프로그래밍 나두 할 수 있다!
I CAN DO PROGRAMMING

•••

다섯번째 모임
~Fifth Meeting~

오늘의 주제

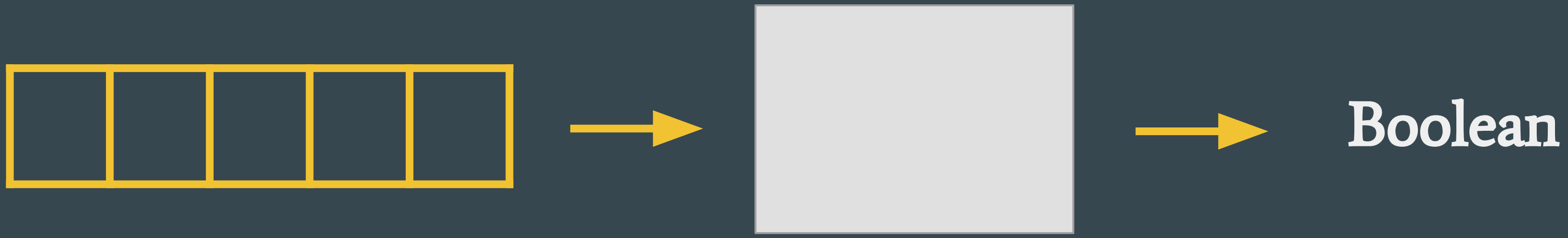
- Search method
  - array
  - Running times
  - Linear search
  - Binary search

Array 기억을 되살려 봅시다

Computer can’t just see all the memory at glance

It has to go through all of the memories one by one either starting from the left or right or even from the middle

We develop some search algorithms to make it effeicnet and less time consuming


Boolean

Running time

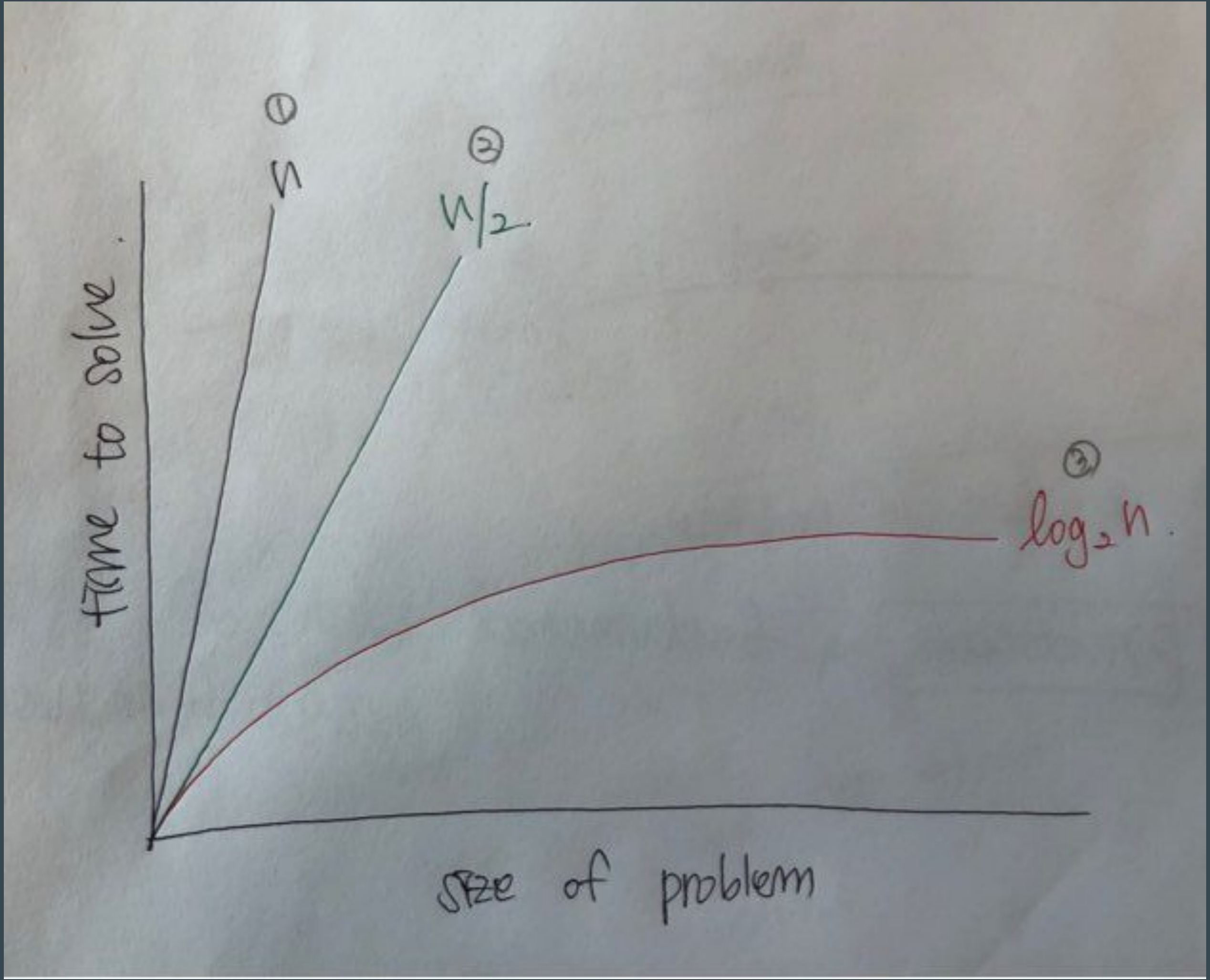
How long it takes

How many steps it takes

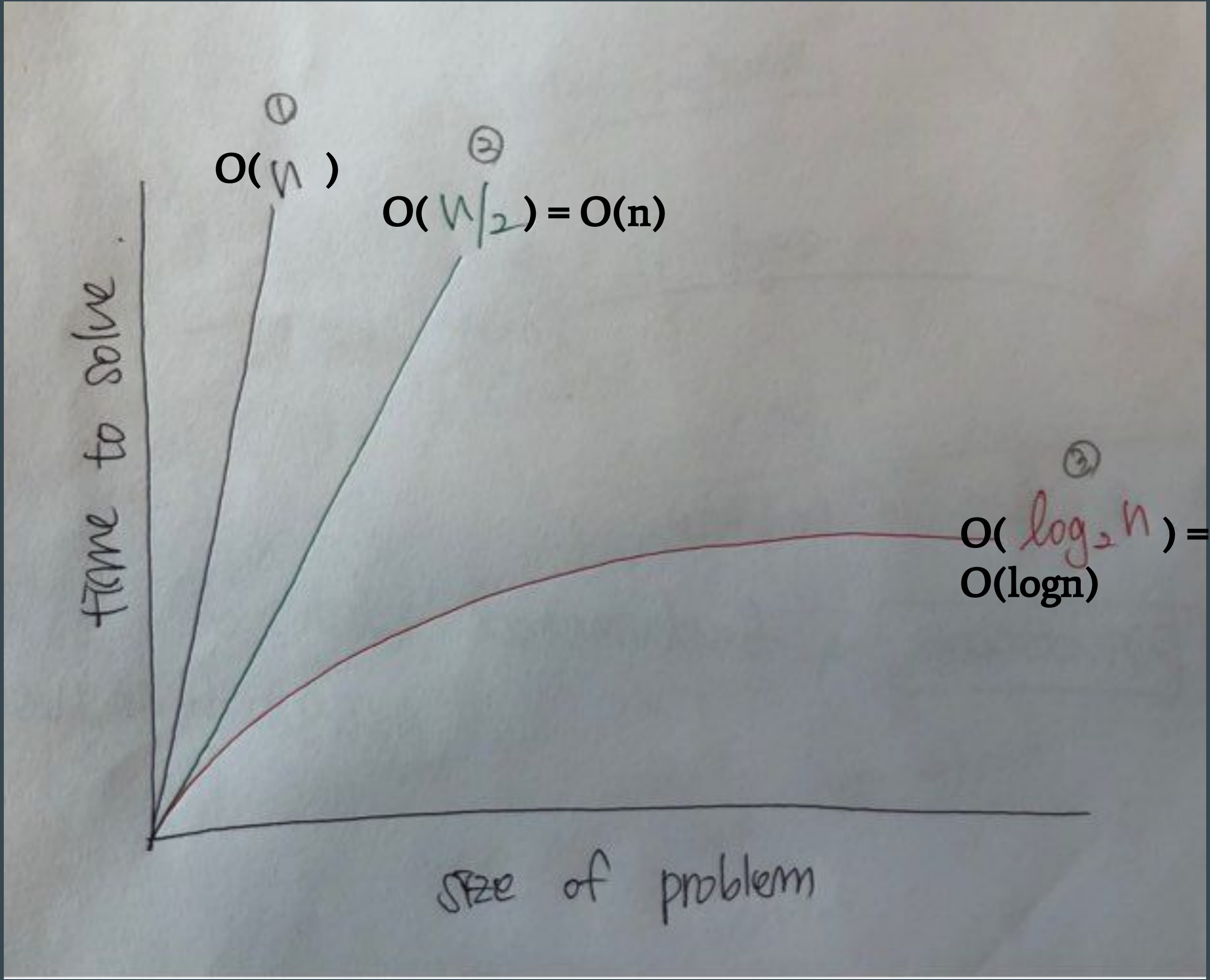
How many iteration it takes

Big O notation to describe the running time of the algorithm

Just roughly how fast it is or how slow it is



①
n
②
n/2
③
log₂ n
time to solve
size of problem


①
O(n)
②
O(n/2) = O(n)
③
O($\log_2 n$) =
O(logn)
time to solve
size of problem

Big O notation

O(n2)

O(n)

O(n log n)

O(log n)

O(1)

O is the upper bound of the running time (worst case scenario)

Omega notation

Ω

The lower bound of the running time

$\Omega(1)$

$\Omega(n)$

$\Omega(\log n)$

$\Omega(n \log n)$

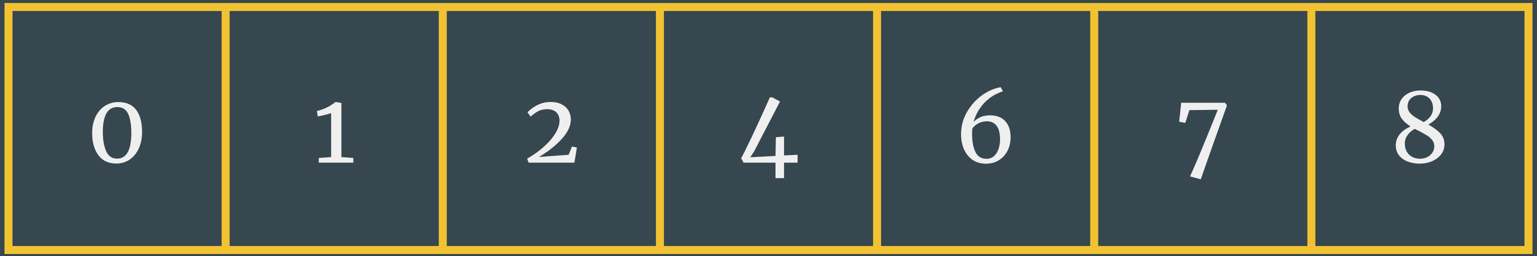
Search

4를 찾아봅시다

2	6	1	8	7	0	4

Linear search

이번엔 4를 어떻게 찾아볼까요?



Pseudo code

```
For i from 0 to n - 1
    If number is at the ith box
        Return true
Return false
```

Algorithmically step by step

Linear search

4를 찾는 가장 효율적인 방법이 무엇인가요? Running time은 무엇인가요?

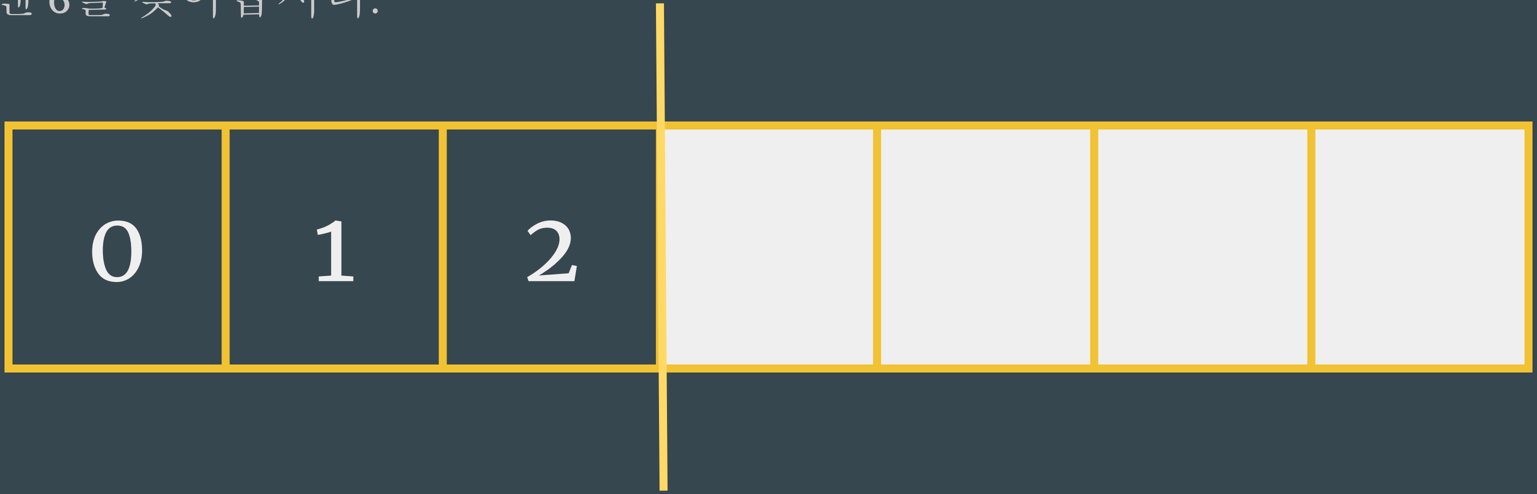
0	1	2	4	6	7	8

What is the worst case scenario? O(n)

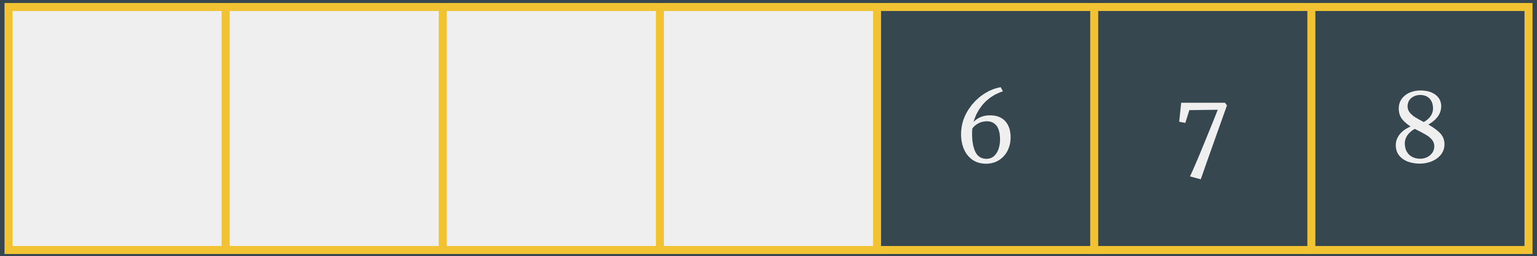
What is the best case scenario? O(1)

Binary search

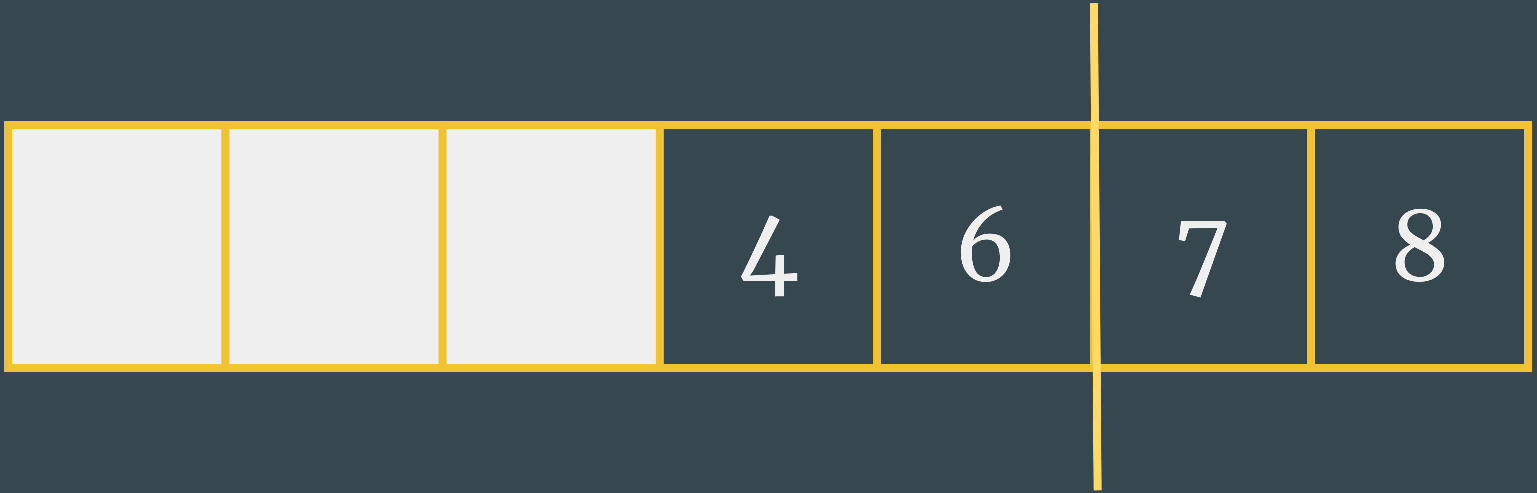
이번엔 6을 찾아봅시다.



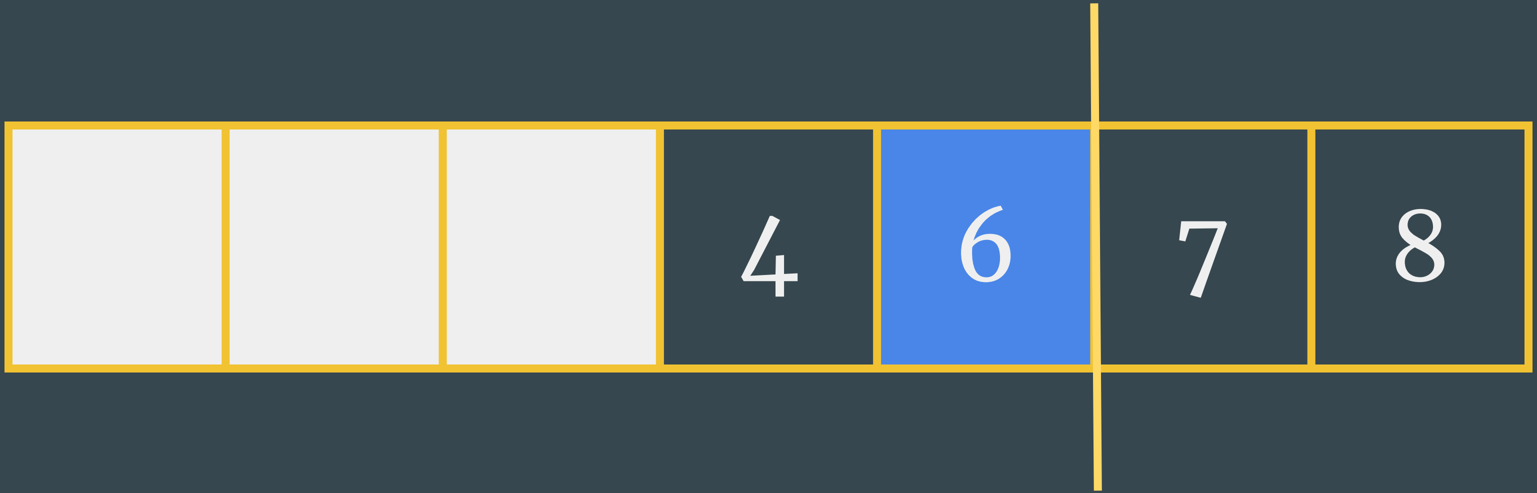
Binary search



Binary search



Binary search



Binary search

만약에 9를 찾고 싶다면 어떻게 찾아야 할까요? 9를 찾을 수 있나요?

0	1	2	4	6	7	8

Binary search

6을 찾는 가장 효율적인 방법이 무엇인가요? Running time?

0	1	2	4	6	7	8

What is the worst case scenario? O(log n)

What is the best case scenario? O(1)

Sort

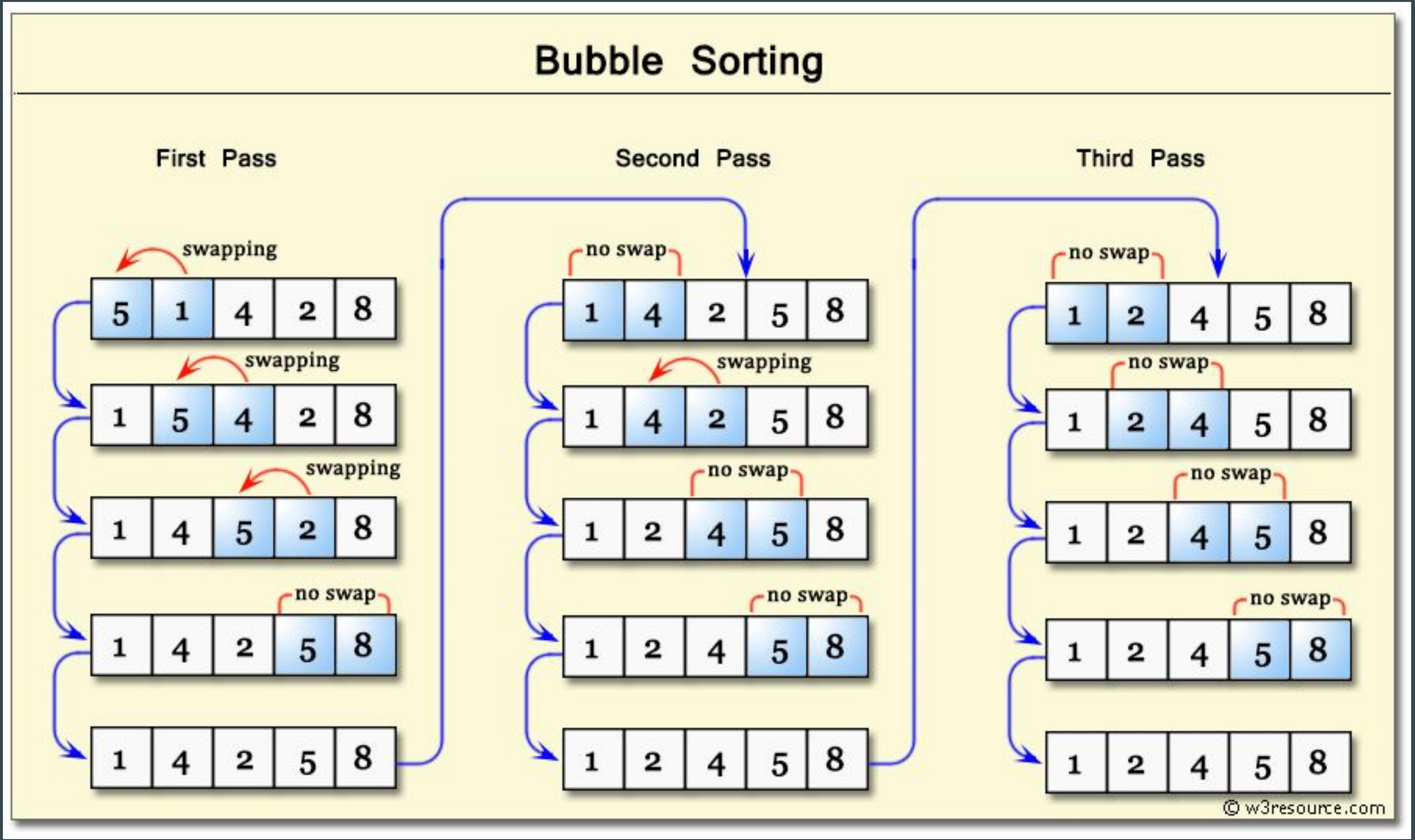
만약 array가 정열되어 있지 않다면 어떻게 문제를 해결할 수 있을까요?

Bubble sort

Insertion sort

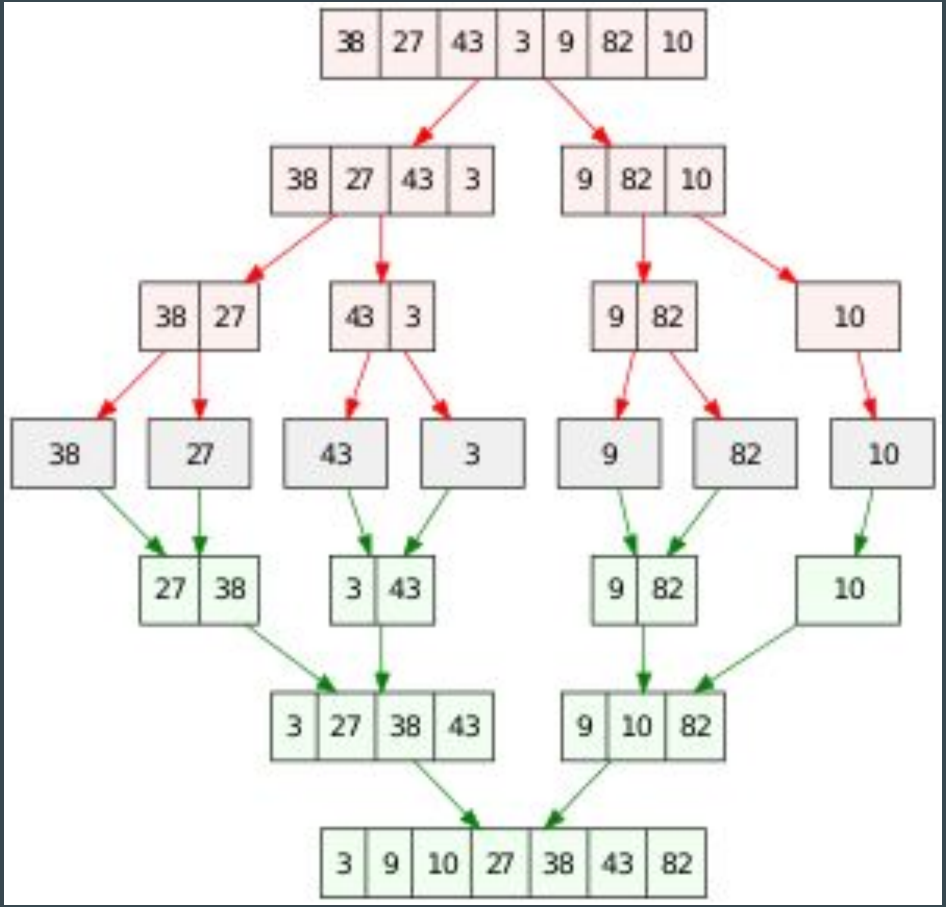
Merge sort

Bubble Sort

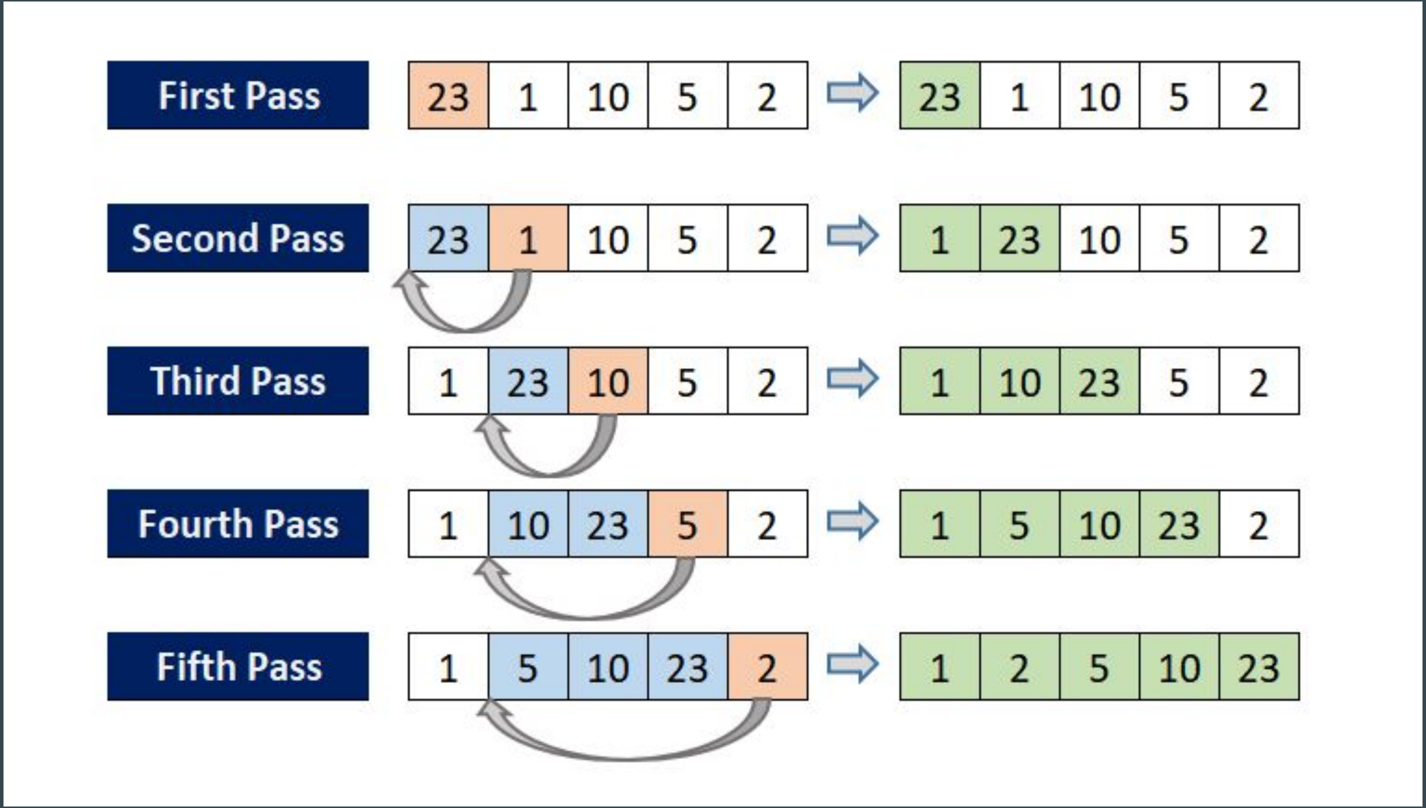


Bubble Sorting
First Pass
Second Pass
Third Pass
swapping
no swap
5 1 4 2 8
1 5 4 2 8
1 4 5 2 8
1 4 2 5 8
1 4 2 5 8
1 4 2 5 8
1 2 4 5 8
1 2 4 5 8
1 2 4 5 8
© w3resource.com

Merge Sort


38 27 43 3 9 82 10
38 27 43 3
9 82 10
38 27
43 3
9 82
10
38
27
43
3
9
82
10
27 38
3 43
9 82
10
3 27 38 43
9 10 82
3 9 10 27 38 43 82

Insertion Sort


First Pass
23 1 10 5 2
23 1 10 5 2
Second Pass
23 1 10 5 2
1 23 10 5 2
Third Pass
1 23 10 5 2
1 10 23 5 2
Fourth Pass
1 10 23 5 2
1 5 10 23 2
Fifth Pass
1 5 10 23 2
1 2 5 10 23