CSIT113
Problem Solving

Workshop - Week 9

Sorting Practice

- Consider the following list of numbers:

9	11	8	9	13	18	3	10

- Sort it using:
  1. Selection sort
  2. Insertion sort
  3. Bubble sort
- Count the number of operations each time.
- Which is the best sort?

Selection Sort.

- Selection sort uses the following strategy:
  1. Start with the whole list.
  2. Find the smallest element in the list.
  3. Swap the first element with the smallest.
  4. Shorten the list by ignoring its first element (because it is in the right place).
  5. If the list has only one element, stop.
  6. Otherwise, go to step 1. with the shorter list.

- Start with the first element

- Find the smallest element in the list: 7 comparisons

- Swap with the first element: 1 swap

- Shorten the list by one. Finish 1 cycle

- Start with the first element

- Find the smallest element: 6 comparisons

- Swap: 1 swap

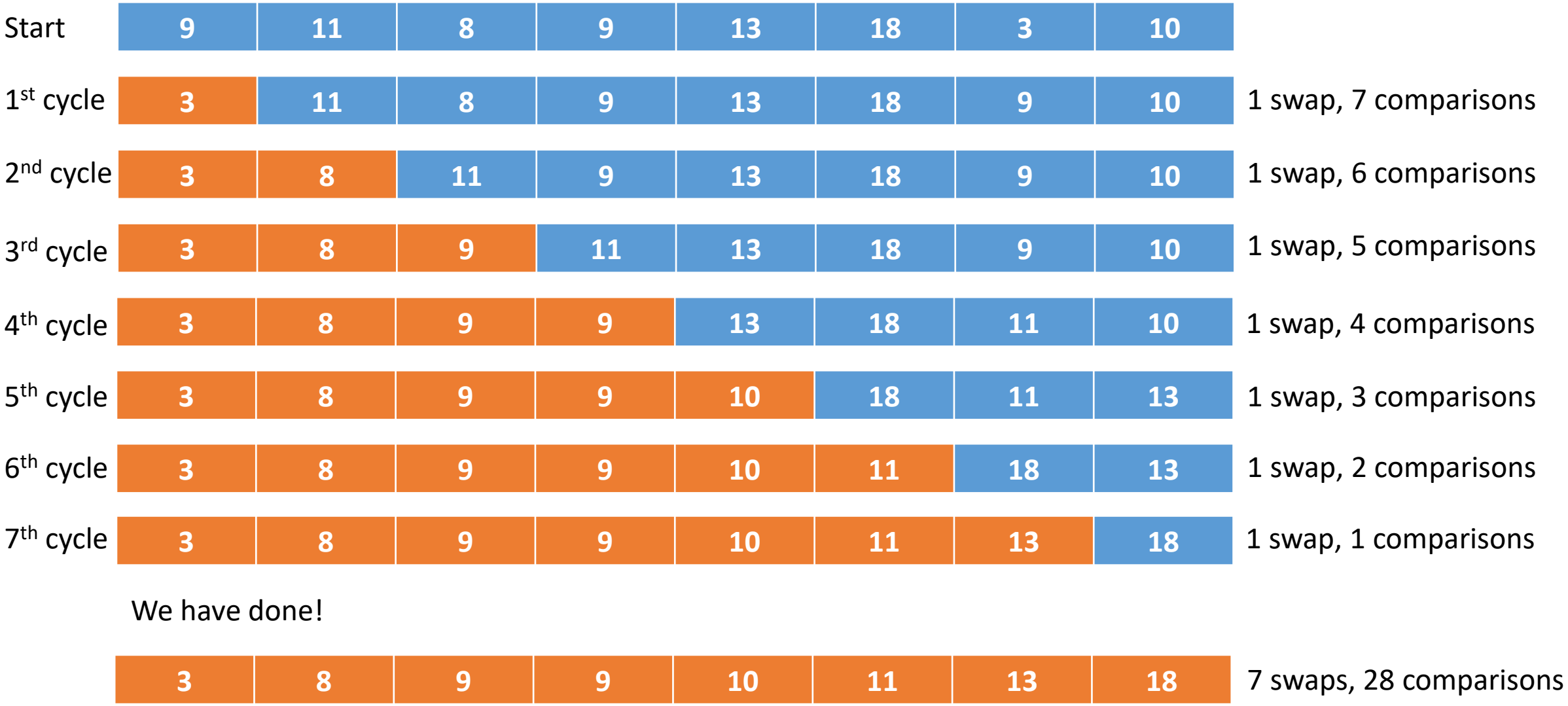
- Shorten list by one, finish 2 cycle

- Start with first element

- Find the smallest element: 5 comparisons

- Swap with first element: 1 swap

- Shorten the list by one: finish 3 cycle


Start
9 11 8 9 13 18 3 10
1st cycle
3 11 8 9 13 18 9 10
1 swap, 7 comparisons
2nd cycle
3 8 11 9 13 18 9 10
1 swap, 6 comparisons
3rd cycle
3 8 9 11 13 18 9 10
1 swap, 5 comparisons
4th cycle
3 8 9 9 13 18 11 10
1 swap, 4 comparisons
5th cycle
3 8 9 9 10 18 11 13
1 swap, 3 comparisons
6th cycle
3 8 9 9 10 11 18 13
1 swap, 2 comparisons
7th cycle
3 8 9 9 10 11 13 18
1 swap, 1 comparisons
We have done!
3 8 9 9 10 11 13 18
7 swaps, 28 comparisons

Insertion Sort.

- Insertion sort uses the following strategy:
  1. Start with the second element in the list.
  2. Insert it in the right place in the preceding list.
  3. Repeat with the next unsorted element.
  4. Keep going until we have placed the last element in the list.

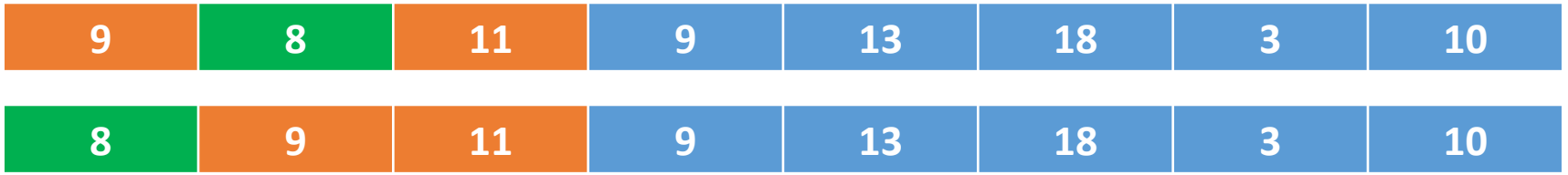
- Start with the second element

- Compare to the preceding list (only one element 9): 1 comparison
- It is in the right position. So do nothing!

- Finish 1st cycle

- Start with the next element

- Compare to the preceding list: 2 comparisons
- Insert into the right position: 2 swaps

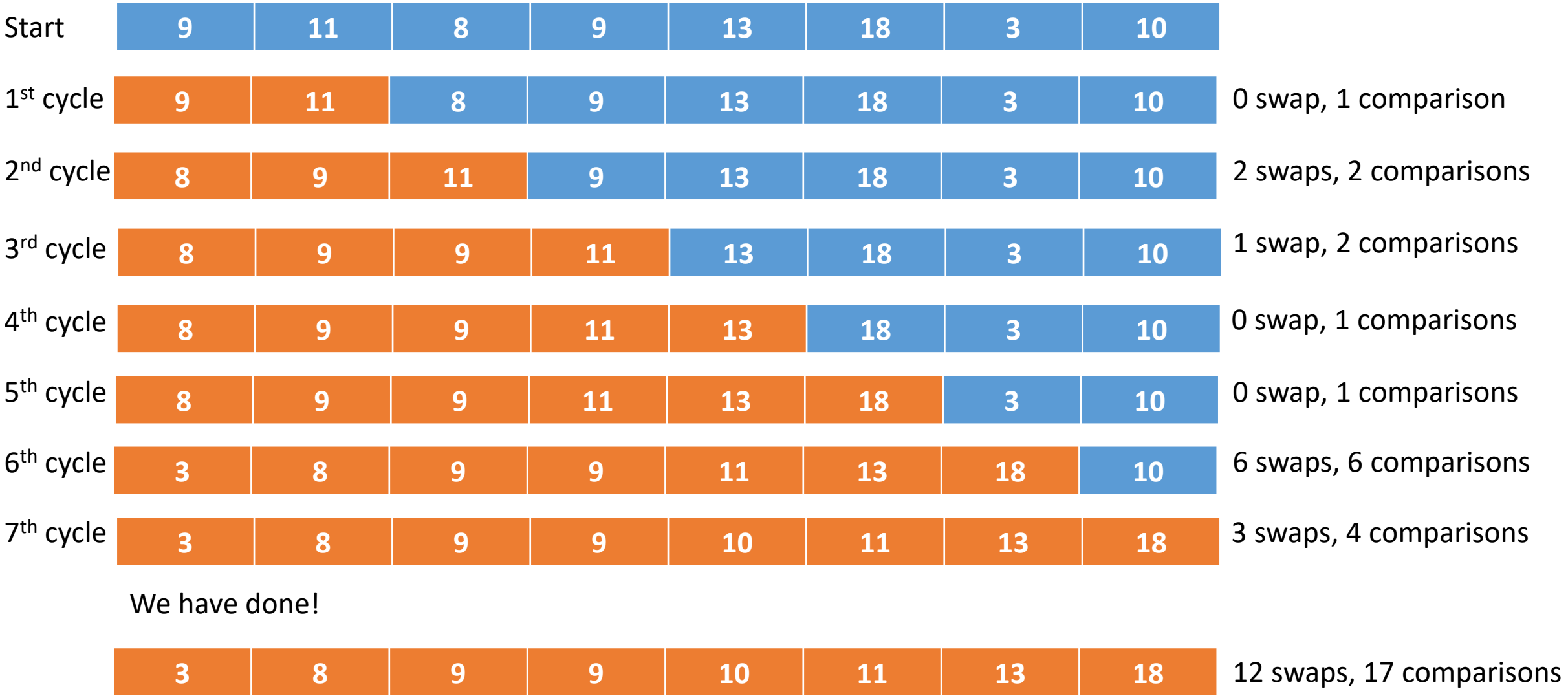


- Finish 2st cycle

- Start with the next element

- Compare to the preceding list: 2 comparisons
- Insert into the right position: 1 swap

- Finish 3rd cycle


Start
9 11 8 9 13 18 3 10
1st cycle
9 11 8 9 13 18 3 10
0 swap, 1 comparison
2nd cycle
8 9 11 9 13 18 3 10
2 swaps, 2 comparisons
3rd cycle
8 9 9 11 13 18 3 10
1 swap, 2 comparisons
4th cycle
8 9 9 11 13 18 3 10
0 swap, 1 comparisons
5th cycle
8 9 9 11 13 18 3 10
0 swap, 1 comparisons
6th cycle
3 8 9 9 11 13 18 10
6 swaps, 6 comparisons
7th cycle
3 8 9 9 10 11 13 18
3 swaps, 4 comparisons
We have done!
3 8 9 9 10 11 13 18
12 swaps, 17 comparisons

Bubble Sort.

- Bubble sort uses the following strategy

1. Compare the first element with the next one.
2. If they are in the wrong order swap them.
3. Continue comparing until the end of the list.
4. Shorten the list by one – the last element is now in the right place.
5. Repeat from step 1.

- Compare the first element with second element: 1 comparison

- They are in the right position. So do nothing!
- Compare the second with the third elements: 1 comparison

- They are not in the right position, swap them: 1 swap

- Compare the 3rd and 4th elements: 1 comparison

- They are not in the right position, swap them: 1 swap

- Compare the 4th and 5th elements: 1 comparison

- They are in the right position. So do nothing!
- Compare the 5th and 6th elements: 1 comparison

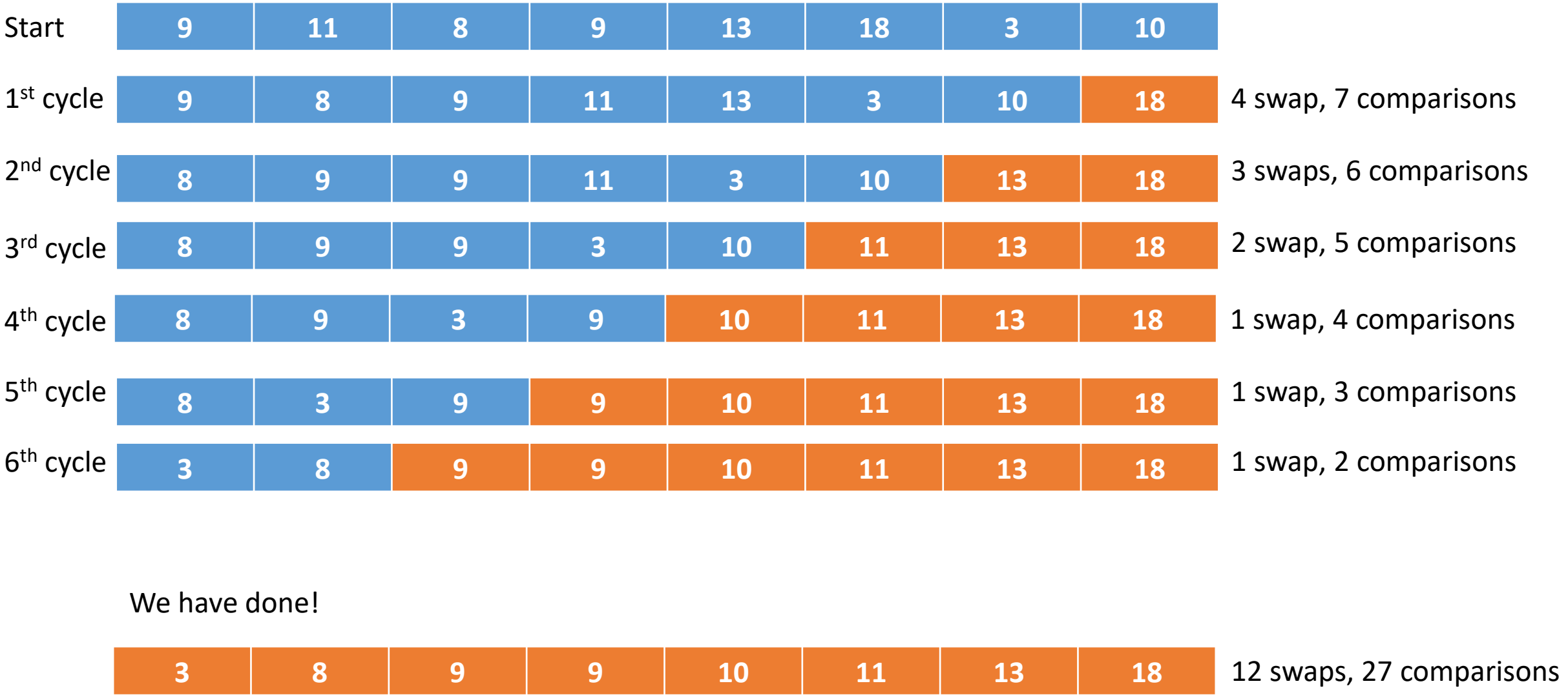
- They are in the right position, do nothing!
- Compare the 6th and 7th elements: 1 comparison

- They are not in the right position, swap them: 1 swap

- Compare the 7th and 8th elements: 1 comparison

- They are not in the right position, swap them: 1 swap

- Complete the 1st cycle: 4 swaps, 7 comparisons


Start
9 11 8 9 13 18 3 10
1st cycle
9 8 9 11 13 3 10 18
4 swap, 7 comparisons
2nd cycle
8 9 9 11 3 10 13 18
3 swaps, 6 comparisons
3rd cycle
8 9 9 3 10 11 13 18
2 swap, 5 comparisons
4th cycle
8 9 3 9 10 11 13 18
1 swap, 4 comparisons
5th cycle
8 3 9 9 10 11 13 18
1 swap, 3 comparisons
6th cycle
3 8 9 9 10 11 13 18
1 swap, 2 comparisons
We have done!
3 8 9 9 10 11 13 18
12 swaps, 27 comparisons

Note

- Well, I have cheated since I finish when I see the list is sorted before the algorithm terminates.
- In this lecture, you just finish when you see the list is sorted.
- But in the real work (assignments), you need to complete all steps in the algorithms to make sure your list is sorted.

Summary

- Start

9	11	8	9	13	18	3	10

- Finish

3	8	9	9	10	11	13	18

	Number of Swaps	Number of Comparisons	Total number of operations
Selection sort	7	28	35
Insertion sort	12	17	29
Bubble sort	12	27	39

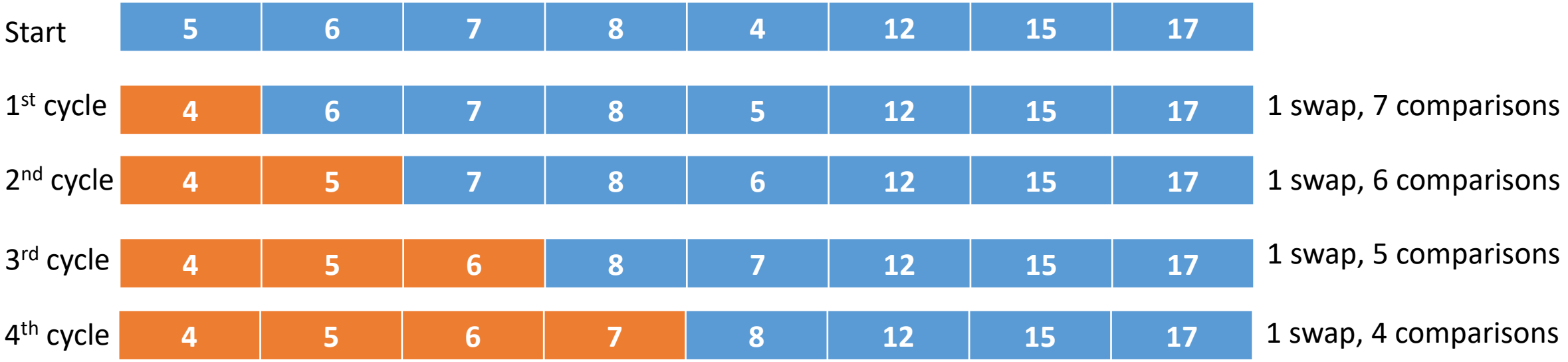
Sorting Practice

- Consider the following list of numbers:

5	6	7	8	4	12	15	17

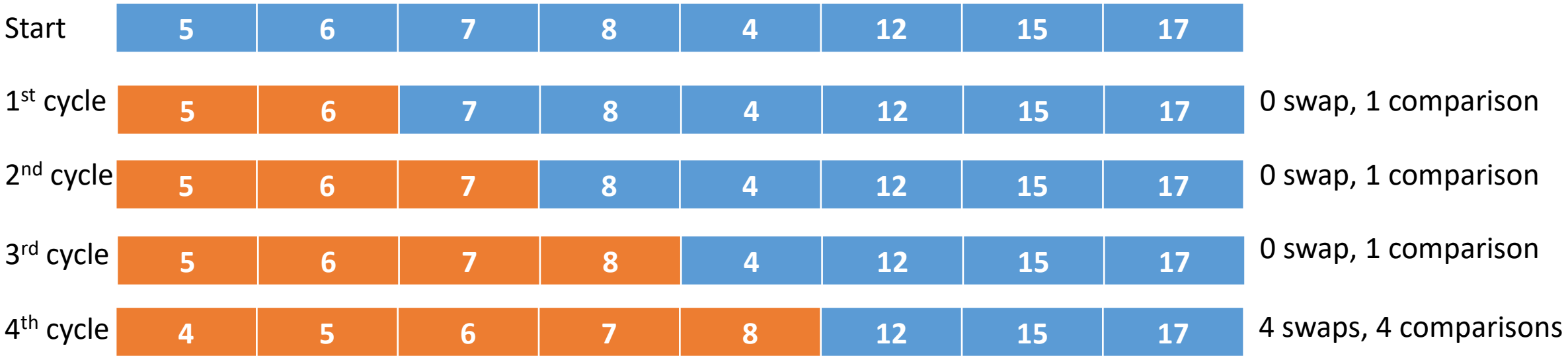
- Sort it using:
  1. Selection sort
  2. Insertion sort
  3. Bubble sort
- Count the number of operations each time.
- Does this change the best sort?

Selection sort



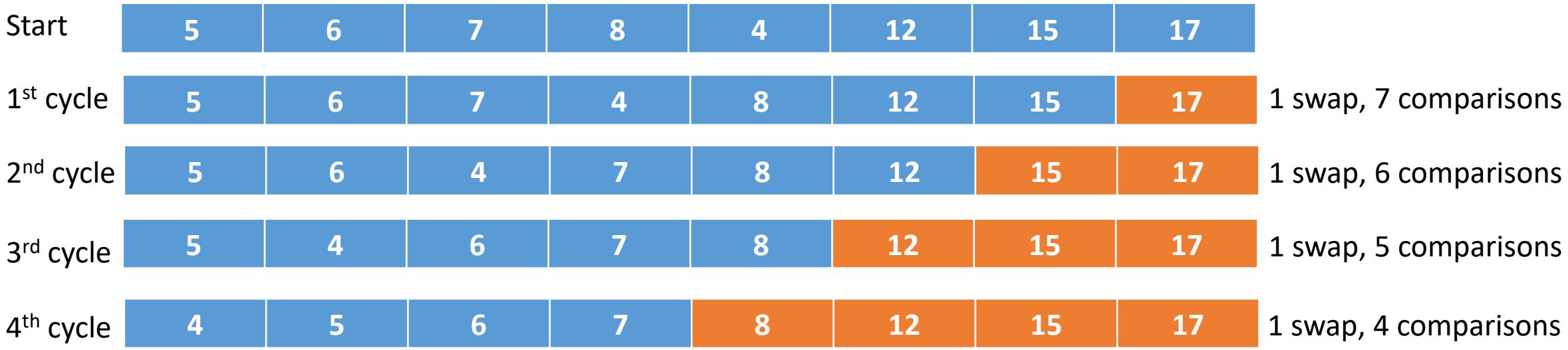
We have done!

Insertion sort



We have done!

Bubble sort



We have done!

Summary

- Start

5	6	7	8	4	12	15	17

- Finish

4	5	6	7	8	12	15	17

	Number of Swaps	Number of Comparisons	Total number of operations
Selection sort	4	22	26
Insertion sort	4	7	11
Bubble sort	4	22	26

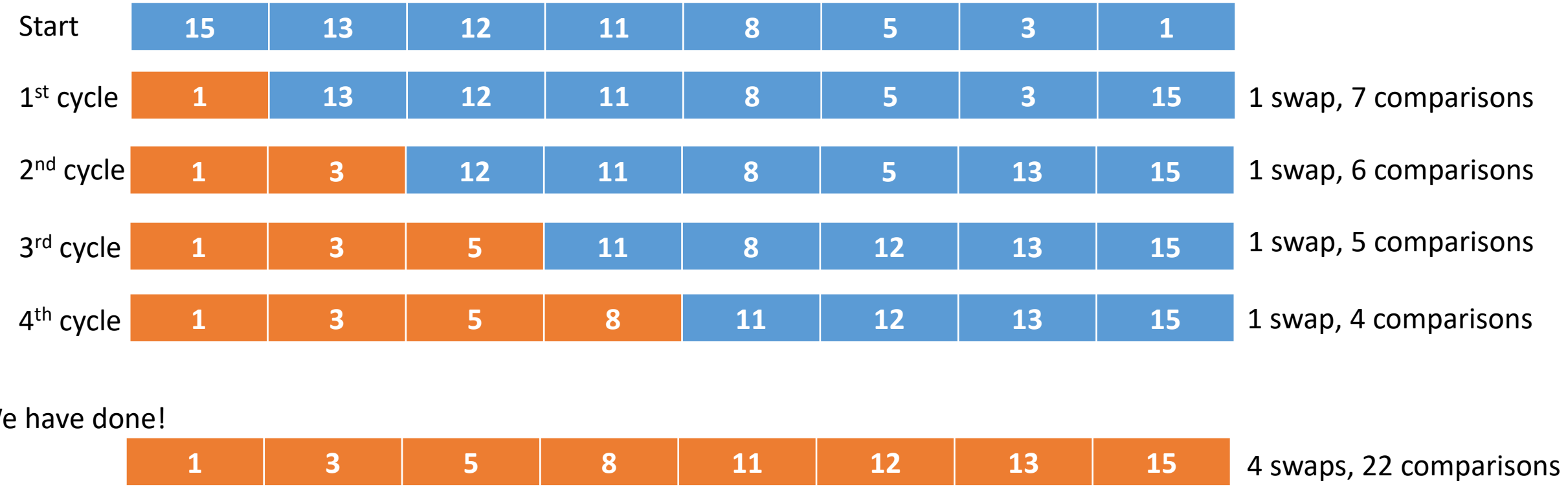
Sorting Practice

- Consider the following list of numbers:

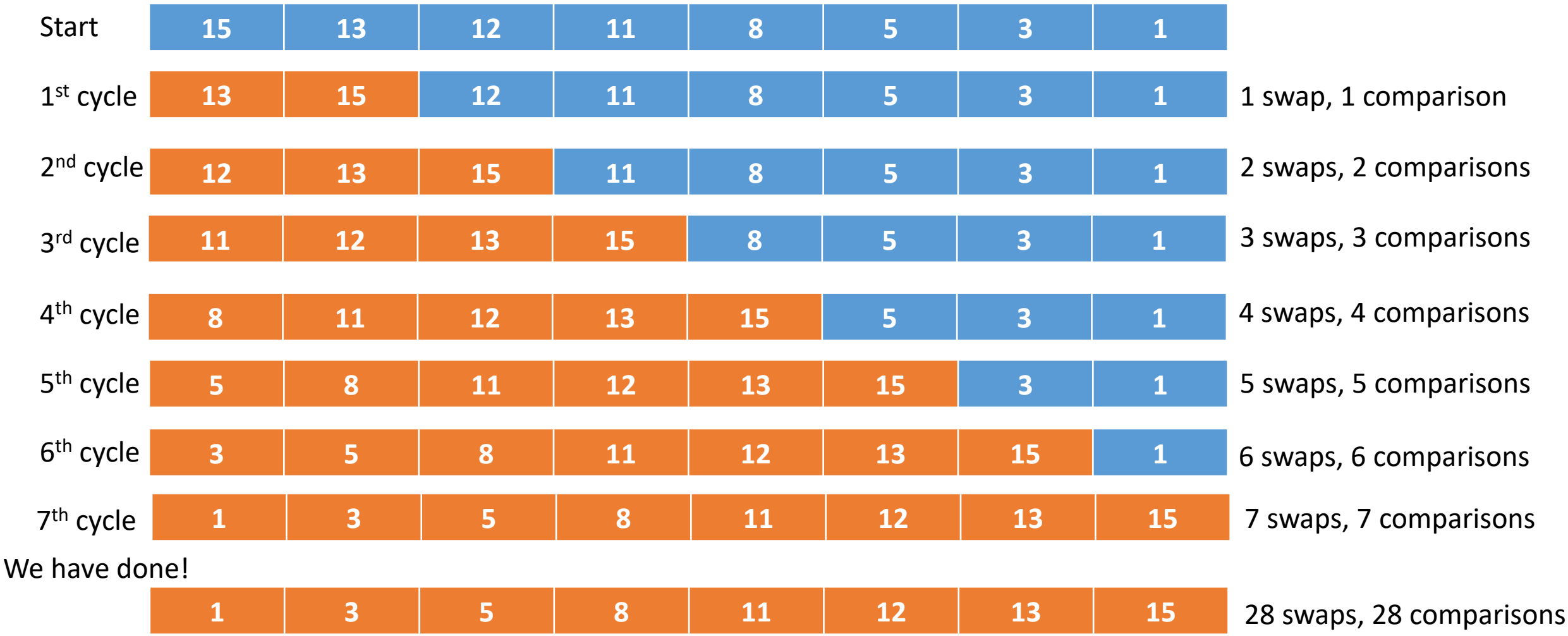
15	13	12	11	8	5	3	1

- Sort it using:
  1. Selection sort
  2. Insertion sort
  3. Bubble sort
- Count the number of operations each time.
- Does this change the best sort?

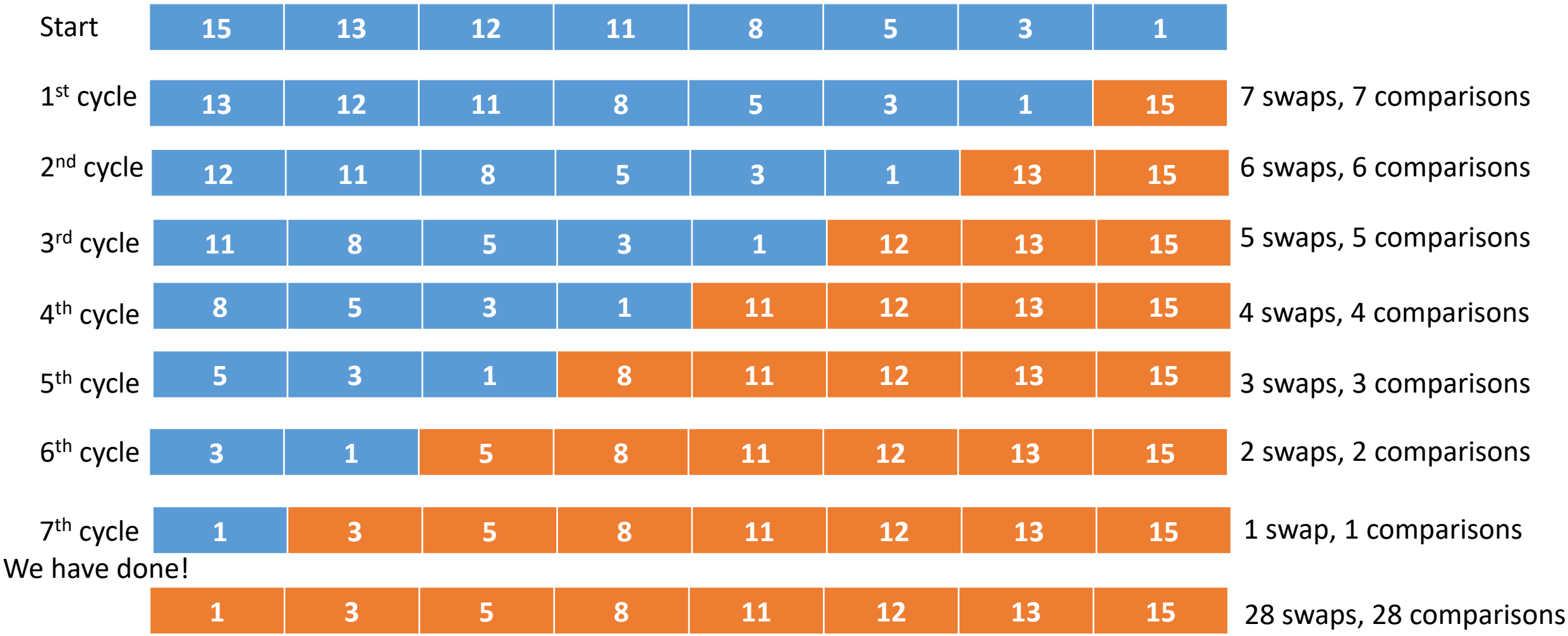
Selection sort



Insertion sort



Bubble sort


Start
15 13 12 11 8 5 3 1
1st cycle
13 12 11 8 5 3 1 15
7 swaps, 7 comparisons
2nd cycle
12 11 8 5 3 1 13 15
6 swaps, 6 comparisons
3rd cycle
11 8 5 3 1 12 13 15
5 swaps, 5 comparisons
4th cycle
8 5 3 1 11 12 13 15
4 swaps, 4 comparisons
5th cycle
5 3 1 8 11 12 13 15
3 swaps, 3 comparisons
6th cycle
3 1 5 8 11 12 13 15
2 swaps, 2 comparisons
7th cycle
1 3 5 8 11 12 13 15
1 swap, 1 comparisons
We have done!
1 3 5 8 11 12 13 15
28 swaps, 28 comparisons

Summary

- Start

15	13	12	11	8	5	3	1

- Finish

1	3	5	8	11	12	13	15

	Number of Swaps	Number of Comparisons	Total number of operations
Selection sort	4	22	26
Insertion sort	28	28	56
Bubble sort	28	28	56

Ball breaking.

- We have two bowling balls and a multi storey building.
- We know that if we drop a ball from a sufficiently high window it will break.
- Let us define a *test* as the process of dropping a ball from a specific floor.
- Our aim is to find out which is the lowest floor from which dropping a ball breaks it. Let’s call this the *critical* floor.
- We also want to do this with the smallest number of tests.

The questions.

- What is the maximum number of tests that I need to conduct to find the critical floor in a 100 floor building?
- What order should I test the floors in?
- What is the highest floor you can reach with *n* tests?

A simpler case.

- Before thinking about the problem with two balls start with a single ball.
- What strategy makes sure that you exactly identify the critical floor?
- If I am allowed to conduct *n* tests what is the highest floor I can test and still guarantee that I find the critical floor.
- If my building is 100 floors high what is the worst case value for the number of tests?

- Drop the ball from each floor in turn: 1, 2, 3 … until the ball breaks.
- If the critical floor is floor n you conduct n tests to find it.
- A 100 floor building will need at most 100 tests.

The real problem.

- Now that you have the answers for a single ball think about how you can make use of a second ball to increase the floor coverage for a given number of floors.
- We still have to exactly identify the critical floor.
- If we are allowed to perform at most *n* tests, what is the highest floor we can test and still be sure of finding the critical floor?
- Hint: there are two cases to consider:
  - Before the first ball breaks;
  - After the first ball breaks.

The strategy to follow is as follows:

- Test floor n with ball 1 (1 test)
- if the ball breaks test floors 1 up to n – 1 with the second ball (n – 1 tests for a total of n)
- if the ball survives we can use it to test floor $2n$ -1 (this equals n + (n – 1) (2 tests)
- if it breaks we test floors n + 1 up to $2n$ -2 with the second ball (n -2 tests for a total of n)
- if it survives use ball 1 to test floor $3n$ -3 (3 tests so far)
- If we repeat this process, reducing the added number of floors by one each time ball 1 survives we can test a total of n + (n -1) + (n -2) +…+ 2 + 1
- =$n(n$ +1)/2 floors with n tests.

- We need to solve $n(n + 1)/2 >= 100$
- The smallest value of n which this is true for is 14
- Test floors in the order 14, 27, 39, 50, 60, 69, 77, 84, 90, 95, 99, 100.
- If the first ball breaks, test the floors between the last two floors tested in order.