CS4311
DESIGN AND ANALYSIS OF ALGORITHMS

Tutorial: Solution to Assignment 1

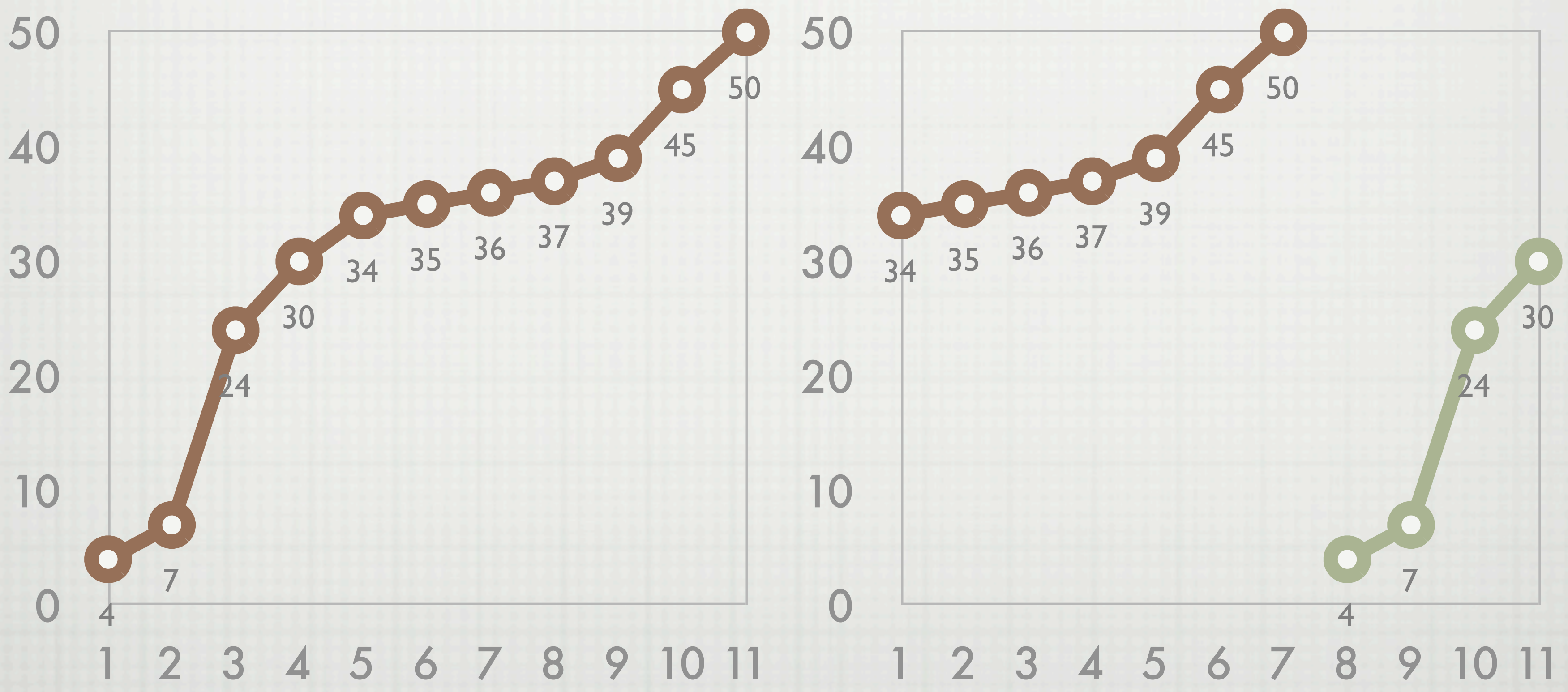
Speaker: 劉富翃 (FOGA)

OUTLINE

- ☐Solution for question 1
- ☐Solution for question 2
- ☐Solution for question 3
- ☐Solution for question 4

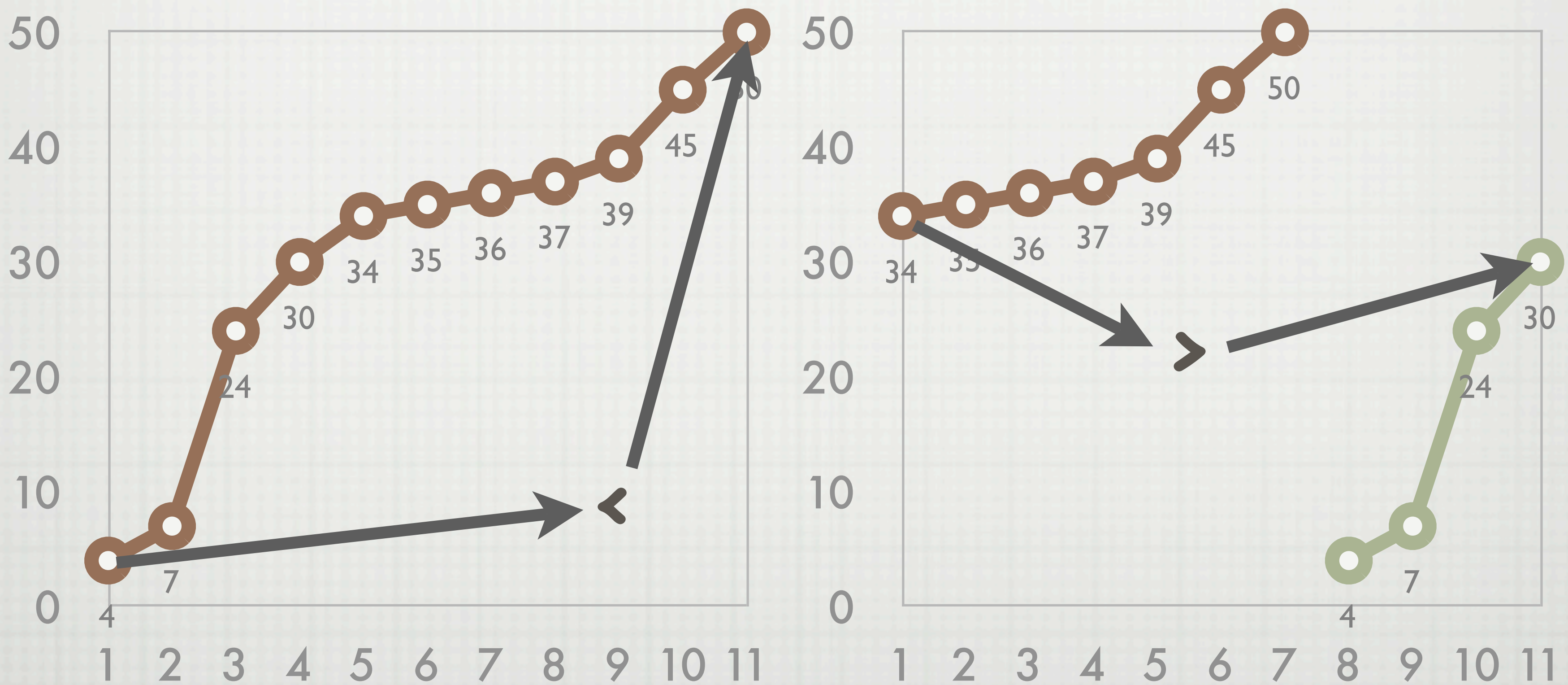
QUESTION 1

- Two situations:



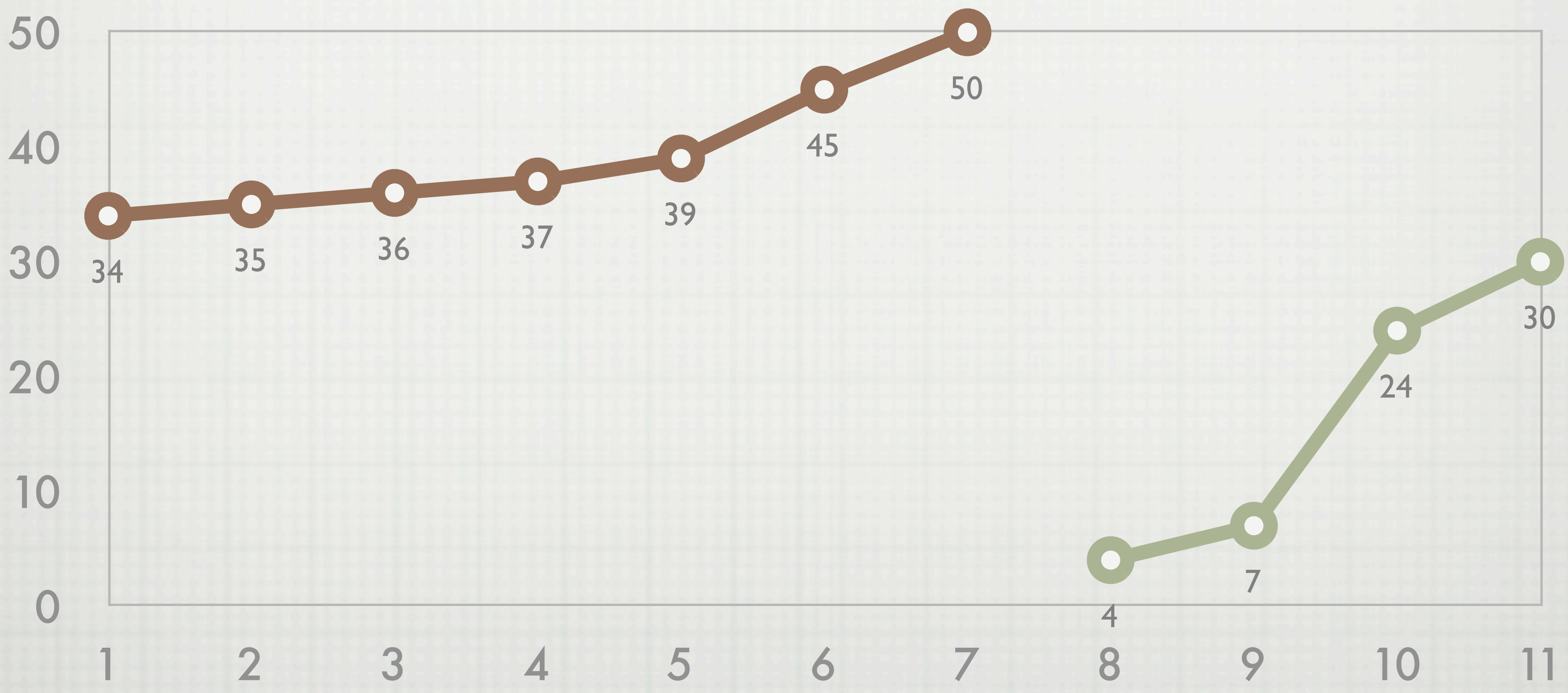
QUESTION 1

- Two situations:



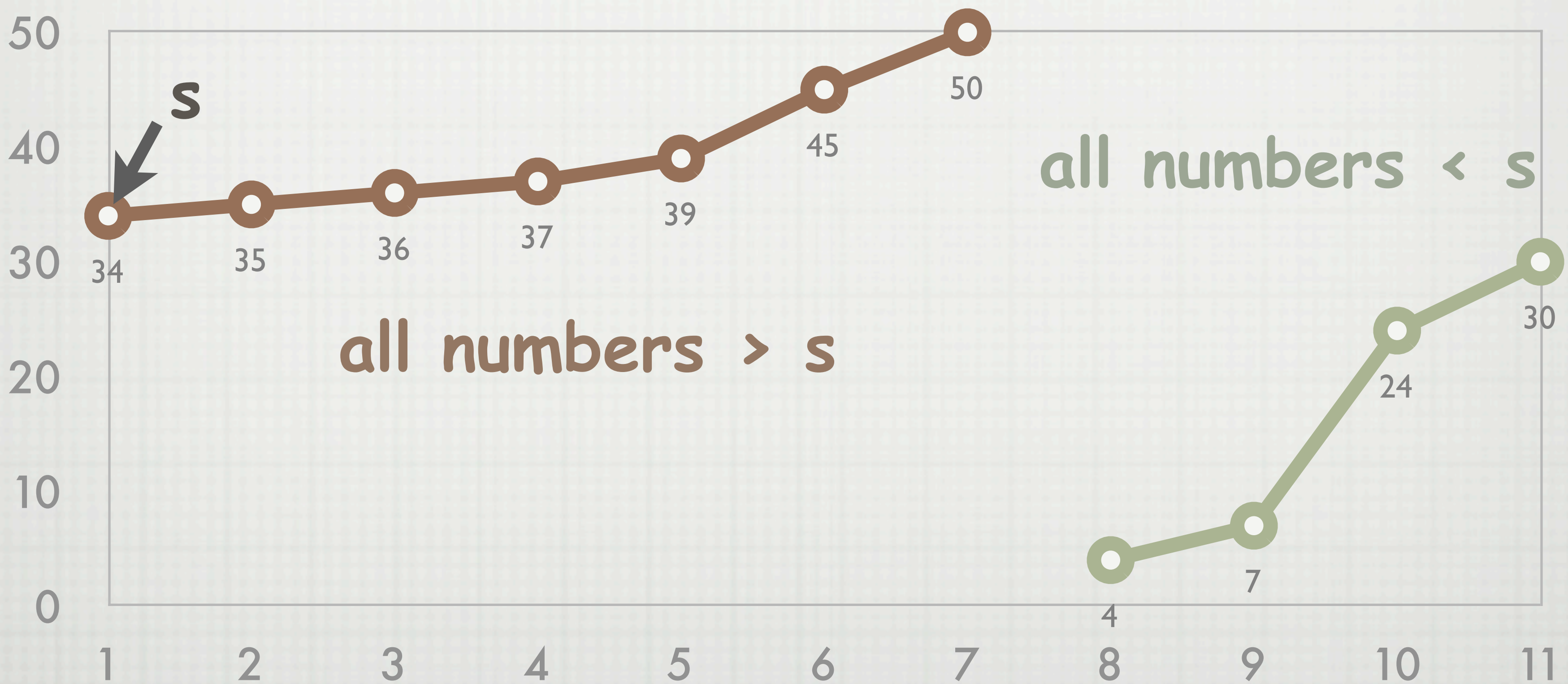
QUESTION 1

- Two cases in 2nd situation:



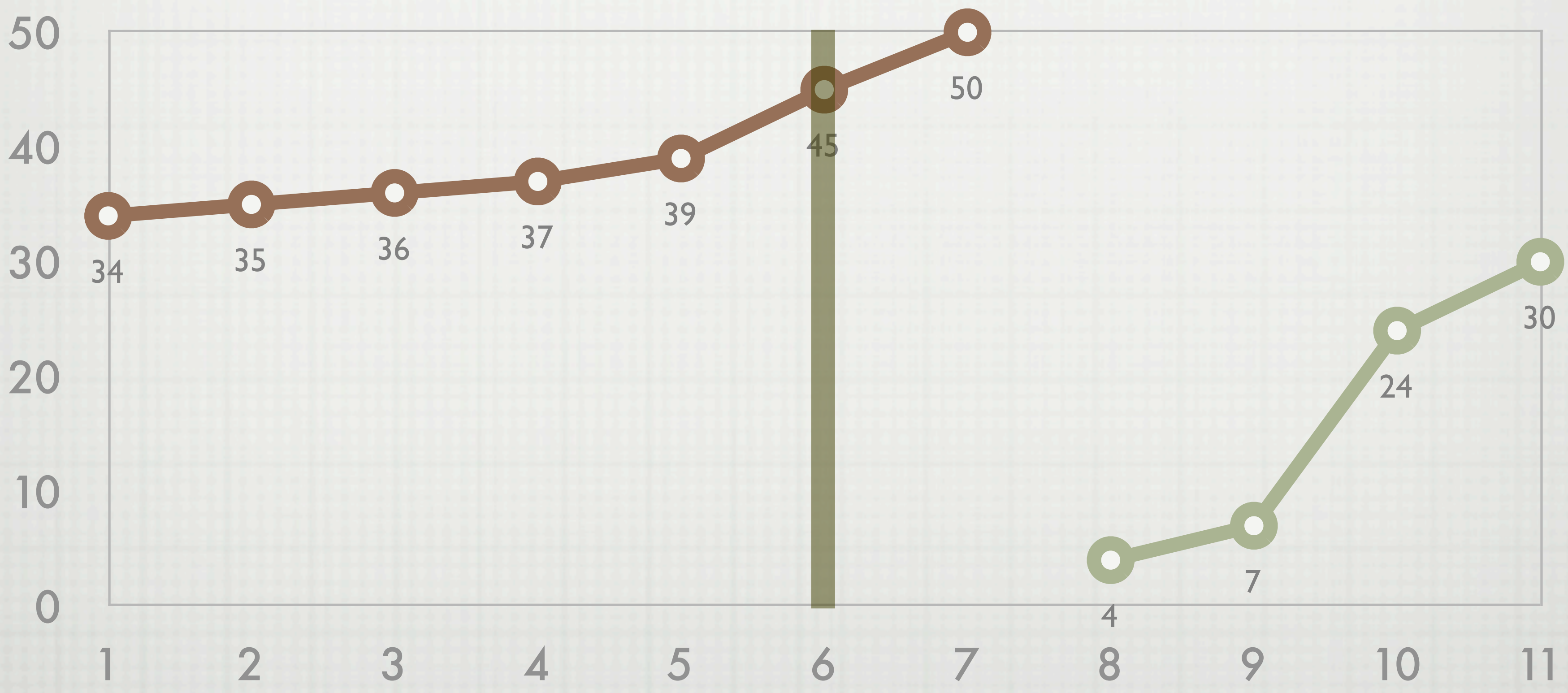
QUESTION 1

- Two cases in 2nd situation:



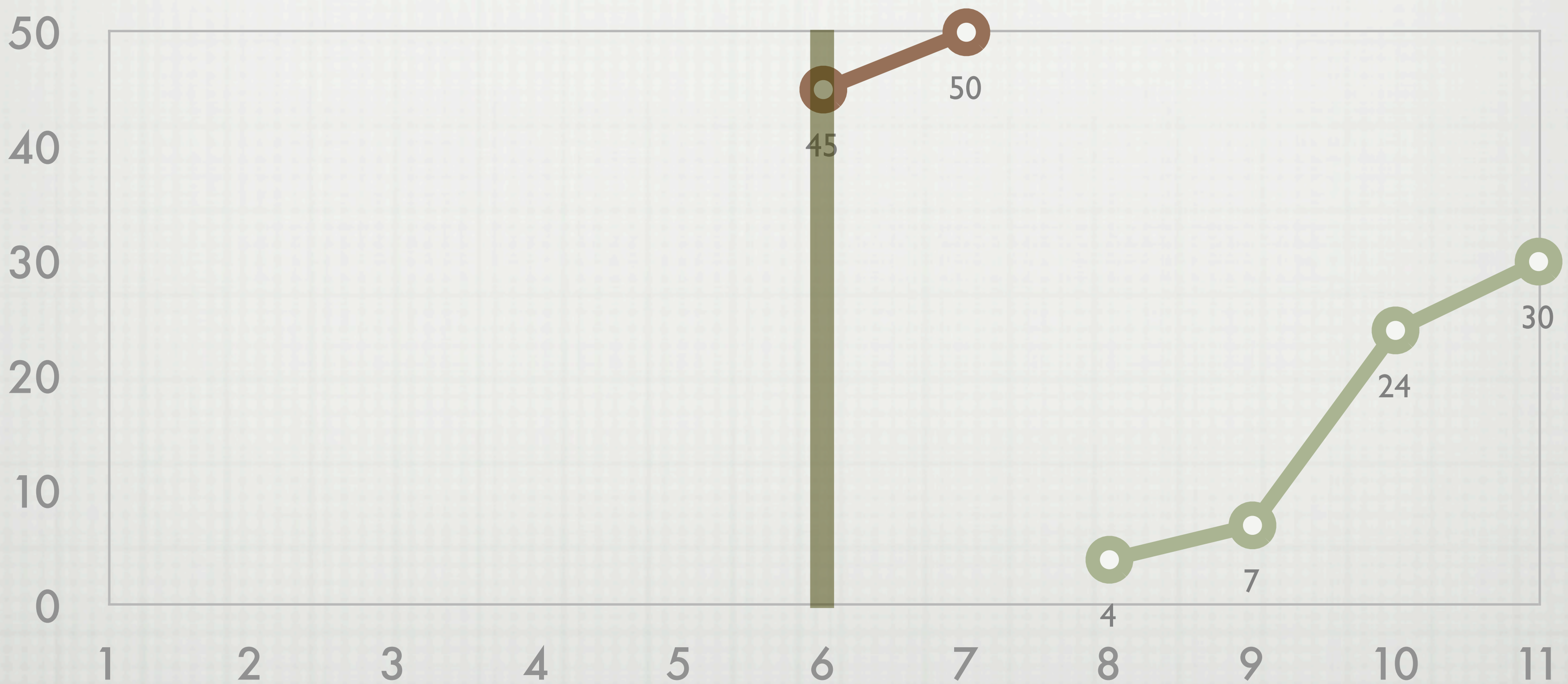
QUESTION 1

- Do modified binary search



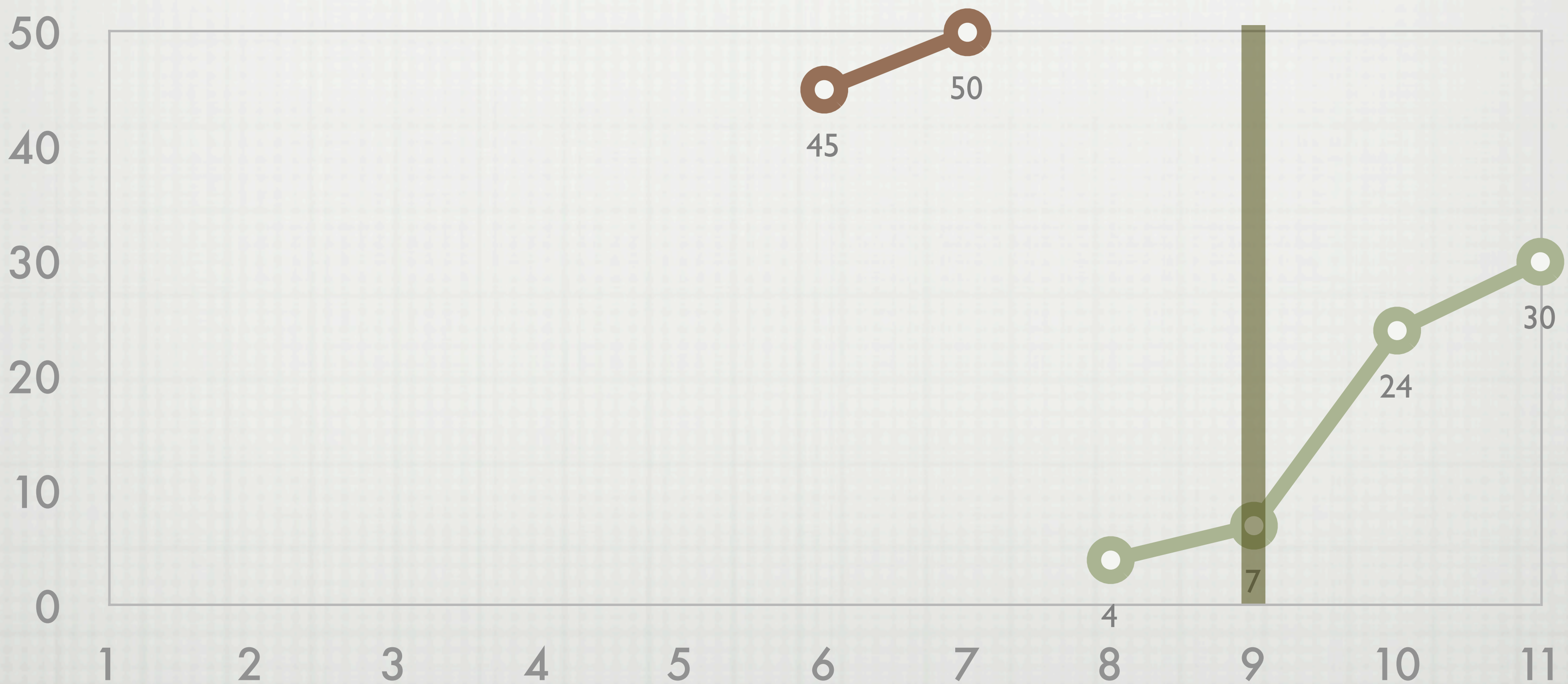
QUESTION 1

- Do modified binary search



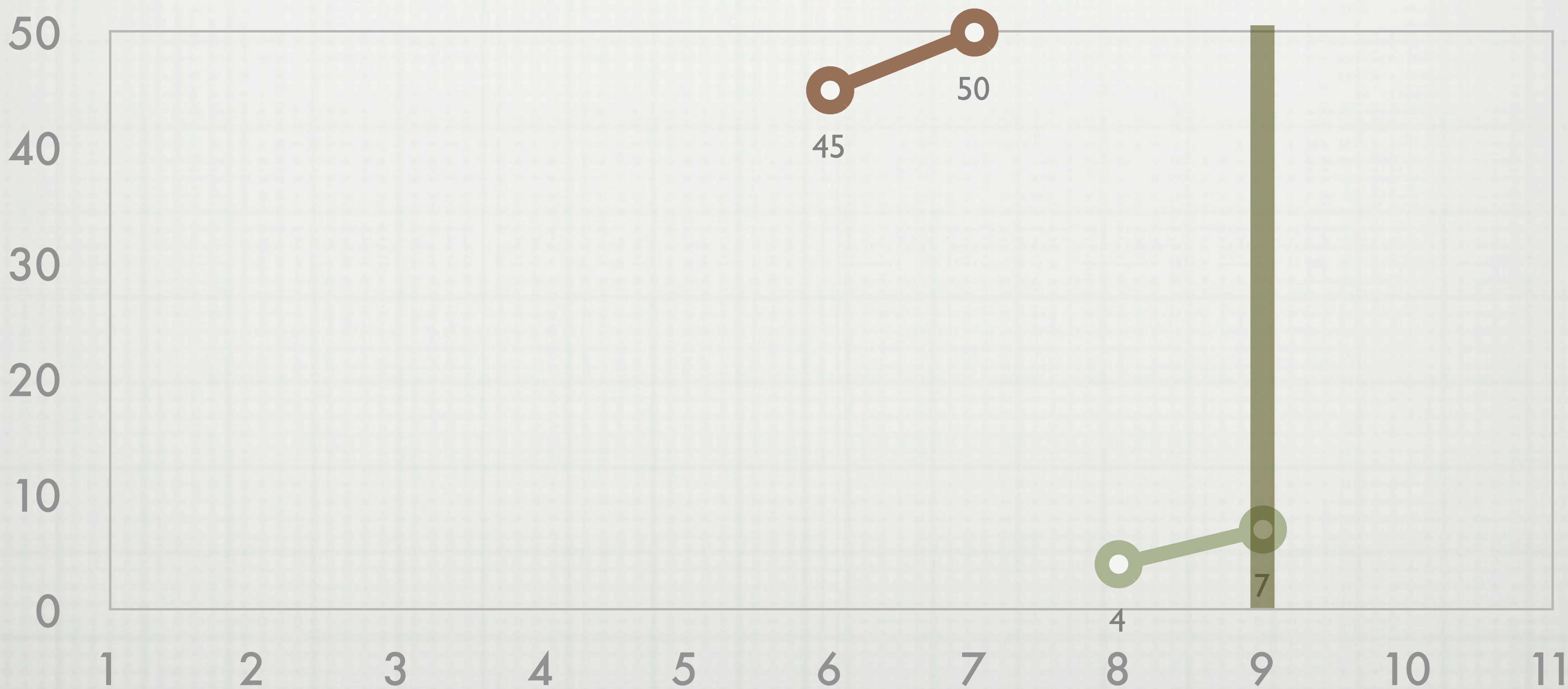
QUESTION 1

- Do modified binary search



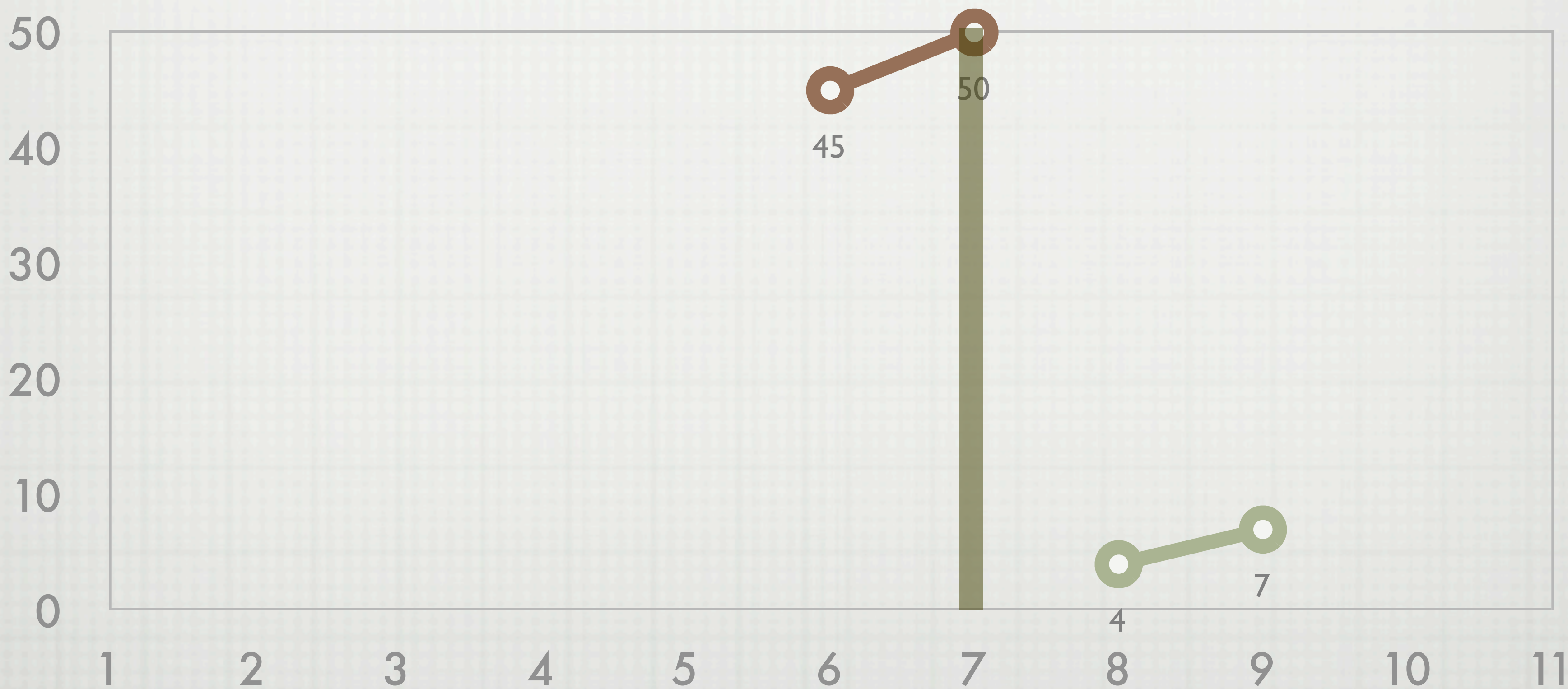
QUESTION 1

- Do modified binary search



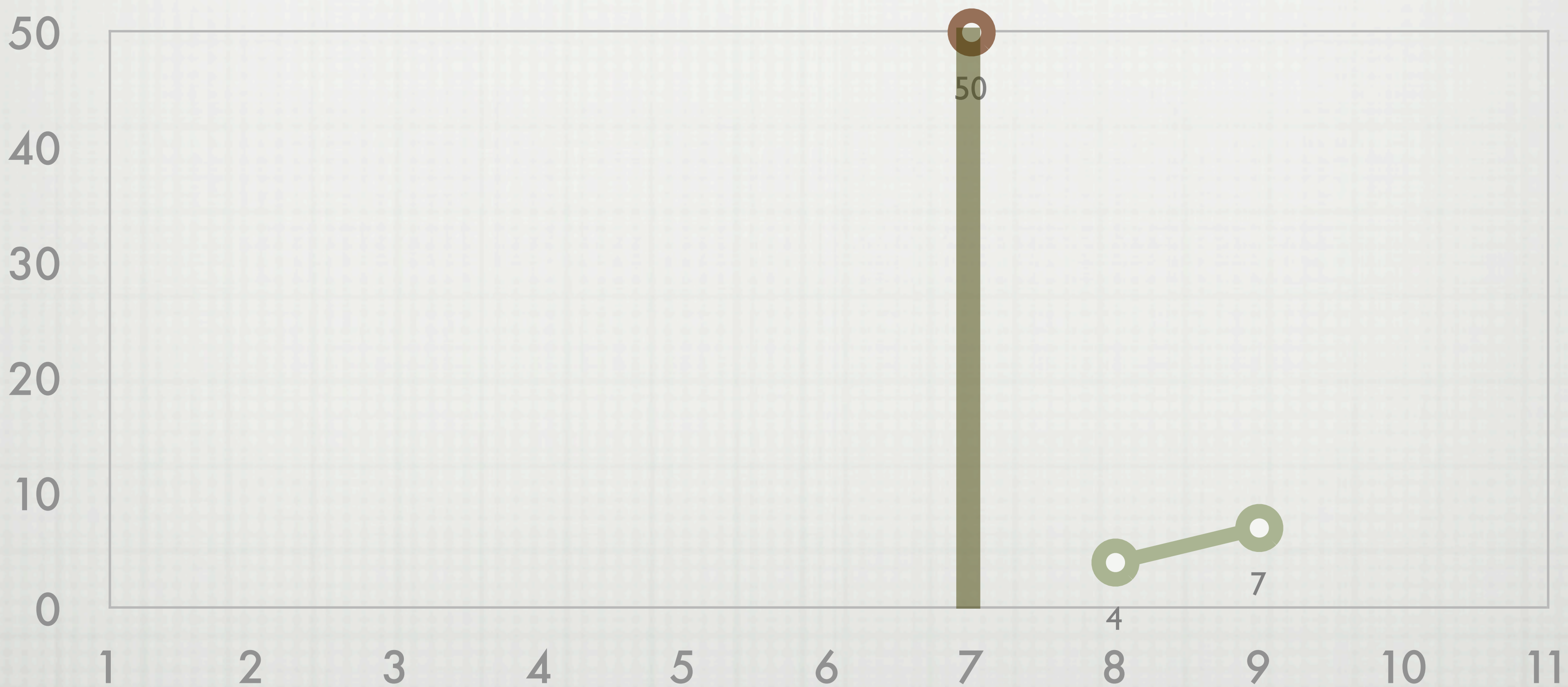
QUESTION 1

- Do modified binary search



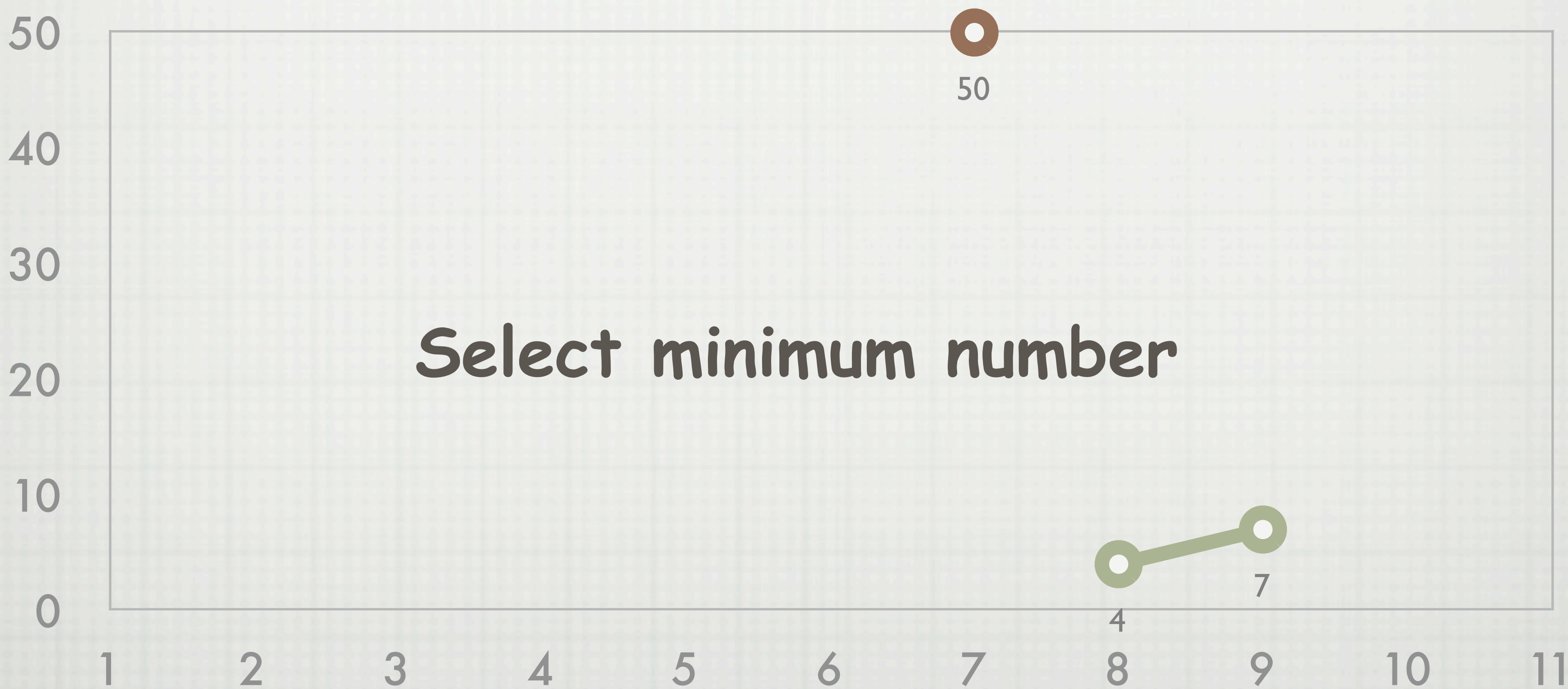
QUESTION 1

- Do modified binary search



QUESTION 1

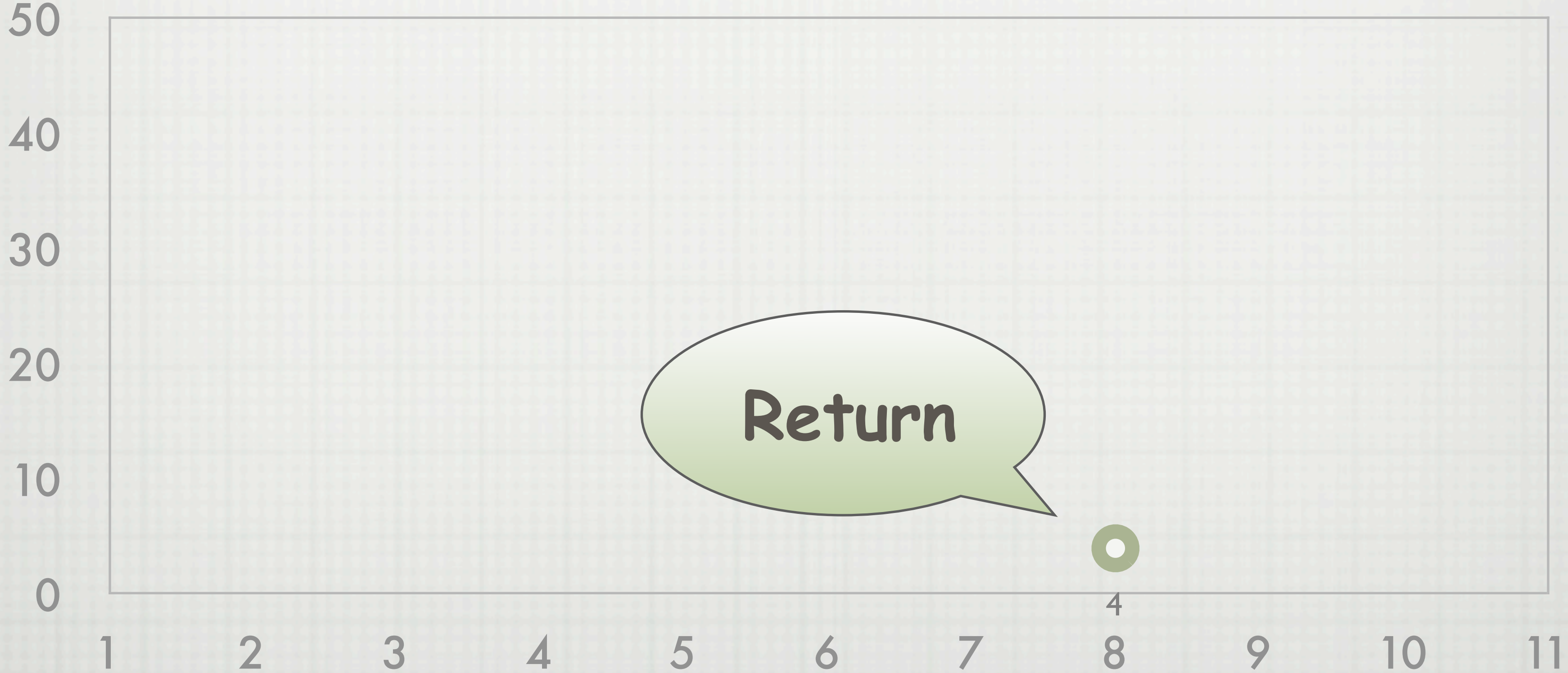
- Do modified binary search



Select minimum number

QUESTION 1

- Do modified binary search



QUESTION 1

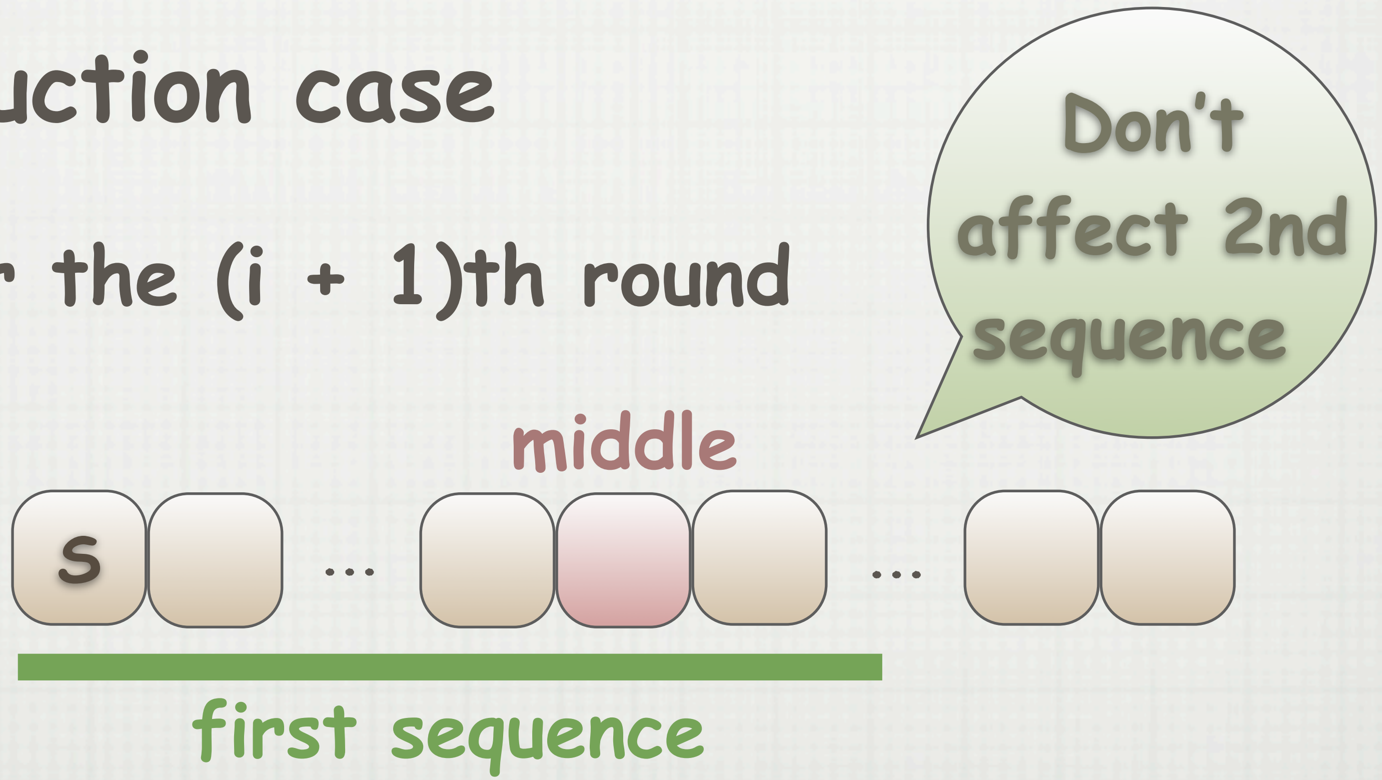
- Prove by induction
- Inductive statement
  - The minimum number is still remained at the end of each round

QUESTION 1

- **Base case**
  - Initially, the minimum number is in the sequence
- **Induction case**
  - Suppose the statement is true for the ith round...

QUESTION 1

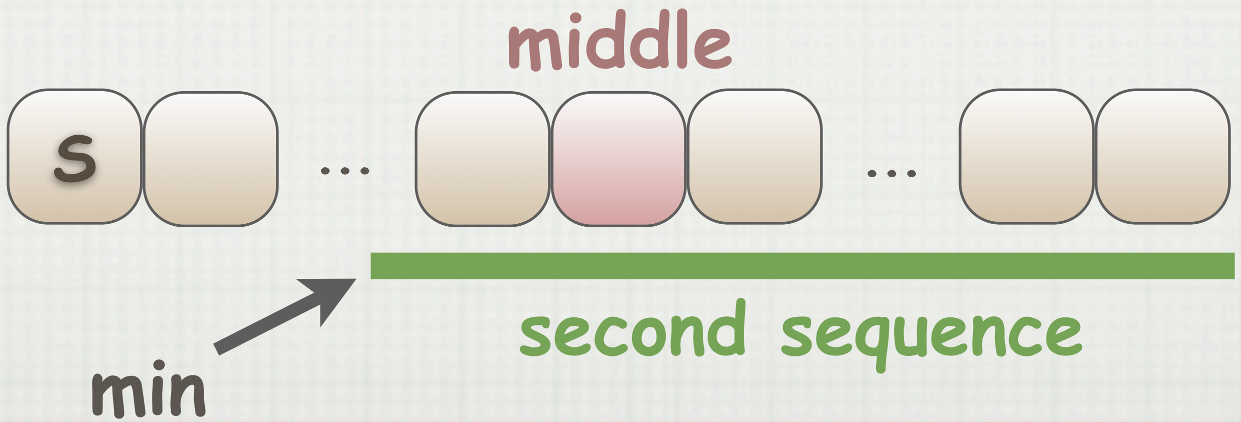
- Induction case
  - For the (i + 1)th round



middle > s

QUESTION 1

- Induction case
  - For the (i + 1)th round



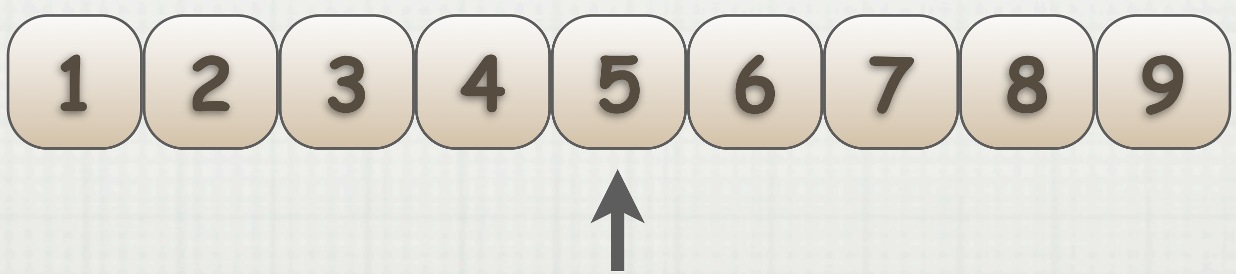
middle < s

QUESTION 2

- **Observation:**
  - count = 0 iff the number of factors is even
  - count = 1 iff the number of factors is odd
- **Physical meaning of count:**
  - count = 0 iff n is not a square number
  - count = 1 iff n is a square number

QUESTION 2

- Do binary search
  - Suppose n = 9



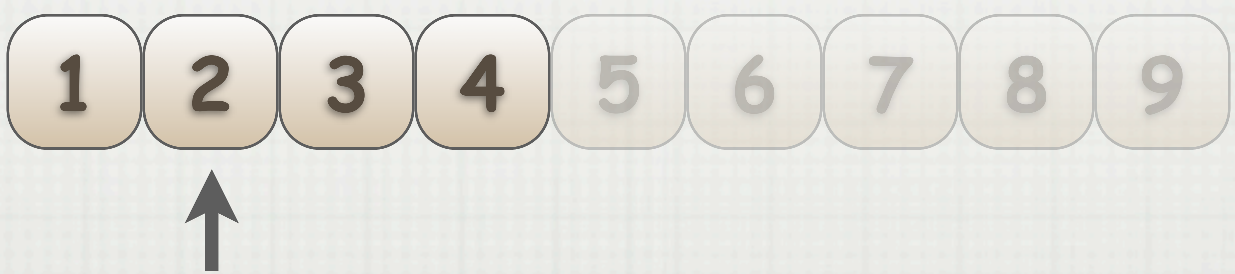
5 * 5 = 25 > 9

QUESTION 2

- Do binary search
  - Suppose n = 9

QUESTION 2

- Do binary search
  - Suppose n = 9



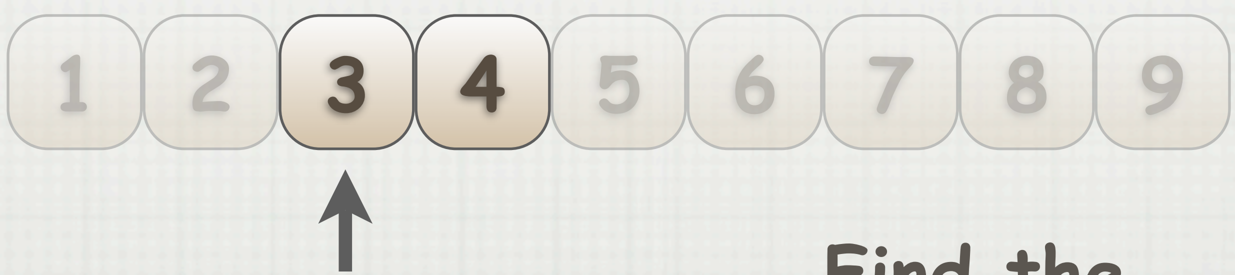
2 * 2 = 4 < 9

QUESTION 2

- Do binary search
  - Suppose n = 9

QUESTION 2

- Do binary search
  - Suppose n = 9



3 * 3 = 9

Find the
square root of n

QUESTION 2

- **The numbers we are concerned is halved in the end of each round, so the time complexity is O(log n).**

QUESTION 2

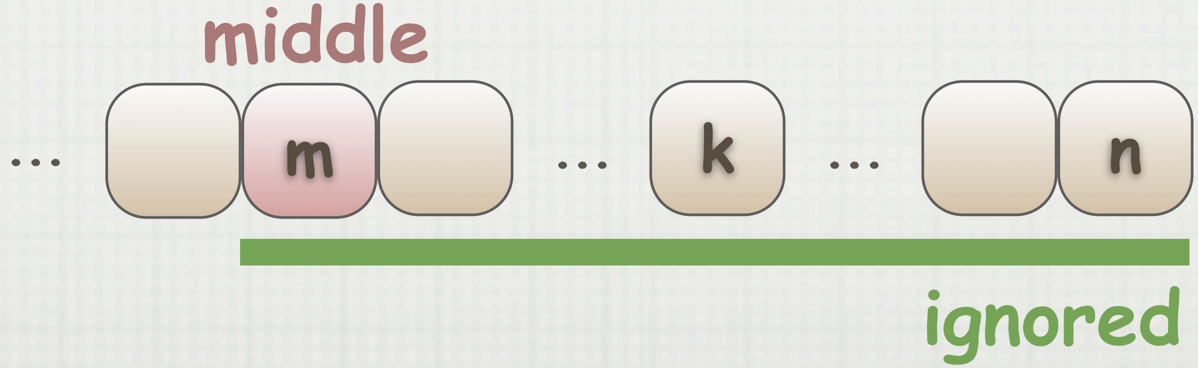
- Prove by contradiction
- Suppose our algorithm isn't correct
  - Our algorithm returns count = 1 and there's no integer k such that $n = k^2$
  - Our algorithm returns count = 0 and there's an integer k such that $n = k^2$

QUESTION 2

- First case
  - If our algorithm outputs count = 1, then there's some k such that $k^2 = n$. Contradiction!

QUESTION 2

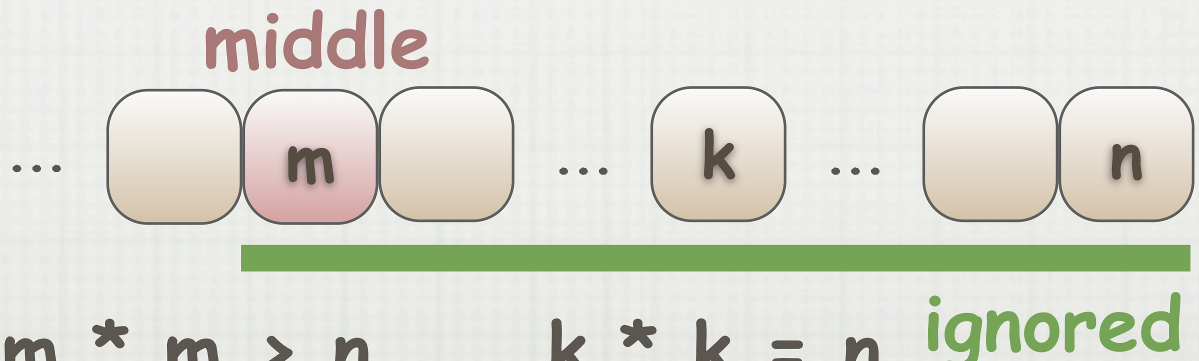
- Second case
  - The integer k must be ignored in some phase, suppose it's ith phase.



QUESTION 2

- Second case

  - The integer k must be ignored in some phase, suppose it's ith phase.



$m * m > n$ $k * k = n$

Contradiction!

QUESTION 3

- Prove by induction
- Inductive statement:
  - At the ith phase, the first to the ith maximum numbers are at the correct position

QUESTION 3

- At each position
  - Select larger number and put it on the right side
- See all positions from left to right
  - The maximum number will be at the current right-most position

QUESTION 3

- Base case
  - For the first phase, since we see all numbers from left to right, the maximum number will be at the right-most position

QUESTION 3

- **Induction case**
  - Suppose the 1st~ith maximum numbers are at correct positions, we'll never do swap on them, and those numbers can be ignored. We still see all the positions from left to right for the remaining sequence, thus the (i + 1)th maximum number will be at the right-most position in the remaining sequence

QUESTION 3

- ☐ Number of inverted pairs <= number of swap operations
- ☐ Number of swap operations <= number of inverted pairs

QUESTION 3

- ☐ If x and y form an inverted pair, then the algorithm swaps x and y once and only once
- ☐ If the algorithm swaps x and y, then x and y form an inverted pair

QUESTION 3

- Forward direction
  - If x and y are a inverted pair, then they must be swapped, otherwise the sequence will not be sorted. After swapping, x and y are not inverted, thus x and y will not swap again.

QUESTION 3

- Backward direction

  - This is just what the algorithm said.

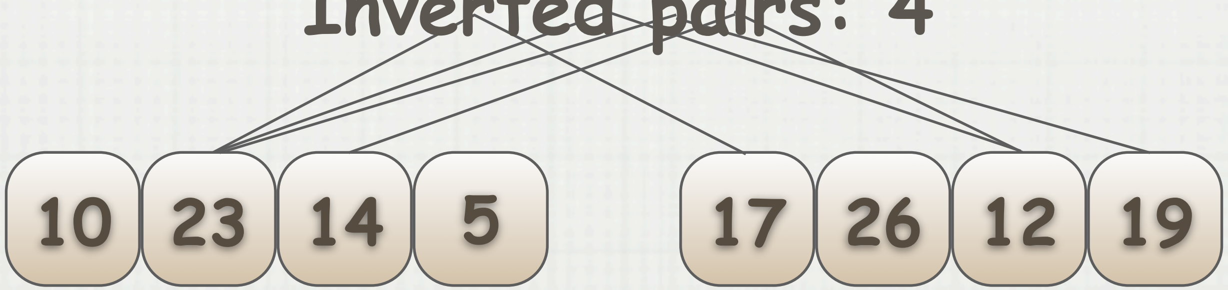
↓

3. **if** $A[j] > A[j+1]$
4. { Swap the entries $A[j]$ and $A[j+1]$; }

QUESTION 3

- Compute inverted pair

Inverted pairs: 4



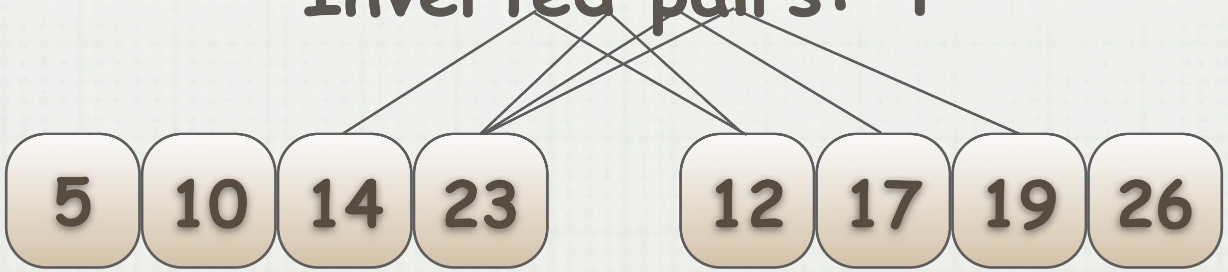
Inverted pairs: 4 Inverted pairs: 3

Total inverted pairs = 4 + 3 + 4 = 11

QUESTION 3

- **Compute inverted pair**

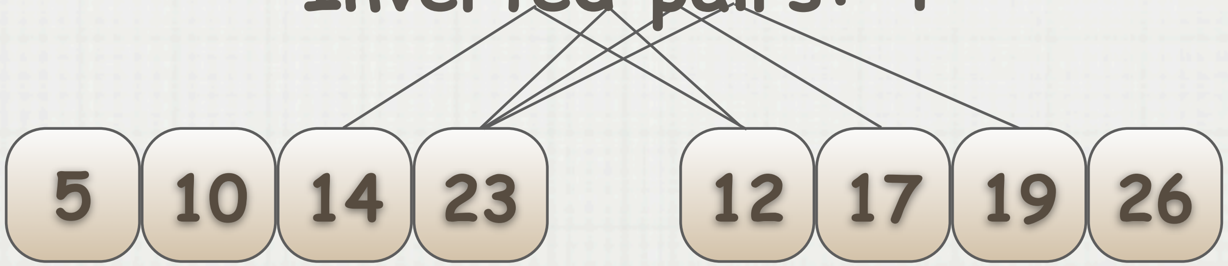
Inverted pairs: 4



QUESTION 3

- Compute inverted pair

Inverted pairs: 4



Inverted pairs: 4

Inverted pairs: 3

QUESTION 3

- Do modified merge

Inverted pairs: 0

5	10	14	23

Inverted pairs: 4

12	17	19	26

Inverted pairs: 3

QUESTION 3

- Do modified merge

Inverted pairs: 0

5

10 14 23

12 17 19 26

Inverted pairs: 4

Inverted pairs: 3

QUESTION 3

- Do modified merge

Inverted pairs: 0

5	10

14	23

12	17	19	26

Inverted pairs: 4

Inverted pairs: 3

QUESTION 3

- Do modified merge

Inverted pairs: 2

5	10	12

14	23

17	19	26

Inverted pairs: 4

Inverted pairs: 3

QUESTION 3

- Do modified merge

Inverted pairs: 2

5	10	12	14

23

17 19 26

Inverted pairs: 4

Inverted pairs: 3

QUESTION 3

- Do modified merge

Inverted pairs: 3

5 10 12 14 17

23

19 26

Inverted pairs: 4

Inverted pairs: 3

QUESTION 3

- Do modified merge

Inverted pairs: 4

5 10 12 14 17 19

23

26

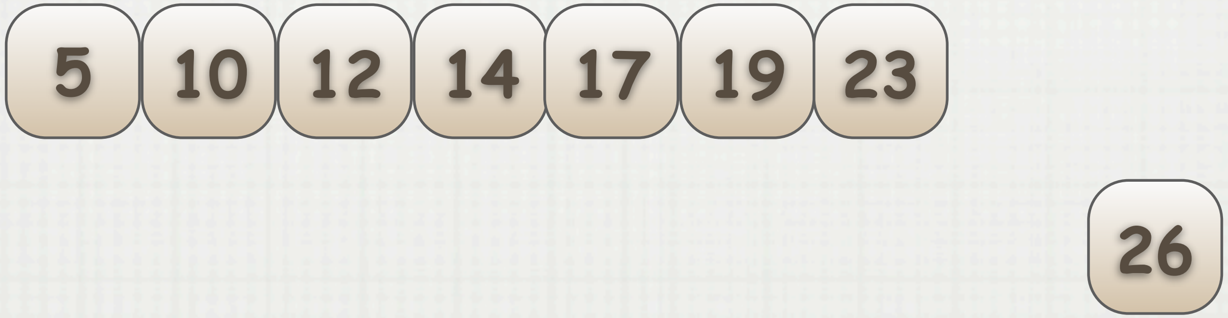
Inverted pairs: 4

Inverted pairs: 3

QUESTION 3

- Do modified merge

Inverted pairs: 4



Inverted pairs: 4

Inverted pairs: 3

QUESTION 3

- Do modified merge

Inverted pairs: 4

5	10	12	14	17	19	23	26

Inverted pairs: 4   Inverted pairs: 3

Total inverted pairs = 4 + 3 + 4 = 11

QUESTION 3

- The modified merge runs in O(n)
- We modify merge sort by calling modified merge to compute inverted pairs. The running time is O(nlogn)

QUESTION 4

a. Use Master theorem case 1

$T(n) = \theta(n^{\log_2 9})$

b. Use Master theorem case 3

$T(n) = \theta(n^3)$ (by choosing $c = 7/8$)

c. Use substitution method

$T(n) = \theta(\log n)$

QUESTION 4

d. Use recursion tree method

$T(n) = \theta(n)$

Cannot use Master theorem because
$a = 0.5 < 1$

e. Use Master theorem case 2

$T(n) = \theta(n \log n)$

QUESTION 4

- You can use recursion tree method in all the 5 questions, but it is also important to you to practice on Master theorem!