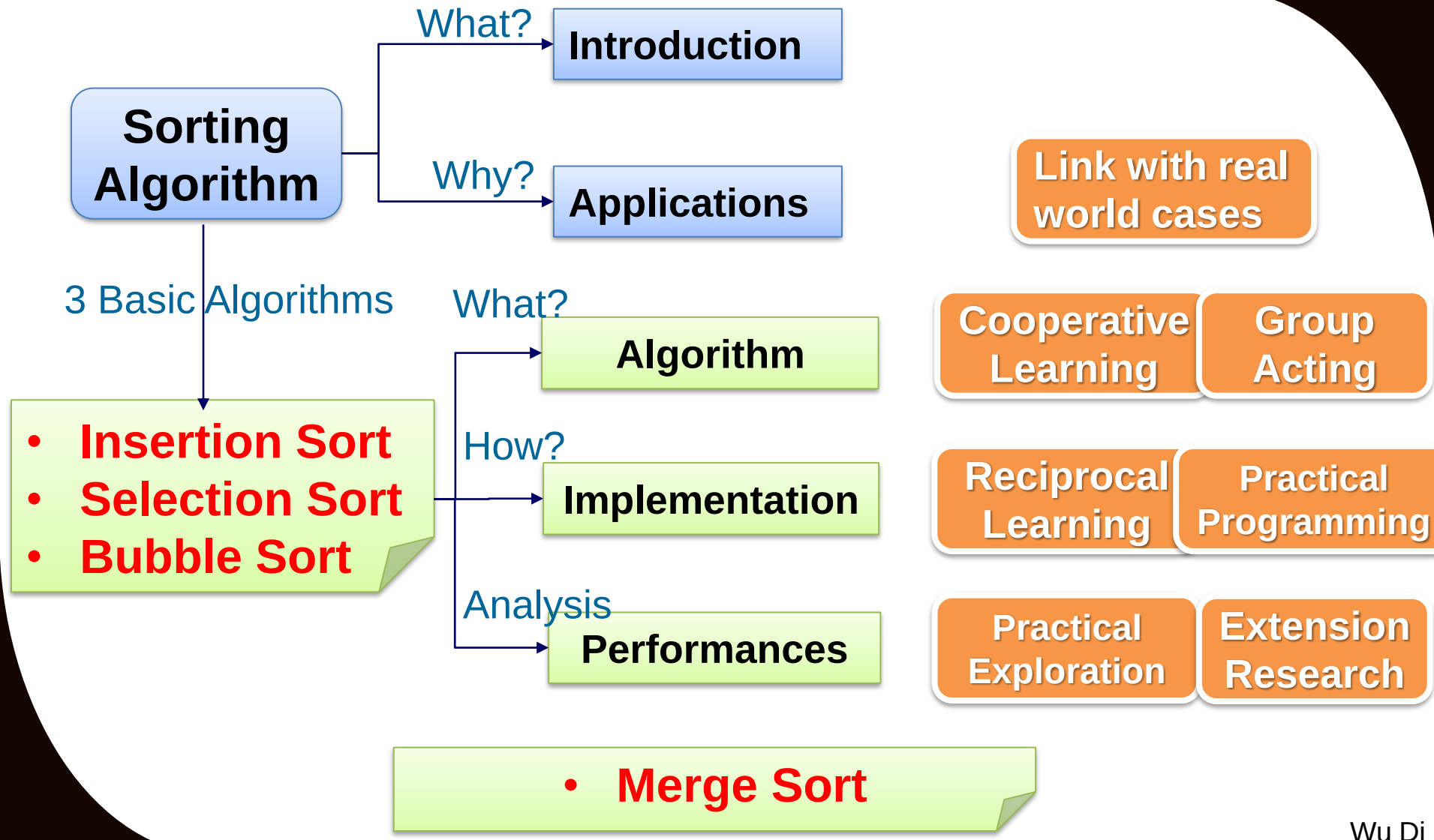
IB computer science

Sorting Algorithms:

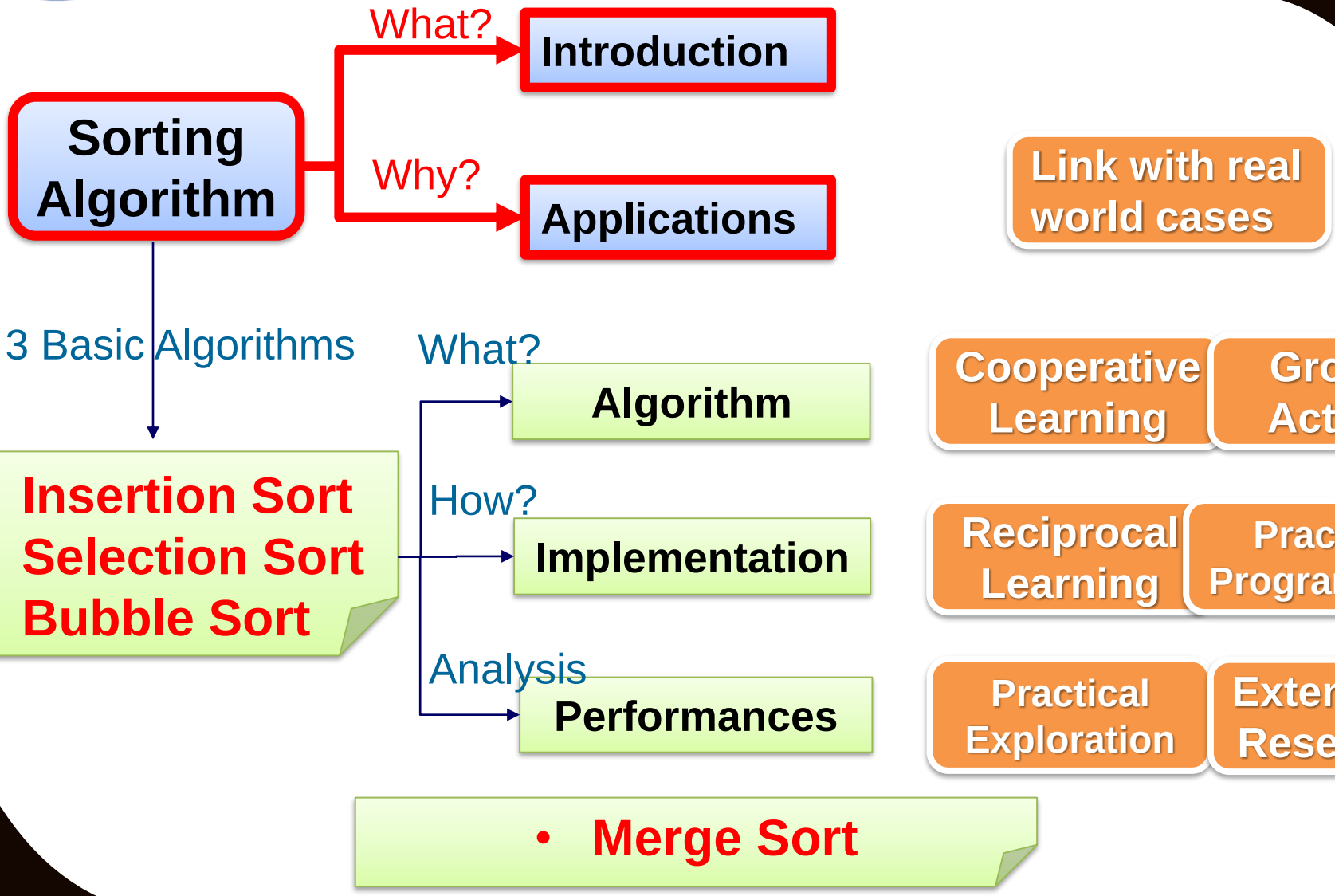
- Insertion Sort
- Selection Sort
- Bubble Sort
- Merge Sort

Wu Di
武迪

Lesson Orientation



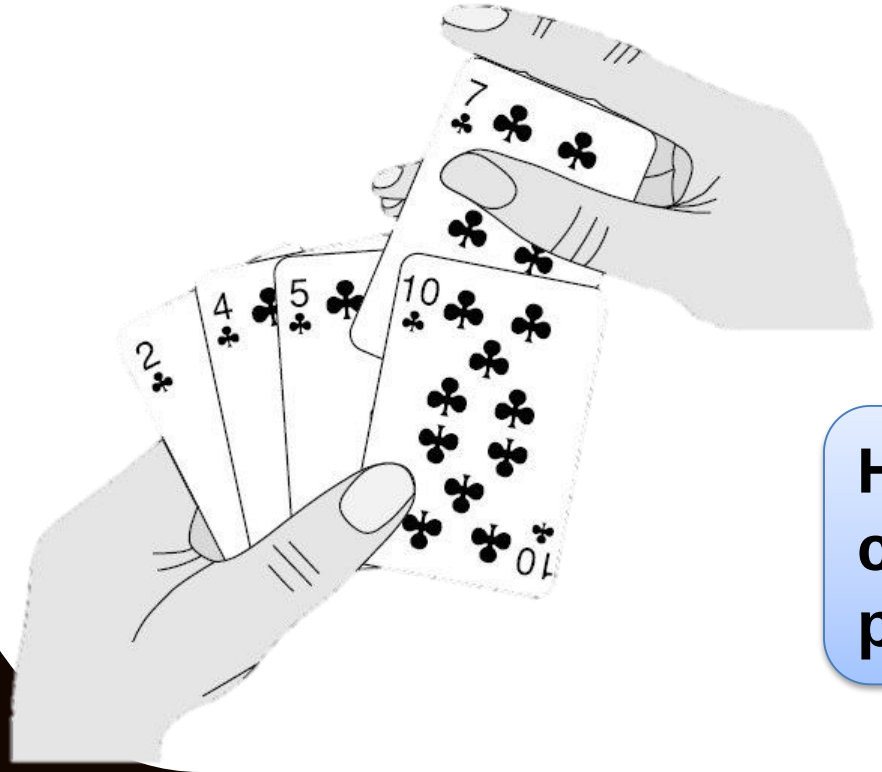
Lesson Orientation



Introduction of Sorting Algorithm

A sorting algorithm is an algorithm that:

- puts elements of an array (a list)
- in a certain order, e.g., ascending numerical order

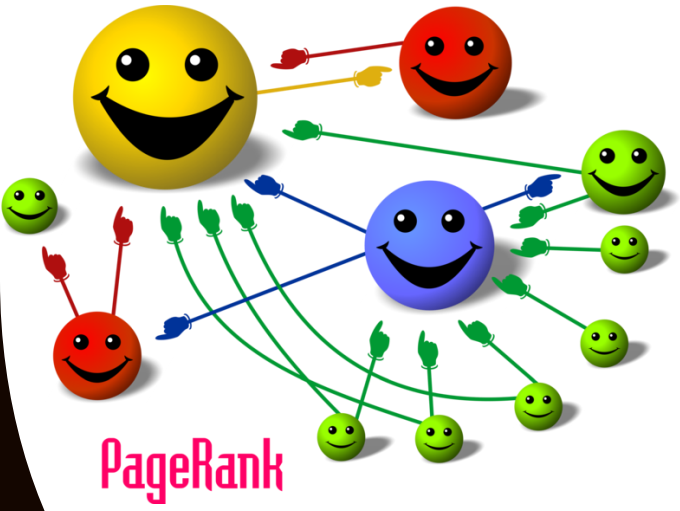


Link with real world cases

How do you draw and collate cards (抓牌和理牌) during a poker game?

Applications of Sorting Algorithm

- *Commercial computing*
- *Search for information*
- *Operations research* (运筹学)
- ……



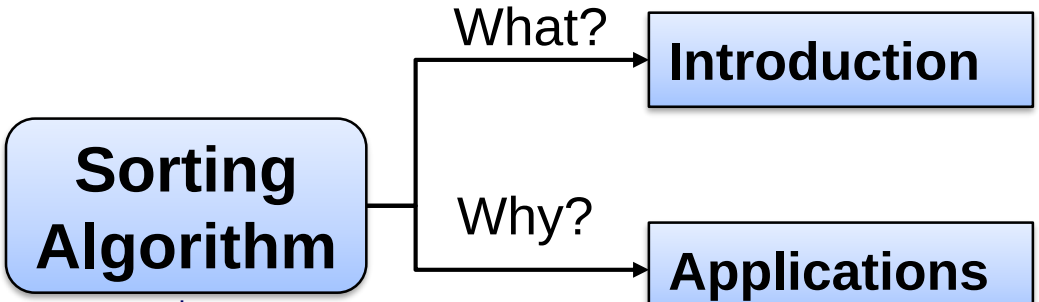
Link with real world cases

Google PageRank

Google

Sorting application in our daily life?

Lesson Orientation



Link with real world cases

3 Basic Algorithms

What?

Algorithm

- Insertion Sort
- Selection Sort
- Bubble Sort

How?

Implementation

Analysis

Performances

Cooperative Learning

Group Acting

Reciprocal Learning

Practical Programming

Practical Exploration

Extension Research

- Merge Sort

3 Basic Algorithms

How are we going to learn?

Step 1

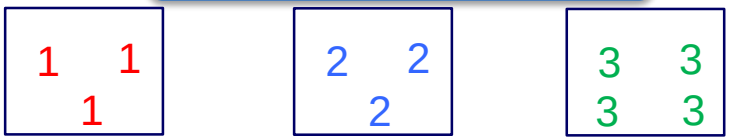
Teacher: Brief Introduction of the 3 basic algorithm with examples
You: Do you best to capture all the key points for the 3 algorithms

Step 2

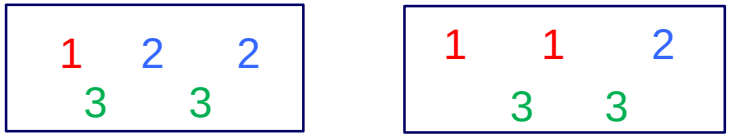
Cooperative Learning

Step 3

Subtopic groups



Jigsaw groups



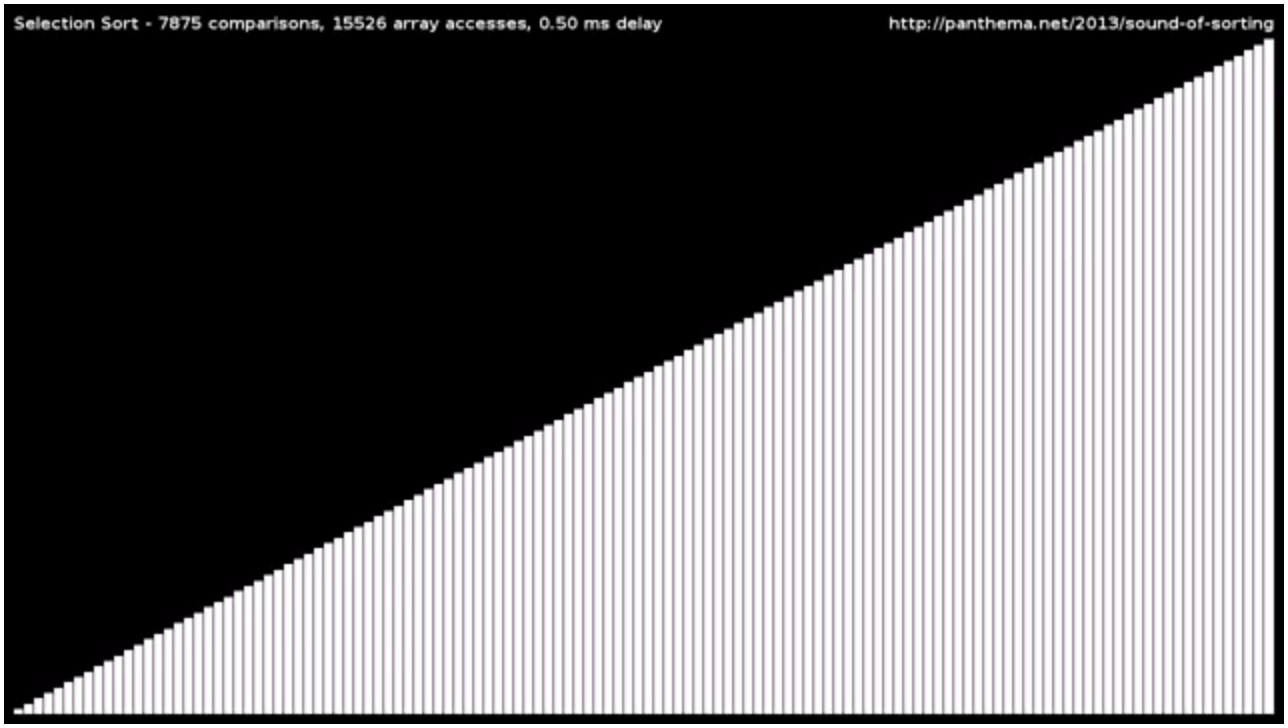
Group Learning:
- Complete the pseudocode
- Demo how it works with poker
- Complete the trace table

Group 1: Insertion Sort
Group 2: Selection Sort
Group 3: Bubble Sort

Group Teaching and Learning:
- Teach new group your subtopic with learning materials
- New group will do group acting of an algorithm based on drawing lots

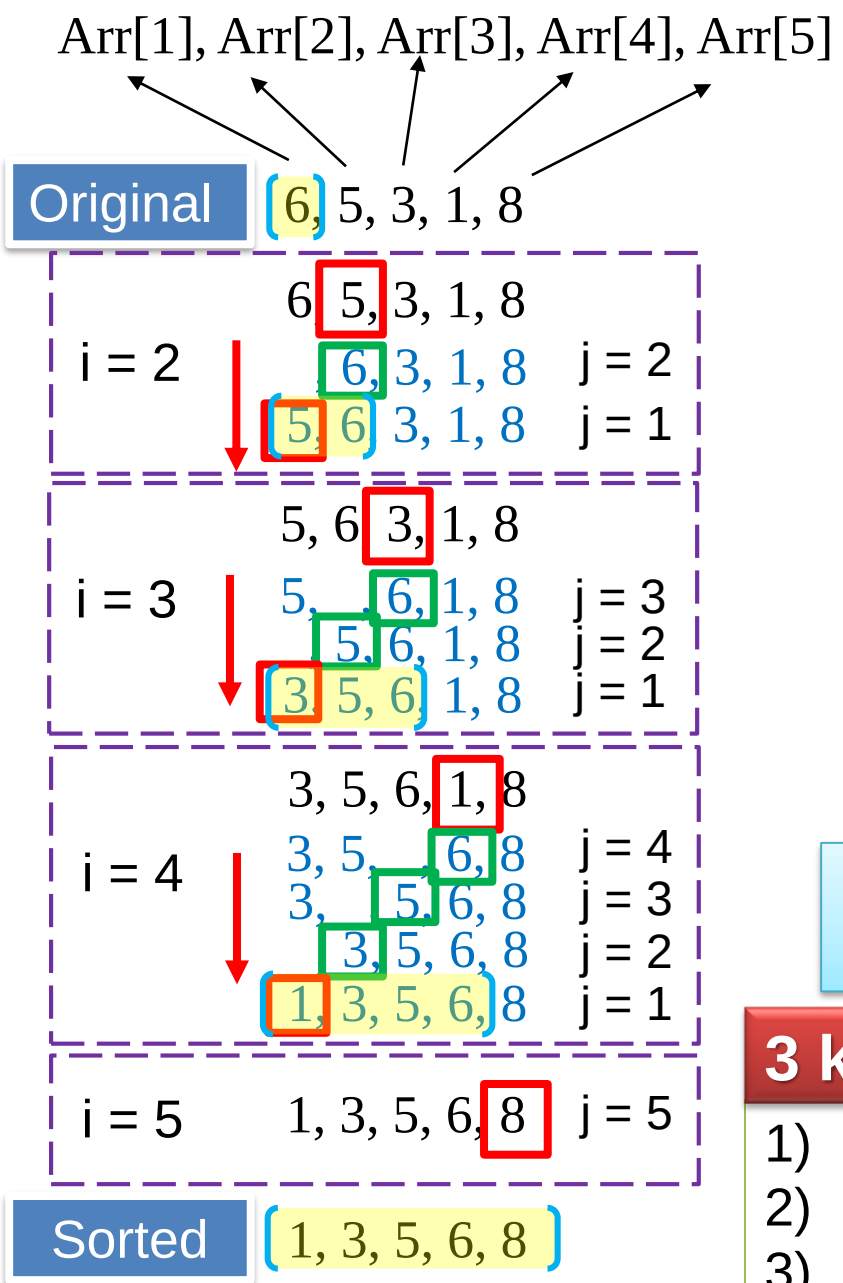
Group Acting

3 Basic Algorithms: Insertion



Insertion Sort

maintains a ***sorted sub-array***, and repetitively **inserts new elements** into it




```
// Arr: An Array which is one-based
N = LENGTH(Arr)
FOR i = 2 TO N
        CurValue = Arr[i]
        j = i
        WHILE j > 1 AND CurValue < Arr[j-1]
                Arr[j] = Arr[j-1]
                j = j -1
        ENDWHILE
        Arr[j] = CurValue
ENDFOR
```

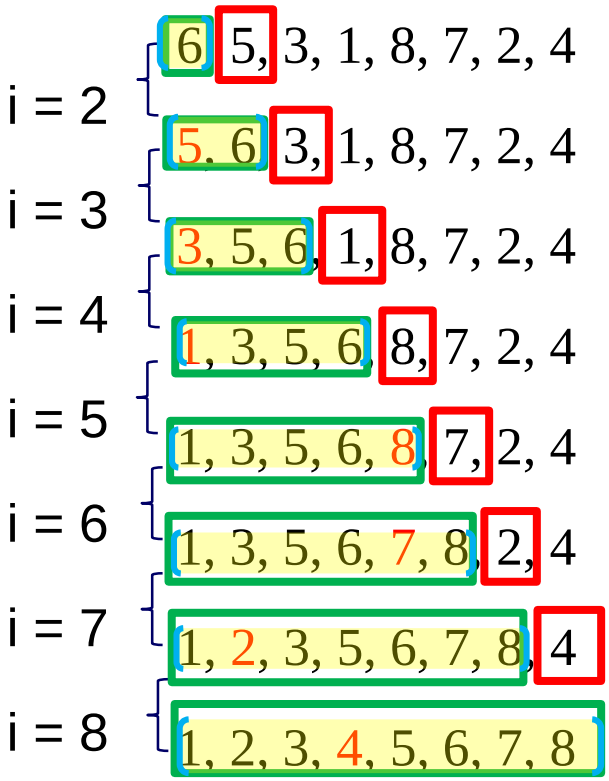
Compare

Insertion

maintains a ***sorted sub-array***, and repetitively ***inserts new elements*** into it

3 key points

1) Sorted sub-array
2) New elements to be inserted i→
3) How to Make space for inserting ←j




```
// Arr: An Array which is one-based
N = LENGTH(Arr)
FOR i = 2 TO N
        CurValue = Arr[i]
        j = i
        WHILE j > 1 AND CurValue < Arr[j-1]
                Arr[j] = Arr[j-1]
                j = j -1
        ENDWHILE
        Arr[j] = CurValue
ENDFOR
```

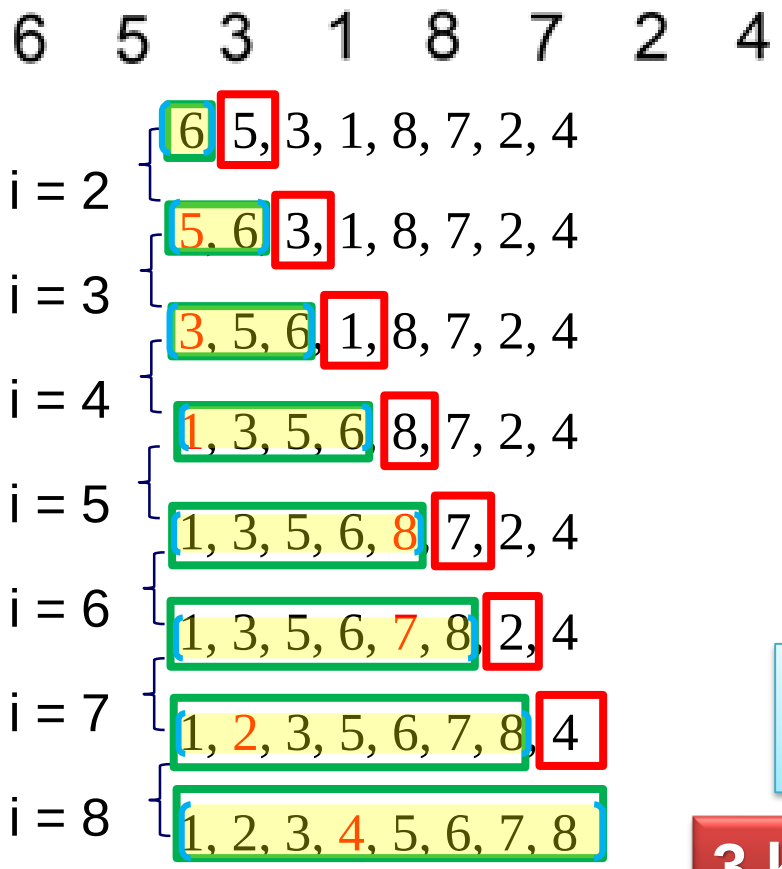
Compare

Insertion

maintains a ***sorted sub-array***, and repetitively ***inserts new elements*** into it

3 key points

1) Sorted sub-array
2) New elements to be inserted i→
3) How to Make space for inserting ←j




```
// Arr: An Array which is one-based
N = LENGTH(Arr)
FOR i = 2 TO N
        CurValue = Arr[i]
        j = i
        WHILE j > 1 AND CurValue < Arr[j-1]
                Arr[j] = Arr[j-1]
                j = j -1
        ENDWHILE
        Arr[j] = CurValue
ENDFOR
```

Compare

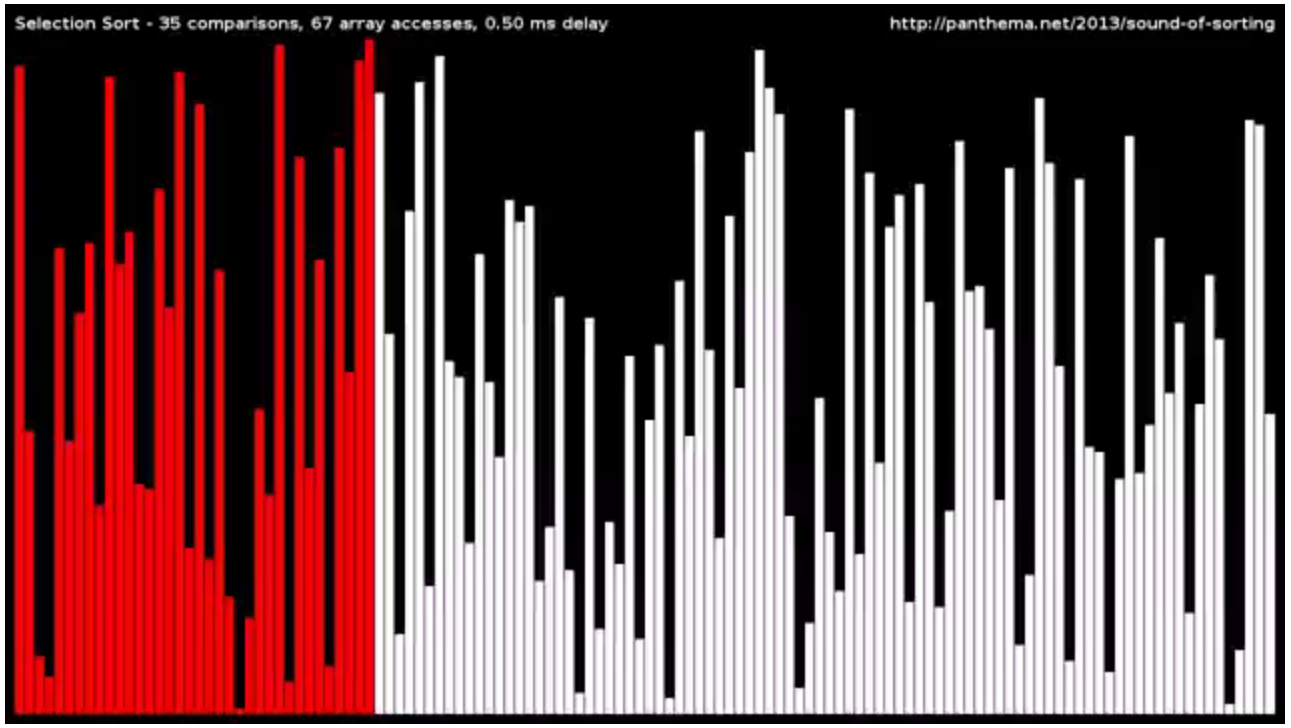
Insertion

maintains a ***sorted sub-array***, and repetitively ***inserts new elements*** into it

3 key points

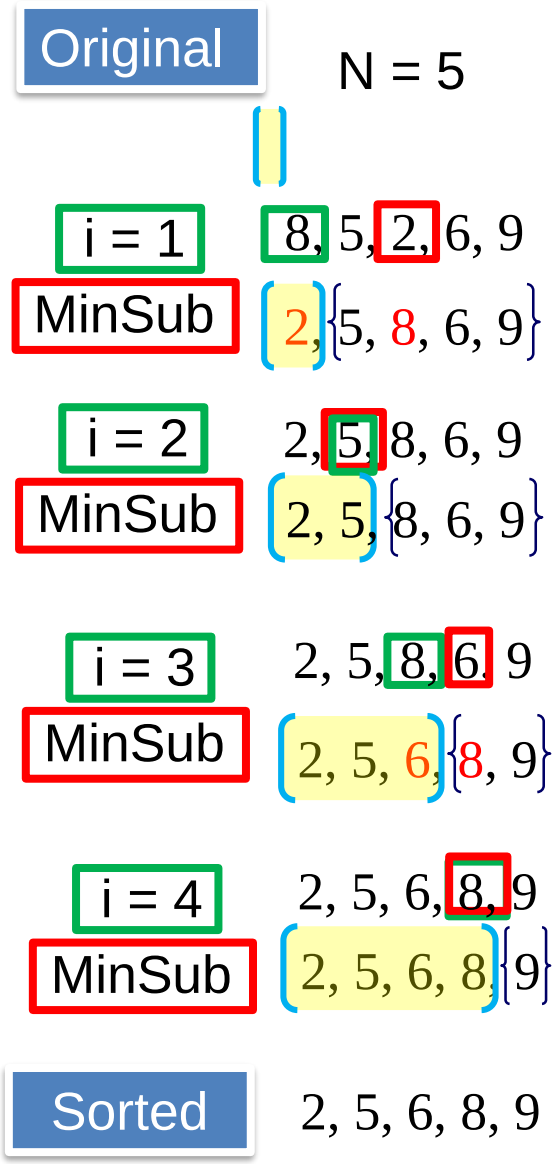
1) Sorted sub-array
2) New elements to be inserted i→
3) How to Make space for inserting ←j

3 Basic Algorithms: Selection



Selection Sort

repetitively ***pick up the smallest*** element and put it into the ***right position***




```
// Arr: An Array which is one-based
N = LENGTH(Arr)
FOR i = 1 TO N-1
        MinSub = i
        FOR j = i + 1 TO N
                IF  Arr[j] < Arr[MinSub]
                        MinSub = j
                ENDIF
        ENDFOR
        Temp = Arr[i]
        Arr[i] = Arr[MinSub]
        Arr[MinSub] = Temp
ENDFOR
```

Compare

Swap

Selection

repetitively **pick up the smallest** element and **put it into the right position**

3 key points

1) Sorted sub-array
2) How to pick up the smallest — MinSub
3) How to put it into the right position

i = 1 {8, 5, 2, 6, 9, 3, 1, 4, 0, 7}

i = 2 {0} {5, 2, 6, 9, 3, 1, 4, 8, 7}

i = 3 {0, 1} {2, 6, 9, 3, 5, 4, 8, 7}

i = 4 {0, 1, 2} {6, 9, 3, 5, 4, 8, 7}

i = 5 {0, 1, 2, 3} {9, 6, 5, 4, 8, 7}

i = 6 {0, 1, 2, 3, 4} {6, 5, 9, 8, 7}

i = 7 {0, 1, 2, 3, 4, 5} {6, 9, 8, 7}

i = 8 {0, 1, 2, 3, 4, 5, 6} {9, 8, 7}

i = 9 {0, 1, 2, 3, 4, 5, 6, 7} {8, 9}

{0, 1, 2, 3, 4, 5, 6, 7, 8} 9

```
// Arr: An Array which is one-based
N = LENGTH(Arr)
FOR i = 1 TO N-1
        MinSub = i
        FOR j = i + 1 TO N
                IF  Arr[j] < Arr[MinSub]
                        MinSub = j
                ENDIF
        ENDFOR
        Temp = Arr[i]
        Arr[i] = Arr[MinSub]
        Arr[MinSub] = Temp
ENDFOR
```

Compare

Swap

repetitively **pick up the smallest** element and **put it into the right position**

3 key points

1) Sorted sub-array
2) How to pick up the smallest — MinSub
3) How to put it into the right position

i = 1 {8, 5, 2, 6, 9, 3, 1, 4, 0, 7}

i = 2 {0} {5, 2, 6, 9, 3, 1, 4, 8, 7}

i = 3 {0, 1} {2, 6, 9, 3, 5, 4, 8, 7}

i = 4 {0, 1, 2} {6, 9, 3, 5, 4, 8, 7}

i = 5 {0, 1, 2, 3} {9, 6, 5, 4, 8, 7}

i = 6 {0, 1, 2, 3, 4} {6, 5, 9, 8, 7}

i = 7 {0, 1, 2, 3, 4, 5} {6, 9, 8, 7}

i = 8 {0, 1, 2, 3, 4, 5, 6} {9, 8, 7}

i = 9 {0, 1, 2, 3, 4, 5, 6, 7} {8, 9}

{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

8
5
2
6
9
3
1
4
0
7

```
// Arr: An Array which is one-based
N = LENGTH(Arr)
FOR i = 1 TO N-1
        MinSub = i
        FOR j = i + 1 TO N
                IF  Arr[j] < Arr[MinSub]
                        MinSub = j
                ENDIF
        ENDFOR
        Temp = Arr[i]
        Arr[i] = Arr[MinSub]
        Arr[MinSub] = Temp
ENDFOR
```

Compare

Swap

Red is current min.
Yellow is sorted list.
Blue is current item.

repetitively **pick up the smallest** element and **put it into the right position**

3 key points

1) Sorted sub-array
2) How to pick up the smallest — MinSub
3) How to put it into the right position

3 Basic Algorithms: Bubble

Bubble Sort

repetitively **compares adjacent pairs** of elements and **swaps** if necessary

Original

N = 5

i = 1	6, 5, 3, 1, 8	j = 1
	5, 6, 3, 1, 8	j = 2
	5, 3, 6, 1, 8	j = 3
	5, 3, 1, 6, 8	j = 4
	5, 3, 1, 6, 8	
i = 2	5, 3, 1, 6, 8	j = 1
	3, 5, 1, 6, 8	j = 2
	3, 1, 5, 6, 8	j = 3
	3, 1, 5, 6, 8	
i = 3	1, 3, 5, 6, 8	j = 1
	1, 3, 5, 6, 8	j = 2
	1, 3, 5, 6, 8	
i = 4	1, 3, 5, 6, 8	j = 1
	1, 3, 5, 6, 8	

```
// Arr: An Array which is one-based
N = LENGTH(Arr)
FOR i = 1 TO N-1
    FOR j = 1 To N-i
        IF Arr[j] > Arr[j+1]
            Temp = Arr[j]
            Arr[j] = Arr[j+1]
            Arr[j+1] = Temp
        ENDIF
    ENDFOR
ENDFOR
```

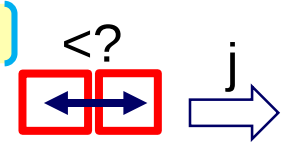
Compare

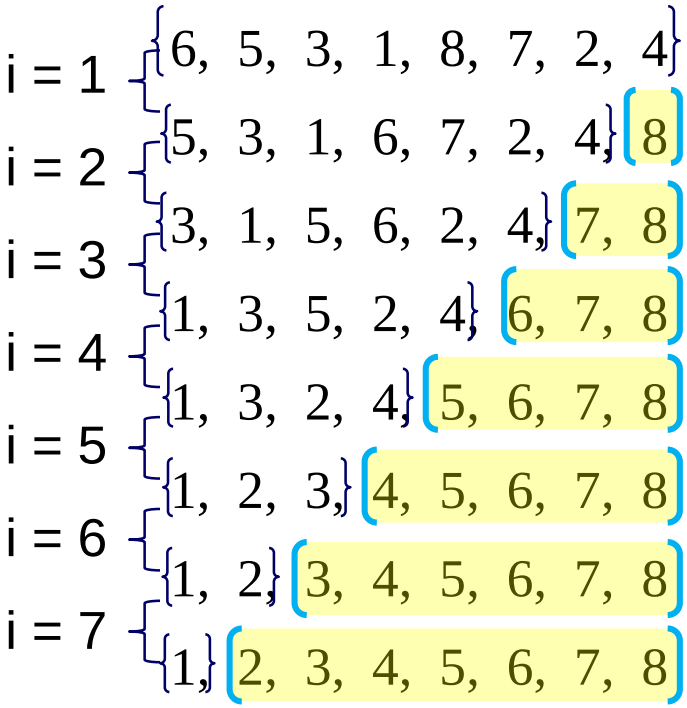
Swap

repetitively **compares adjacent pairs** of elements and **swaps** if necessary

3 key points

1) Sorted sub-array
2) Result of each inner loop
3) Why called bubble sort?






```
// Arr: An Array which is one-based
N = LENGTH(Arr)
FOR i = 1 TO N-1
    FOR j = 1 To N-i
        IF Arr[j] > Arr[j+1]
            Temp = Arr[j]
            Arr[j] = Arr[j+1]
            Arr[j+1] = Temp
        ENDIF
    ENDFOR
ENDFOR
```

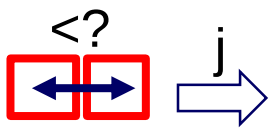
Compare

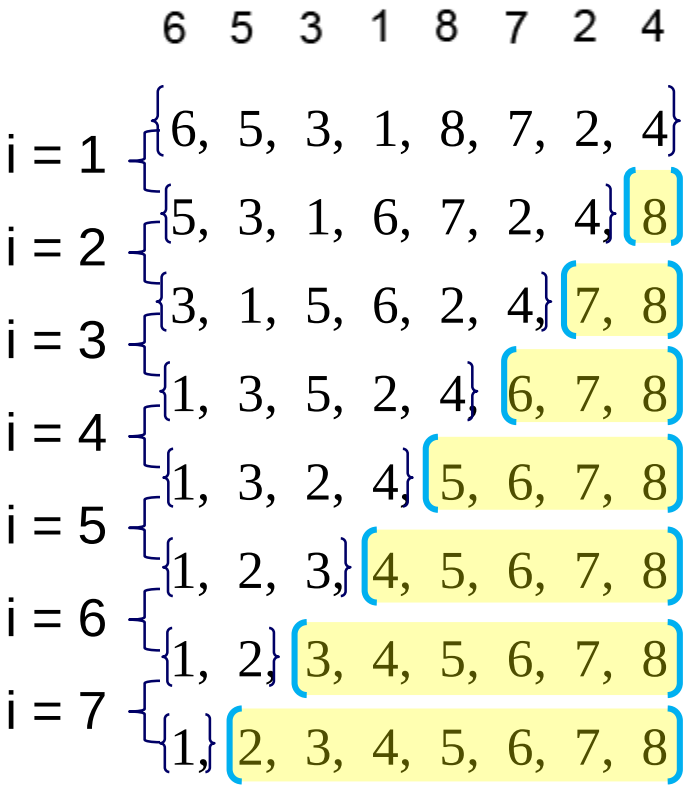
Swap

repetitively **compares adjacent pairs** of elements and **swaps** if necessary

3 key points

1) Sorted sub-array
2) Result of each inner loop
3) Why called bubble sort?






```
// Arr: An Array which is one-based
N = LENGTH(Arr)
FOR i = 1 TO N-1
    FOR j = 1 To N-i
        IF Arr[j] > Arr[j+1]
            Temp = Arr[j]
            Arr[j] = Arr[j+1]
            Arr[j+1] = Temp
        ENDIF
    ENDFOR
ENDFOR
```

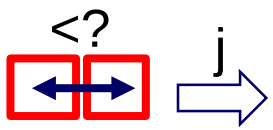
Compare

Swap

repetitively **compares adjacent pairs** of elements and **swaps** if necessary

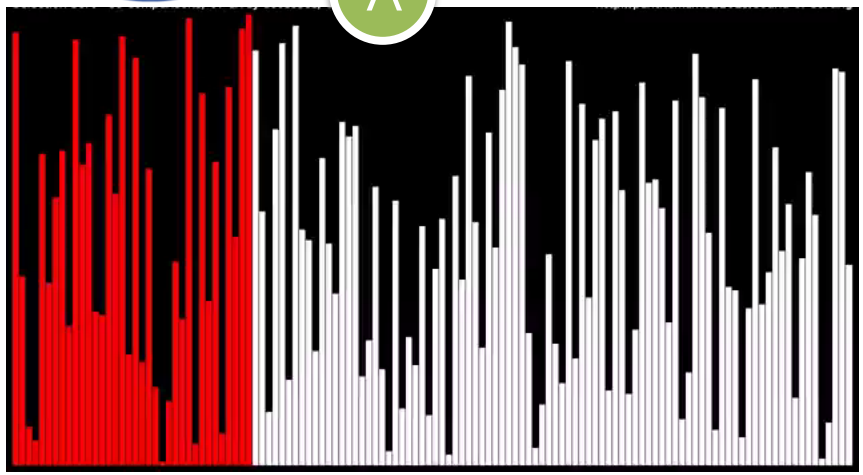
3 key points

1) Sorted sub-array
2) Result of each inner loop
3) Why called bubble sort?



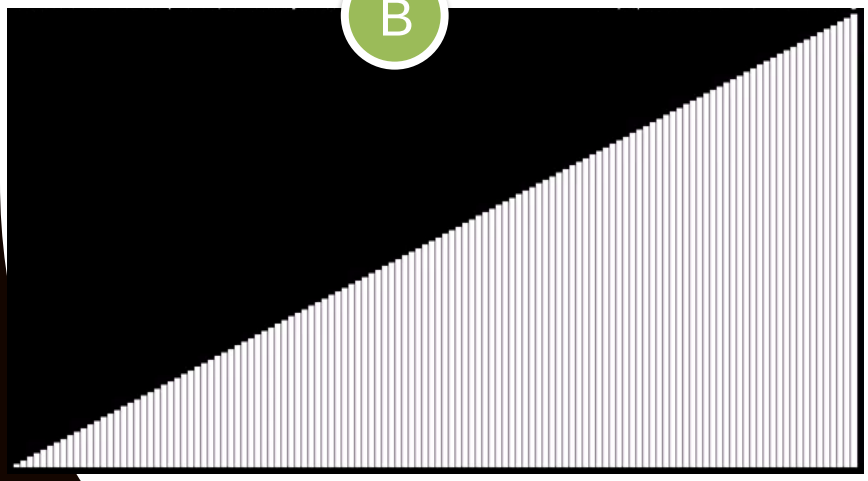
Quiz: 3 Basic Algorithms

A



C

B



1 maintains a sorted sub-array, and repetitively inserts new elements into it

2 repetitively compares adjacent pairs of elements and swaps if necessary

3 repetitively pick up the smallest element and put it into the right position

Insertion 1 B

Selection 3 A

Bubble 2 C

3 Basic Algorithms

How are we going to learn?

Step 1

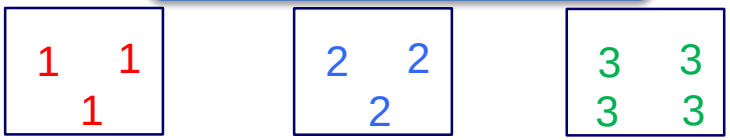
Teacher: Brief Introduction of the 3 basic algorithm with examples
You: Do you best to capture all the key points for the 3 algorithms

Step 2

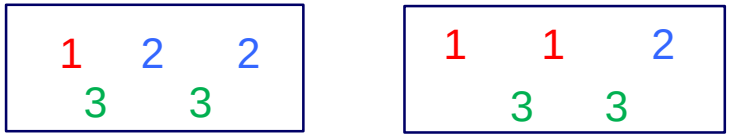
Cooperative Learning

Step 3

Subtopic groups



Jigsaw groups



Group Learning:
- Demo how it works with poker
- Complete the pseudocode
- Complete the trace table

Group 1: Insertion Sort
Group 2: Selection Sort
Group 3: Bubble Sort

Group Teaching and Learning:
- Teach new group your subtopic with learning materials
- New group will do group acting of an algorithm based on drawing lots

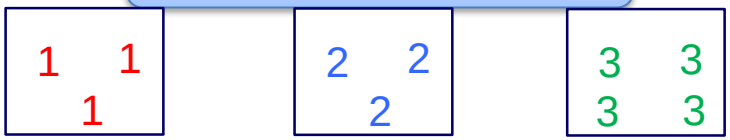
Group Competition

3 Basic Algorithms

Step 2

Cooperative Learning

Subtopic groups



Group 1: Insertion Sort
Group 2: Selection Sort
Group 3: Bubble Sort

Group Learning: 4 minutes

- Complete the pseudocode: 1 min
- Demo how it works with poker: 1 min
- Complete the trace table: 2 min

Handout: Group Learning Activity

Array: {7,9,3,2}

1: Arthur, Enzo, David

2: Tom, Matt, Margret

3: Alex, Charles, Doris, Michael

3 Basic Algorithms

Step 3

Subtopic groups

1 1 1	2 2 2	3 3 3 3

Arthur	David	Enzo
Tom	Margret	Matt
Alex Doris	Charles Michael	

1
2
3

Cooperative Learning

Jigsaw groups

1 2 2 3 3	1 1 2 3 3

Arthur Charles	Margret Michael	Matt
Alex Doris	David Tom	Enzo

A
B

Group Teaching and Learning:

- Teach new group (A,B,C) your subtopic with learning materials
  - ✓ Each subtopic: 1~2 minutes * 3
- New group will do group acting of an algorithm based on drawing lots
  - ❖ All group members should be involved in the acting
  - ❖ Show your creativities and imagination
  - ✓ 3 minutes preparation + 2 minutes show for each group * 2

3 Basic Algorithms

Step 3

Jigsaw groups

Cooperative Learning

1 2 2
3 3

1 1 2
3 3

Arthur Margret Matt
Charles Michael

A

Alex David Enzo
Doris Tom

B

Insertion

Selection

Bubble

Group Teaching and Learning:

- Teach new group (A,B,C) your subtopic with learning materials
  - ✓ Each subtopic: 1~2 minutes * 3
- New group will do group acting of an algorithm based on drawing lots
  - ❖ All group members should be involved in the acting
  - ❖ Show your creativities and imagination
  - ✓ 3 minutes preparation + 2 minutes show for each group * 2

3 Basic Algorithms

Step 3

Jigsaw groups

Cooperative Learning

Draw lots time (抽签)!!!

NOTE: Prize for your excellent performances

1 2 2
3 3

1 1 2
3 3

A

Arthur Margret Matt
Charles Michael

B

Alex David Enzo
Doris Tom

- New group will do group acting of an algorithm based on drawing lots
  - ❖ All group members should be involved in the acting
  - ❖ Show your creativities and imagination
  - ✓ 3 minutes preparation + 2 minutes show for each group * 2

Hint: show Comparison and Swap

3 Basic Algorithms

Step 3

Jigsaw groups

Cooperative Learning

1 2 2
3 3

1 1 2
3 3

3 minutes preparation

A
Arthur Margret Matt
Charles Michael

B
Alex David Enzo
Doris Tom

- Insertion
- Selection
- Bubble

Prize for best performance!

- New group will do group acting of an algorithm based on drawing lots
  - ❖ All group members should be involved in the acting
  - ❖ Show your creativities and imagination
  - ✓ 3 minutes preparation + 2 minutes show for each group * 2

Hint: show Comparison and Swap

3 Basic Algorithms

Step 3

Jigsaw groups

Cooperative Learning

1 2 2
3 3

1 1 2
3 3

Arthur Margret Matt
Charles Michael

A

Alex David Enzo
Doris Tom

B

Show Time!!!

Insertion

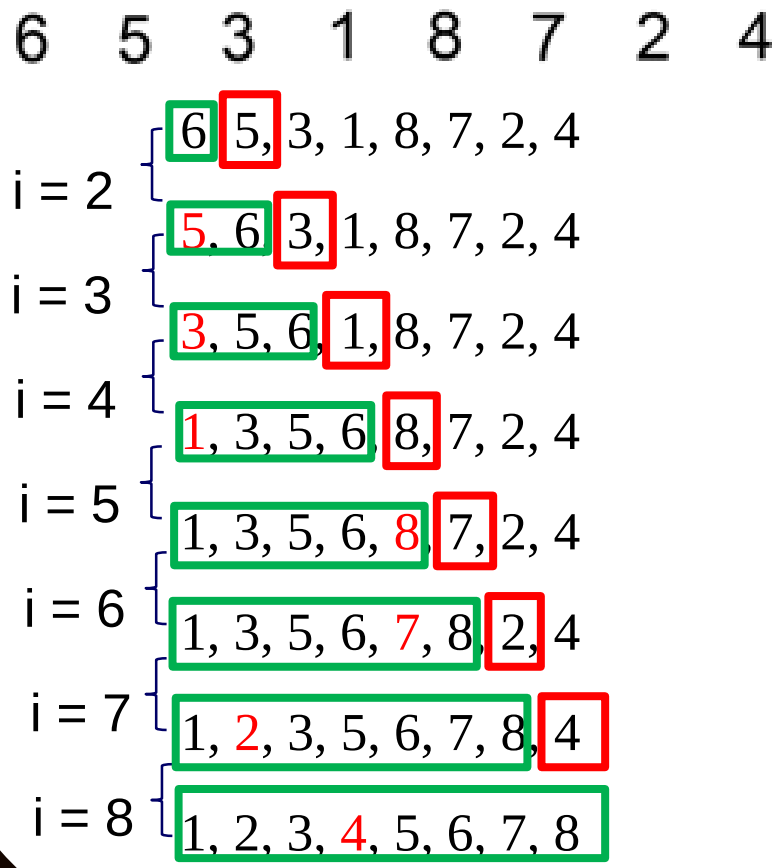
Selection

Bubble

Prize for best performance!

- New group will do group acting of an algorithm based on drawing lots
  - ❖ All group members should be involved in the acting
  - ❖ Show your creativities and imagination
  - ✓ 3 minutes preparation + 2 minutes show for each group * 2

3 Basic Algorithms: Insertion




```
// Arr: An Array which is one-based
N = LENGTH(Arr)
FOR i = 2 TO N
        CurValue = Arr[i]
        j = i
        WHILE j > 1 AND CurValue < Arr[j-1]
                Arr[j] = Arr[j-1]
                j = j -1
        ENDWHILE
        Arr[j] = CurValue
ENDFOR
```

Compare

maintains a sorted sub-array, and repetitively inserts new elements into it

3 Basic Algorithms: Selection

i = 1 {8, 5, 2, 6, 9, 3, 1, 4, 0, 7}
i = 2 {0, 5, 2, 6, 9, 3, 1, 4, 8, 7}
i = 3 {0, 1, 2, 6, 9, 3, 5, 4, 8, 7}
i = 4 {0, 1, 2, 3, 9, 6, 5, 4, 8, 7}
i = 5 {0, 1, 2, 3, 4, 6, 5, 9, 8, 7}
i = 6 {0, 1, 2, 3, 4, 5, 6, 9, 8, 7}
i = 7 {0, 1, 2, 3, 4, 5, 6, 9, 8, 7}
i = 8 {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
i = 9 {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

8
5
2
6
9
3
1
4
0
7

```
// Arr: An Array which is one-based
N = LENGTH(Arr)
FOR i = 1 TO N-1
        MinSub = i
        FOR j = i + 1 TO N
                IF  Arr [j] < Arr [MinSub]
                        MinSub = j
                ENDIF
        ENDFOR
        Temp = Arr [i]
        Arr [i] = Arr [MinSub]
        Arr [MinSub] = Temp
ENDFOR
```

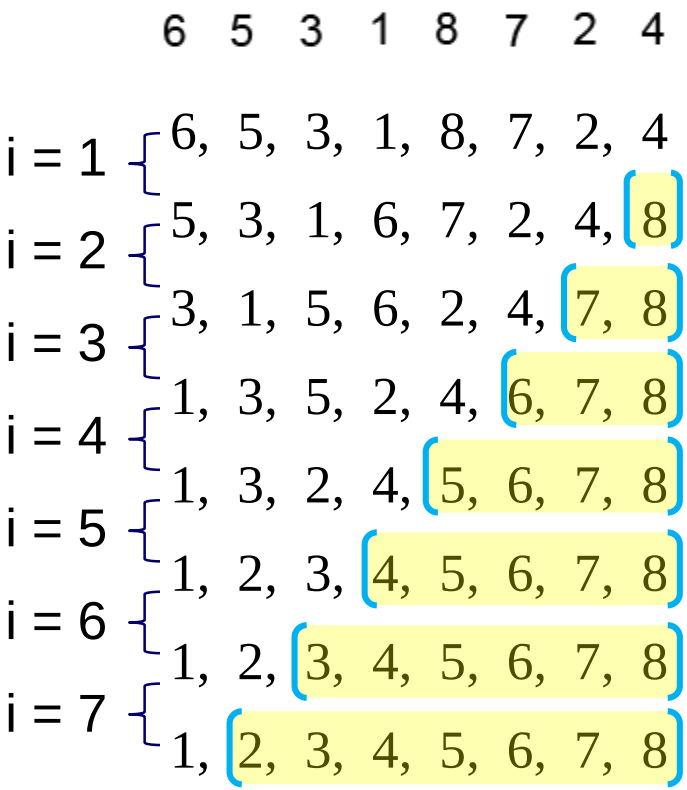
Compare

Swap

Red is current min.
Yellow is sorted list.
Blue is current item.

repetitively pick up the smallest element and put it into the right position

3 Basic Algorithms: Bubble



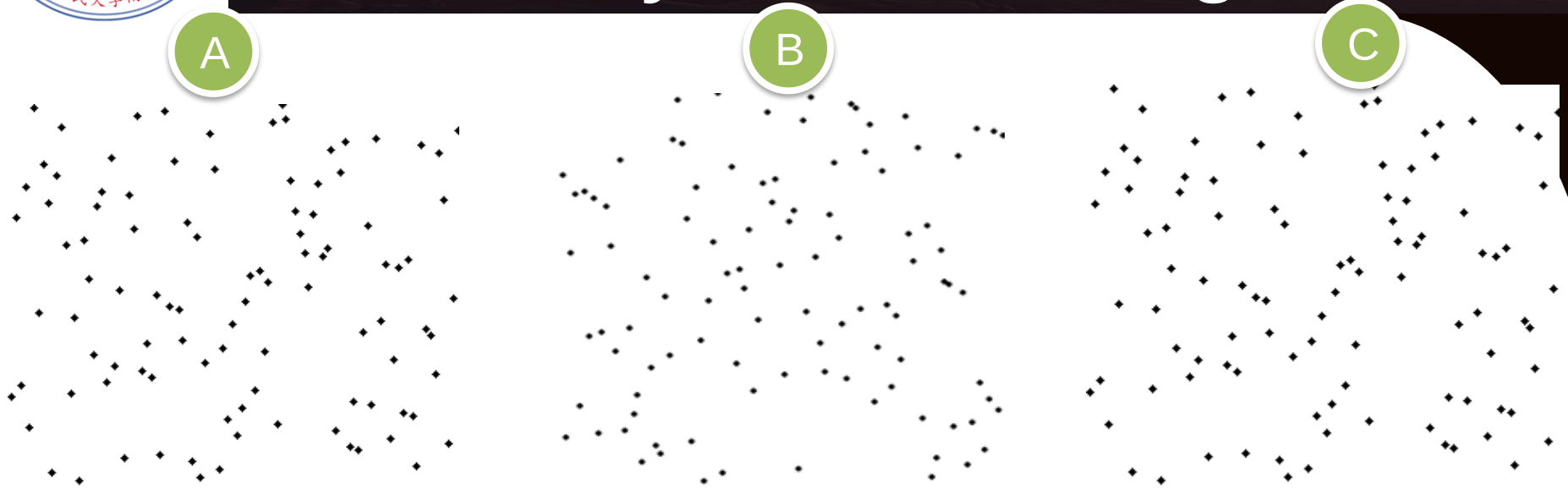

```
// Arr: An Array which is one-based
N = LENGTH(Arr)
FOR i = 1 TO N-1
    FOR j = 1 To N-i
        IF Arr [j] > Arr [j+1]
            Temp = Arr [j]
            Arr [j] = Arr [j+1]
            Arr [j+1] = Temp
        ENDIF
    ENDFOR
ENDFOR
```

Compare

Swap

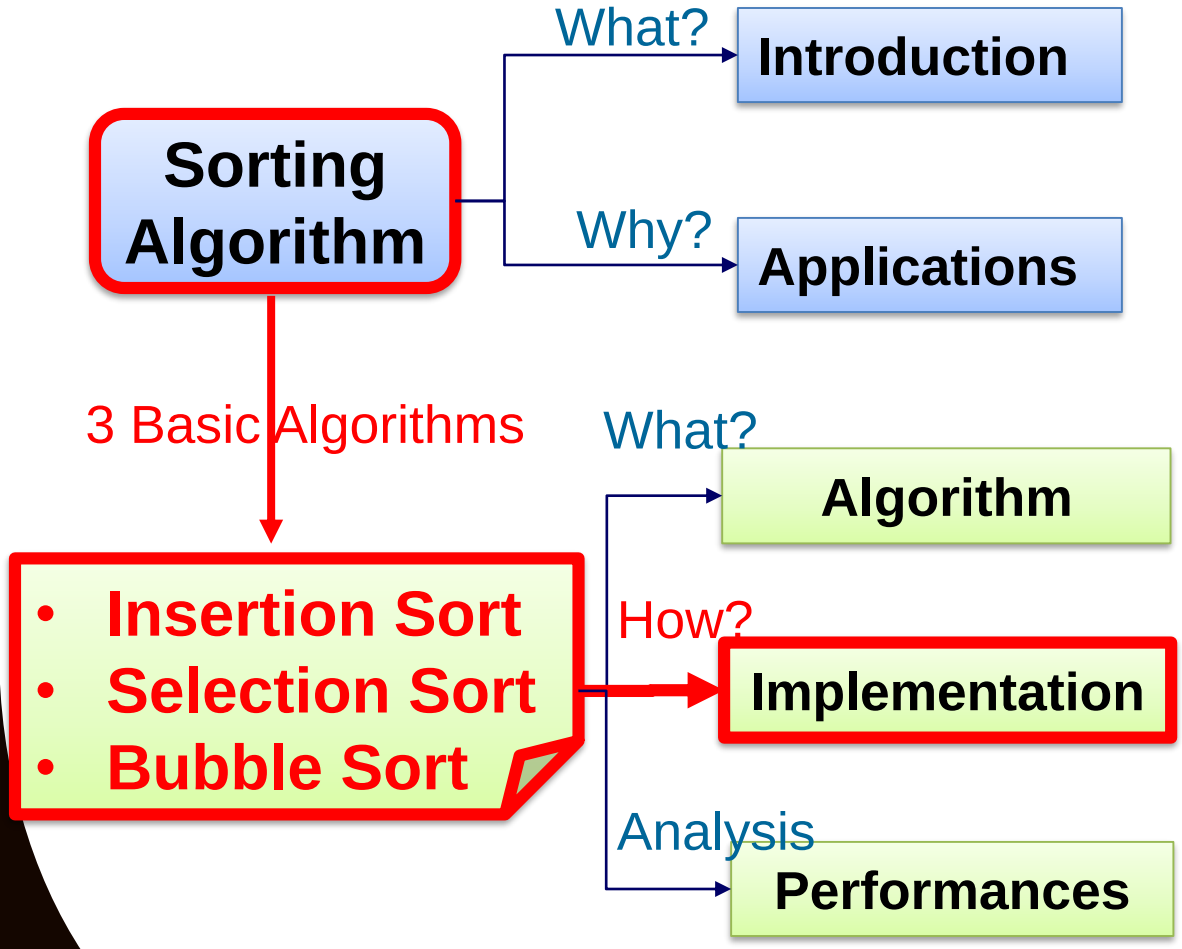
repetitively compares adjacent pairs of elements and swaps if necessary

Summary: 3 Basic Algorithms



Insertion Sort	maintains a sorted sub-array, and repetitively inserts new elements into it
Selection Sort	repetitively pick up the smallest element and put it into the right position
Bubble Sort	repetitively compares adjacent pairs of elements and swaps if necessary

Lesson Orientation



Link with real world cases

Cooperative Learning

Group Acting

Reciprocal Learning

Practical Programming

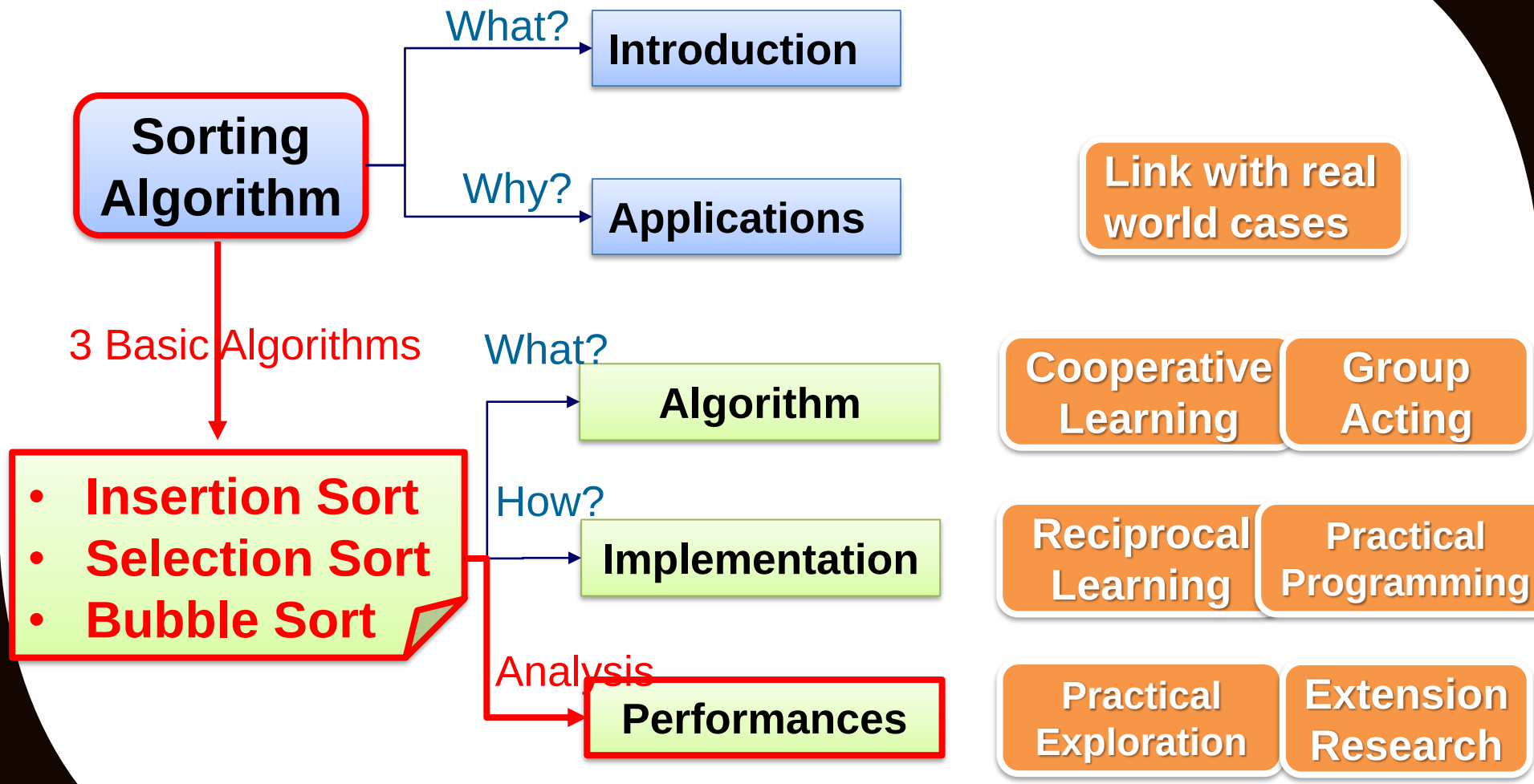
Practical Exploration

Extension Research

Implementation

- Open you Python IDE
- Download the code template from our wiki:
  - Home: 5. Topics and materials:
  - Sorting Algorithms: insertion, selection, bubble and merge sort
  - 2: The code: 3 basic algorithms for you to complete (with pseudocode a
- Try to complete the core part of each algorithm
  - Insertion
  - Selection
  - Bubble
- Test your algorithms
- Submit your code to your wiki project page

Lesson Orientation



Performance Analysis

In terms of the size of the array(N): Big O notation - $O(f(N))$

	Computational complexity Time usage		Space complexity Memory usage
	Comparison	Swaps	
Insertion			
Selection			
Bubble			

Implementation

- Run your algorithms with different scales of input
  - See how the running time varies with increasing data input
  - Plot the Running Time v.s. N (number of input data) graph

Performance Analysis: Bubble

- Outer loop 1
  - Inner loop: N-1 Comparison
    N-1 Swap
- Outer loop 2
  - Inner loop: N-2 Comparison
    N-2 Swap
- Outer loop N-i
  - Inner loop: i Comparison
    i Swap

C_1: time required to do 1 Swap
C_2: time required to do 1 Comparison
k: constant time to declare, initialize

Array size: N

```
N = LENGTH(Arr)
FOR i = 1 TO N-1
    FOR j = 1 To N-i
        IF Arr [j] > Arr [j+1]
            Temp = Arr [j]
            Arr [j] = Arr [j+1]
            Arr [j+1] = Temp
        ENDIF
    ENDFOR
ENDFOR
```

Compare

Swap

Swap: $C_1\left((N-1)+(N-2)+\ \ldots+1\right)=C_1\frac{N(N-1)}{2}$

Comparison: $C_2\left((N-1)+(N-2)+\ \ldots+1\right)=C_2\frac{N(N-1)}{2}$

Total: $(C_1+C_2)\frac{N(N-1)}{2}+k \rightarrow O(N^2)$

Memory usage

In place + N,i,j,Temp

$O(1)$

Performance Analysis

In terms of the size of the array(N): Big O notation - $O(f(N))$

	Computational complexity Time usage		Space complexity Memory usage
	Comparison	Swaps	
Insertion			
Selection			
Bubble	$\frac{N(N-1)}{2} \sim O(N^2)$	$\frac{N(N-1)}{2} \sim O(N^2)$	$O(1)$

Homework

Theory Deduction

Based on the analysis of bubble sort → Complete the comparison table on the handout

Extensions

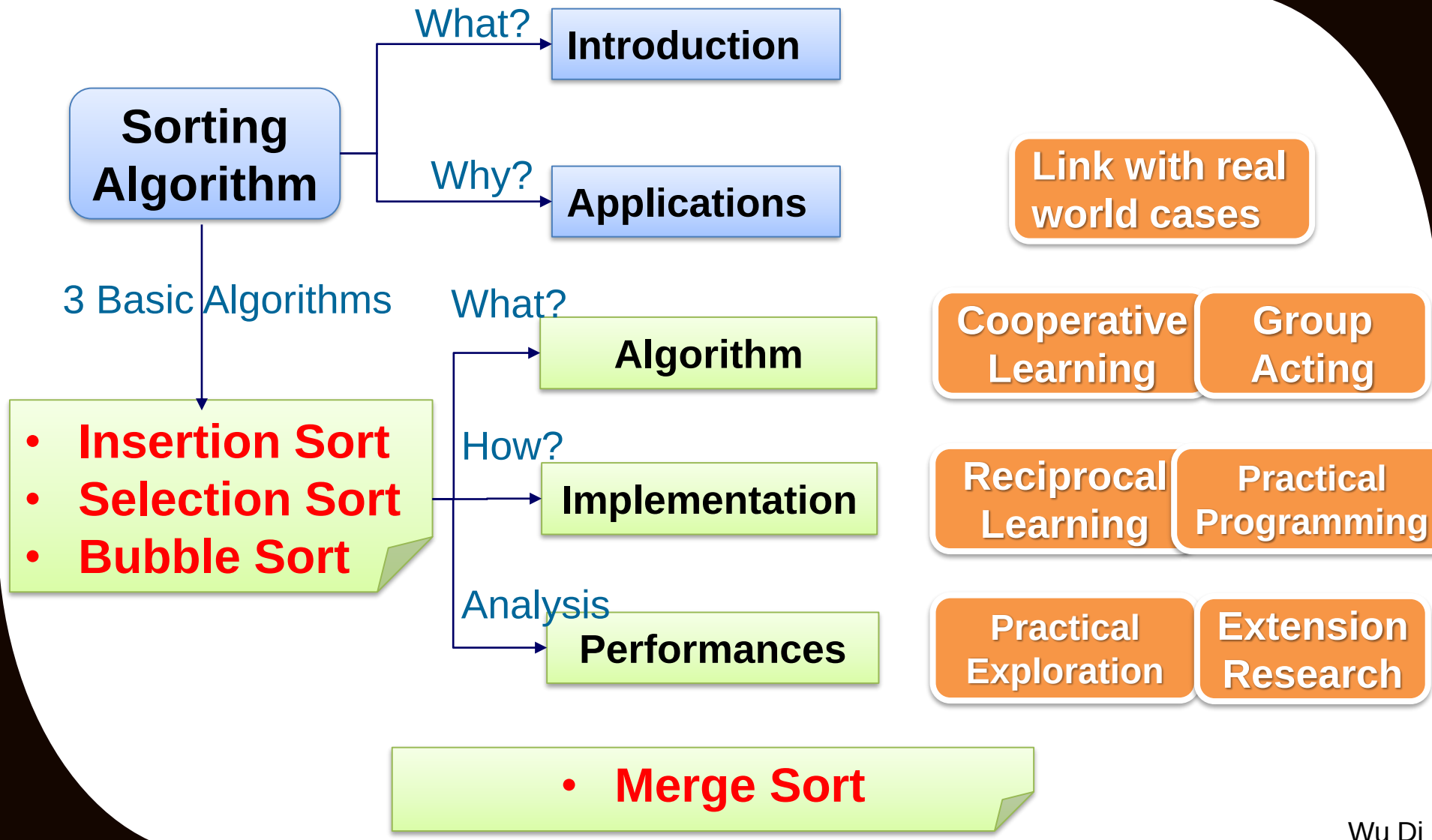
How to improve the 3 basic algorithms: Insertion, Selection, Bubble?

Algorithm **Implementation** **Performance**

- **Shell Sort**
- **Heap Sort**
- **Merge Sort**
- **Quick Sort**
- **……**

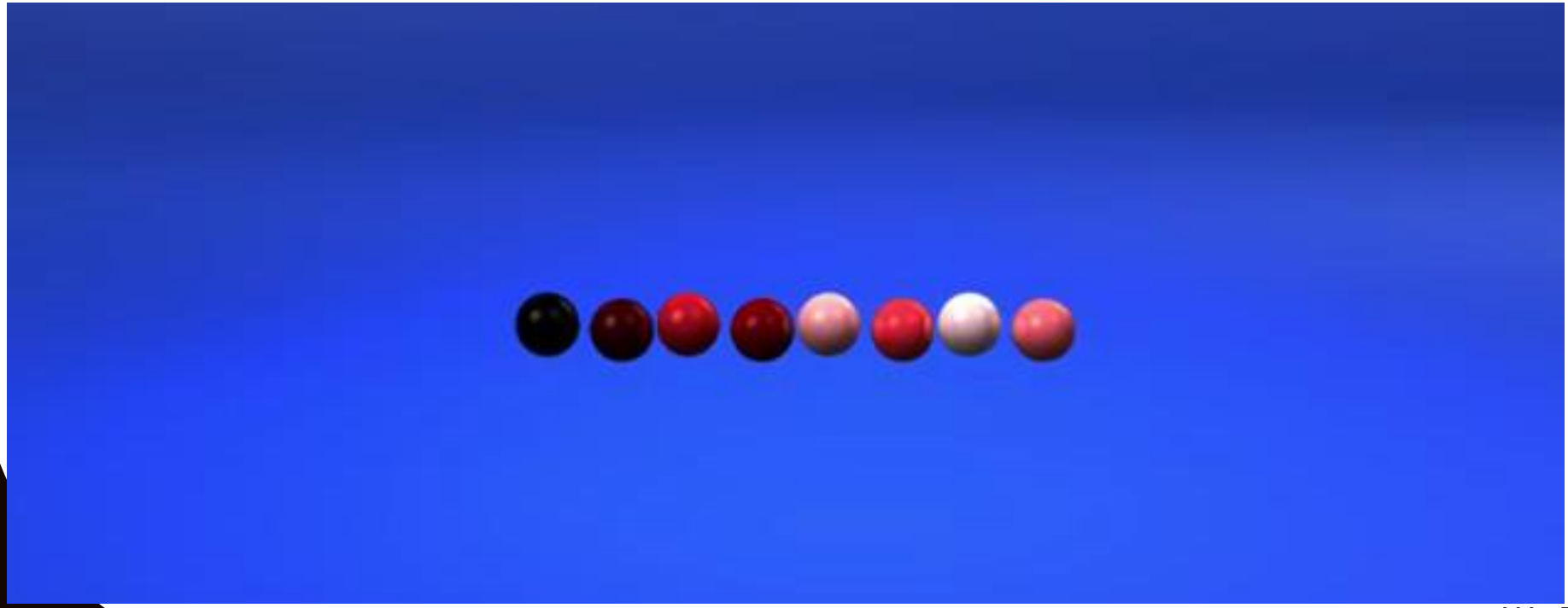
http://en.wikipedia.org/wiki/Sorting_algorithm

Lesson Orientation



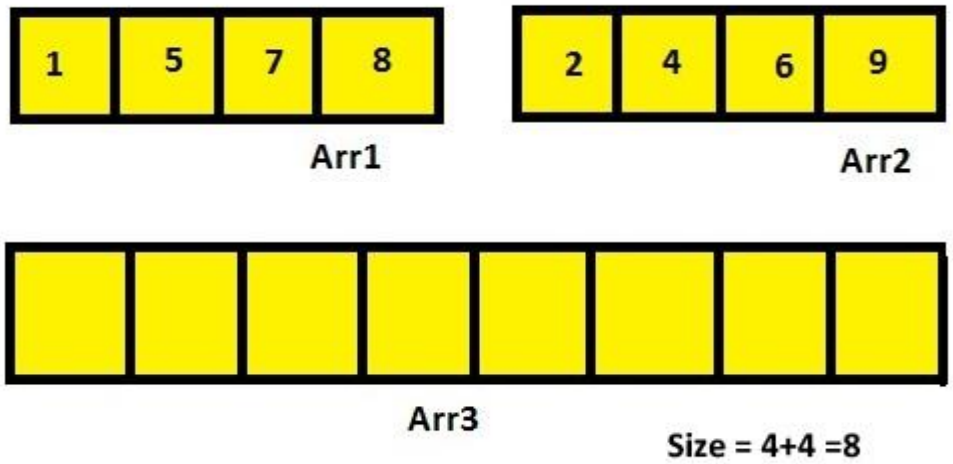
Merge sort

- What is merging?
  - Given two sorted list
  - How to combine them into one sorted list?

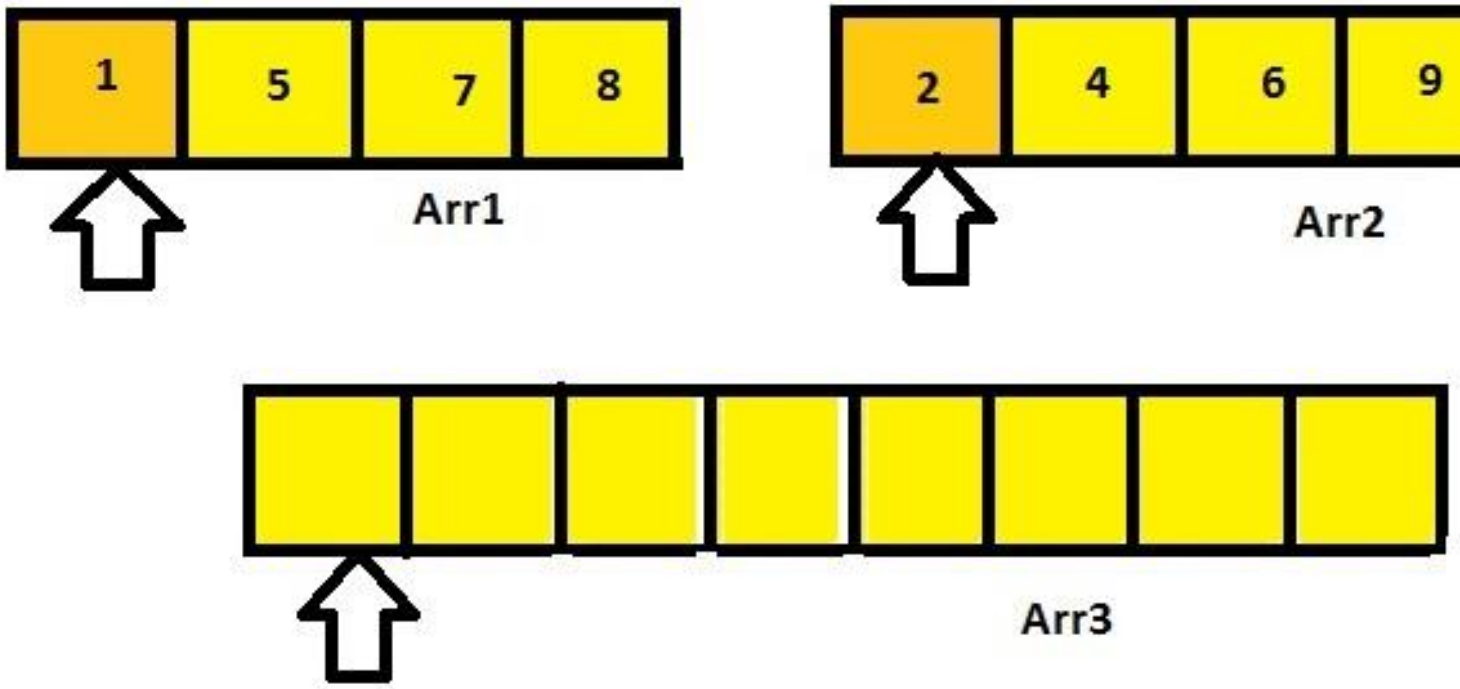


Merge sort

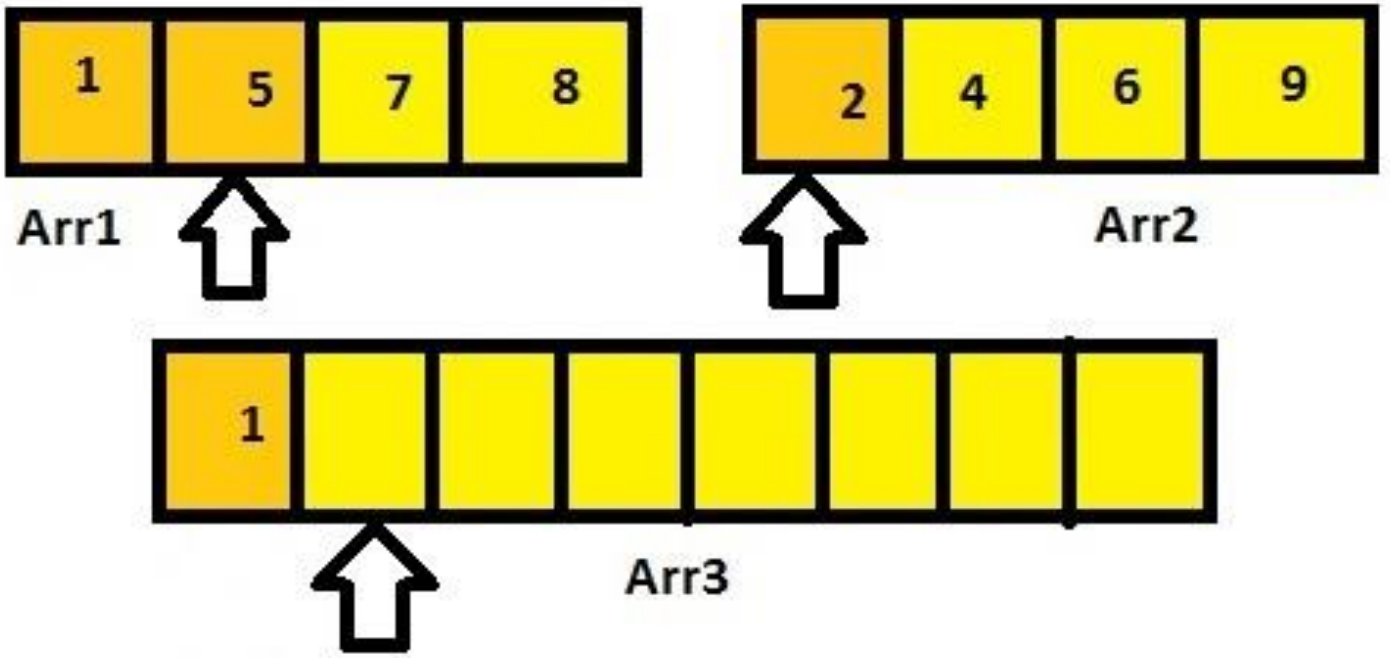
- What is merging?
  - Given two sorted list
  - How to combine them into one sorted list?



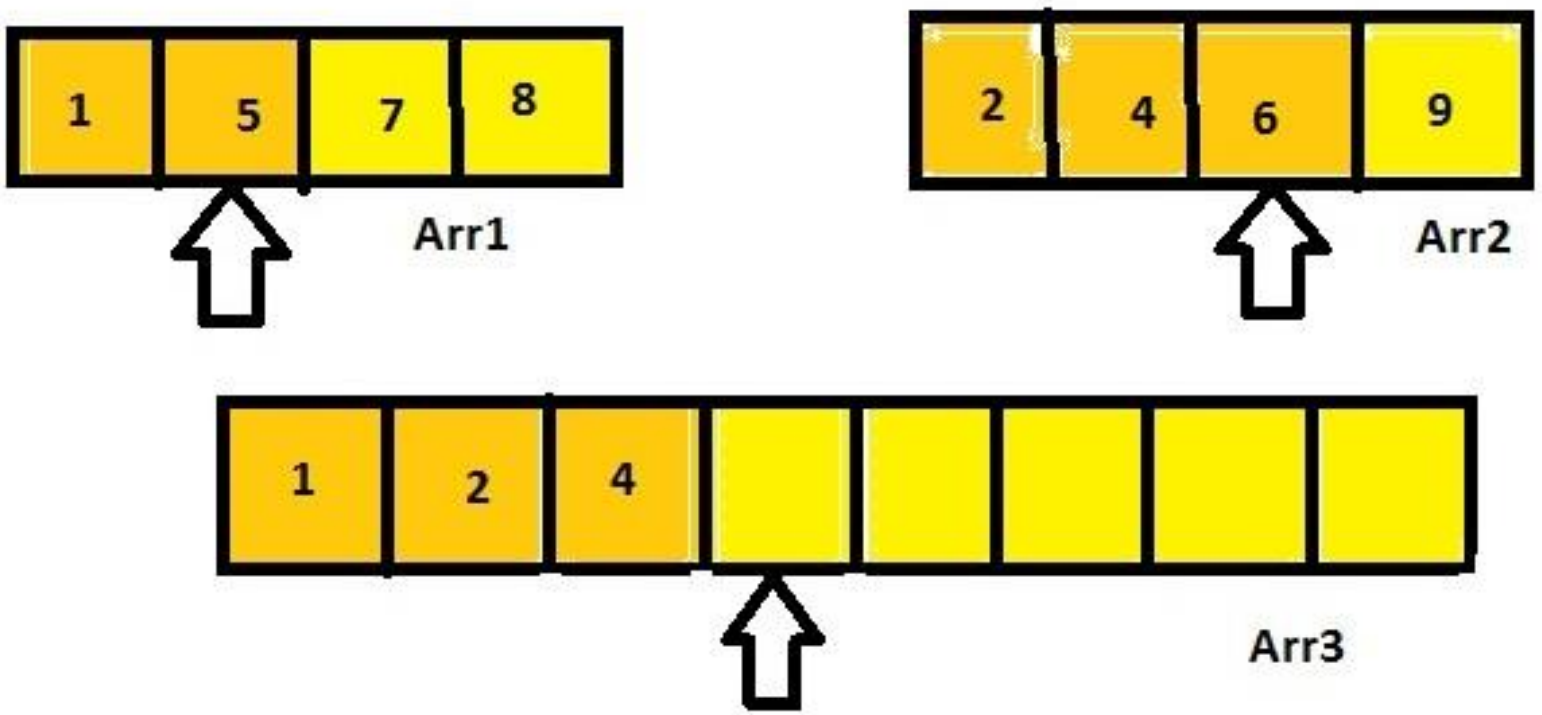
Merge sort



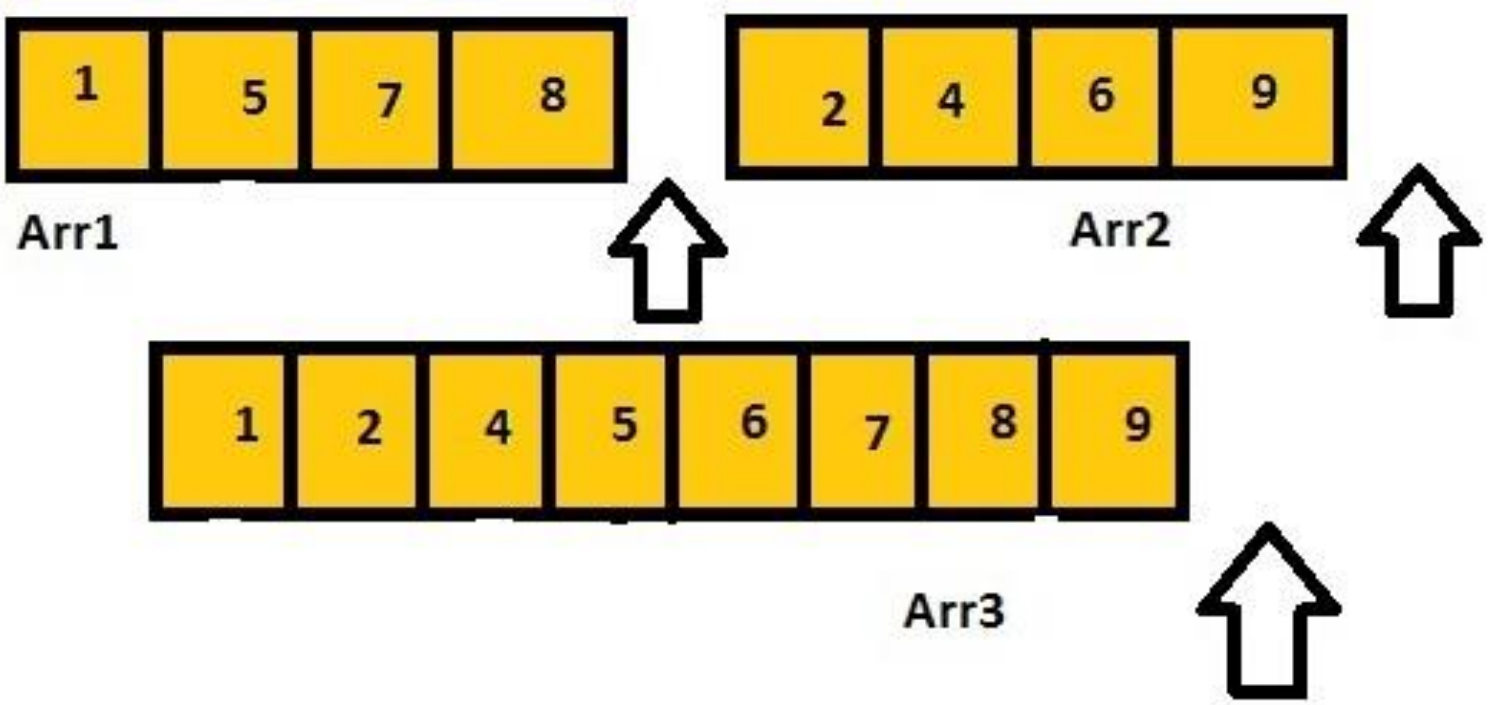
Merge sort



Merge sort



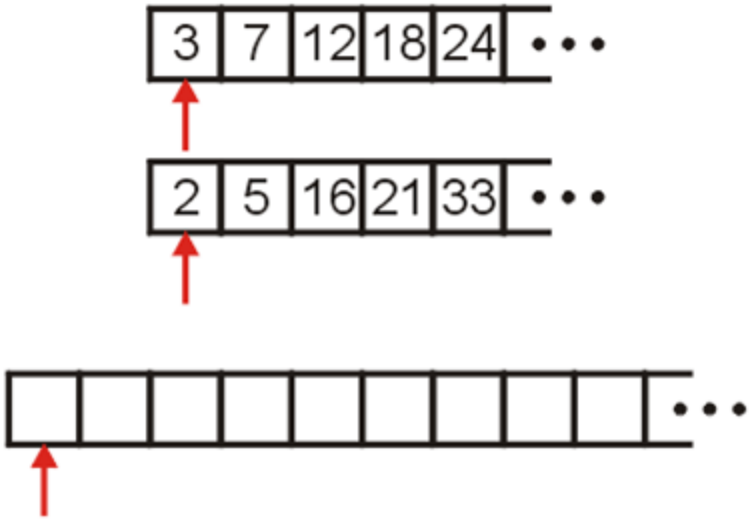
Merge sort



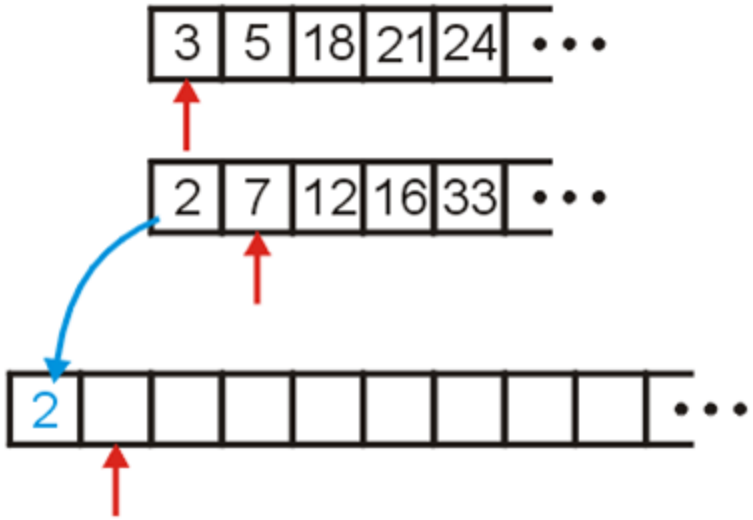
Merge sort

The merging algorithm assumes the two lists are in order.

It creates a third merged list from the two original lists.



We define three references at the front of each array



We keep picking the smallest element and move it to a temporary array, incrementing the corresponding indices.

Merge sort

Merge sort: divide and conquer

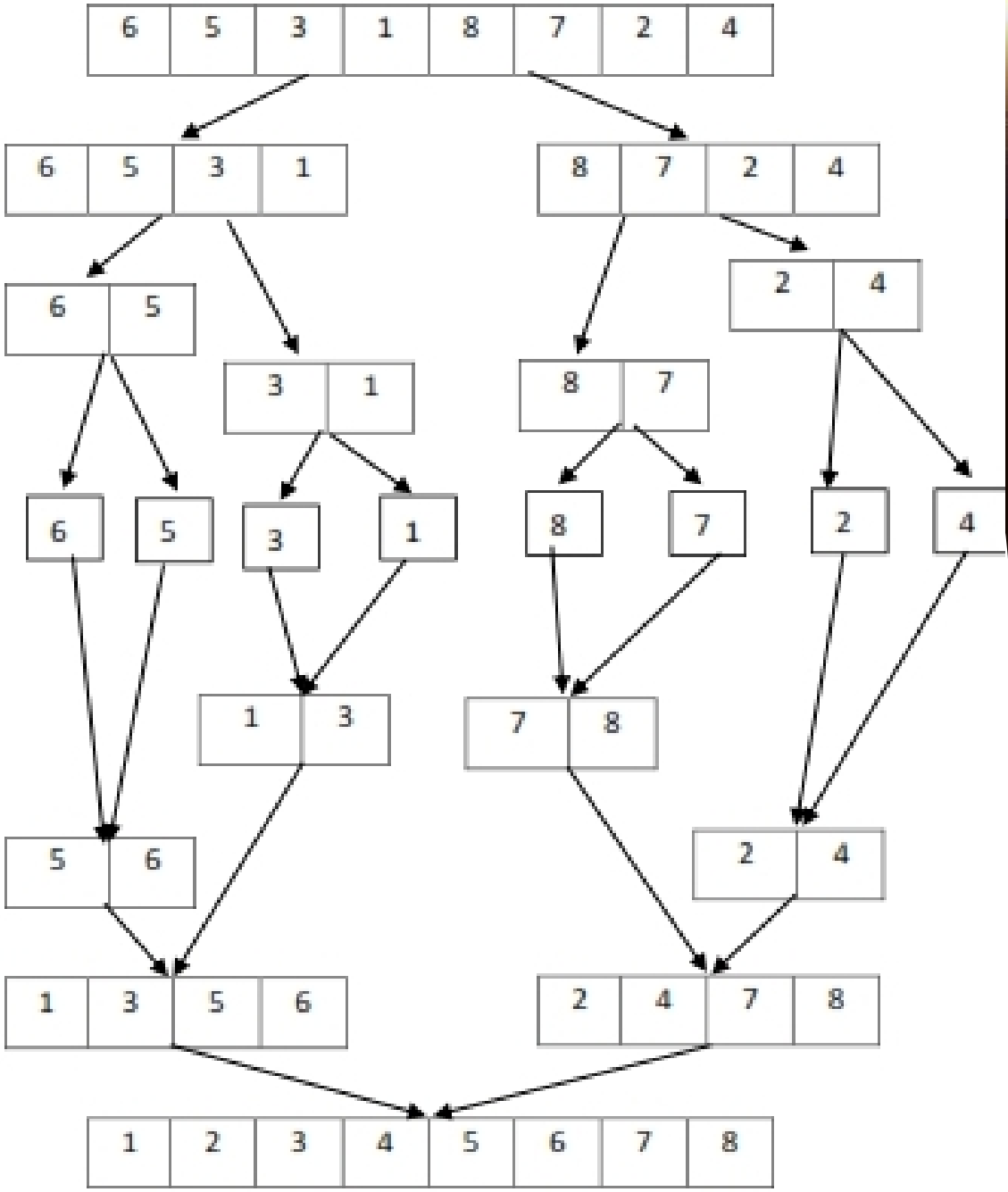
$O(n \log n)$

- if n = 1 ,done
- **recursively** sort A(1,n/2)and A(n/2+1,n)
- merge the two sort list

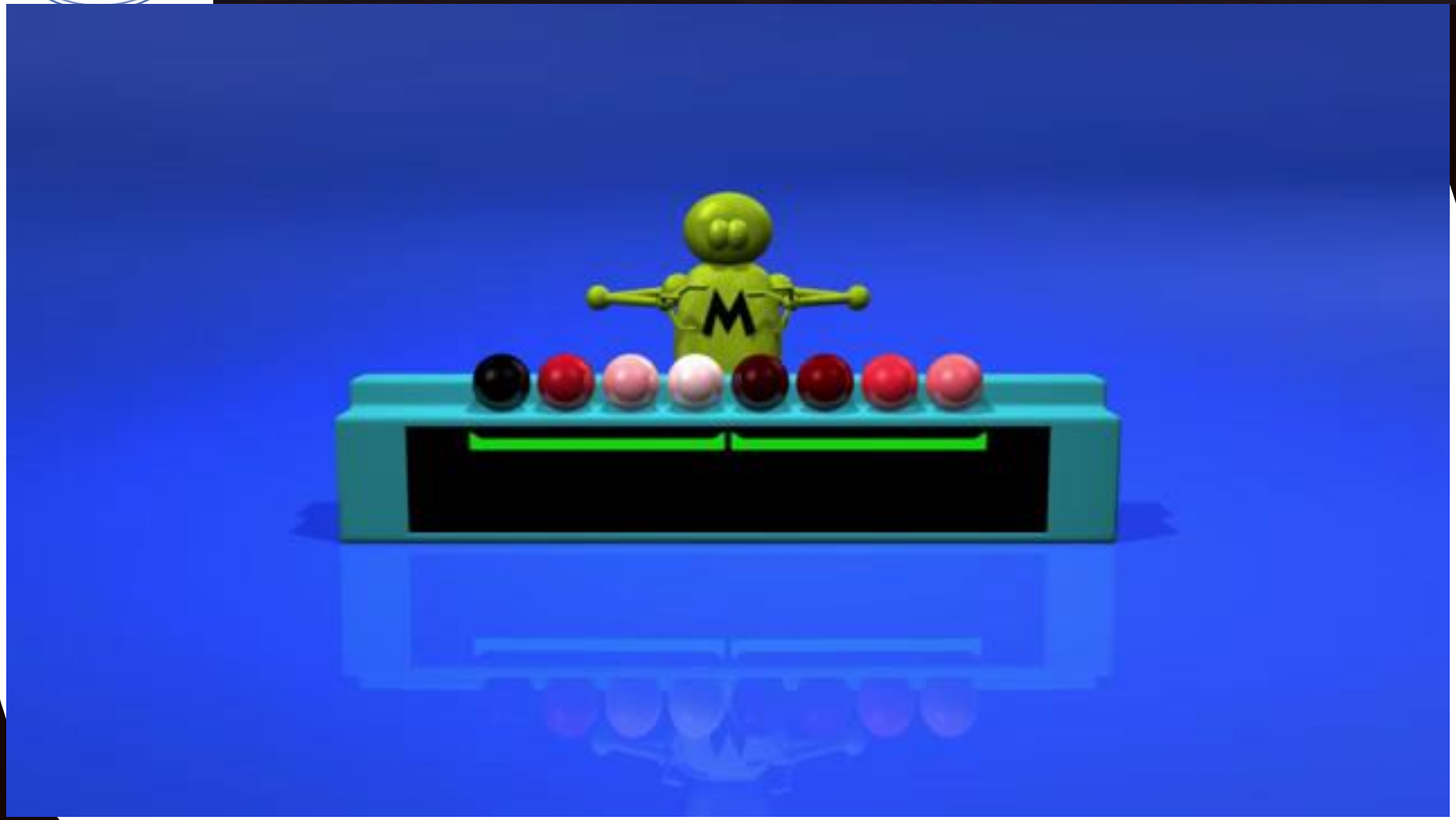
6 5 3 1 8 7 2 4

```
MERGE_SORT( A, low, high )
if low < high then
      mid = ( low+high )/2
      MERGE_SORT( A, low, mid )
      MERGE_SORT( A, mid+1, high )-
      MERGE( A, low, mid, mid+1, high )
END MERGE_SORT
```

6 5 3 1 8 7 2 4


6 5 3 1 8 7 2 4
6 5 3 1
8 7 2 4
6 5
3 1
8 7
2 4
6
5
3
1
8
7
2
4
1 3
7 8
5 6
2 4
1 3 5 6
2 4 7 8
1 2 3 4 5 6 7 8

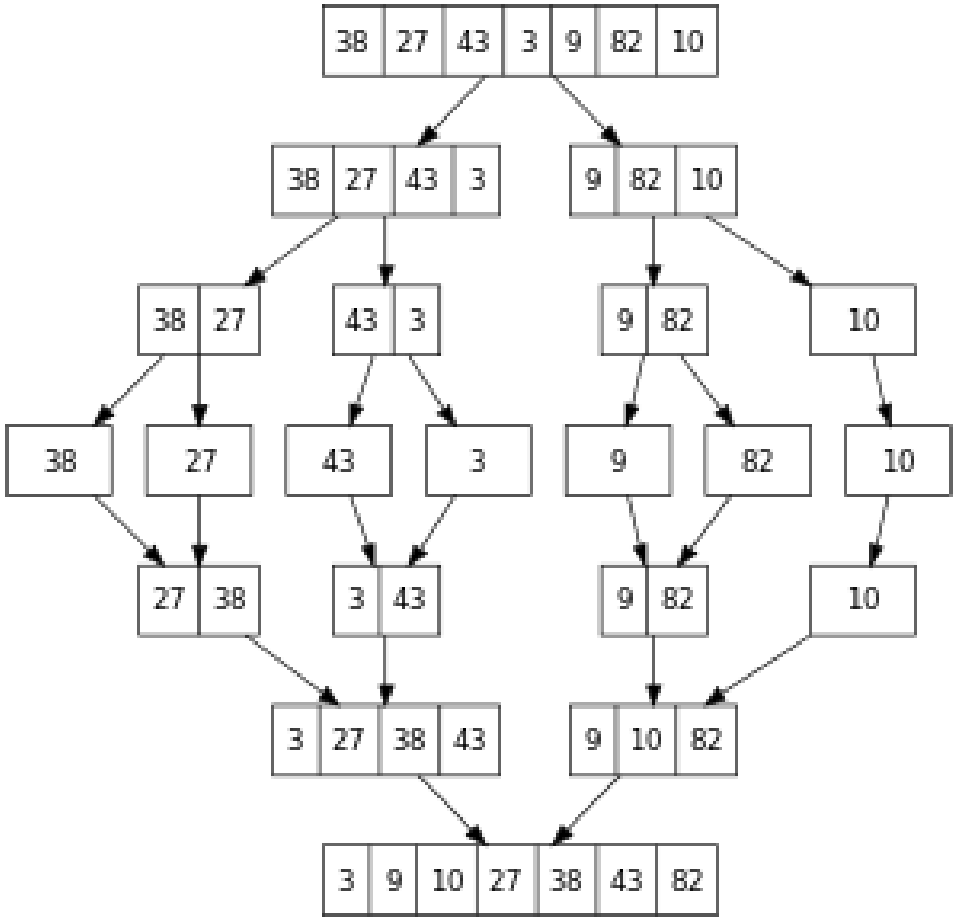
Merge sort

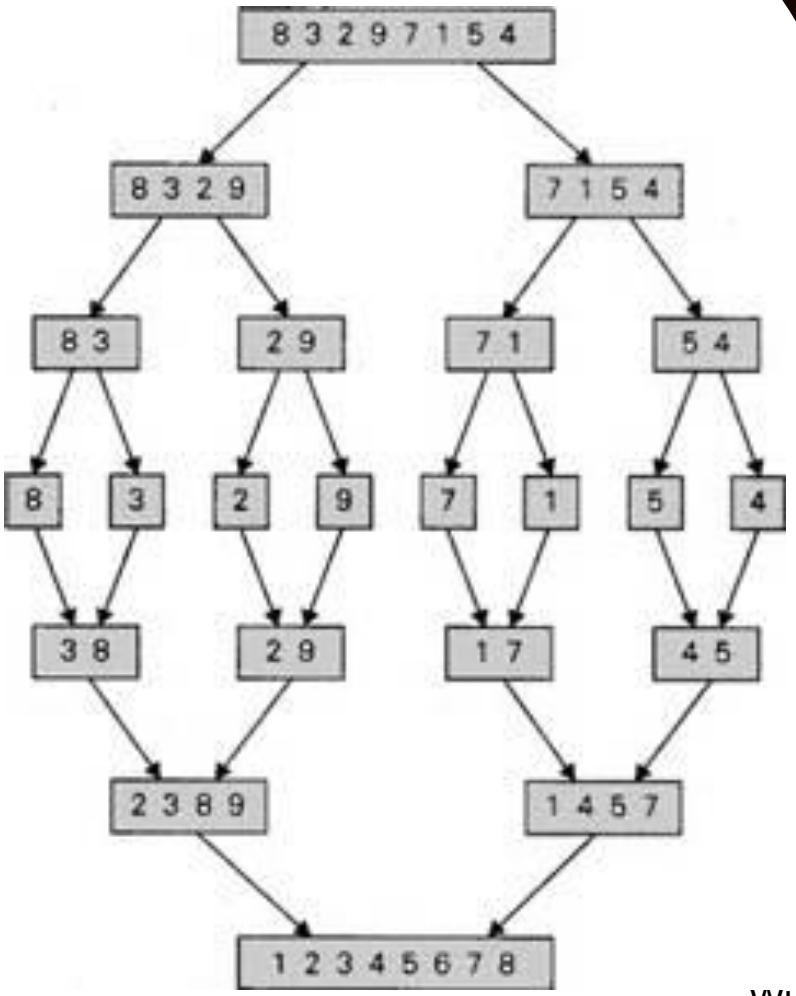


Merge sort

Merge sort: divide and conquer

$O(n \log n)$

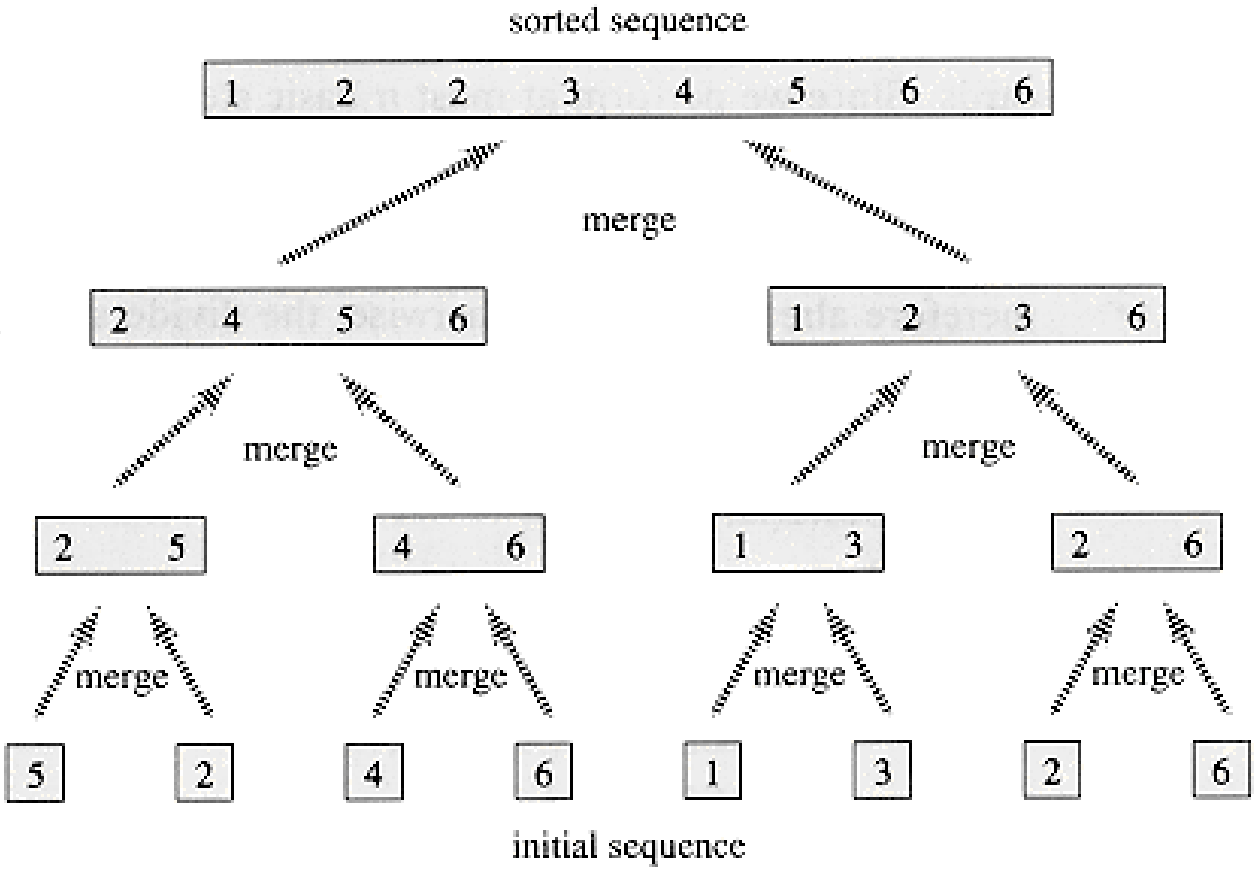




Merge sort

Merge sort: divide and conquer

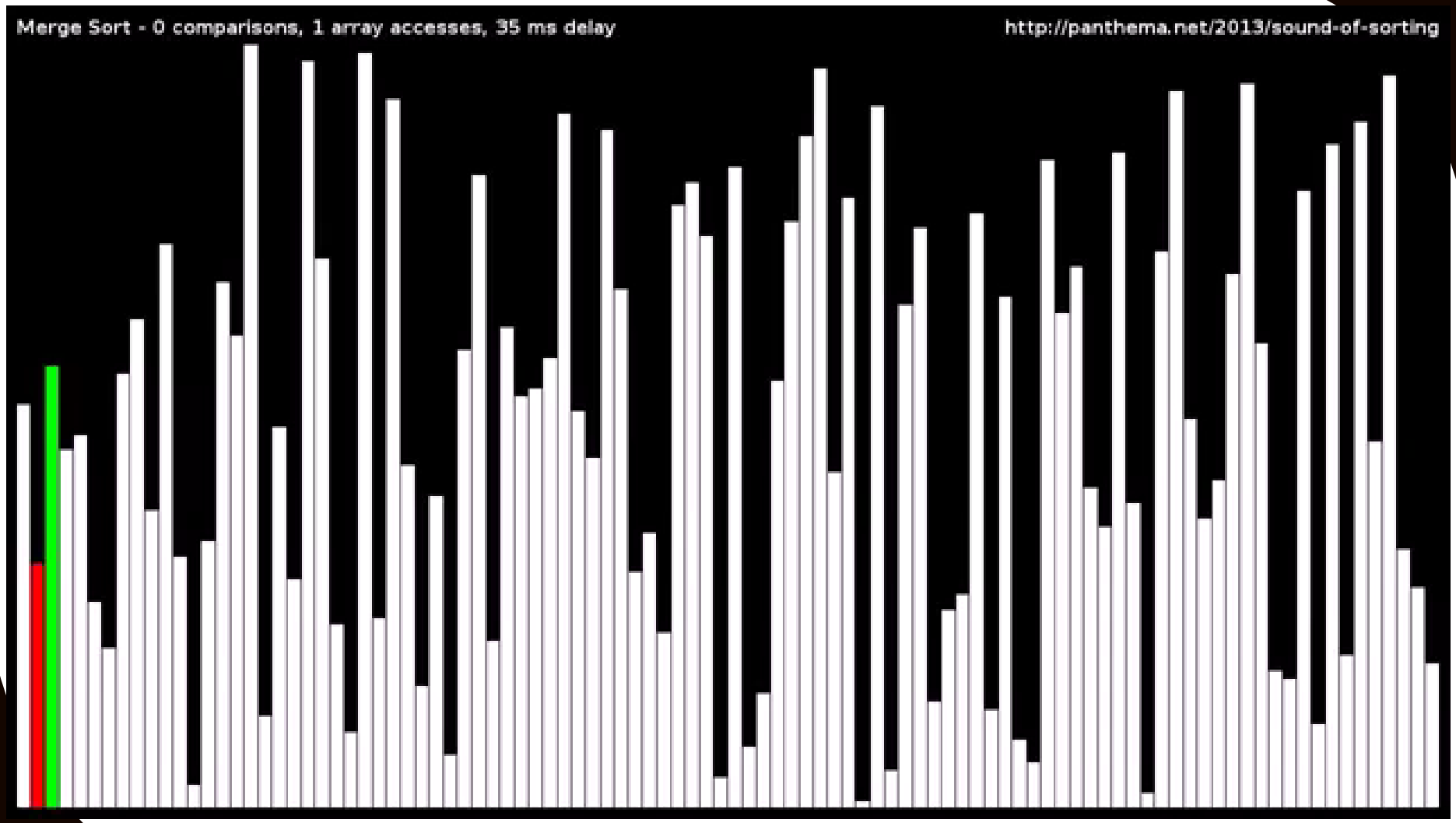
$O(n \log n)$



Merge sort

- http://interactivepython.org/runestone/static/pythonds/SortSearch/TheMergeSort.html
- Step by step running

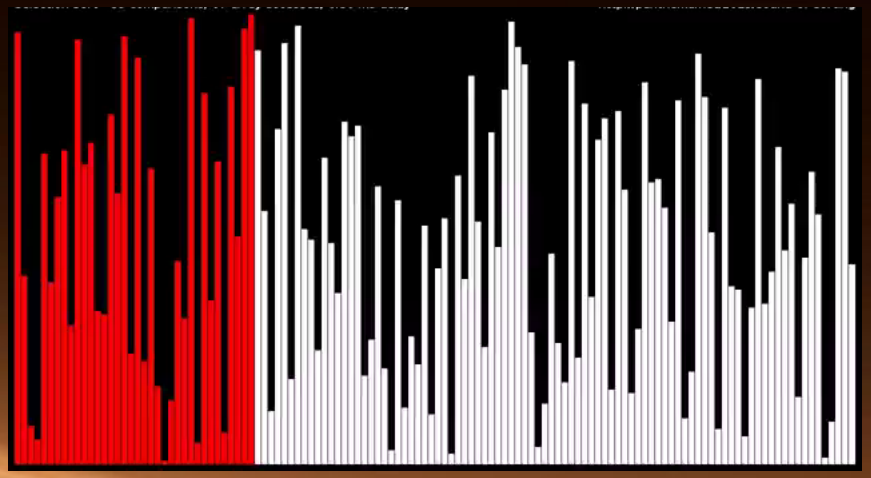
Merge sort

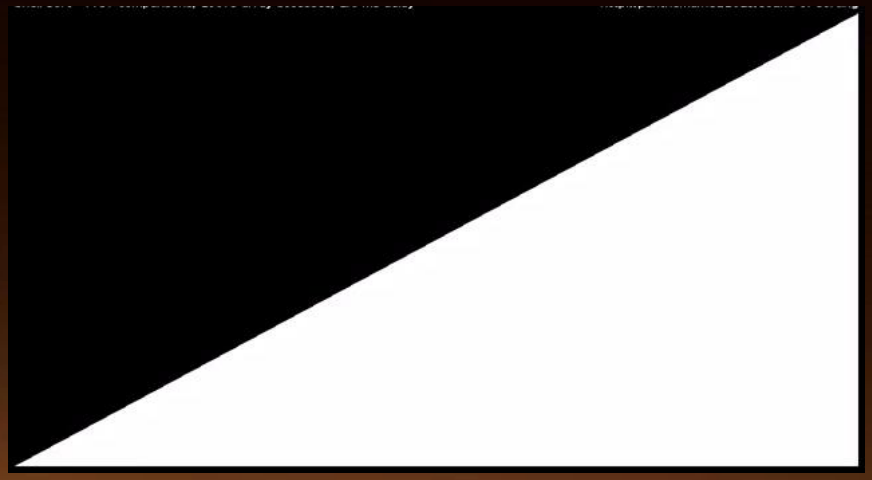

Merge Sort - 0 comparisons, 1 array accesses, 35 ms delay
http://panthema.net/2013/sound-of-sorting

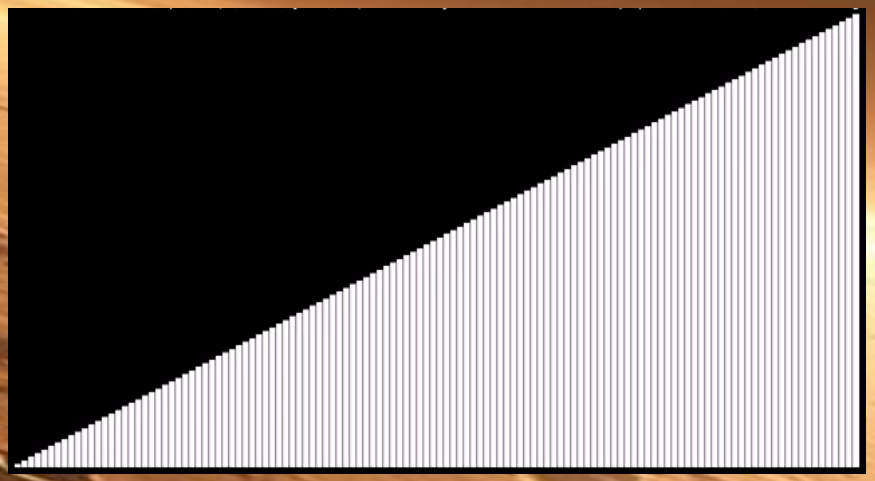
Homework

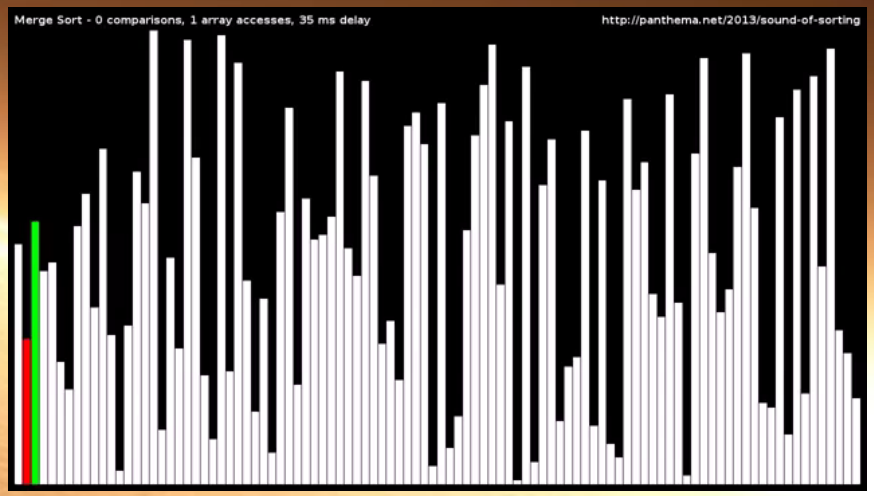
- Complete your handout: exercises
- Implement and test each algorithm by yourself, especially the merge sort. Think about recursion.
- Compare the time complexity of the introduced 4 algorithms, try to plot the running time v.s. Data scale figure using matplotlib

Thank you very much!









CompSci Guide

	Assessment statement	Obj	Teacher's notes
4.2.1	Describe the characteristics of standard algorithms on linear arrays.	2	These are: sequential search, binary search, bubble sort, selection sort.