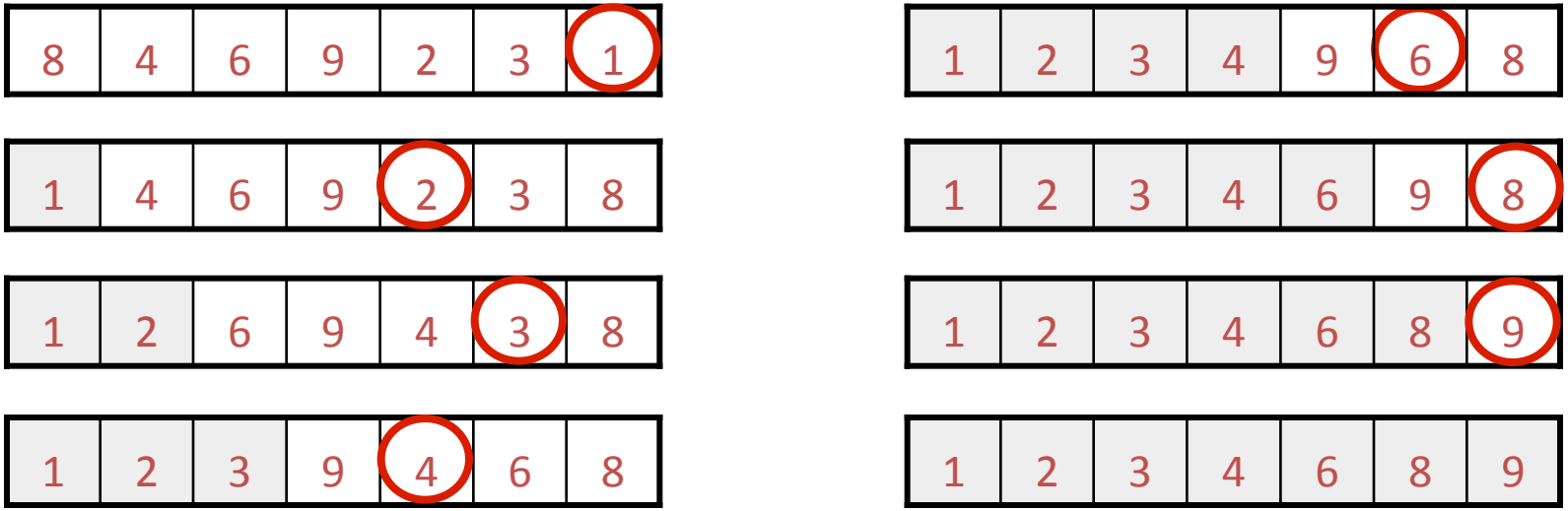
- **Sorting takes an unordered collection and makes it an ordered one.**



Selection Sort

- Idea:
  - Find the smallest element in the array
  - Exchange it with the element in the first position
  - Find the second smallest element and exchange it with the element in the second position
  - Continue until the array is sorted
- Disadvantage:
  - Running time depends only slightly on the amount of order in the file

Example


8 4 6 9 2 3 1
1 4 6 9 2 3 8
1 2 6 9 4 3 8
1 2 3 9 4 6 8
1 2 3 4 9 6 8
1 2 3 4 6 9 8
1 2 3 4 6 8 9
1 2 3 4 6 8 9

Selection Sort

```
void Selection Sort(int[] A)
  for i from 0 to n − 2 do
    min = i
    for j from i + 1 to n − 1 do
      if A[j] < A[min] then
        min = j
      end if
    end for
    {Swap A[i] and A[min]}
    tmp = A[i]
    A[i] = A[min]
    A[min] = tmp
  end for
```

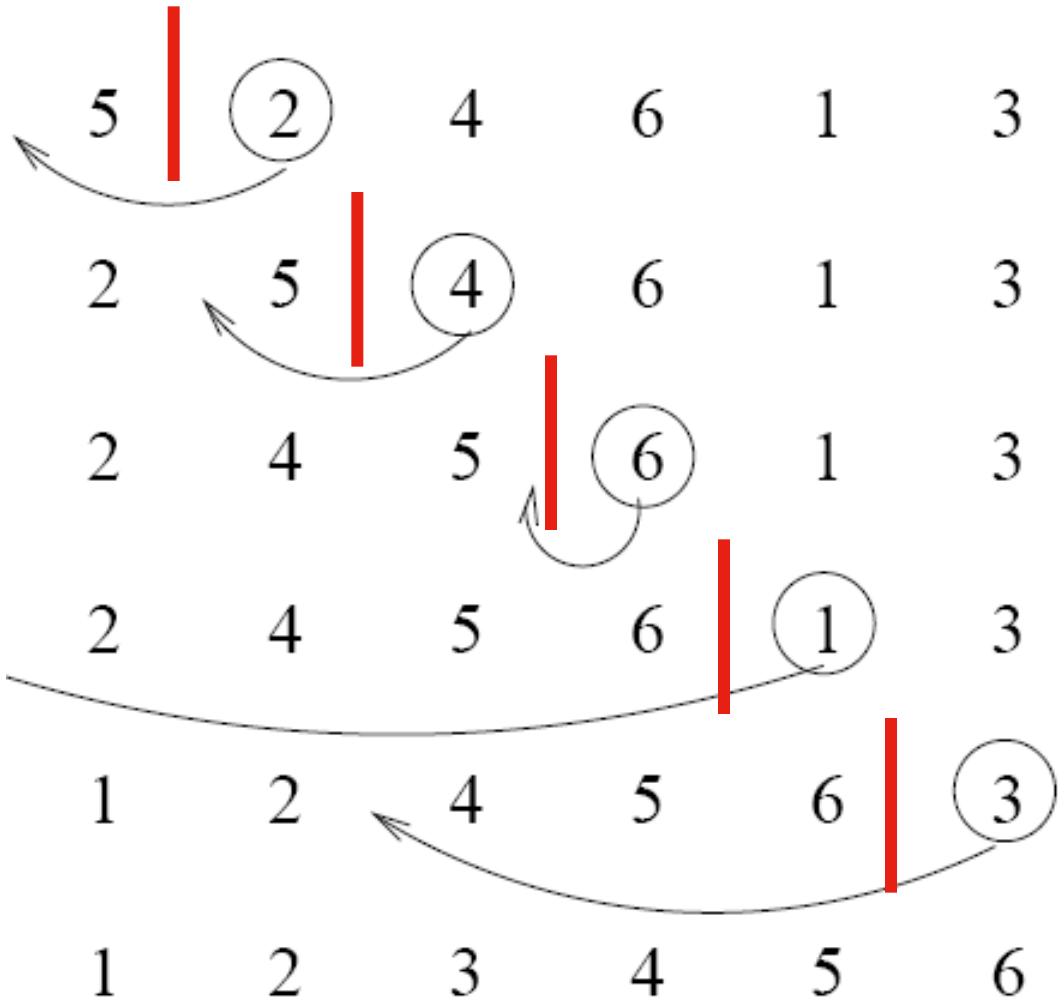
Insertion Sort

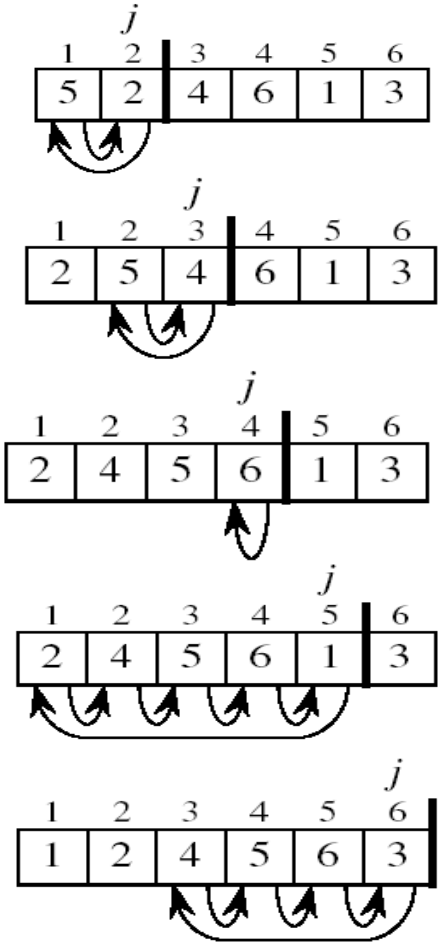
- Idea: like sorting a hand of playing cards
  - Start with an empty left hand and the cards facing down on the table.
  - Remove one card at a time from the table, and insert it into the correct position in the left hand
    - compare it with each of the cards already in the hand, from right to left
  - The cards held in the left hand are sorted
    - these cards were originally the top cards of the pile on the table

Insertion Sort

```
void Insertion Sort(int [] A)
  for i from 1 to n − 1 do
    j = i
    while ((j > 0) and (A[j] < A[j − 1])) do
      {Swap A[j − 1] and A[j]}
      tmp = A[j]
      A[j] = A[j − 1]
      A[j − 1] = tmp
      j = j − 1
    end while
  end for
```

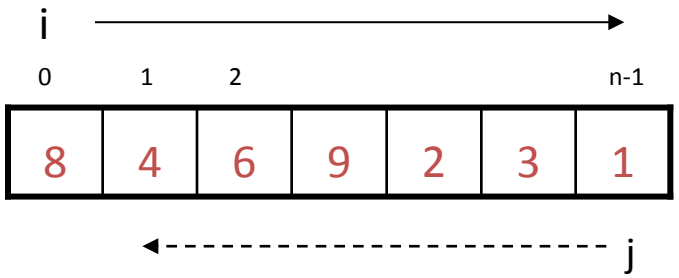
Insertion Sort





Bubble Sort

- Idea:
  - Repeatedly pass through the array
  - Swaps adjacent elements that are out of order



Bubble Sort

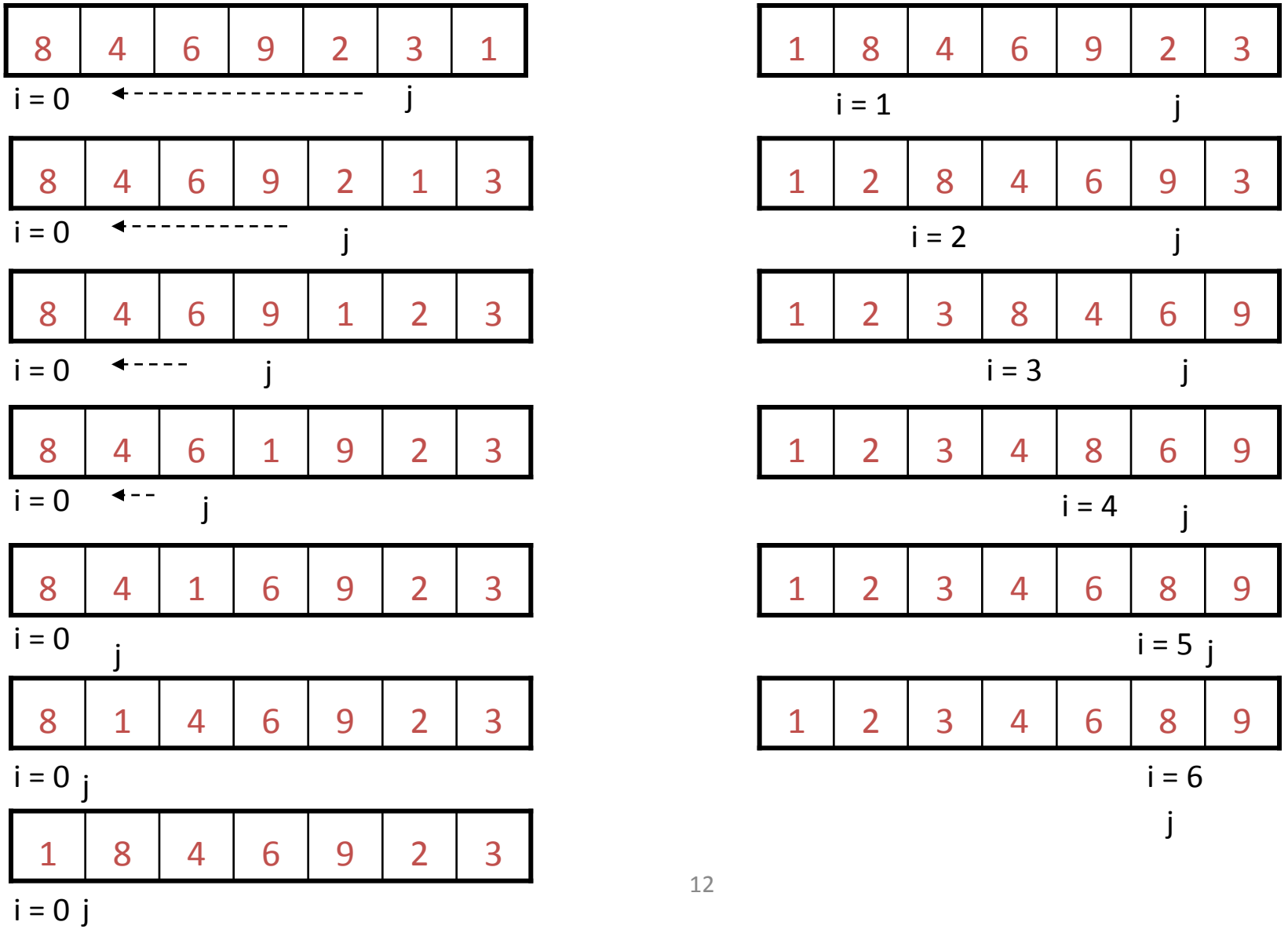
```
void Bubble Sort(int [] A)
  for i from 0 to n − 1 do
    for j from n − 2 down to i do
      if A[j] > A[j + 1] then
        {Swap A[j] and A[j + 1]}
        tmp = A[j]
        A[j] = A[j + 1]
        A[j + 1] = tmp
      end if
    end for
  end for
```

```
void Bubble Sort(int [] A)
  for i from 0 to n − 1 do
    for j from n − 2 down to i do
      if A[j] > A[j + 1] then
        {Swap A[j] and A[j + 1]}
        tmp = A[j]
        A[j] = A[j + 1]
        A[j + 1] = tmp
      end if
    end for
  end for
```

$$T(n) = c_1(n+1) + c_2 \sum_{i=0}^{n-1}(n-i) + c_3\sum_{i=0}^{n-1}(n-i-1) + c_4 \sum_{i=0}^{n-1}(n-i-1)$$

Thus, $T(n) = \Theta(n^2)$

Example


8 4 6 9 2 3 1
i = 0
j
8 4 6 9 2 1 3
i = 0
j
8 4 6 9 1 2 3
i = 0
j
8 4 6 1 9 2 3
i = 0
j
8 4 1 6 9 2 3
i = 0
j
8 1 4 6 9 2 3
i = 0 j
1 8 4 6 9 2 3
i = 0 j
1 8 4 6 9 2 3
i = 1
j
1 2 8 4 6 9 3
i = 2
j
1 2 3 8 4 6 9
i = 3
j
1 2 3 4 8 6 9
i = 4
j
1 2 3 4 6 8 9
i = 5 j
1 2 3 4 6 8 9
i = 6
j

Repeat "Bubble Up" How Many Times?

- **If we have N elements…**
- **And if each time we bubble an element, we place it in its correct location…**
- **Then we repeat the "bubble up" process N –1 times.**
- **This guarantees we' ll correctly place all N elements.**

Compare these two bubble sort solutions

A

```
void Bubble Sort(int [] A)
  for i from 0 to n − 1 do
    for j from n − 2 down to i do
      if A[j] > A[j + 1] then
        {Swap A[j] and A[j + 1]}
        tmp = A[j]
        A[j] = A[j + 1]
        A[j + 1] = tmp
      end if
    end for
  end for
```

B

```
exchanges = true;
while (exchanges) {
  exchanges = false;
  i = 0;
  while (i < A.length-1) {
    if (A[i] > A[i+1]) {
      Swap A[i] and A[i+1]
      exchanges = true;
    }
    i = i+1;
  }
}
```

Reducing the Number of Comparisons

- **On the N^{th} iteration, we only need to do MAX-N comparisons.**

Already Sorted Collections?

- **What if the collection was already sorted?**
- **What if only a few elements were out of place and after a couple of "bubble ups," the collection was sorted?**

- **We want to be able to detect this and "stop early"!**

Using a Boolean "Flag"

- **We can use a boolean variable to determine if any swapping occurred during the inner loop.**
- **If no swapping occurred, then we know that the collection is already sorted!**
- **This boolean "flag" needs to be reset after each inner loop.**

Bubble Sort Loop Invariant (inner Loop)

```
void Bubble Sort(int [] A)
  for i from 0 to n − 1 do
    for j from n − 2 down to i do
      if A[j] > A[j + 1] then
        {Swap A[j] and A[j + 1]}
        tmp = A[j]
        A[j] = A[j + 1]
        A[j + 1] = tmp
      end if
    end for
  end for
```

Each time we put the smallest elements at the beginning, and once we put them there, we don't move them again

- The variable i starts at the first index in the array and increases to n-1
- Our invariant is: Every element to the left of i is in the correct place
- That is, for all h < i, if h < k, then a[h] <= a[k]
- When this is combined with the loop exit test, i == n-1, we know that *all* elements of the array are in the correct place

Animations

- http://www.sorting-algorithms.com
- https://www.youtube.com/watch?v=ZZuD6iUe3Pc

- Some of the slides from:
- http://www.cse.unr.edu/~bebis/CS477/Lect/InsertionSortBubbleSortSelectionSort.ppt