# CS2_3b 演習

Archer Shu

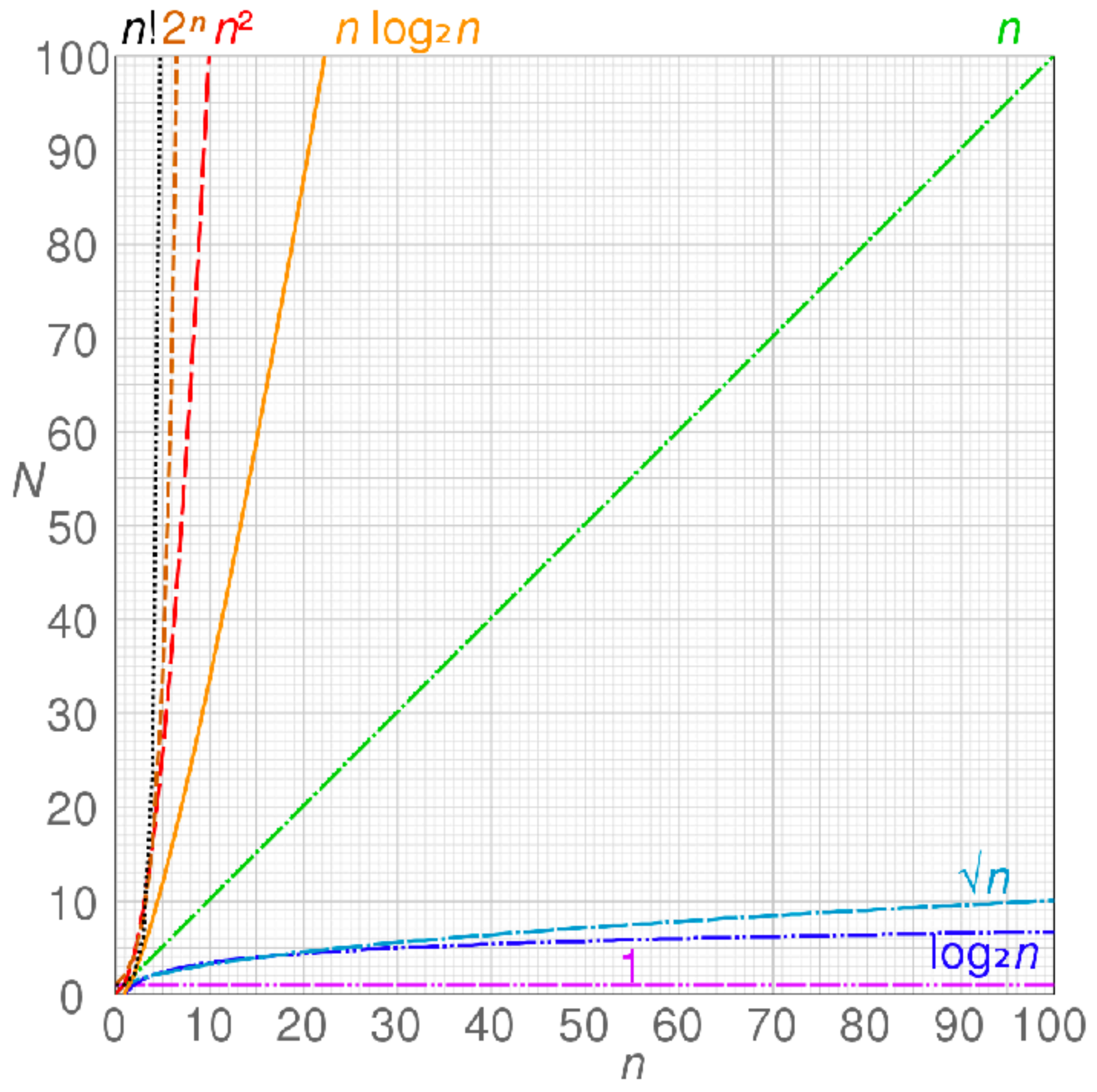
- P, NP, NP-hard問題
- アルゴリズム演習： Valid Anagram
- 自習

# Time Complexity
# 時間複雑性

- P: Polynomial time
  多項式時間

- NP: Non-deterministic Polynomial time
  非決定性多項式時間

**NP != Not P**

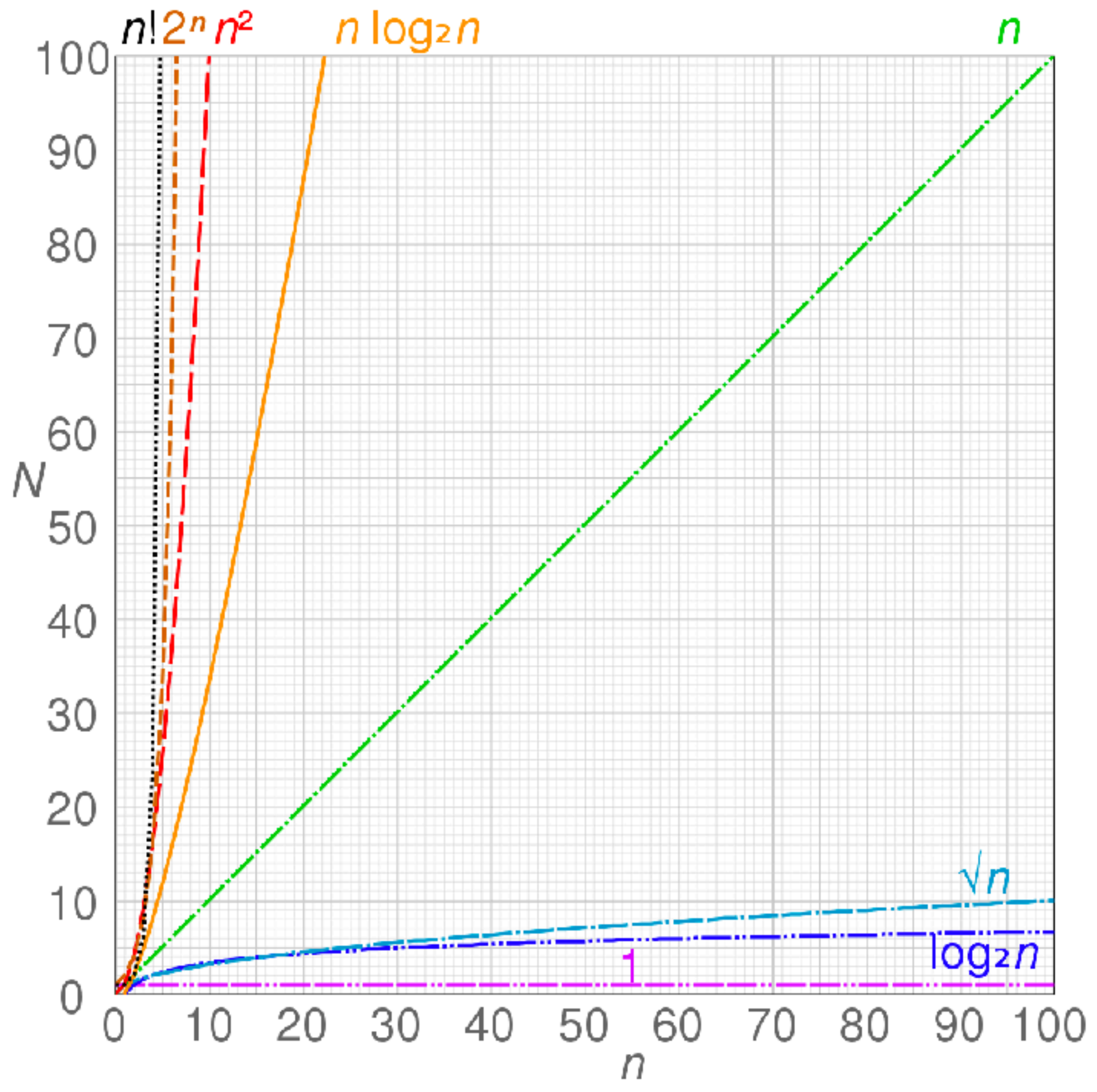


# Time Complexity
# 時間複雑性

- P: Polynomial time
  多項式時間 $\boxed{\text{解く}}$

- NP: Non-deterministic Polynomial time
  非決定性多項式時間 $\boxed{\text{検証}}$

**NP != Not P**

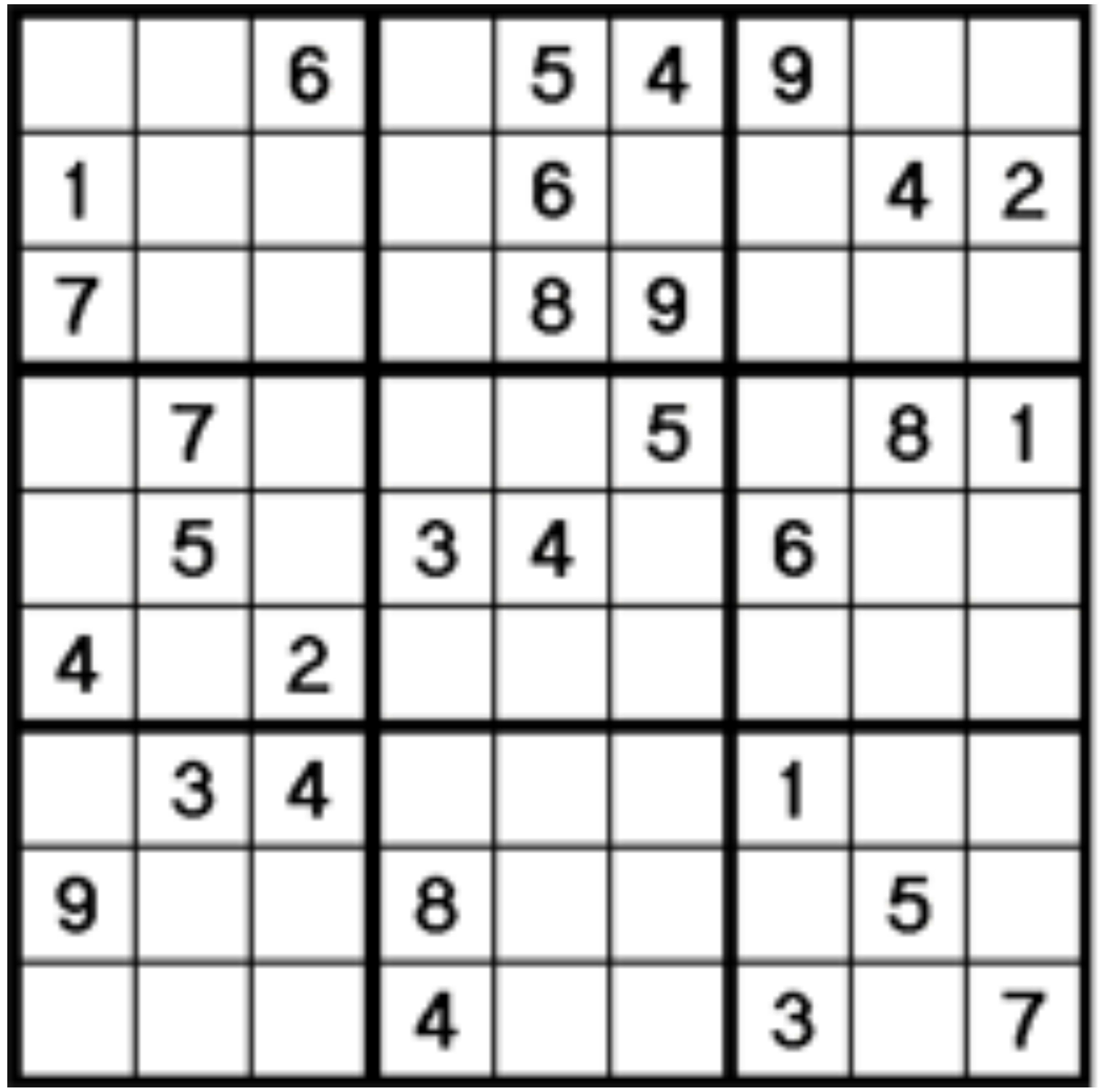


100^2 milliseconds to years

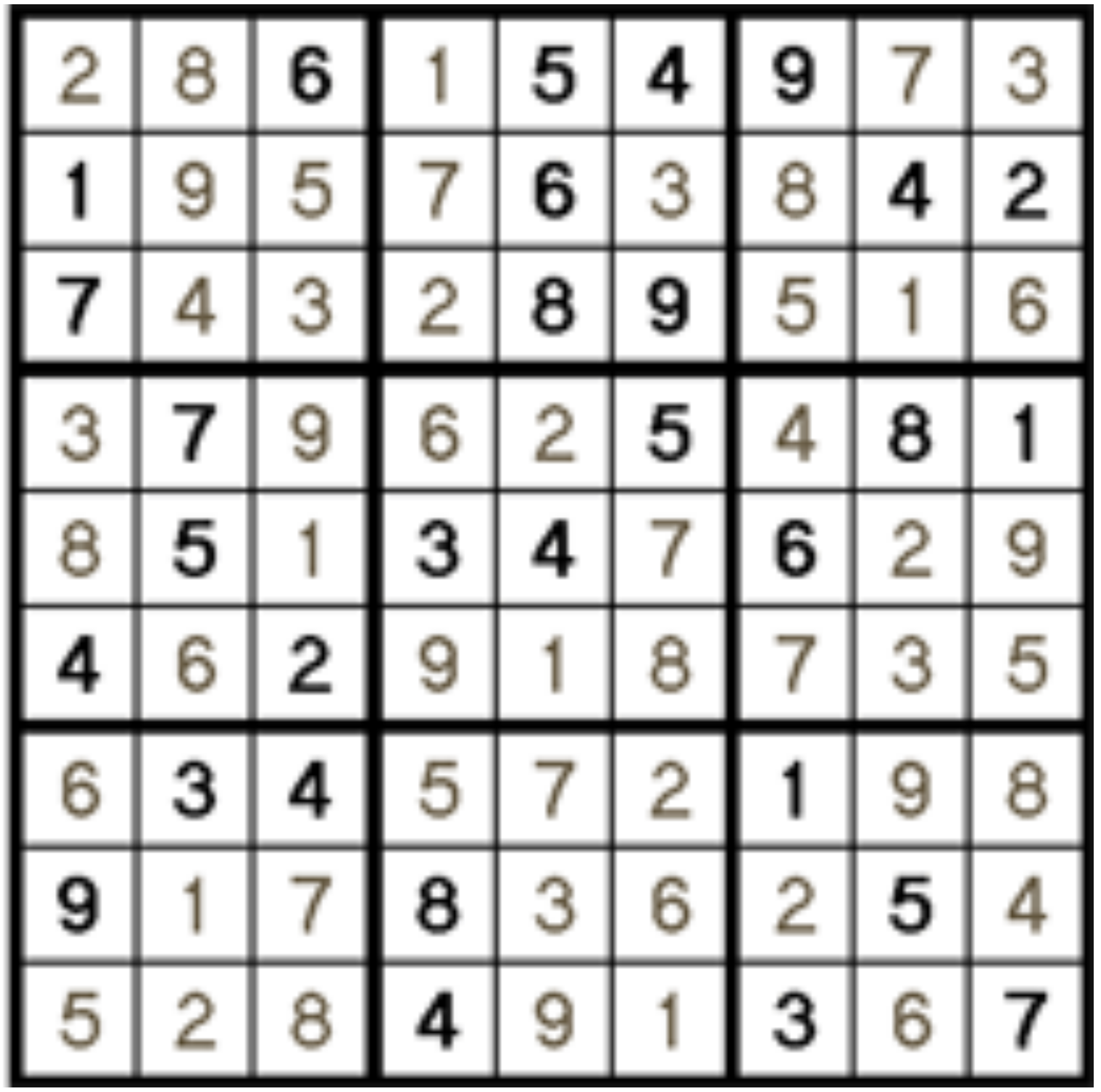
2^100 milliseconds to years

100! milliseconds to years

# NP問題の例: 数独



| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | 6 | | 5 | 4 | 9 | | |
| 1 | | | | 6 | | | 4 | 2 |
| 7 | | | | 8 | 9 | | | |
| | 7 | | | | 5 | | 8 | 1 |
| | 5 | | 3 | 4 | | 6 | | |
| 4 | | 2 | | | | | | |
| | 3 | 4 | | | | 1 | | |
| 9 | | | 8 | | | | 5 | |
| | | | 4 | | | 3 | | 7 |

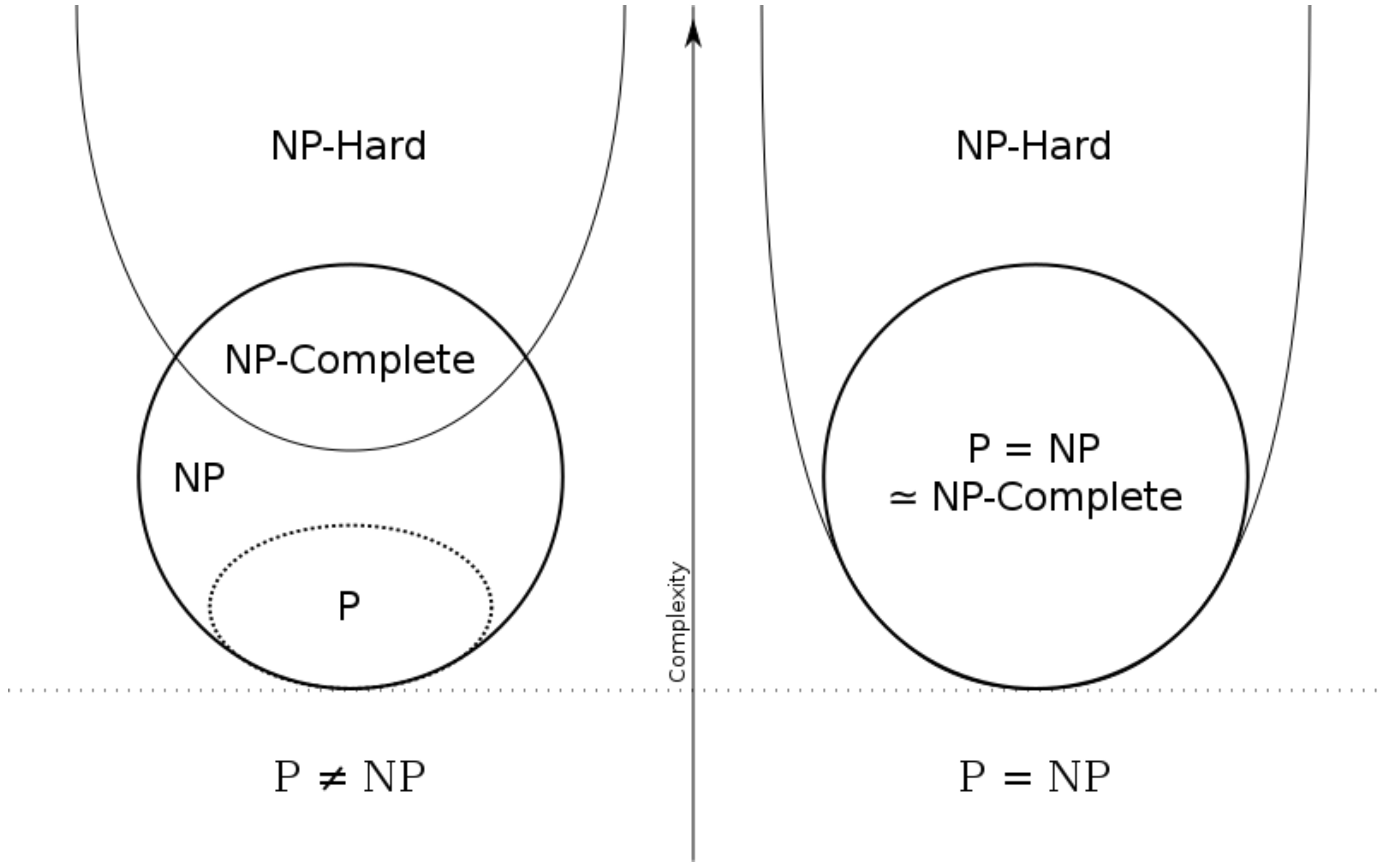


| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 2 | 8 | 6 | 1 | 5 | 4 | 9 | 7 | 3 |
| 1 | 9 | 5 | 7 | 6 | 3 | 8 | 4 | 2 |
| 7 | 4 | 3 | 2 | 8 | 9 | 5 | 1 | 6 |
| 3 | 7 | 9 | 6 | 2 | 5 | 4 | 8 | 1 |
| 8 | 5 | 1 | 3 | 4 | 7 | 6 | 2 | 9 |
| 4 | 6 | 2 | 9 | 1 | 8 | 7 | 3 | 5 |
| 6 | 3 | 4 | 5 | 7 | 2 | 1 | 9 | 8 |
| 9 | 1 | 7 | 8 | 3 | 6 | 2 | 5 | 4 |
| 5 | 2 | 8 | 4 | 9 | 1 | 3 | 6 | 7 |

# P ≠ NP

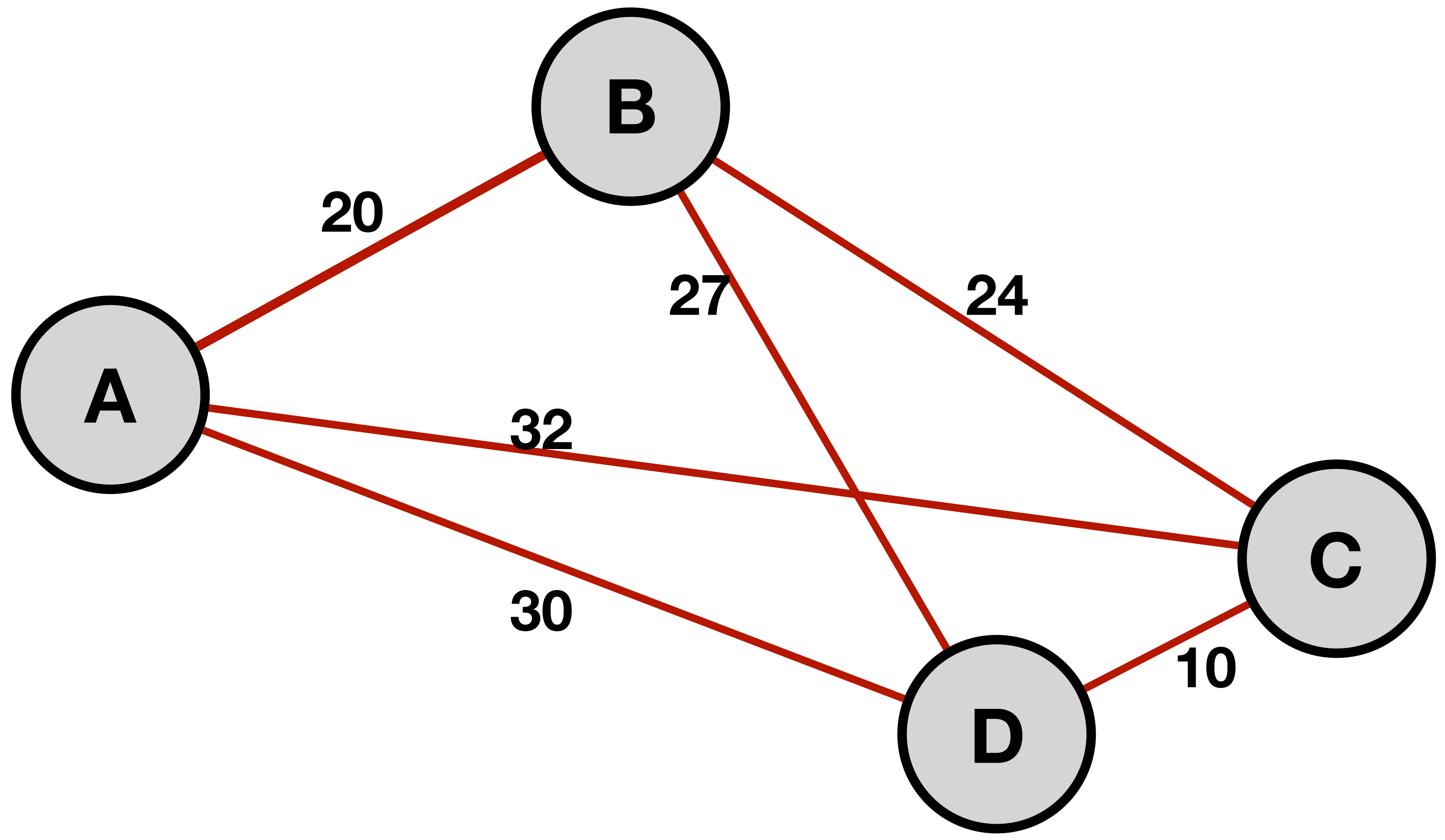
- Millennium prize problems
  ミレニアム懸賞問題

- 応用例：RSA暗号



# TSP (Traveling Salesman Problem)
# 巡回セールスマン問題

- Condition
  - Starting from a city, the salesman must travel to all cities once before returning home
  - The distance between each city is given, and is assumed to be the same in both directions.
  - Only the links shown are to be used
- Question
  - Minimum distance to be travelled.



**NP-hard**

# Valid Anagram

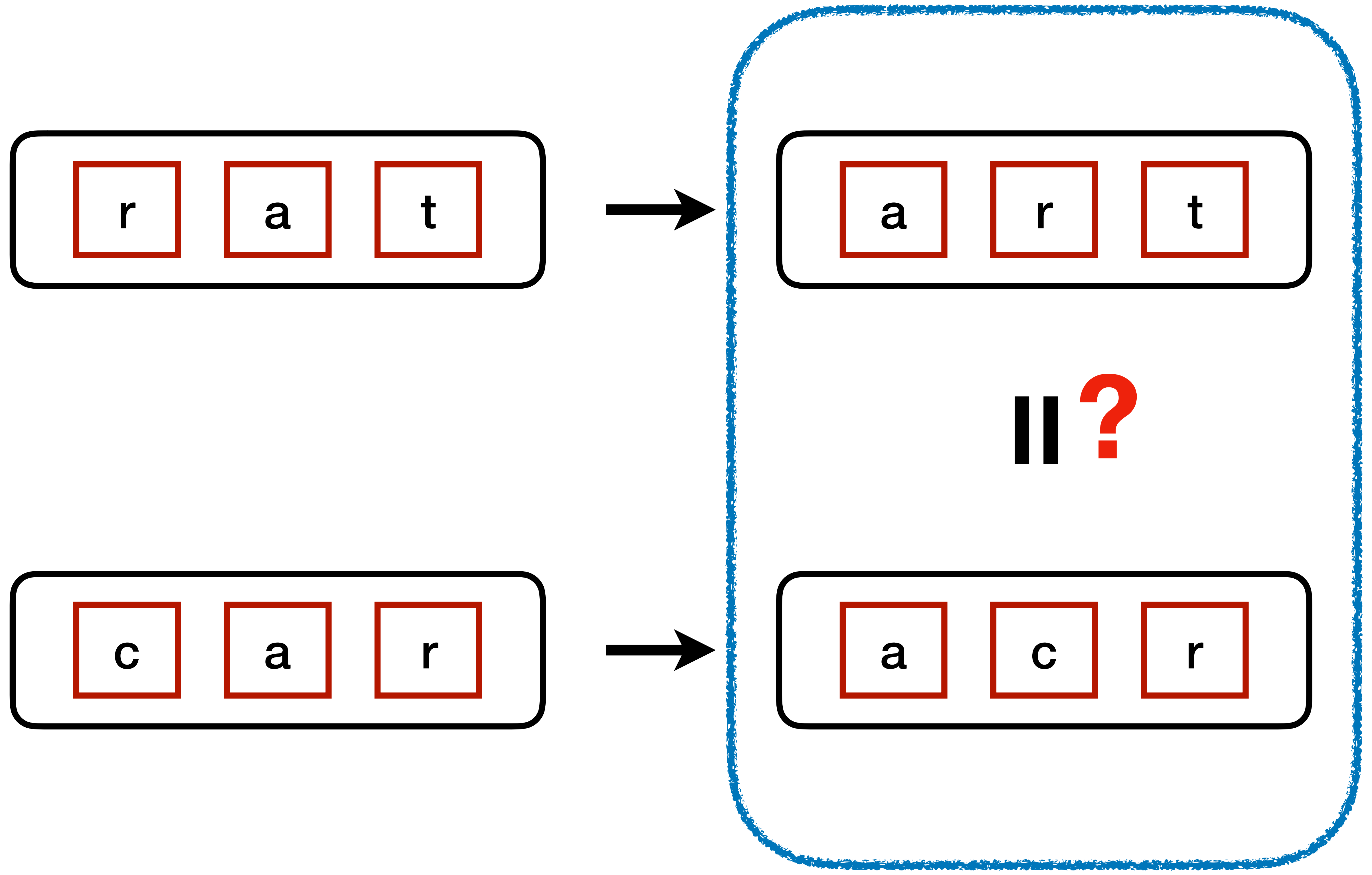
- LeetCode 242
- Anagram: アナグラム、(語句の)つづり換え
- E.g. “titech” & “tchite”

## Solution One: Sort & Compare

sort s

sort t

sorted s equals t?



```
class Solution:
  def isAnagram(self, s: str, t: str) -> bool:
      return sorted(s) == sorted(t)
```

Selection Sort

Insertion Sort

Heap Sort

Merge Sort

Quick Sort

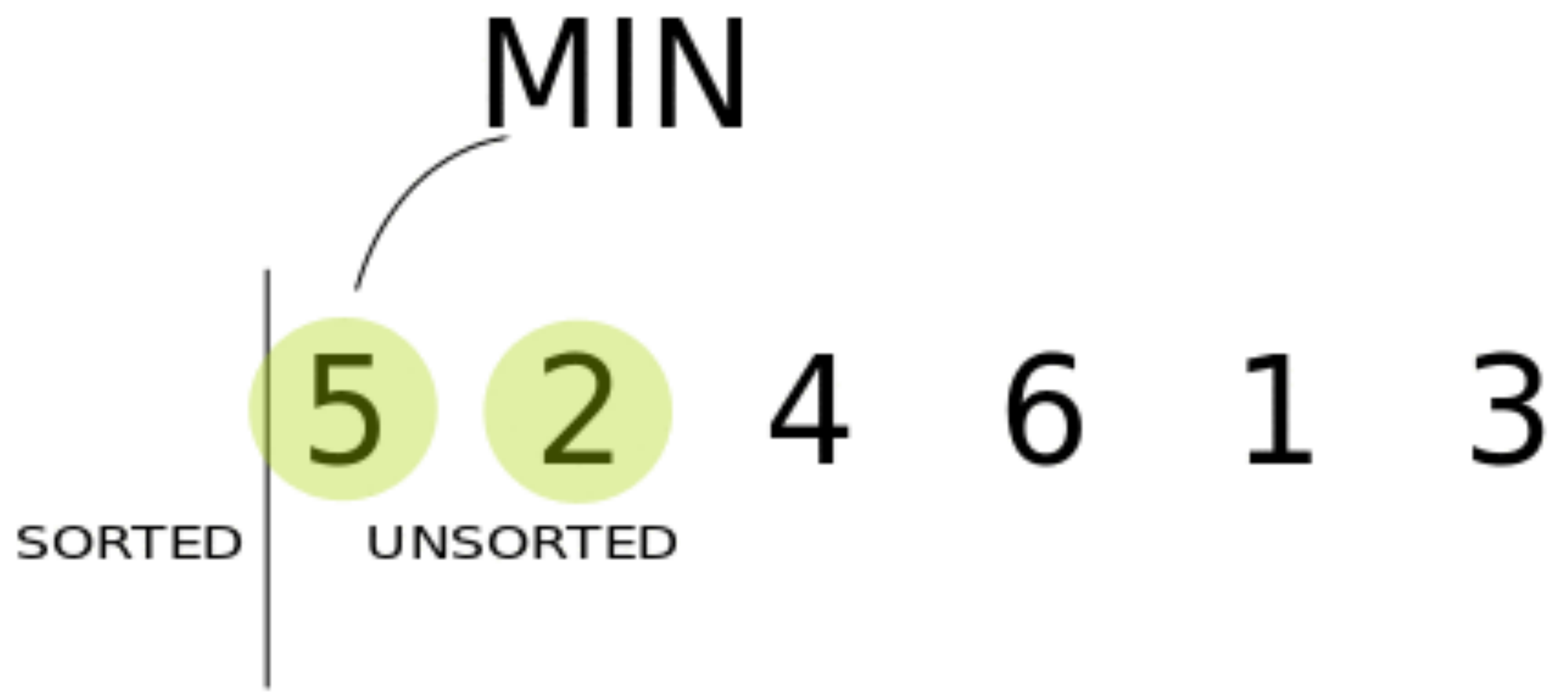
…

## Selection Sort

- $O(n^2)$

```
def sort(nums):
    for i in range(len(nums)):
        for j in range(i+1, len(nums)):
            if(nums[i] > nums[j]):
                nums[i], nums[j] = nums[j], nums[i]
    return nums
```

```
sort([5,2,4,6,1,3])
```



# Selection Sort

- O($n^2$)

```
def sort(nums):
    for i in range(len(nums)):
        for j in range(i+1, len(nums)):
            if(nums[i] > nums[j]):
                nums[i], nums[j] = nums[j], nums[i]
    return nums
```

```
sort([5,2,4,6,1,3])
```

$\longrightarrow$

```
def sort(str):
    lst = list(str)
    for i in range(len(lst)):
        for j in range(i+1, len(lst)):
            if(lst[i] > lst[j]):
                lst[i], lst[j] = lst[j], lst[i]
    return "".join(lst)
```

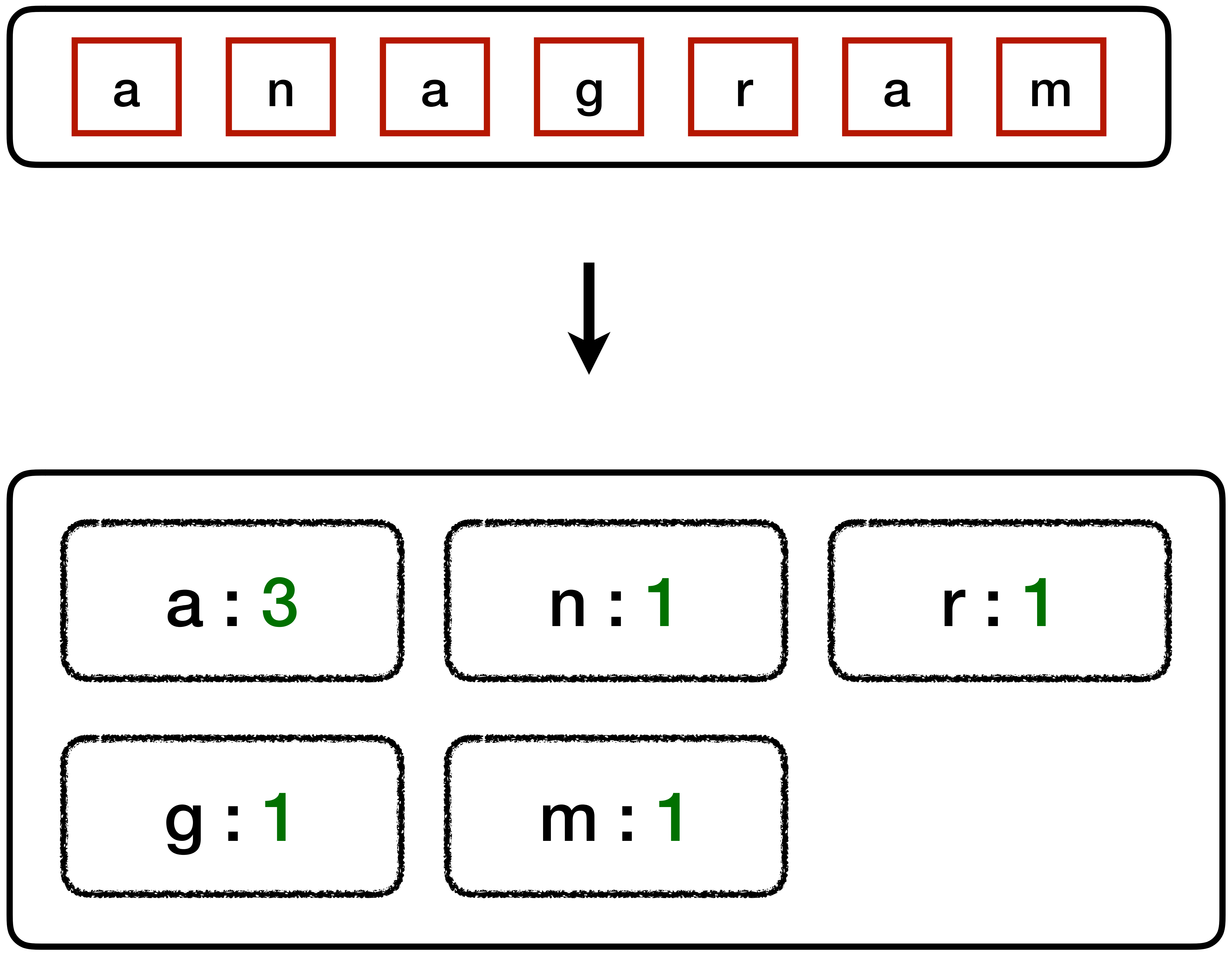
```
sort(“anagram”)
```

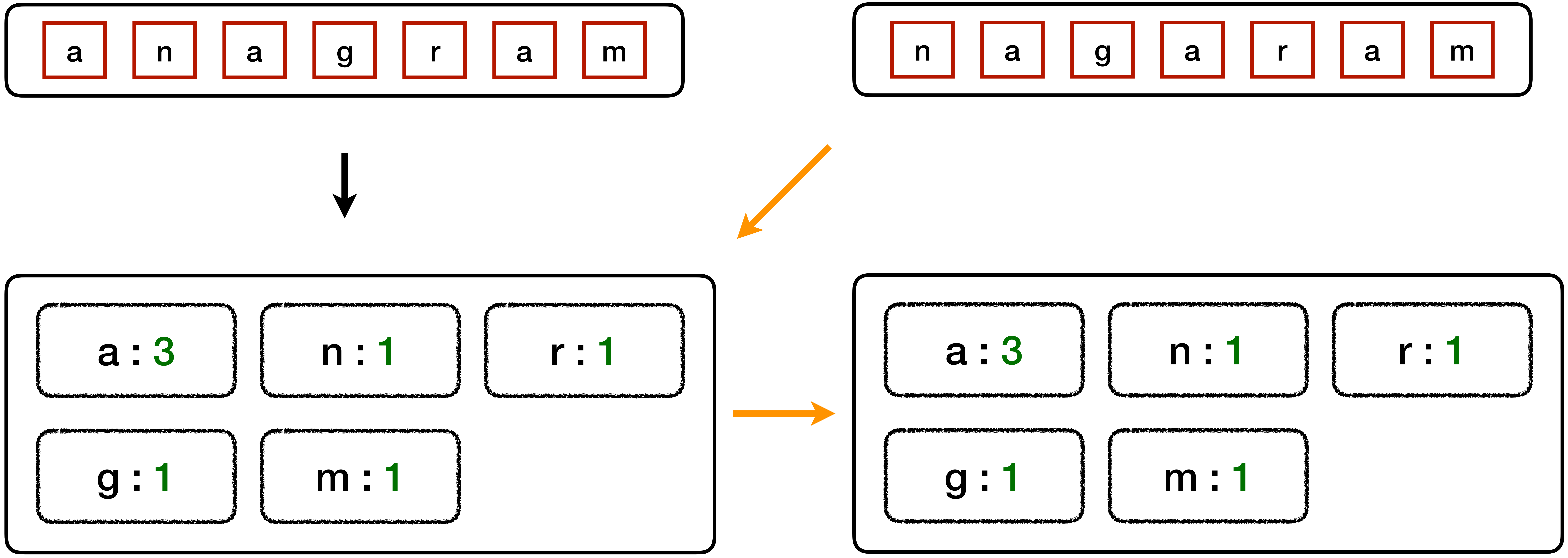
- sorted( )
    - timsort
    - O($nlogn$)  (faster than selection sort)

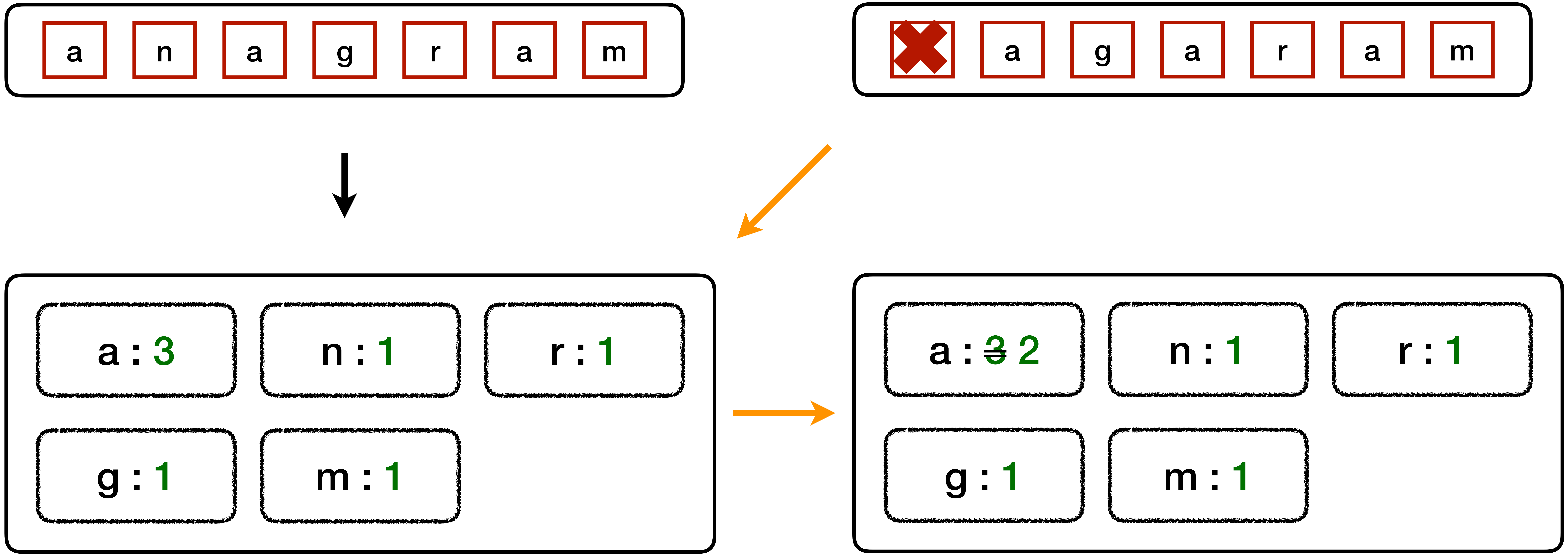
```
class Solution:
    def isAnagram(self, s: str, t: str) -> bool:
        return sorted(s) == sorted(t)
```

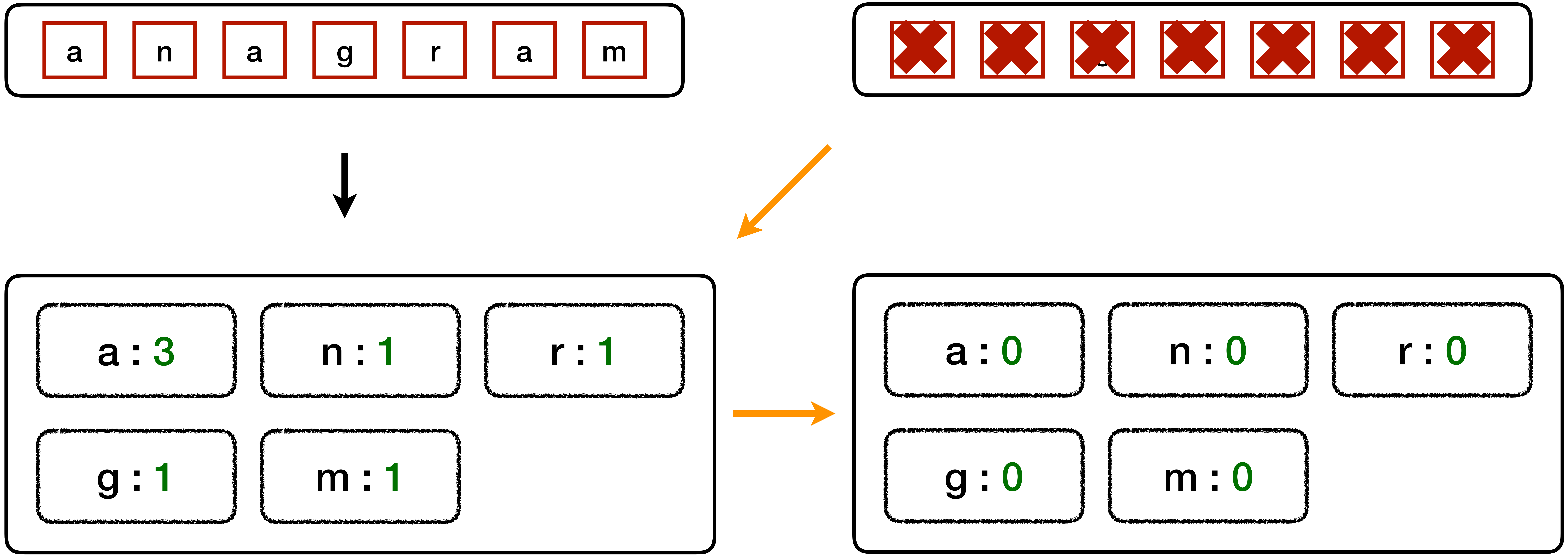
# HashMap

- A data structure that the time complexity of searching is O(1)




a n a g r a m
n a g a r a m
a : 3
n : 1
r : 1
g : 1
m : 1
a : 3
n : 1
r : 1
g : 1
m : 1


a
n
a
g
r
a
m
a
g
a
r
a
m
a : 3
n : 1
r : 1
g : 1
m : 1
a : ~~3~~ 2
n : 1
r : 1
g : 1
m : 1


a
n
a
g
r
a
m
a : 3
n : 1
r : 1
g : 1
m : 1
a : 0
n : 0
r : 0
g : 0
m : 0

# HashMap

- O(n) (faster than sort)

```
class Solution:
    def isAnagram(self, s: str, t: str) -> bool:
        h={}
        for ch in s:
            if ch not in h:
                h[ch] = 0
            h[ch] += 1

        for ch in t:
            if ch not in h:
                h[ch] = 0
            h[ch] -= 1

        for key in h.keys():
            if h[key] != 0:
                return False

        return True
```

**Try Other Method~**

## Practise Website

- 日本語
    - Atcoder: https://atcoder.jp/
    - paiza: https://paiza.jp/
- English
    - LeetCode: https://leetcode.com/