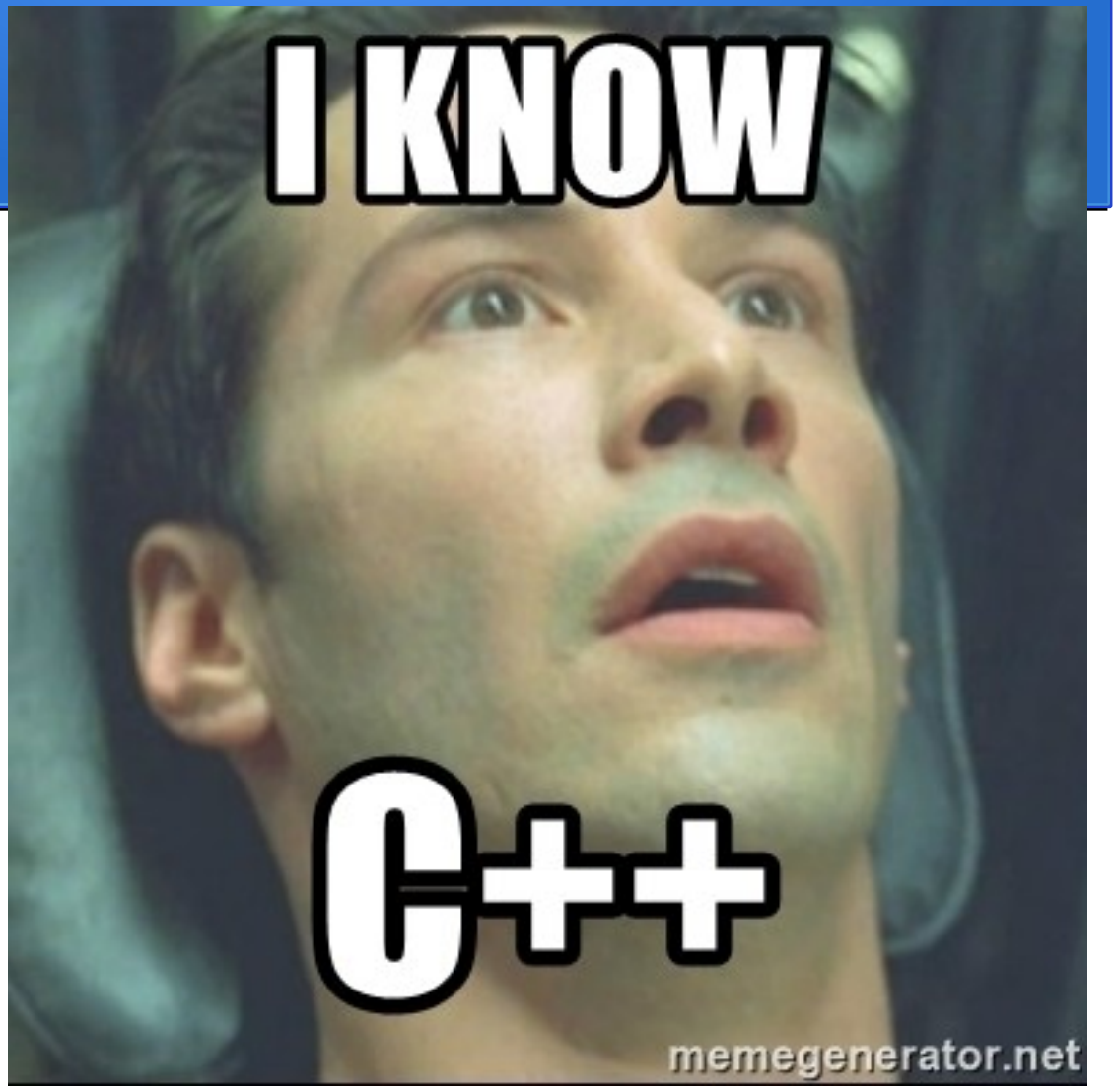

I KNOW
C++
memegenerator.net

CS2308
Gentry Atkinson

Lecture 2.3
Sorting

What is Sorting?

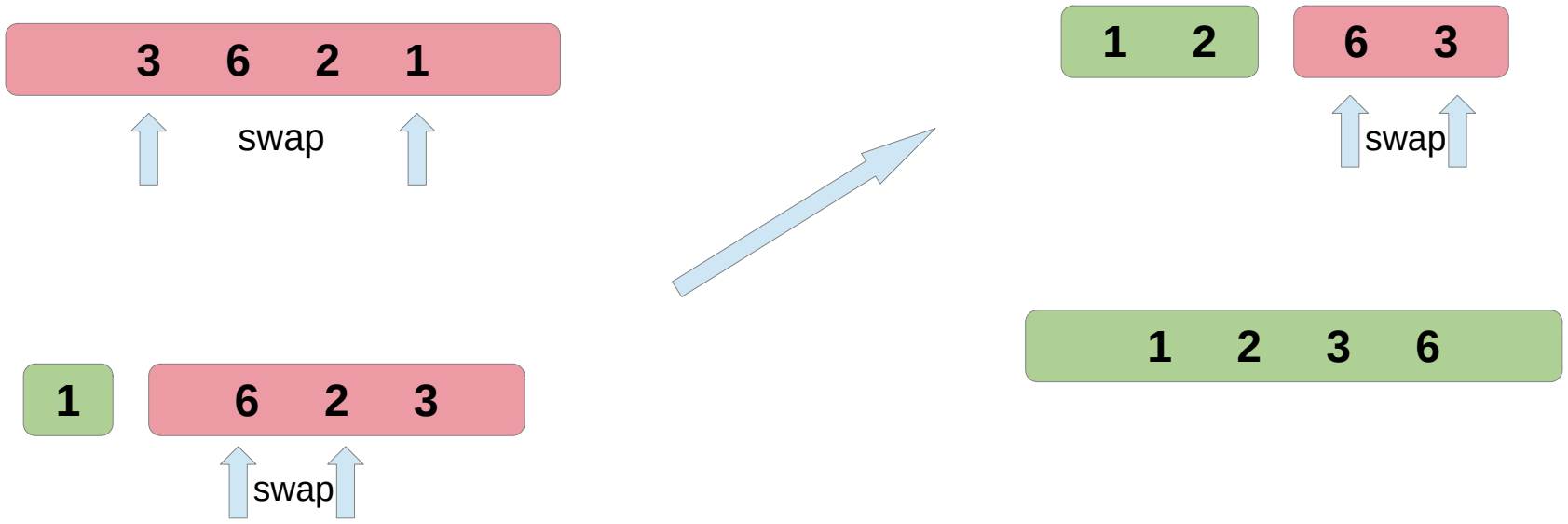
- Putting a list of values into an ordered sequence is one of the most fundamental problems of data processing.
- To be able to order values, we have to have some way of deciding which order they should be in.
- Examples of ordered values:
  - Numerical: 1 is before 2…
  - Alphabetical: a is before b…
  - Chronological: Jan 1 is before Jan 2…

3 Basic Strategies

1) For each position in the list, find the right value to put there.
2) For each value in the list, find the right place to put it.
3) For each pair of values in the list, swap them if they are in the wrong order.

Selection Sort

- Repeatedly find the smallest value in the unsorted portion of the list, and move it into the lower position in the list.



Selection Sort Pseudo Code

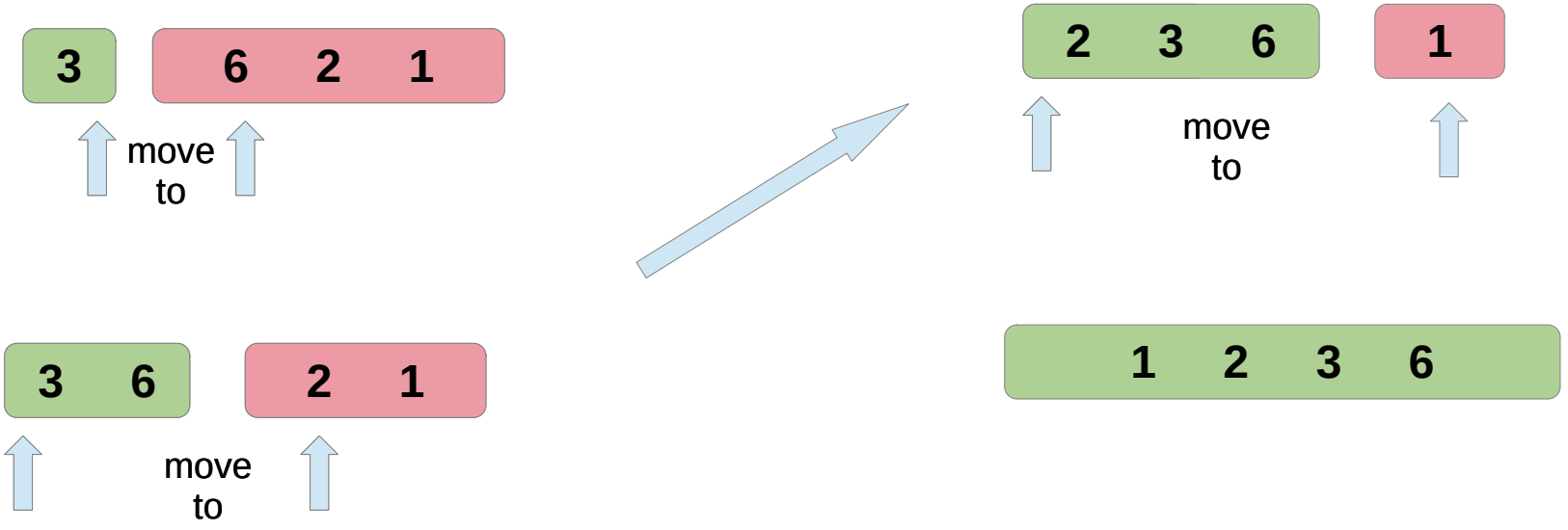
- Given an unsorted list **L** of size **n**.
- For integer **i** from 0 to **n-1**:
  - Set **min_position** to **i**
  - For integer **j** from **i+1** to **n-1**:
    - If **Lj** < **Li**, set **min_position** to **j**
  - Swap **Li** and **Lmin_position**
- Output the sorted list **L**

Efficiency of Selection Sort

- The outer loop will run **n-1** times.
- The inner loop will run **n-2** times when when **i=0** and **1** time when **i=n-1**.
- The average number of iterations for the inner loop is:
  - ((n-2)+ 1)/2 = 0.5n-0.5
- We can calculate the total number of runs for the inner loop as (n-1)(0.5n-0.5) = $0.5n^2$-n+0.5
- Dropping constants and keeping only the highest order operation we see the Selection sort is **O(n^2)**

Insertion Sort

- Repeatedly move values to an earlier position in the list until they are in the correct position.



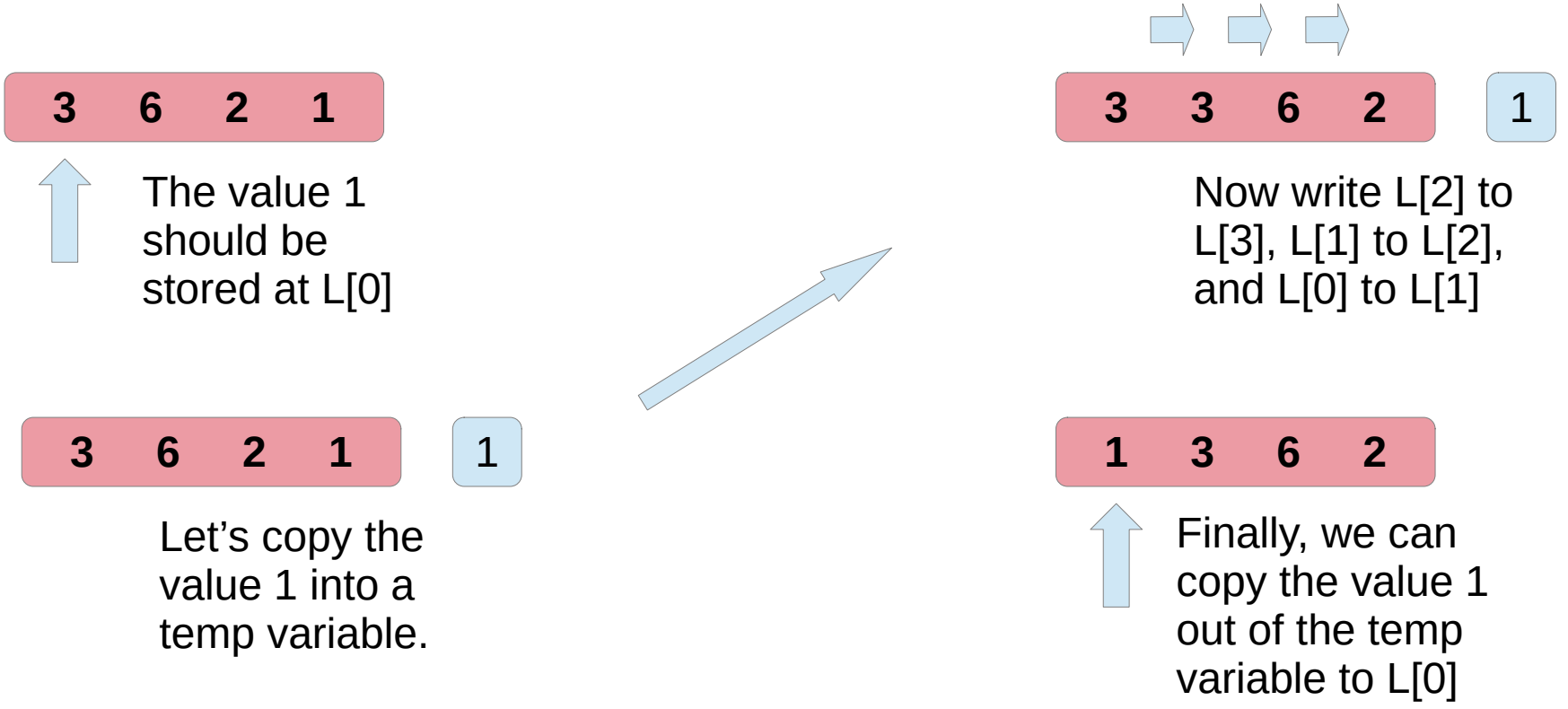
Insertion Sort Pseudo Code

- Given a list **L** of length **n**.
- For integer **i** with values 1 to **n-1**:
  - Set **j** to **i**
  - While **j>0** and **Lj > Li**:
    - Subtract **1** from **j**
  - Move **Li** to position **j** in **L**
- Output the sorted list **L**

Complexity of Insertion Sort

- The outer loop will run n-1.
- The inner loop will run 0 to n-1 times for every one iteration of the outer loop.
- This tells us that the average case is **$O(n^2)$**, just like Selection sort.
- What is the complexity of Insertion sort if it is given a pre-sorted list?

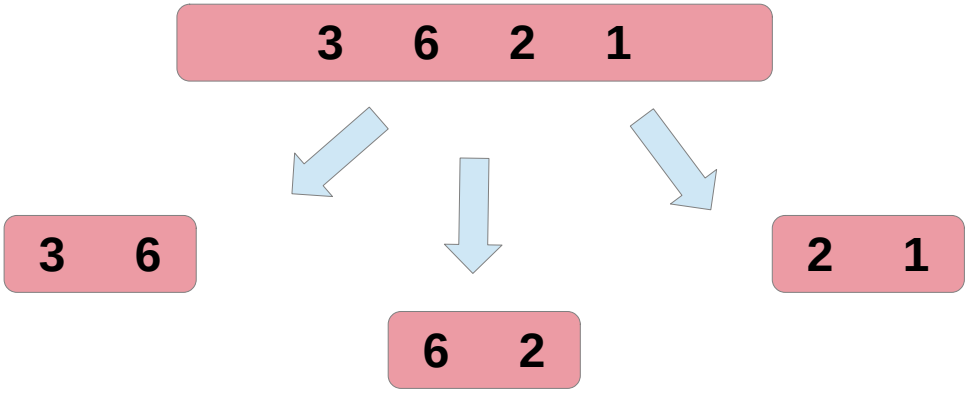
What is the Cost of Moving an Array Element?



We had to write 5 values to perform 1 insertion!

Bubble Sort

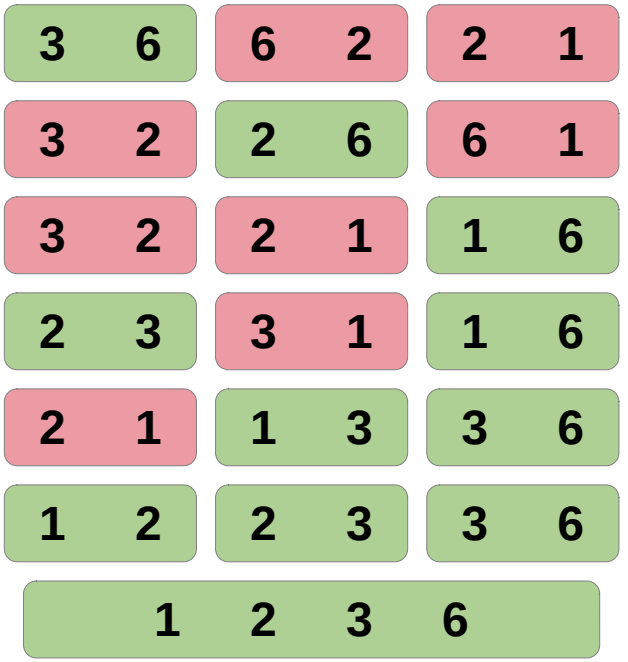
- Let’s leverage the intuition that every list can be seen as several 2-element lists...



Sorting the three 2-element lists will sort the whole list.

Bubble Sort Pseudo Code

- Given a list **L** of length **n**
- For integer **i** from **n-1** down to 1:
  - For integer **j** from 0 to **i-1**:
    - If **Lj** > **Li**, swap the elements stored at positions **i** and **j**.
- Output the sorted list **L**



Complexity of Bubble Sort

- The algorithm will perform n passes over the unsorted list.
- Each pass will perform n-1 comparisons, and as many as n-1 swaps.
- Bubble sort is **$O(n^2)$**

Improving Bubble Sort

- Consider Bubble Sort running on a list that is already sorted...
- The algorithm we wrote will still make n passes over the list, and each pass will make up to n-1 comparisons.
- What if the algorithm could quit as soon as the list was sorted.

Improved Bubble Sort

- Given a list **L** of length **n**
- For integer **i** from **n-1** down to 1:
  - For integer **j** from 0 to **i-1**:
    - Set **done** to true
    - If **Lj** > **Li**, swap the elements stored at positions **i** and **j** and set **done** to false.
  - If **done** is true, break the outer for loop.
- Output the sorted list **L**

New Complexity of Bubble Sort

- On an unsorted list:
  - The algorithm will do n pass.
  - Each pass will do up to n-1 comparisons and swaps.
  - The algorithm is still **$O(n^2)$**
- On a sorted list:
  - The algorithm will only do one pass.
  - That pass will do up to n-1 comparisons and swaps
  - The performance is now **O(n)**.
- On a *nearly* sorted list (e.g. 2 or 3 elements out of place), Bubble Sort is still **O(n)**. Why?

Bubble Sort is the Best Sorting Algorithm.

- Do not tell this to other computer scientists. This is a secret shared only in this classroom.
- Many lists in the real world are nearly sorted to begin with, meaning that Bubble Sort will perform very quickly on them.
- Bubble Sort shows us that breaking a list into several smaller lists makes sorting easier. Remember that in CS3358!

Questions or Comments?