

question @ startlink.io

# 자료구조 2

최백준 choi@startlink.io

---

# 스택

---

# 검열

<https://www.acmicpc.net/problem/3111>

- 텍스트 T에서 A라는 단어를 다음과 같은 알고리즘을 이용해서 모두 지운다
  1. T에 A가 없으면 알고리즘을 종료한다.
  2. T에서 처음 등장하는 A를 찾은 뒤, 삭제한다.
  3. T에 A가 없으면 알고리즘을 종료한다.
  4. T에서 마지막으로 등장하는 A를 찾은 뒤, 삭제한다.
  5. 1번으로 돌아간다.

# 검열

<https://www.acmicpc.net/problem/3111>

- 왼쪽 스택과 오른쪽 스택으로 나눠서 문제를 푼다.
- $A = abc$
- $T = fabaabcabcabccd$

# 검열

<https://www.acmicpc.net/problem/3111>

- 왼쪽 스택과 오른쪽 스택으로 나눠서 문제를 푼다.
- $A = abc$
- $T = \mathbf{f}abaabcbcabccd$
- $L = f$
- $R =$

# 검열

<https://www.acmicpc.net/problem/3111>

- 왼쪽 스택과 오른쪽 스택으로 나눠서 문제를 푼다.
- $A = abc$
- $T = f**a**baabcbcabccd$
- $L = fa$
- $R =$

# 검열

<https://www.acmicpc.net/problem/3111>

- 왼쪽 스택과 오른쪽 스택으로 나눠서 문제를 푼다.
- $A = abc$
- $T = fab$ **a**abcbcabccd
- $L = fab$
- $R =$

# 검열

<https://www.acmicpc.net/problem/3111>

- 왼쪽 스택과 오른쪽 스택으로 나눠서 문제를 푼다.
- $A = abc$
- $T = fabaabcabcabccd$
- $L = faba$
- $R =$



# 검열

<https://www.acmicpc.net/problem/3111>

- 왼쪽 스택과 오른쪽 스택으로 나눠서 문제를 푼다.
- $A = abc$
- $T = faba**a**bcbcabccd$
- $L = fabaa$
- $R =$

# 검열

10

<https://www.acmicpc.net/problem/3111>

- 왼쪽 스택과 오른쪽 스택으로 나눠서 문제를 푼다.
- $A = abc$
- $T = fabaa**b**cbcabccd$
- $L = fabaab$
- $R =$

# 검열

<https://www.acmicpc.net/problem/3111>

- 왼쪽 스택과 오른쪽 스택으로 나눠서 문제를 푼다.
- $A = abc$
- $T = fabaabc**bc**abccd$
- $L = faba**abc**$
- $R =$

# 검열

12

<https://www.acmicpc.net/problem/3111>

- 왼쪽 스택과 오른쪽 스택으로 나눠서 문제를 푼다.
- $A = abc$
- $T = fabaabc**c**bcabccd$
- $L = faba$
- $R =$

# 검열

13

<https://www.acmicpc.net/problem/3111>

- 왼쪽 스택과 오른쪽 스택으로 나눠서 문제를 푼다.
- $A = abc$
- $T = fabaabc**bc**abcc**d**$
- $L = faba$
- $R = d$

# 검열

<https://www.acmicpc.net/problem/3111>

- 왼쪽 스택과 오른쪽 스택으로 나눠서 문제를 푼다.
- $A = abc$
- $T = fabaabc**b**cabcc**d**$
- $L = faba$
- $R = dc$

# 검열

15

<https://www.acmicpc.net/problem/3111>

- 왼쪽 스택과 오른쪽 스택으로 나눠서 문제를 푼다.
- $A = abc$
- $T = fabaabc**c**bcab**c**cd$
- $L = faba$
- $R = dcc$

# 검열

<https://www.acmicpc.net/problem/3111>

- 왼쪽 스택과 오른쪽 스택으로 나눠서 문제를 푼다.
- $A = abc$
- $T = fabaabc**b**ca**b**ccd$
- $L = faba$
- $R = dccb$



# 검열

<https://www.acmicpc.net/problem/3111>

- 왼쪽 스택과 오른쪽 스택으로 나눠서 문제를 푼다.
- $A = abc$
- $T = fabaabc**bc**abccd$
- $L = faba$
- $R = dcc**ba**$

# 검열

<https://www.acmicpc.net/problem/3111>

- 왼쪽 스택과 오른쪽 스택으로 나눠서 문제를 푼다.
- $A = abc$
- $T = fabaabc**bc**abccd$
- $L = faba$
- $R = dc$

# 검열

<https://www.acmicpc.net/problem/3111>

- 왼쪽 스택과 오른쪽 스택으로 나눠서 문제를 푼다.
- $A = abc$
- $T = fabaabc**b**cabccd$
- $L = fabab$
- $R = dc$

# 검열

20

<https://www.acmicpc.net/problem/3111>

- 왼쪽 스택과 오른쪽 스택으로 나눠서 문제를 푼다.
- $A = abc$
- $T = fabaabc**b**cabccd$
- $L = fab**abc**$
- $R = dc$

# 검열

<https://www.acmicpc.net/problem/3111>

- 왼쪽 스택과 오른쪽 스택으로 나눠서 문제를 푼다.
- $A = abc$
- $T = fabaabc**b**cabccd$
- $L = fab$
- $R = dc$

# 검열

<https://www.acmicpc.net/problem/3111>

- 모든 과정이 끝난 후에는 L에 들어있는 것을 R로 옮긴다.
- $A = abc$
- $T = fabaabc**b**cabccd$
- $L = fa$
- $R = dcb$

# 검열

<https://www.acmicpc.net/problem/3111>

- 모든 과정이 끝난 후에는 L에 들어있는 것을 R로 옮긴다.
- $A = abc$
- $T = fabaabc**b**abccd$
- $L = f$
- $R = dc**b**a$

# 검열

<https://www.acmicpc.net/problem/3111>

- 모든 과정이 끝난 후에는 L에 들어있는 것을 R로 옮긴다.
- $A = abc$
- $T = fabaabc**b**cabccd$
- $L = f$
- $R = d$



# 검열

25

<https://www.acmicpc.net/problem/3111>

- 모든 과정이 끝난 후에는 L에 들어있는 것을 R로 옮긴다.
- $A = abc$
- $T = fabaabc**b**cabccd$
- $L =$
- $R = df$

# 검열

26

<https://www.acmicpc.net/problem/3111>

- 모든 과정이 끝난 후에는 L에 들어있는 것을 R로 옮긴다.
- $A = abc$
- $T = fabaabc**b**abccd$
- $L =$
- $R = df$
- 정답: fd

- 문자열  $S$ 에서 폭발 문자열  $T$ 를 모두 지우는 문제

- 문자열: ~~mirkov~~4nizCC44

- 폭발 문자열: C4  $m\bar{r}(\leq) n\bar{z} \underline{C4}$

CCCCCCCC 44444444

$$O(N) \times N$$
$$O(N^2)$$

- 결과: mirkovniz

$$(1,000,000)^2 / 10^8 \approx 10$$

$$10^{12} / 10^8$$

$$10^4 \frac{1}{2}$$

# 문자열 폭발

<https://www.acmicpc.net/problem/9935>

- 스택에 넣는 것은 (문자열의 인덱스, 폭발 문자열에서 인덱스)
- 현재 문자가 폭발 문자열의 첫 번째 문자와 같으면 스택에 넣는다
- 다른 경우에 스택이 비어있으면 그냥 넘어간다
- 다른 경우에 스택이 비어있지 않으면, 스택의 가장 위에 있는 것의 폭발 문자열의 인덱스를 가져온다. 이 인덱스를  $p$ 라고 한다.
- 현재 문자가 폭발 문자열의  $p+1$ 문자와 같으면, 스택에 넣는다. ( $p+1$ 이 폭발 문자열의 마지막 문자면, 폭발 문자열을 찾은 것이다. 스택에서 폭발 문자열을 지운다)
- 다르면, 스택을 모두 비워버린다.

# 문자열 폭발

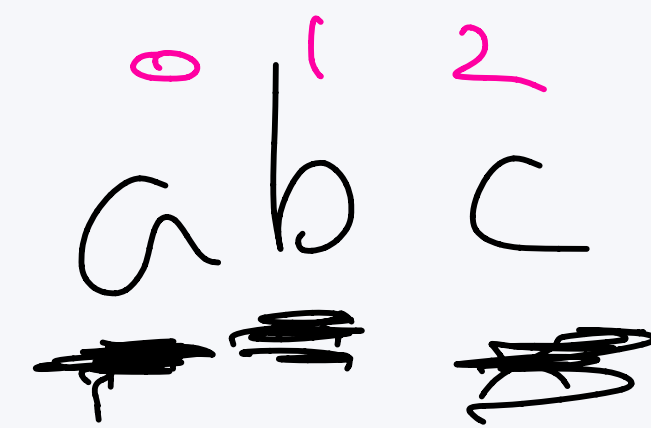
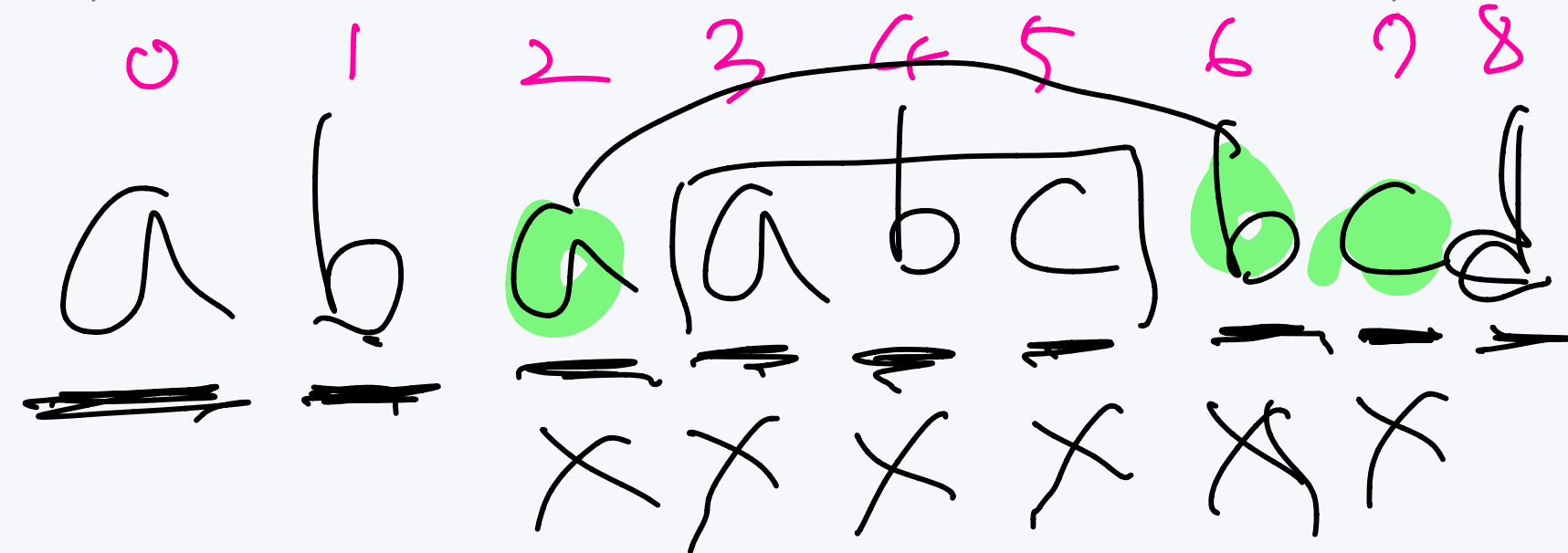
<https://www.acmicpc.net/problem/9935>

29

스택: ( 문자열 인덱스, 폭 인덱스 )

문자: abacabcabcd    폭: abc

- 폭발 문자열의 길이가 1이면, 스택을 사용할 수가 없기 때문에, 그냥 for문을 돌면서 체크해준다.



스택 1

abd

$N \times O(1)$

$O(N)$

# 문자열 폭발

<https://www.acmicpc.net/problem/9935>

- 폭발 문자열이 abc인 예시
- abaabcbcd -> ababcbcd -> abcd
- **a**baabcbcd
- 폭발 문자열의 첫 번째 문자와 같기 때문에, 스택에 넣는다.
- 스택: (0, 0)
- **a**baabcbcd
- 첫 번째 문자와 다르기 때문에, 스택의 가장 위에 있는 것의 폭발 문자열 인덱스를 가져온다.
- 이 인덱스가 0이고, b는 폭발 문자열의 0+1번째 문자와 같기 때문에, 스택에 넣는다.
- 스택: (0, 0), (1, 1)

# 문자열 폭발

<https://www.acmicpc.net/problem/9935>

- aba**a**abcbcd
- 폭발 문자열의 첫 번째 문자와 같기 때문에, 스택에 넣는다.
- 스택: (0, 0), (1, 1), (2, 0)
- aba**a**abcbcd
- 폭발 문자열의 첫 번째 문자와 같기 때문에, 스택에 넣는다.
- 스택: (0, 0), (1, 1), (2, 0), (3, 0)

# 문자열 폭발

<https://www.acmicpc.net/problem/9935>

- abaab**b**cbcd
- 첫 번째 문자와 다르기 때문에, 스택의 가장 위에 있는 것의 폭발 문자열 인덱스를 가져온다.
- 이 인덱스가 0이고, b는 폭발 문자열의 0+1번째 문자와 같기 때문에, 스택에 넣는다.
- 스택: (0, 0), (1, 1), (2, 0), (3, 0), (4, 1)



# 문자열 폭발

<https://www.acmicpc.net/problem/9935>

- abaab**c**bcd
- 첫 번째 문자와 다르기 때문에, 스택의 가장 위에 있는 것의 폭발 문자열 인덱스를 가져온다.
- 이 인덱스가 1이고, c는 폭발 문자열의 0+2번째 문자와 같기 때문에, 스택에 넣는다.
- 스택: (0, 0), (1, 1), (2, 0), (3, 0), (4, 1), (5, 2)
- 2는 폭발 문자열의 마지막 인덱스이기 때문에, 총 3개를 스택에서 뺀다.
- 스택: (0, 0), (1, 1), (2, 0), **(3, 0), (4, 1), (5, 2)**
- 3, 4, 5번째 문자는 사라져야 한다.
- 스택: (0, 0), (1, 1), (2, 0)

# 문자열 폭발

<https://www.acmicpc.net/problem/9935>

- abaabc**b**cd
- 첫 번째 문자와 다르기 때문에, 스택의 가장 위에 있는 것의 폭발 문자열 인덱스를 가져온다.
- 이 인덱스가 0이고, b는 폭발 문자열의 0+1번째 문자와 같기 때문에, 스택에 넣는다.
- 스택: (0, 0), (1, 1), (2, 0), (6, 1)

# 문자열 폭발

<https://www.acmicpc.net/problem/9935>

- abaabcb**cd**
- 첫 번째 문자와 다르기 때문에, 스택의 가장 위에 있는 것의 폭발 문자열 인덱스를 가져온다.
- 이 인덱스가 1이고, c는 폭발 문자열의 0+2번째 문자와 같기 때문에, 스택에 넣는다.
- 스택: (0, 0), (1, 1), (2, 0), (6, 1), (7, 2)
- 2는 폭발 문자열의 마지막 인덱스이기 때문에, 총 3개를 스택에서 뺀다.
- 스택: (0, 0), (1, 1), **(2, 0), (6, 1), (7, 2)**
- 2, 6, 7번째 문자는 사라져야 한다.
- 스택: (0, 0), (1, 1)

# 문자열 폭발

<https://www.acmicpc.net/problem/9935>

- abaabcbcd
- 첫 번째 문자와 다르기 때문에, 스택의 가장 위에 있는 것의 폭발 문자열 인덱스를 가져온다.
- 스택: (0, 0), (1, 1)
- 이 인덱스가 1이고, d는 폭발 문자열의 0+2번째 문자와 다르기 때문에, 스택을 비운다.
- 스택: (비어있음)

# 문자열 폭발

<https://www.acmicpc.net/problem/9935>

- 스택에 넣고 빼는 과정에서 지워져야 하는 문자로 체크하지 않은 글자를 모두 출력하면 된다.
- 이 방법의 예외는 폭발 문자열의 길이가 1일 때이다.
- 이 때는, 한 글자 이기 때문에, 그냥 지우면 된다.

# 문자열 폭발

<https://www.acmicpc.net/problem/9935>

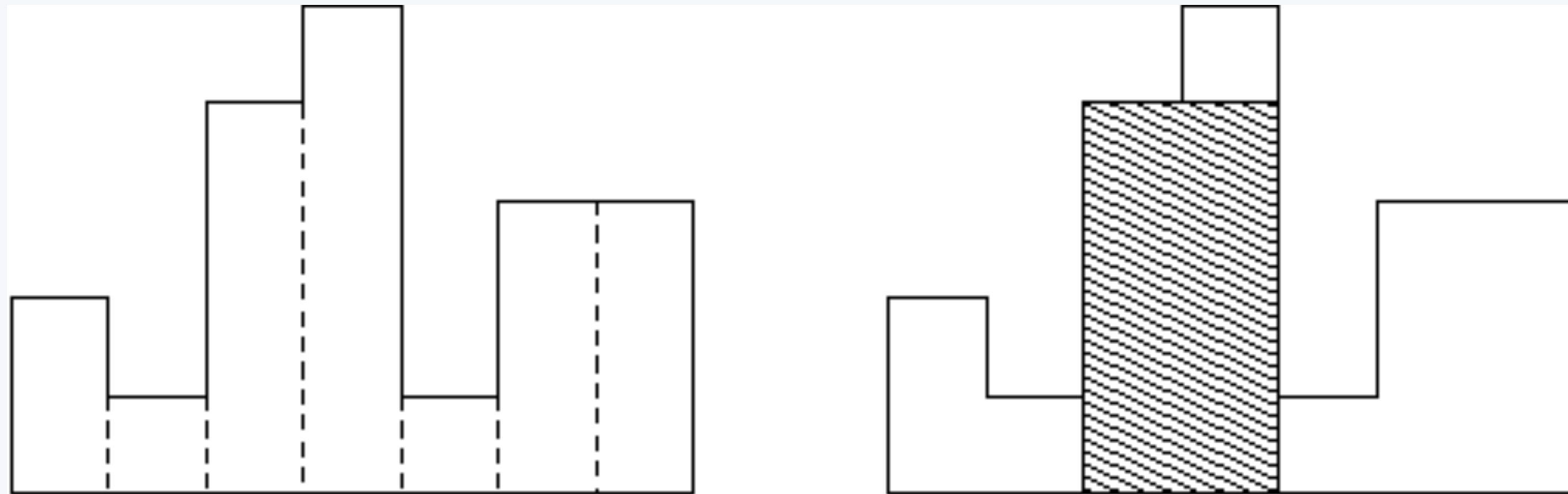
- C/C++: <https://gist.github.com/Baekjoon/fbbac6c2b54946aff40e>
- Java: <https://gist.github.com/Baekjoon/4368405c6c974ee61f74>

# 히스토그램에서 가장 큰 직사각형

39

<https://www.acmicpc.net/problem/6549>

- 히스토그램이 주어졌을 때, 가장 큰 직사각형을 찾는 문제

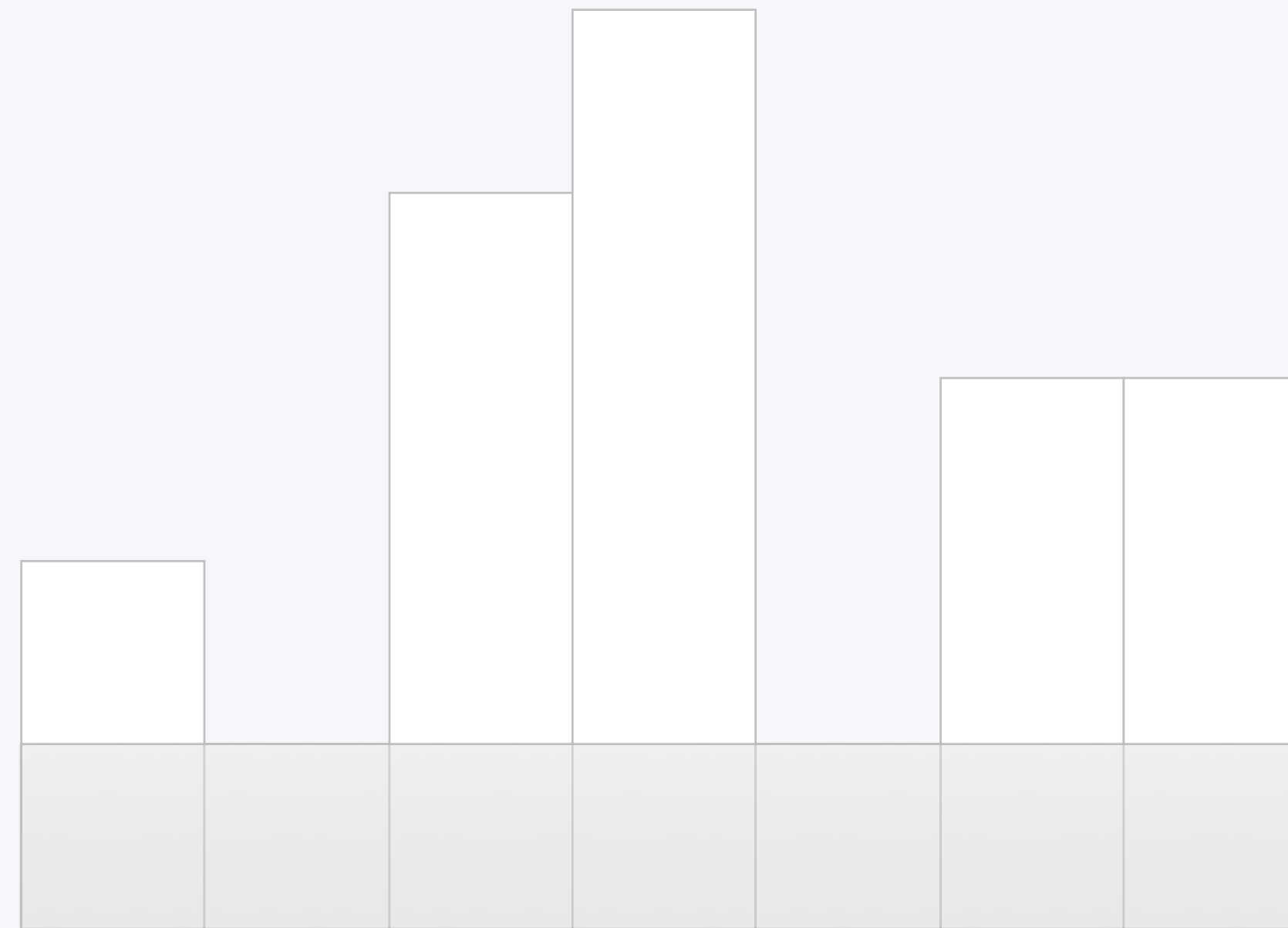


# 히스토그램에서 가장 큰 직사각형

40

<https://www.acmicpc.net/problem/6549>

- 가장 왼쪽 끝과 오른쪽 끝을 변으로 하는 가장 큰 직사각형의 높이는?
- 높이 : 히스토그램에서 가장 높이가 낮은 막대의 높이





# 히스토그램에서 가장 큰 직사각형

<https://www.acmicpc.net/problem/6549>

- 모든 막대  $x$ 에 대해서,  $x$ 를 높이로 하면서, 만들 수 있는 가장 큰 직사각형을 찾아야 함
- $x$ 를 높이로 하면서 만들 수 있는 가장 큰 직사각형은
- $x$ 의 왼쪽에 있는 막대 중에  $x$ 보다 높이가 작은 첫 번째 막대  $left$ 와
- $x$ 의 오른쪽에 있는 막대 중에서  $x$ 보다 높이가 작은 첫 번째 막대  $right$ 를
- 찾아야 한다

# 히스토그램에서 가장 큰 직사각형

<https://www.acmicpc.net/problem/6549>

- 스택에 막대를 넣기 전에
- 스택의 가장 위에 있는 막대  $top$ 과 현재 넣으려고 하는 막대  $x$ 를 비교해야 한다
- $top$ 의 높이가  $x$ 의 높이보다 크면
- $top$ 을 높이로 하는 직사각형은  $x$ 를 지나갈 수 없다
- $top$ 을 높이로 하는 직사각형의  $right$ 는  $x-1$ 이다
- $top$ 을 높이로 하는 직사각형의  $left$ 는  $top$  다음에 스택에 들어있는 막대
- $left$ 와  $right$ 를 구했기 때문에,  $top$ 을 높이로 하는 넓이를 구할 수 있다

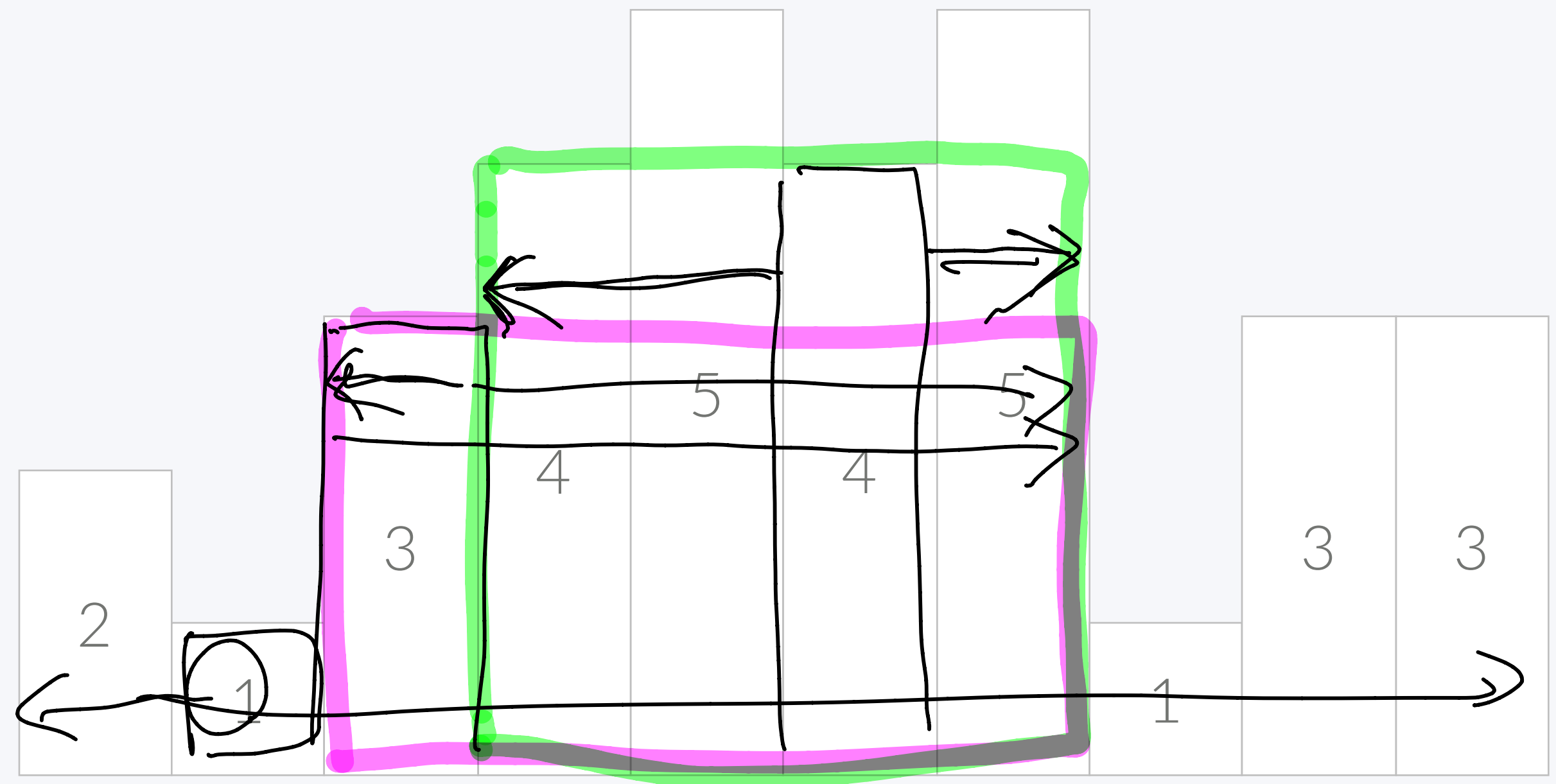
# 히스토그램에서 가장 큰 직사각형

43

<https://www.acmicpc.net/problem/6549>

- 0번 막대, 높이 2
- 스택이 비어있기 때문에, 막대 번호 0을 스택에 넣는다
- 스택: 0

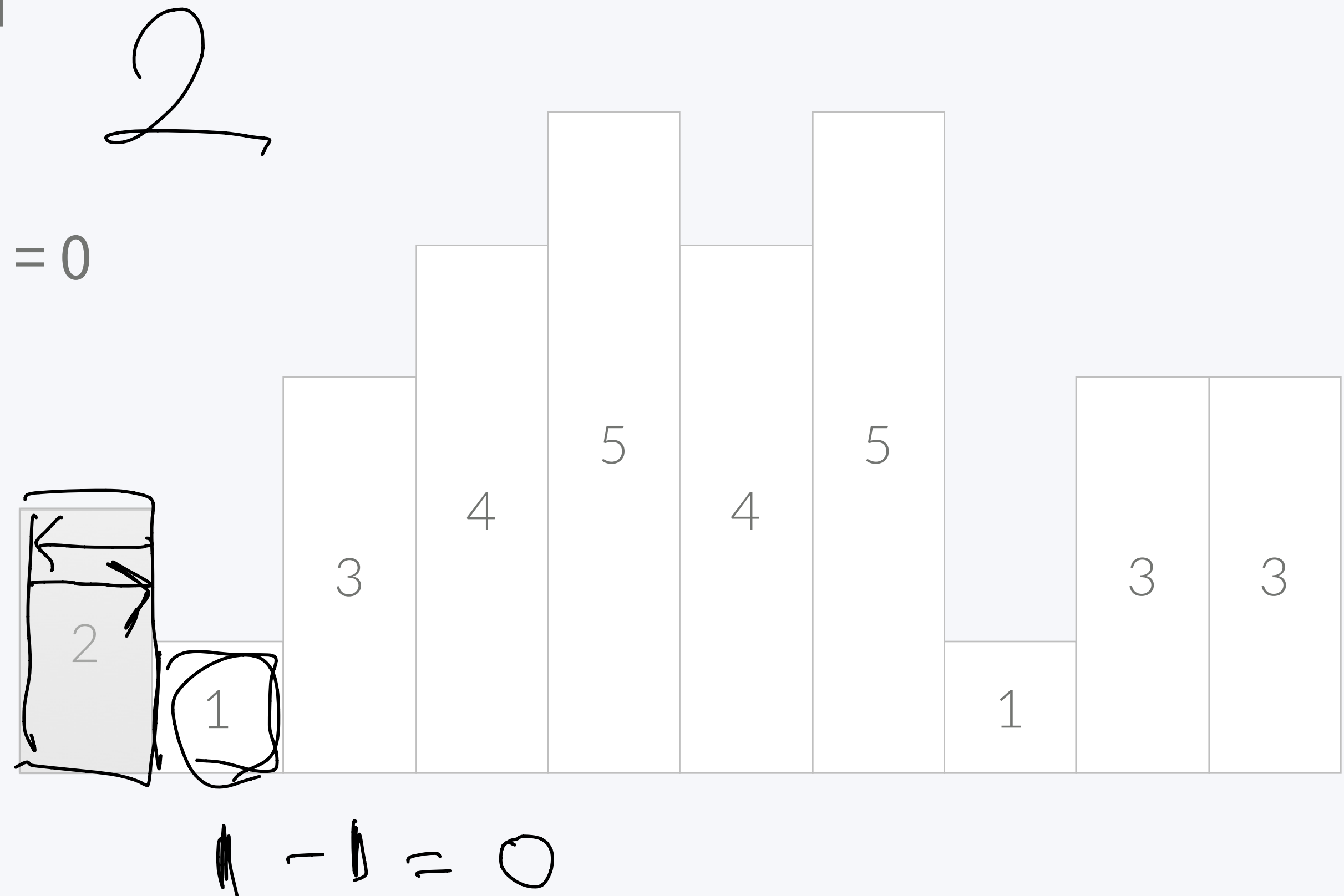
$O(N^2)$



# 히스토그램에서 가장 큰 직사각형

<https://www.acmicpc.net/problem/6549>

- 1번 막대, 높이 1
- 스택의 가장 위에 있는 막대보다 높이가 작다
- 0번 막대의 오른쪽 끝 =  $1 - 1 = 0$
- pop을 하면 스택이 비어있기 때문에 왼쪽 끝 = 0
- 넓이: 2
- 스택: 1

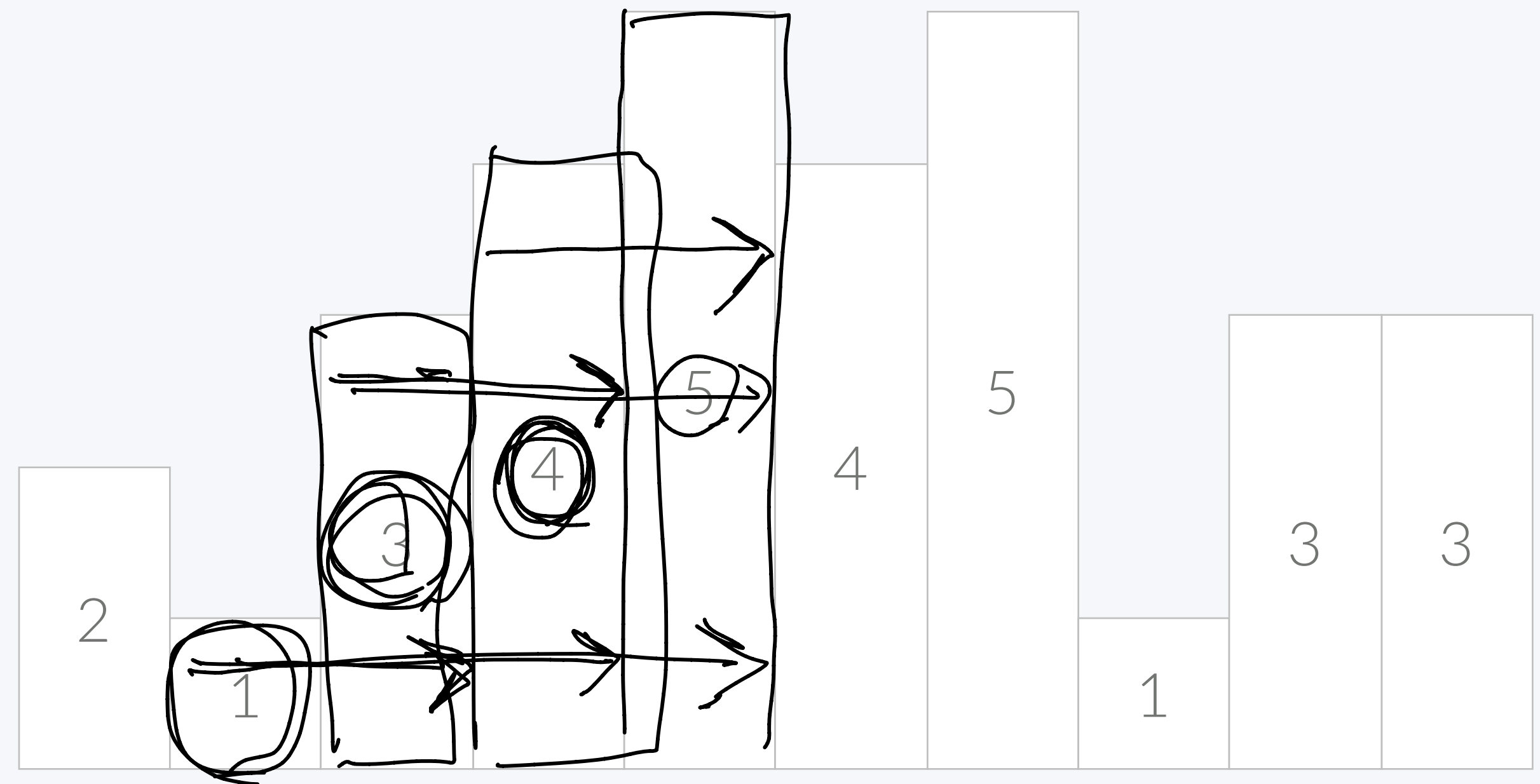


# 히스토그램에서 가장 큰 직사각형

45

<https://www.acmicpc.net/problem/6549>

- 2번 막대, 높이 3
- 스택의 가장 위에 있는 막대 1보다 높이가 크거나 같기 때문에 push
- 스택: 1 2

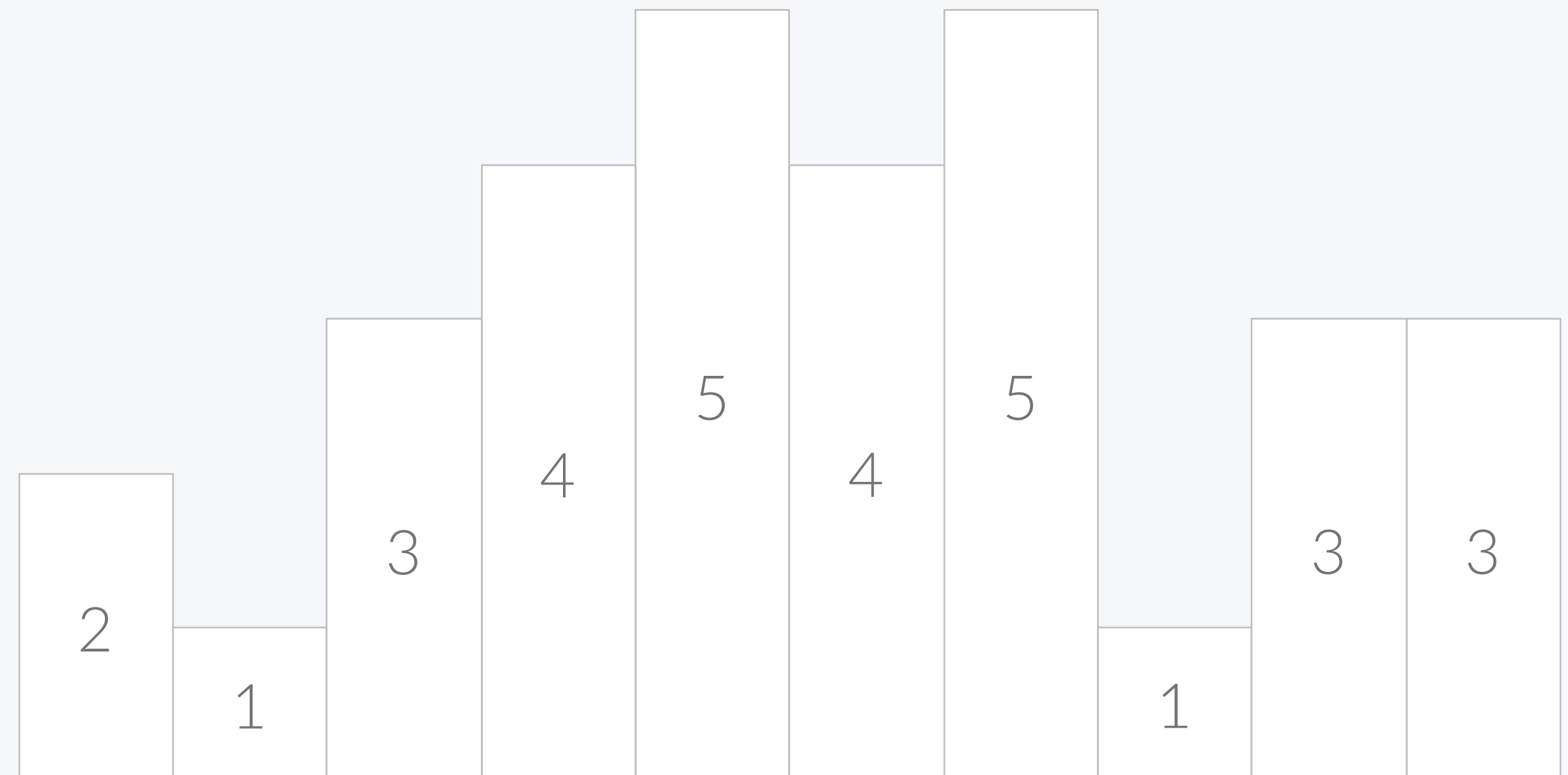


# 히스토그램에서 가장 큰 직사각형

46

<https://www.acmicpc.net/problem/6549>

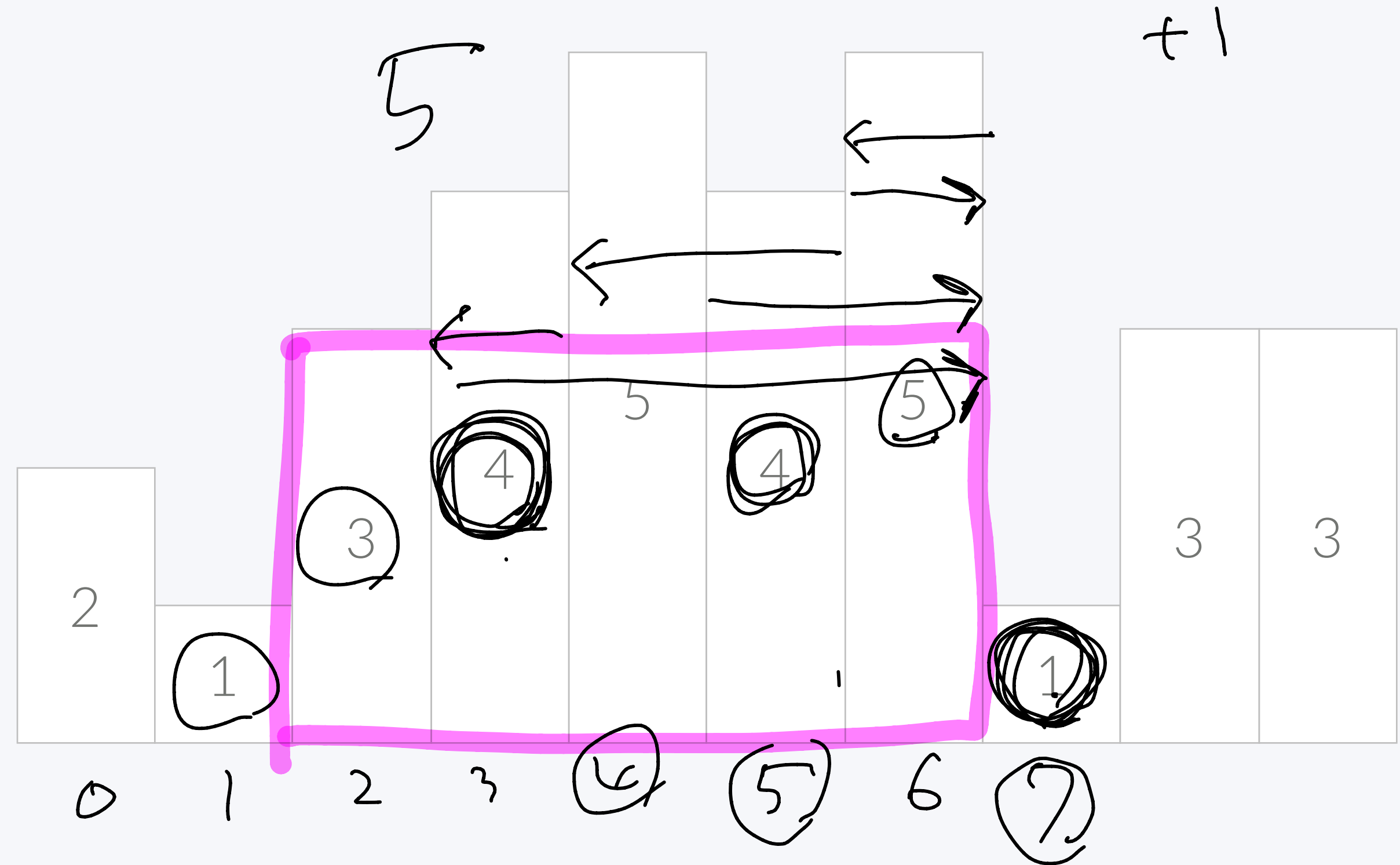
- 3번 막대, 높이 4
- 스택의 가장 위에 있는 막대 2보다 높이가 크거나 같기 때문에 push
- 스택: 1 2 3



# 히스토그램에서 가장 큰 직사각형

<https://www.acmicpc.net/problem/6549>

- 4번 막대, 높이 5
- 스택의 가장 위에 있는 막대 3보다 높이가 크거나 같기 때문에 push 1 2
- 스택: 1 2 3 4

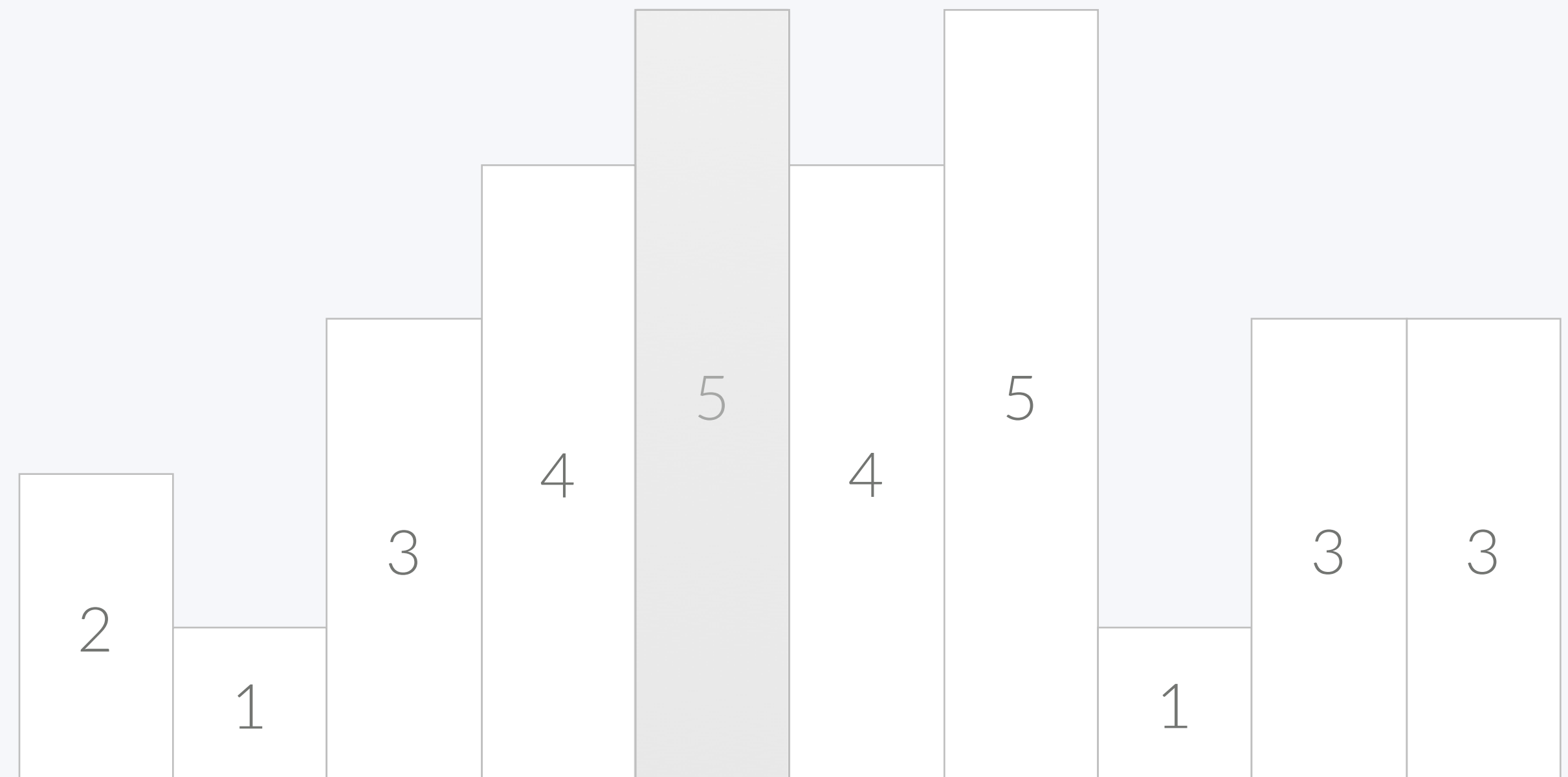


# 히스토그램에서 가장 큰 직사각형

48

<https://www.acmicpc.net/problem/6549>

- 5번 막대, 높이 4
- 스택의 가장 위에 있는 막대 4번 (높이 5)이 현재 높이보다 크다
- $\text{right} = 5 - 1 = 4$ ,  $\text{left} = 3 + 1 = 4$
- 4번 막대로 만들 수 있는 직사각형 넓이: 5
- 스택: 1 2 3



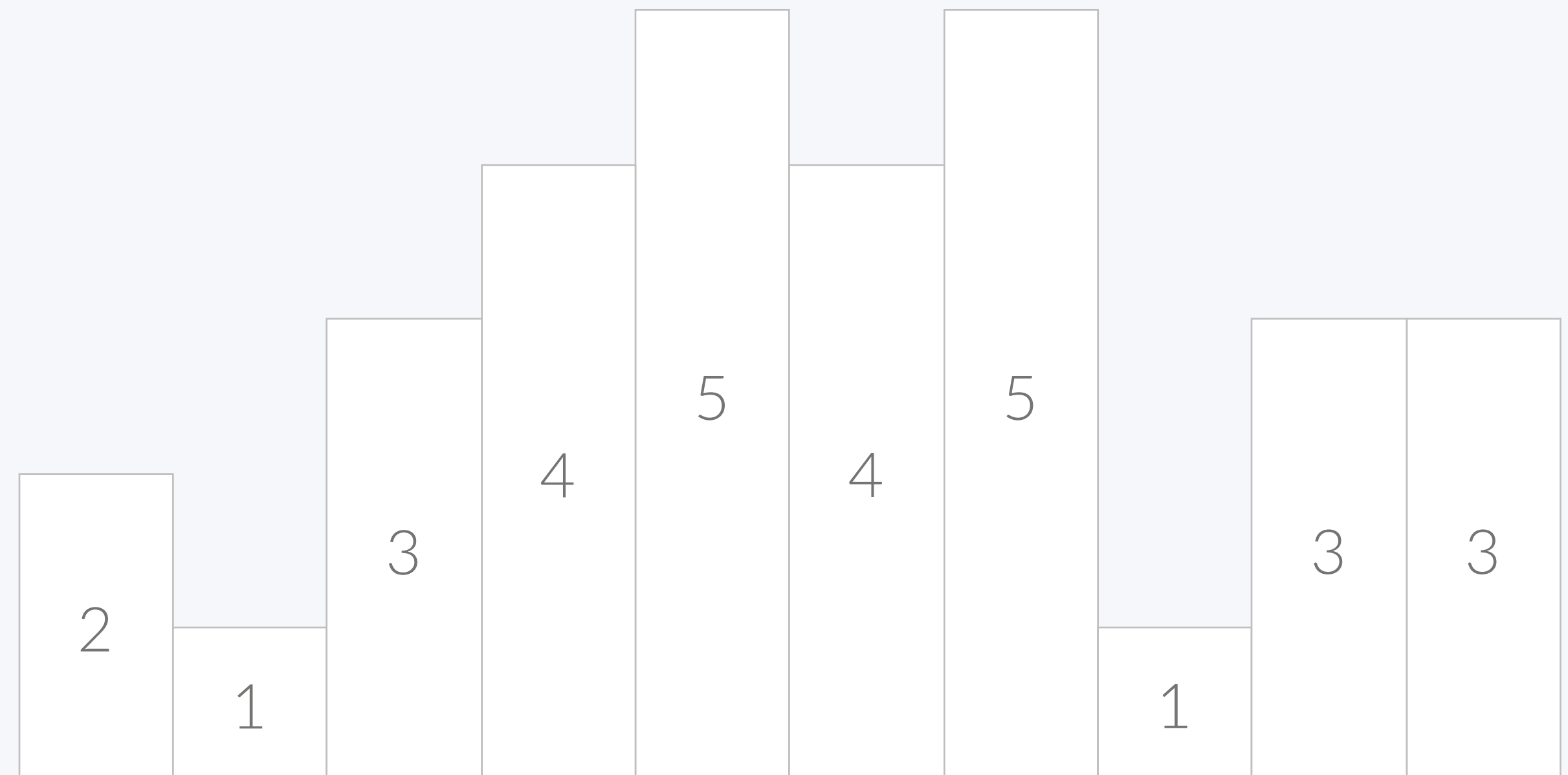


# 히스토그램에서 가장 큰 직사각형

49

<https://www.acmicpc.net/problem/6549>

- 5번 막대, 높이 4
- 스택의 가장 위에 있는 막대 3보다 높이가 크거나 같기 때문에 push
- 스택: 1 2 3 5

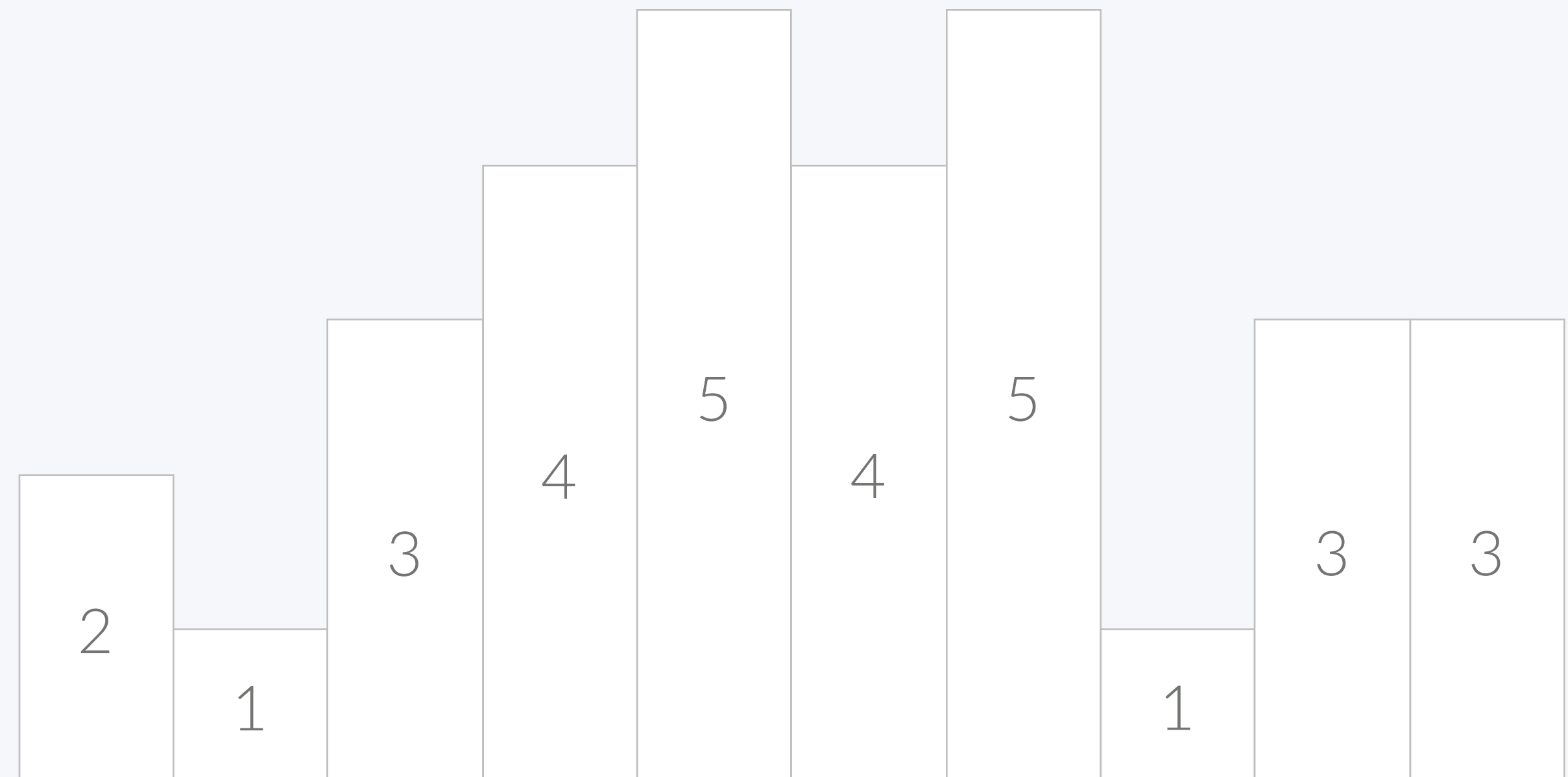


# 히스토그램에서 가장 큰 직사각형

50

<https://www.acmicpc.net/problem/6549>

- 6번 막대, 높이 5
- 스택의 가장 위에 있는 막대 5보다 높이가 크거나 같기 때문에 push
- 스택: 1 2 3 5 6

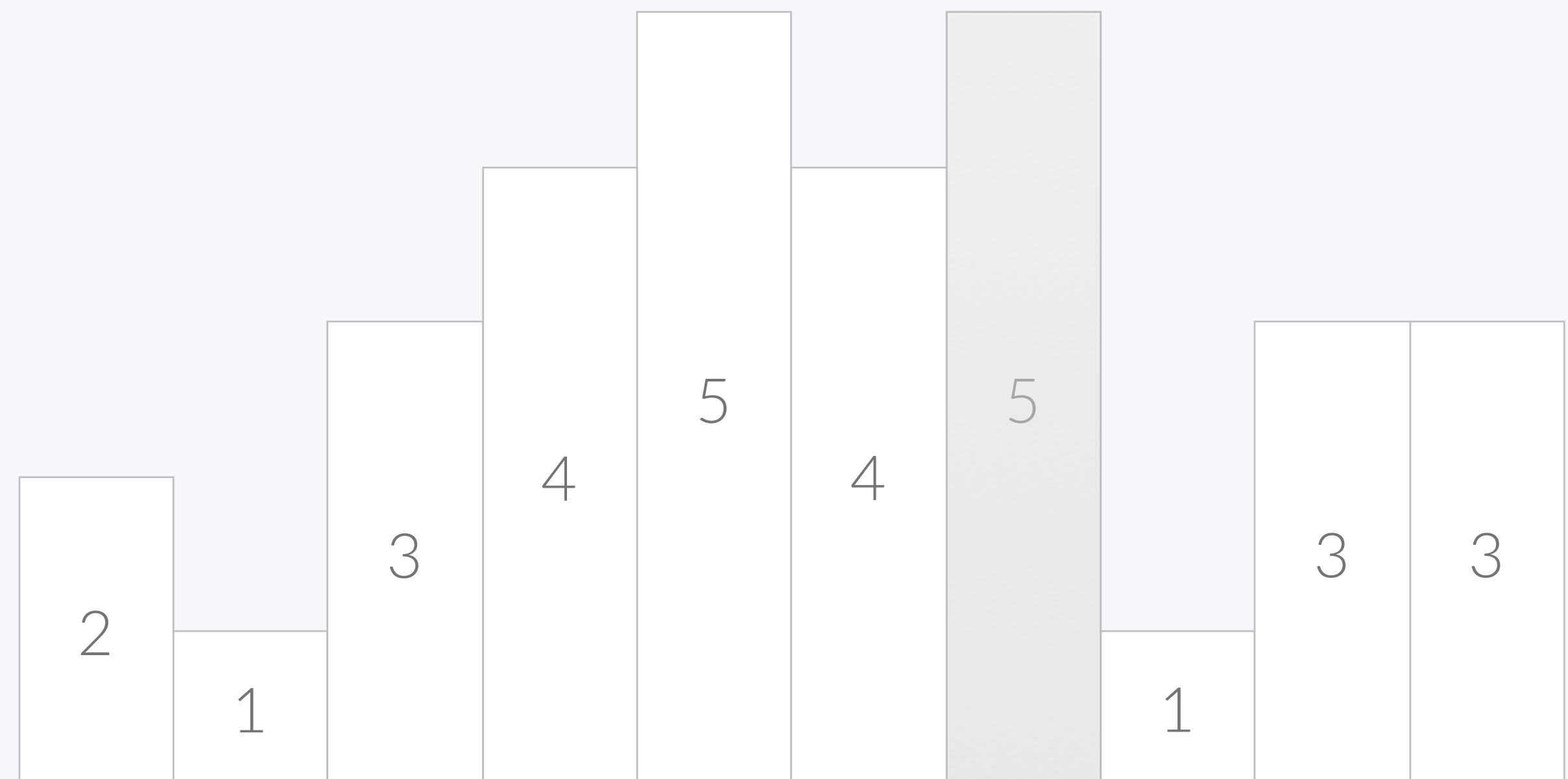


# 히스토그램에서 가장 큰 직사각형

51

<https://www.acmicpc.net/problem/6549>

- 7번 막대, 높이 1
- 스택의 가장 위에 있는 막대 6번 (높이 5)이 현재 높이보다 크다
- $\text{right} = 7 - 1 = 6$ ,  $\text{left} = 5 + 1 = 6$
- 6번 막대로 만들 수 있는 직사각형 넓이: 5
- 스택: 1 2 3 5

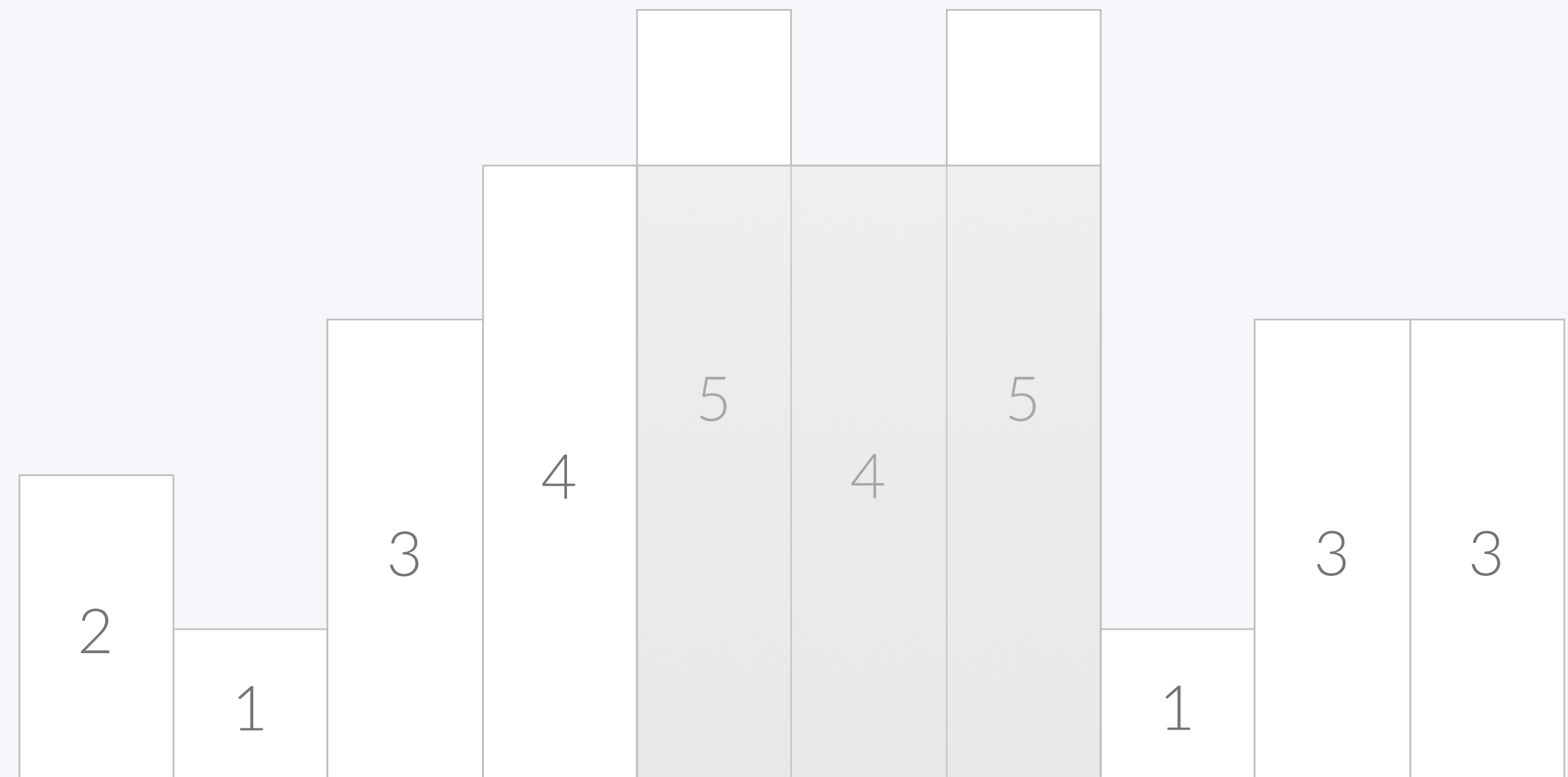


# 히스토그램에서 가장 큰 직사각형

52

<https://www.acmicpc.net/problem/6549>

- 7번 막대, 높이 1
- 스택의 가장 위에 있는 막대 5번 (높이 4)이 현재 높이보다 크다
- $\text{right} = 7 - 1 = 6$ ,  $\text{left} = 3 + 1 = 4$
- 5번 막대로 만들 수 있는 직사각형 넓이: 12
- 스택: 1 2 3

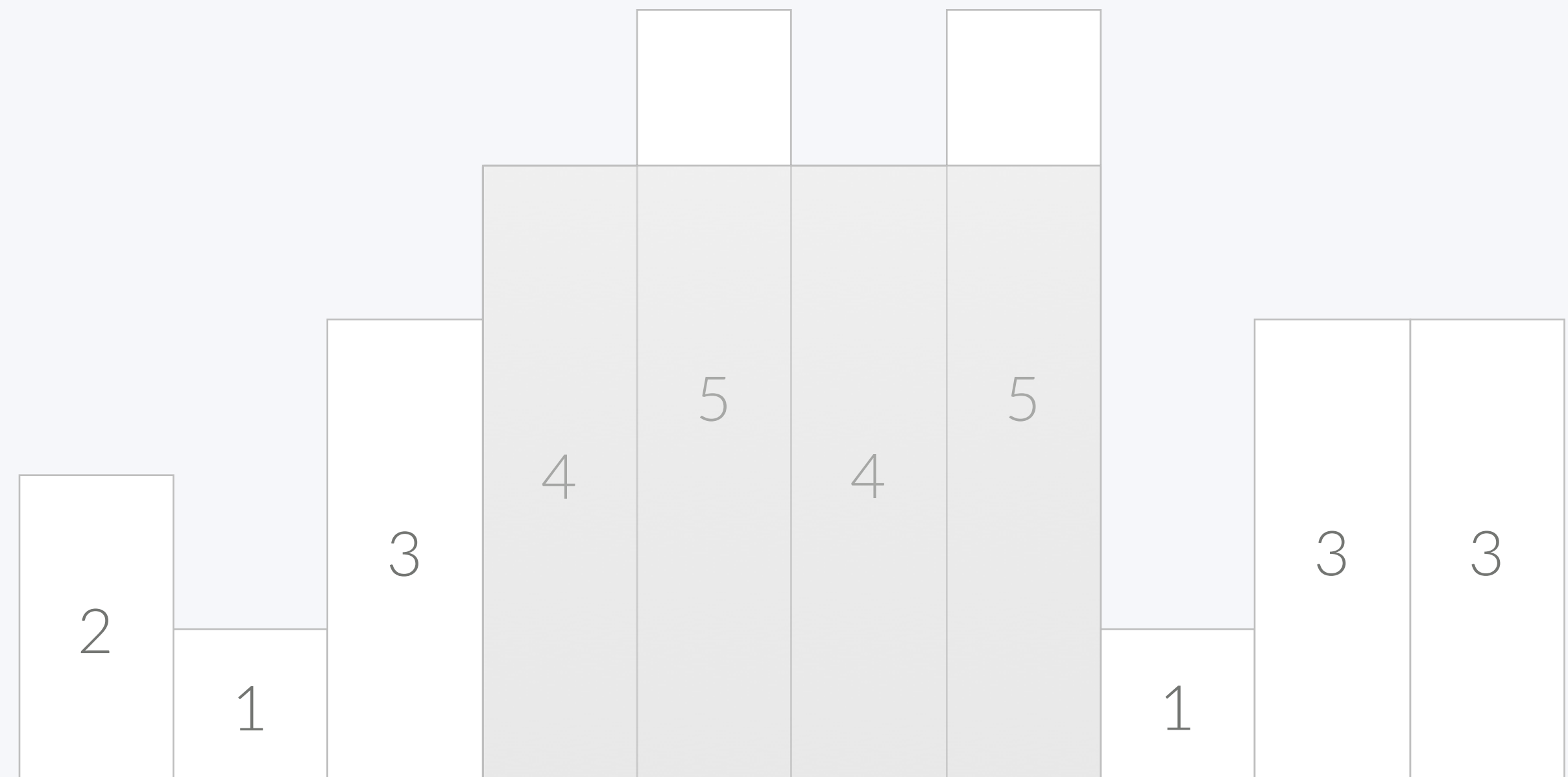


# 히스토그램에서 가장 큰 직사각형

53

<https://www.acmicpc.net/problem/6549>

- 7번 막대, 높이 1
- 스택의 가장 위에 있는 막대 3번 (높이 4)이 현재 높이보다 크다
- $\text{right} = 7 - 1 = 6$ ,  $\text{left} = 2 + 1 = 3$
- 3번 막대로 만들 수 있는 직사각형 넓이: 16
- 스택: 1 2

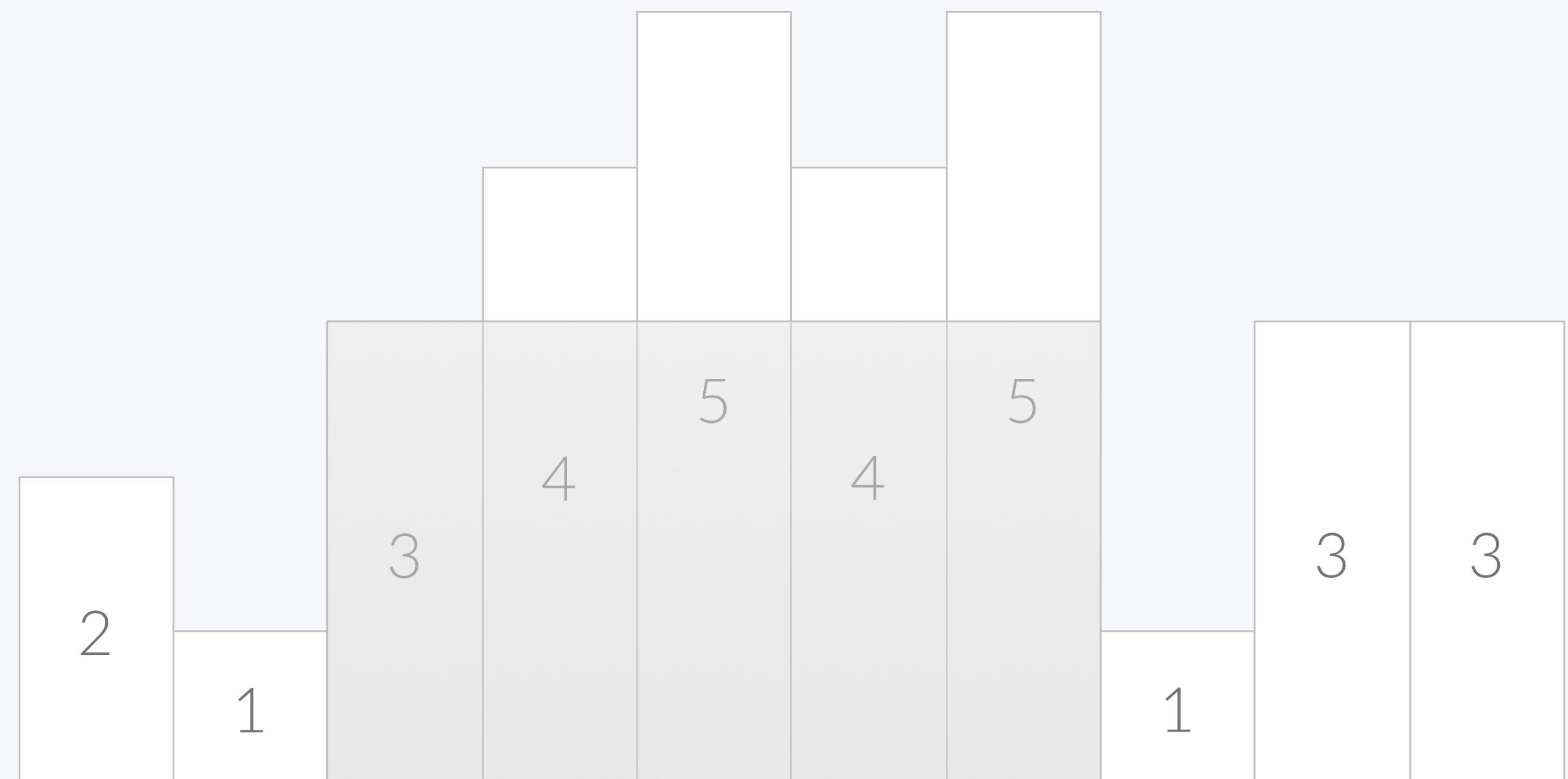


# 히스토그램에서 가장 큰 직사각형

54

<https://www.acmicpc.net/problem/6549>

- 7번 막대, 높이 1
- 스택의 가장 위에 있는 막대 2번 (높이 3)이 현재 높이보다 크다
- $\text{right} = 7 - 1 = 6$ ,  $\text{left} = 1 + 1 = 2$
- 2번 막대로 만들 수 있는 직사각형 넓이: 15
- 스택: 1

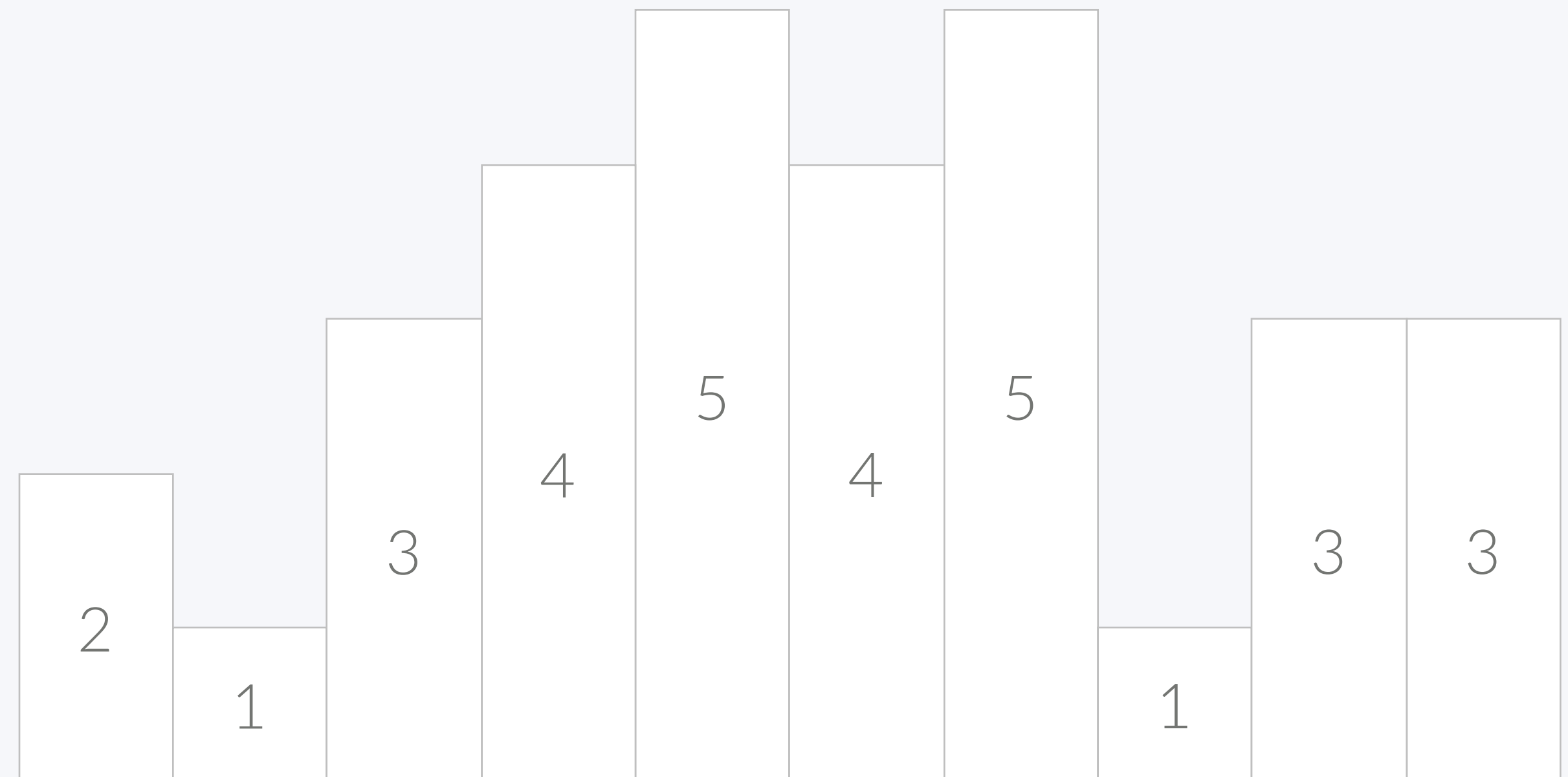


# 히스토그램에서 가장 큰 직사각형

55

<https://www.acmicpc.net/problem/6549>

- 7번 막대, 높이 1
- 스택의 가장 위에 있는 막대 1번 (높이 1)이 현재 높이보다 크거나 같기 때문에
- 스택에 현재 막대 번호 7번을 push
- 스택: 1 7

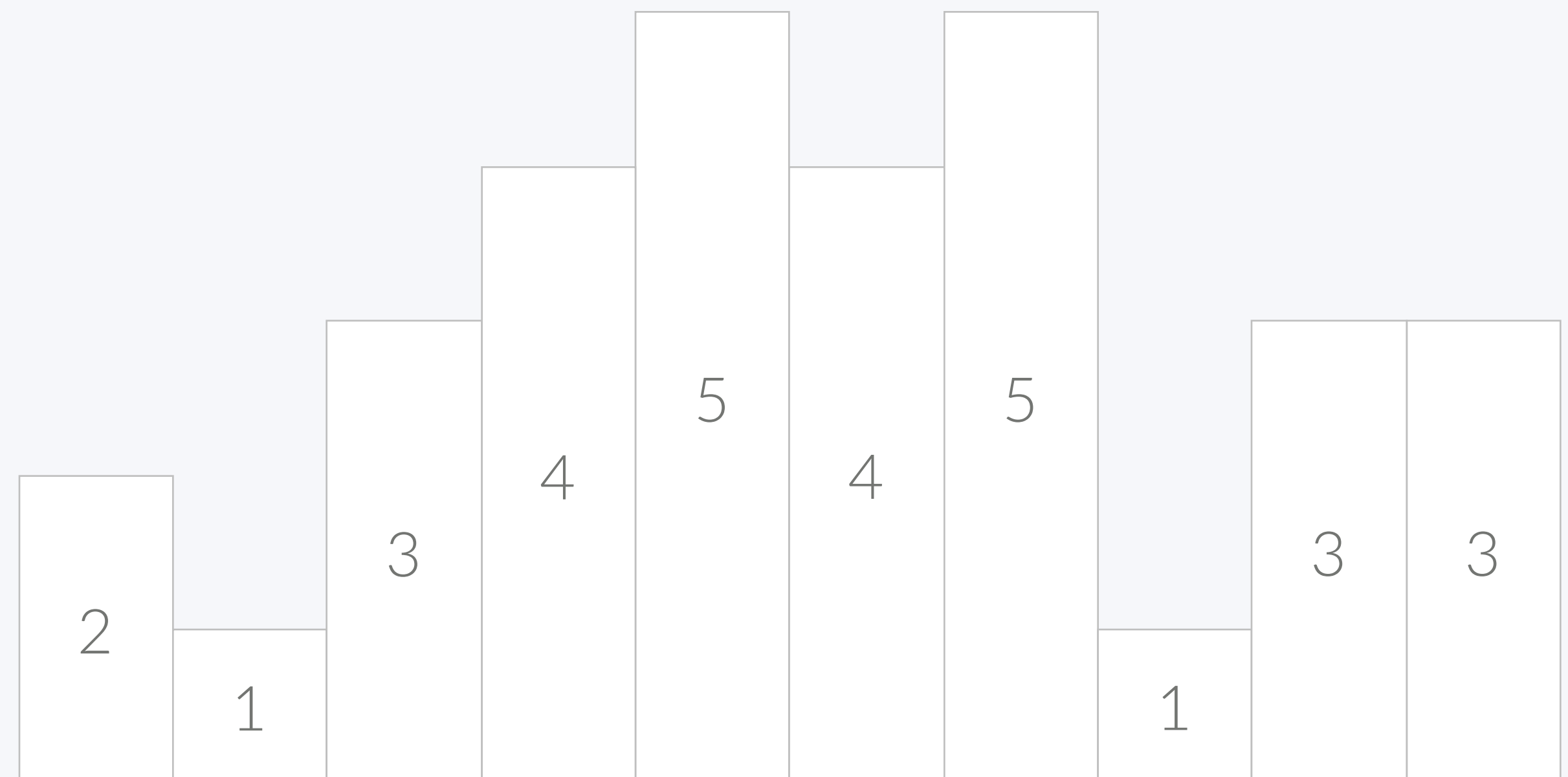


# 히스토그램에서 가장 큰 직사각형

56

<https://www.acmicpc.net/problem/6549>

- 8번 막대, 높이 3
- 스택의 가장 위에 있는 막대 7번 (높이 1)이 현재 높이보다 크거나 같기 때문에
- 스택에 현재 막대 번호 8번을 push
- 스택: 1 7 8



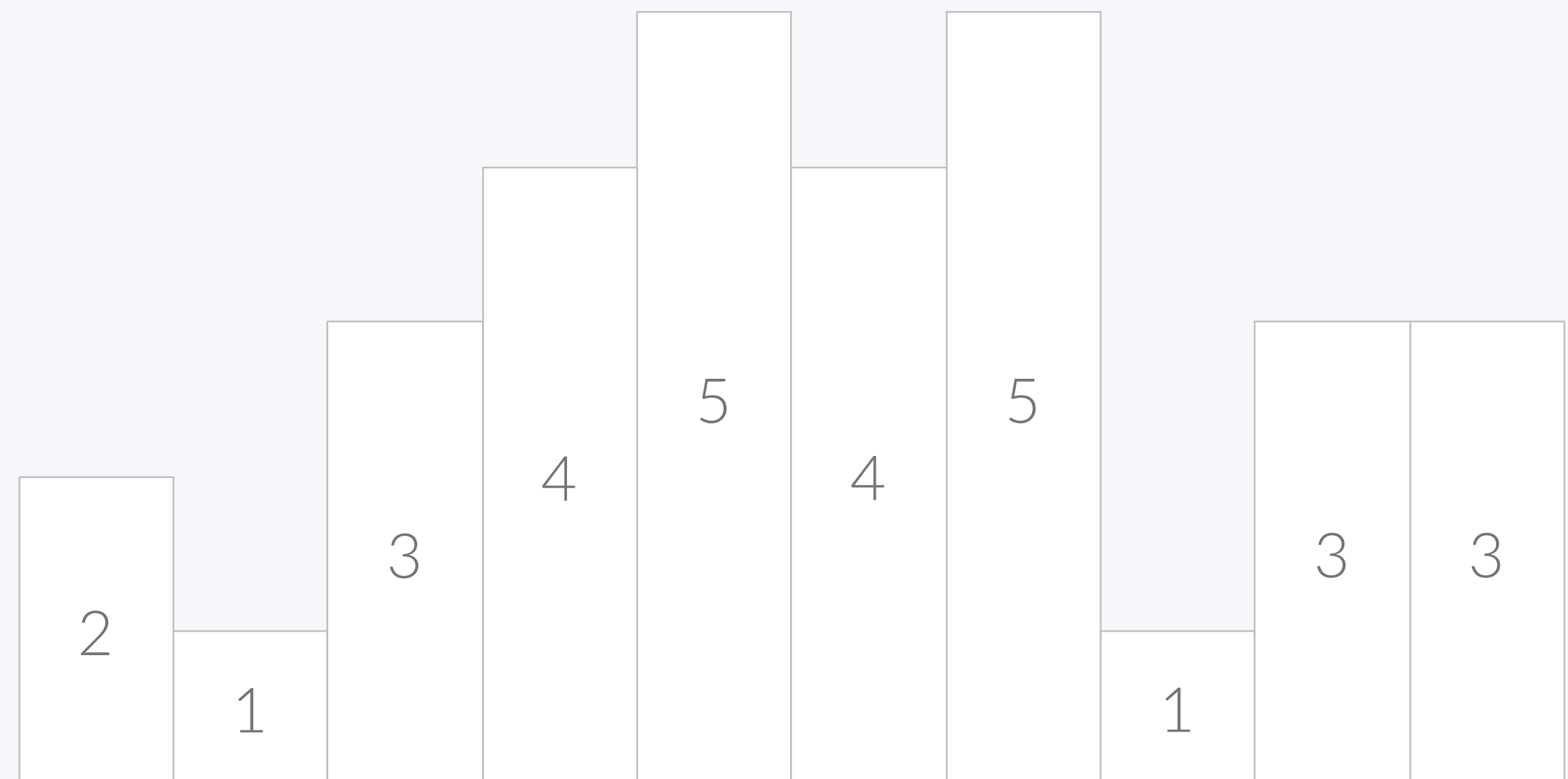


# 히스토그램에서 가장 큰 직사각형

57

<https://www.acmicpc.net/problem/6549>

- 9번 막대, 높이 3
- 스택의 가장 위에 있는 막대 8번 (높이 3)이 현재 높이보다 크거나 같기 때문에
- 스택에 현재 막대 번호 9번을 push
- 스택: 1 7 8 9

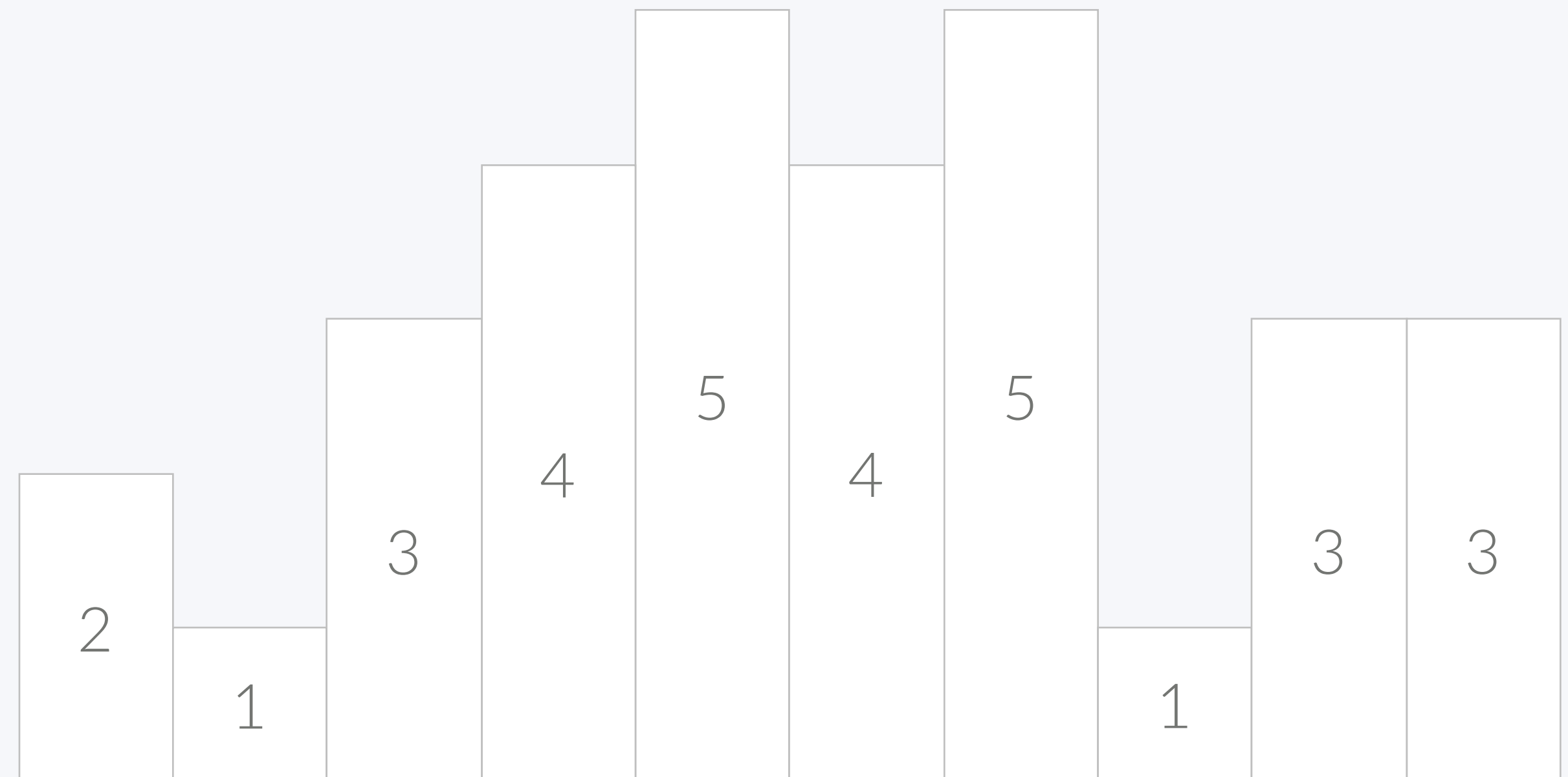


# 히스토그램에서 가장 큰 직사각형

58

<https://www.acmicpc.net/problem/6549>

- 모든 막대가 스택에 들어갔고
- 지금부터는  $\text{right} = n-1$  인 경우를 처리할 차례

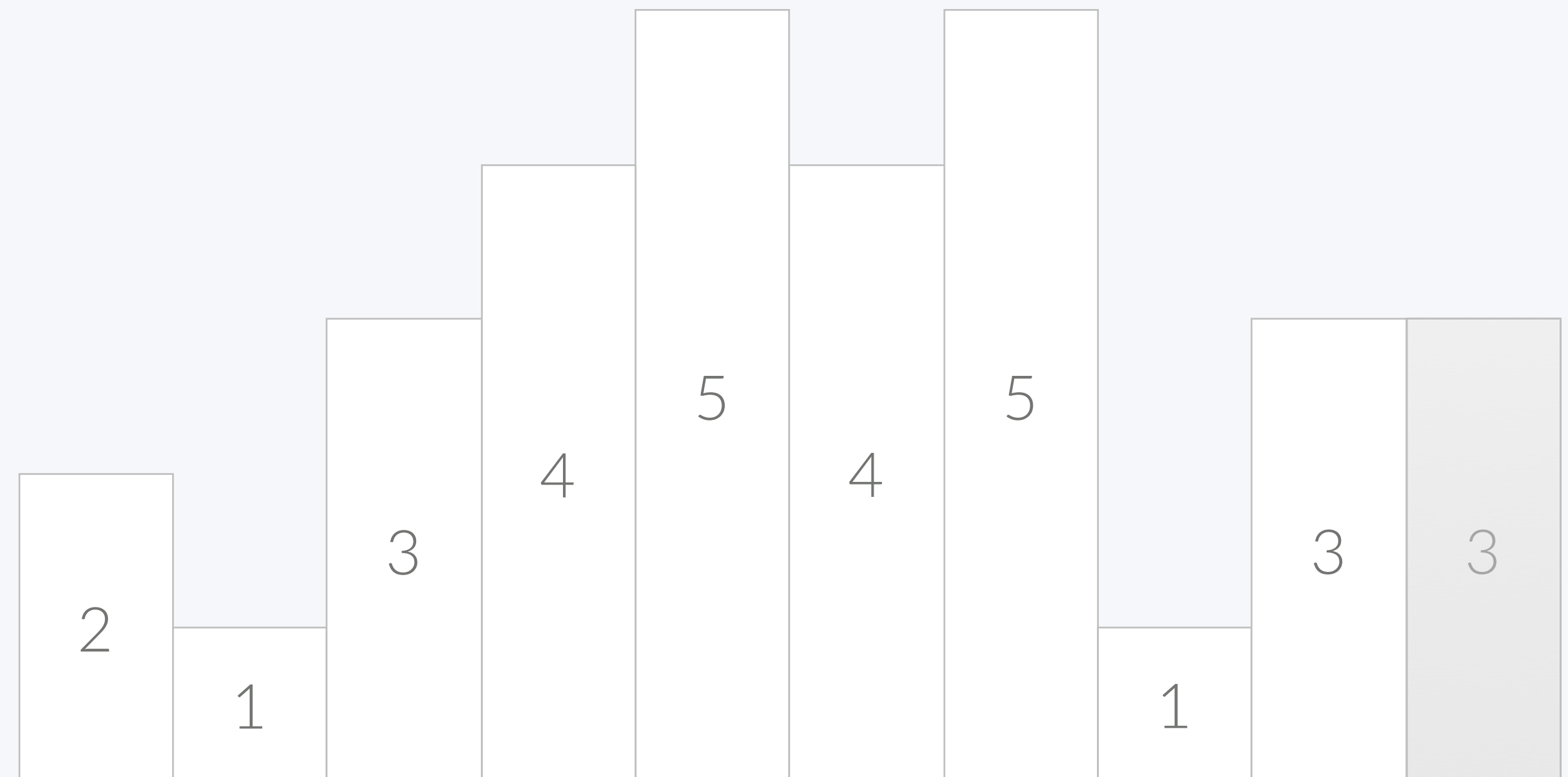


# 히스토그램에서 가장 큰 직사각형

59

<https://www.acmicpc.net/problem/6549>

- 스택: 1 7 8 9
- 9로 만들 수 있는 가장 큰 직사각형
- $\text{Right} = 9$
- $\text{Left} = 8 + 1 = 9$

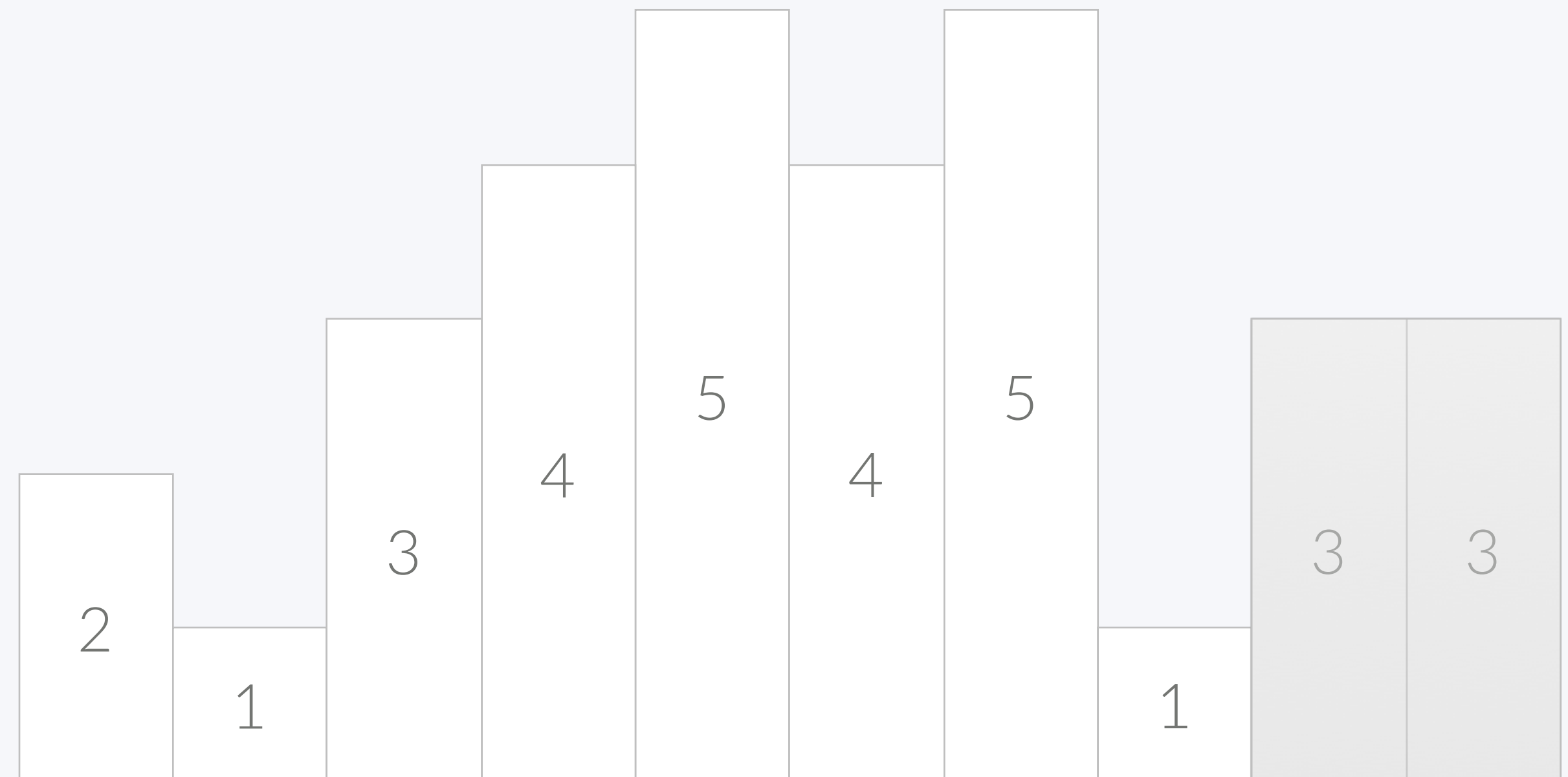


# 히스토그램에서 가장 큰 직사각형

60

<https://www.acmicpc.net/problem/6549>

- 스택: 1 7 8
- 8로 만들 수 있는 가장 큰 직사각형
- $\text{Right} = 9$
- $\text{Left} = 7 + 1 = 8$

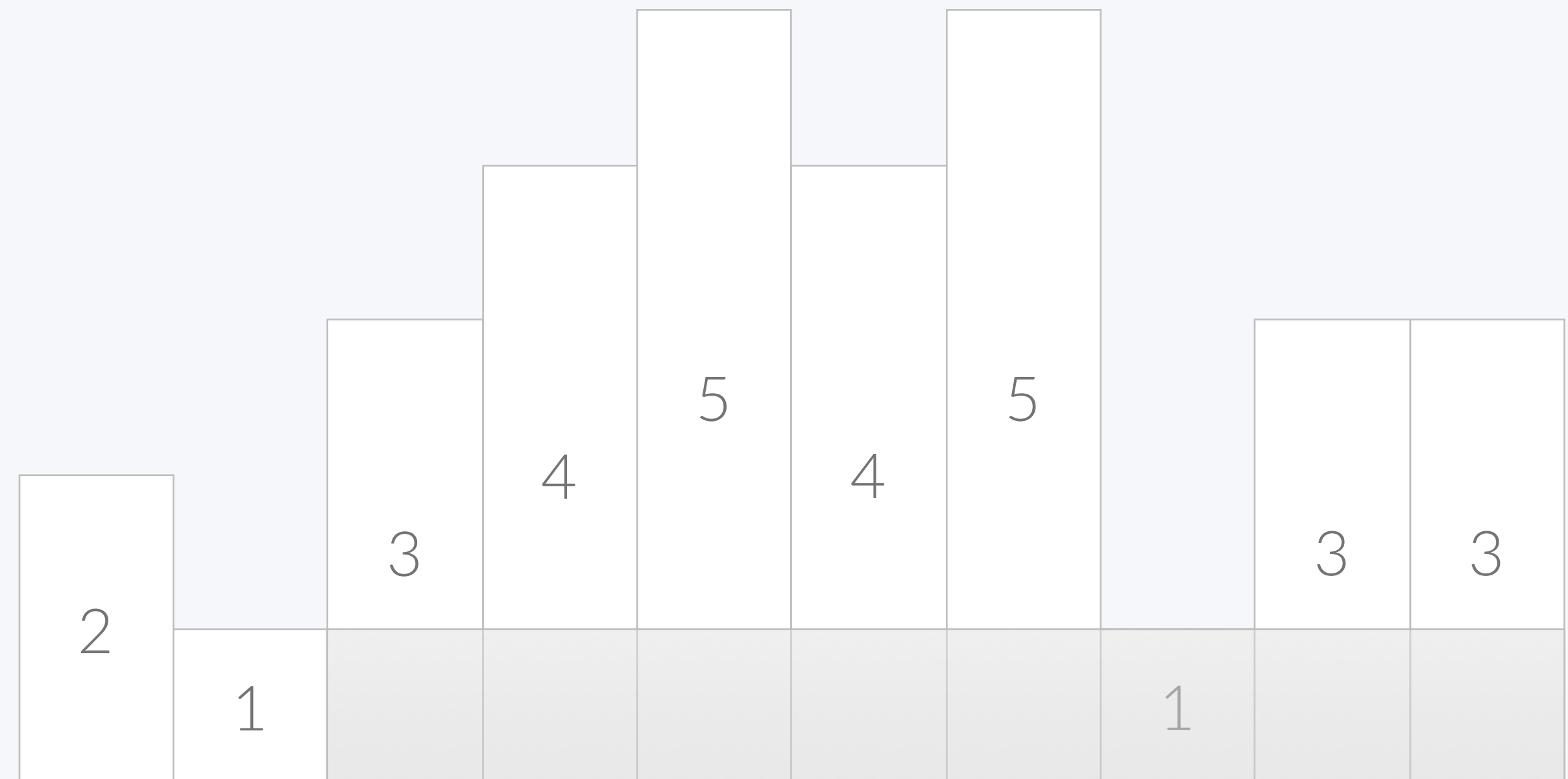


# 히스토그램에서 가장 큰 직사각형

61

<https://www.acmicpc.net/problem/6549>

- 스택: 1 7
- 7로 만들 수 있는 가장 큰 직사각형
- $\text{Right} = 9$
- $\text{Left} = 1 + 1 = 2$

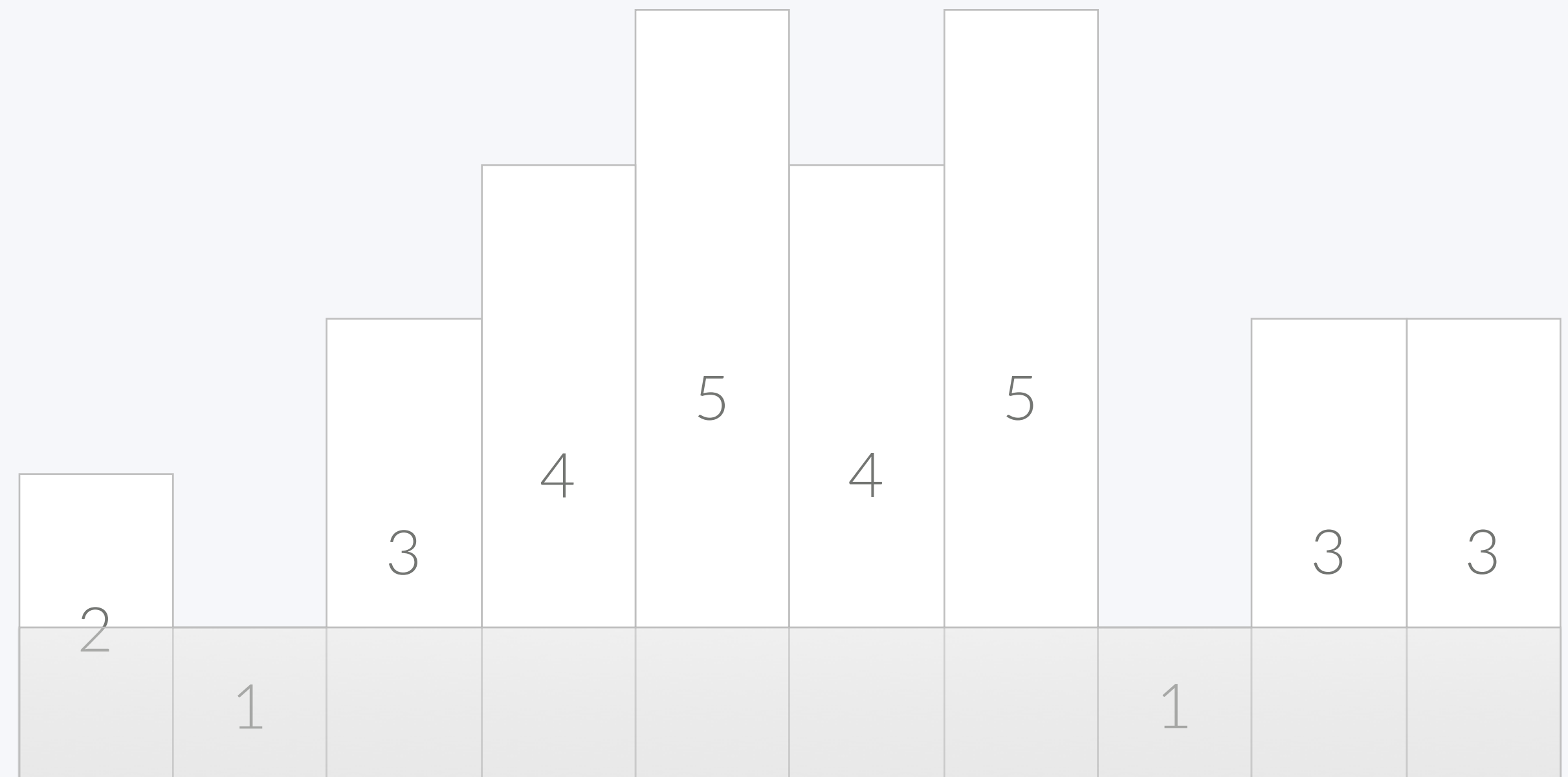


# 히스토그램에서 가장 큰 직사각형

62

<https://www.acmicpc.net/problem/6549>

- 스택: 1
- 1로 만들 수 있는 가장 큰 직사각형
- Right = 9
- Left = 0



# 히스토그램에서 가장 큰 직사각형

<https://www.acmicpc.net/problem/6549>

```
stack<int> s;
int ans = 0;
for (int i=0; i<n; i++) {
    int left = i;
    while(!s.empty() && a[s.top()] > a[i]) {
        int height = a[s.top()];
        s.pop();
        int width = i;
        if (!s.empty()) width = (i - s.top() - 1);
        if (ans < width*height) ans = width*height;
    }
    s.push(i);
}
```

# 히스토그램에서 가장 큰 직사각형

<https://www.acmicpc.net/problem/6549>

```
while (!s.empty()) {  
    int height = a[s.top()];  
    s.pop();  
    int width = n;  
    if (!s.empty()) {  
        width = n-s.top()-1;  
    }  
    if (ans < width*height) {  
        ans = width*height;  
    }  
}
```



# 히스토그램에서 가장 큰 직사각형

65

<https://www.acmicpc.net/problem/6549>

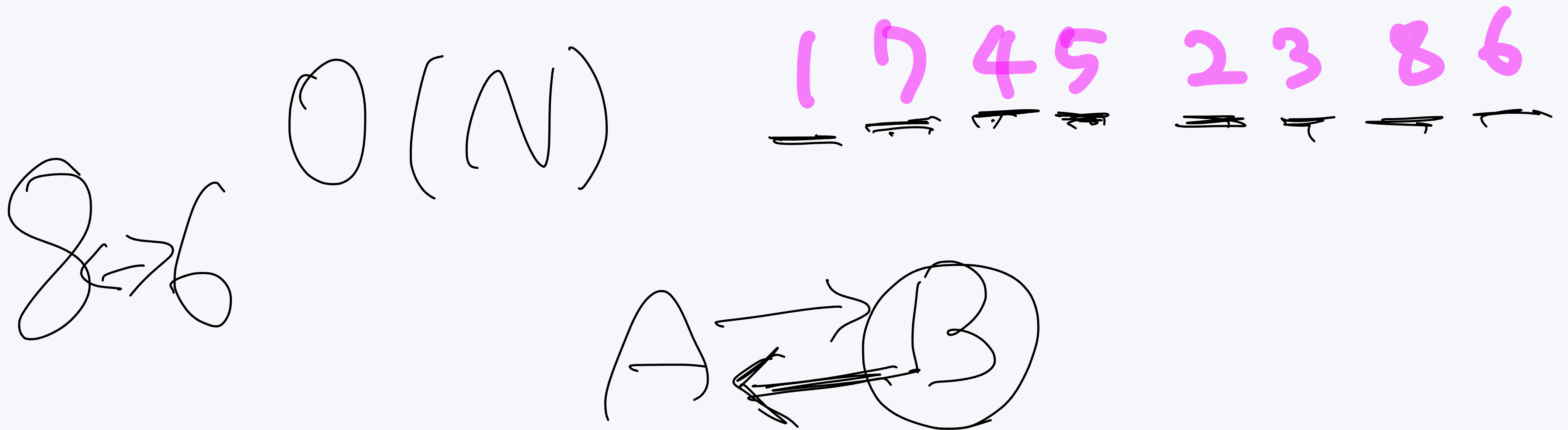
- C/C++: <https://gist.github.com/Baekjoon/12d18034c2cd01a8d0ad>

# 오아시스 재결합

<https://www.acmicpc.net/problem/3015>

66

- N명이 한 줄로 줄을 서있다
- A와 B가 서로 볼 수 있으려면, 두 사람 사이에 A 또는 B보다 키가 큰 사람이 없어야 한다
- 줄에 서있는 사람의 키가 주어졌을 때, 서로 볼 수 있는 쌍의 수 구하기



# 오아시스 재결합

<https://www.acmicpc.net/problem/3015>

- 먼저, 문제에 나와있지 않은 조건을 하나 추가해서 문제를 푼다
- 모든 사람의 키가 모두 서로 다르다고 가정
- A 뒤에 B가 줄을 섰다
- B가 A보다 키가 크다
- .....A.....B.....
- B의 뒤에 있는 사람들은 절대로 A를 볼 수 없다

# 오아시스 재결합

<https://www.acmicpc.net/problem/3015>

- 한 명씩 줄을 서는 경우를 생각
- 줄을 설 때마다 자기 앞에 총 몇 명을 볼 수 있는지 계산해보기
- 스택에 들어있는 사람은 뒤에 줄을 서는 사람이 볼 수도 있는 사람
- 스택에서 이미 나온 사람은 뒤에 줄을 서는 사람이 절대 볼 수 없는 사람
- 따라서, 스택에는 항상 키가 작아지는 순으로 저장되게 됨
- 즉, 스택의 top이 스택에 들어있는 사람 중에서 키가 가장 작은 사람

# 오아시스 재결합

<https://www.acmicpc.net/problem/3015>

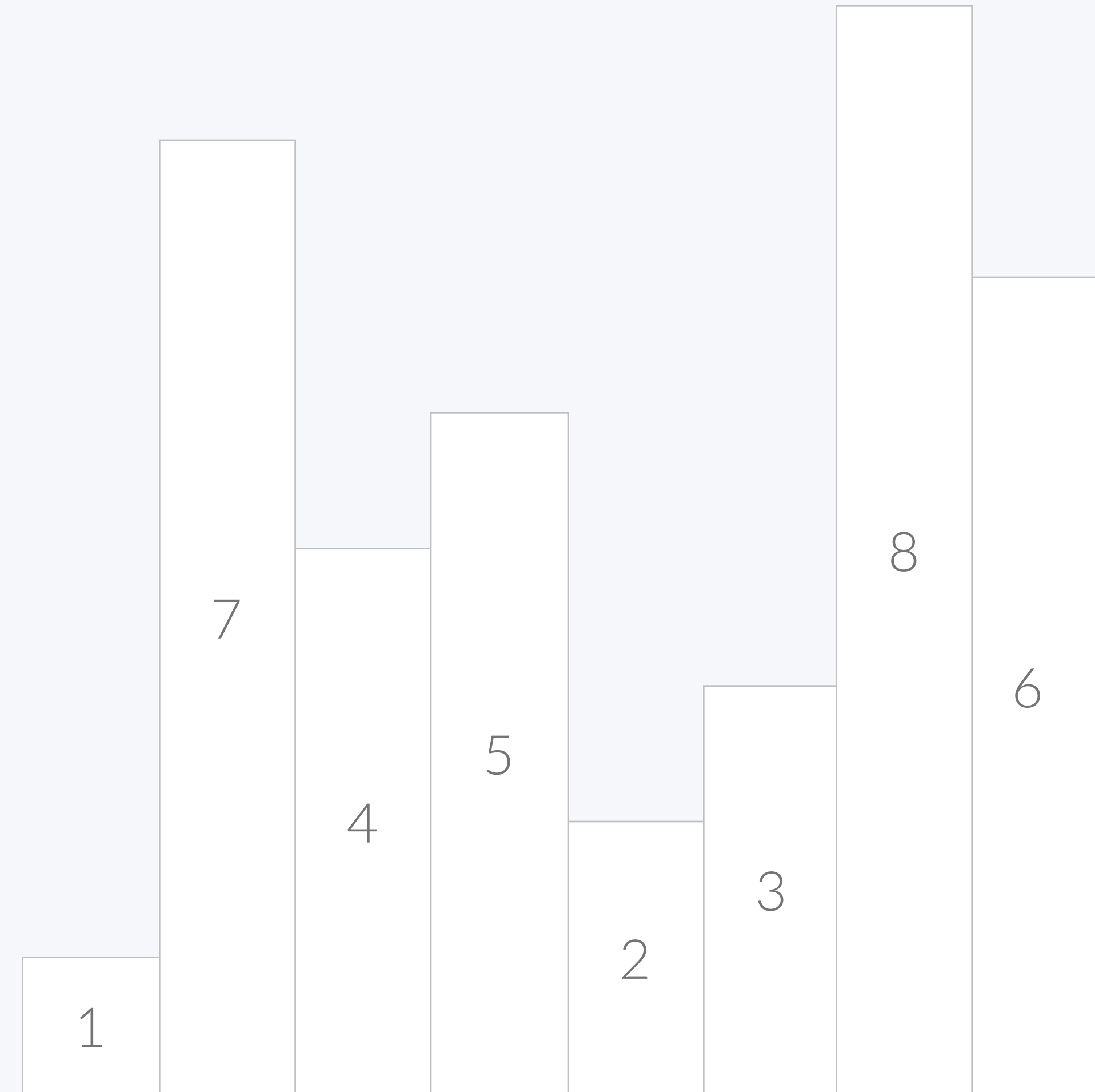
- 각 사람 A가 들어올 때마다 다음을 검사한다
  - 스택의 가장 위에 있는 사람 S와 키를 비교
  - S보다 A가 키가 크면
  - A의 뒤에 줄을 서는 사람은 절대로 A를 볼 수 없음
  - 따라서, 정답에 1을 더하고
  - S를 스택에서 빼고 위의 과정을 반복
- 이 때, 스택이 비어있지 않으면 정답에 1을 더한다
  - 왜냐하면, 이제 스택의 가장 위에 있는 사람은 A보다 키가 큰 사람인데, 그 사람은 볼 수 있기 때문
- 이제, A를 스택에 추가한다.

# 오아시스 재결합

70

<https://www.acmicpc.net/problem/3015>

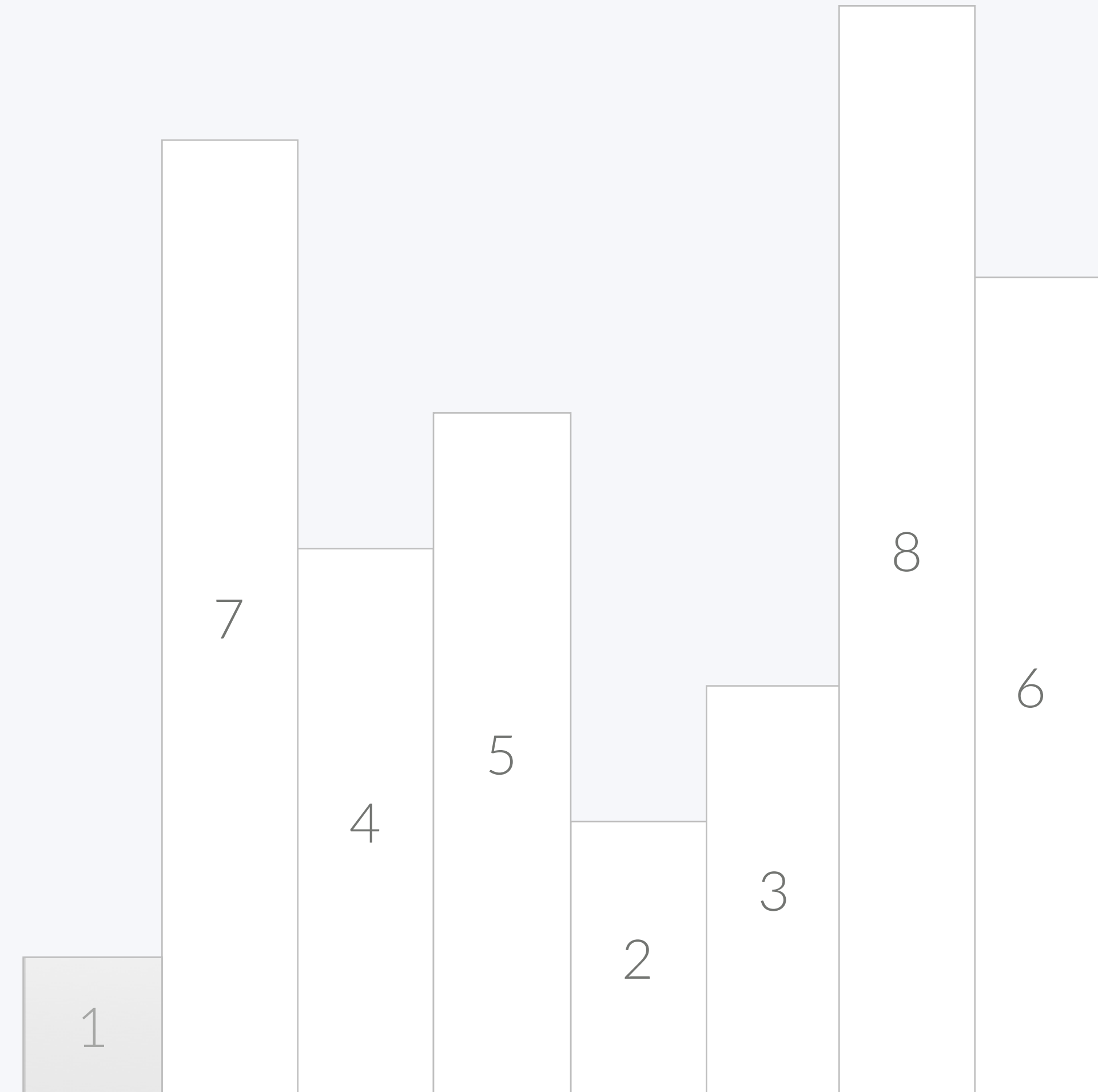
- 1 7 4 5 2 3 8 6의 경우를 생각해보자
- 볼 수 있는 쌍의 수: 11개



# 오아시스 재결합

<https://www.acmicpc.net/problem/3015>

- 스택: 1

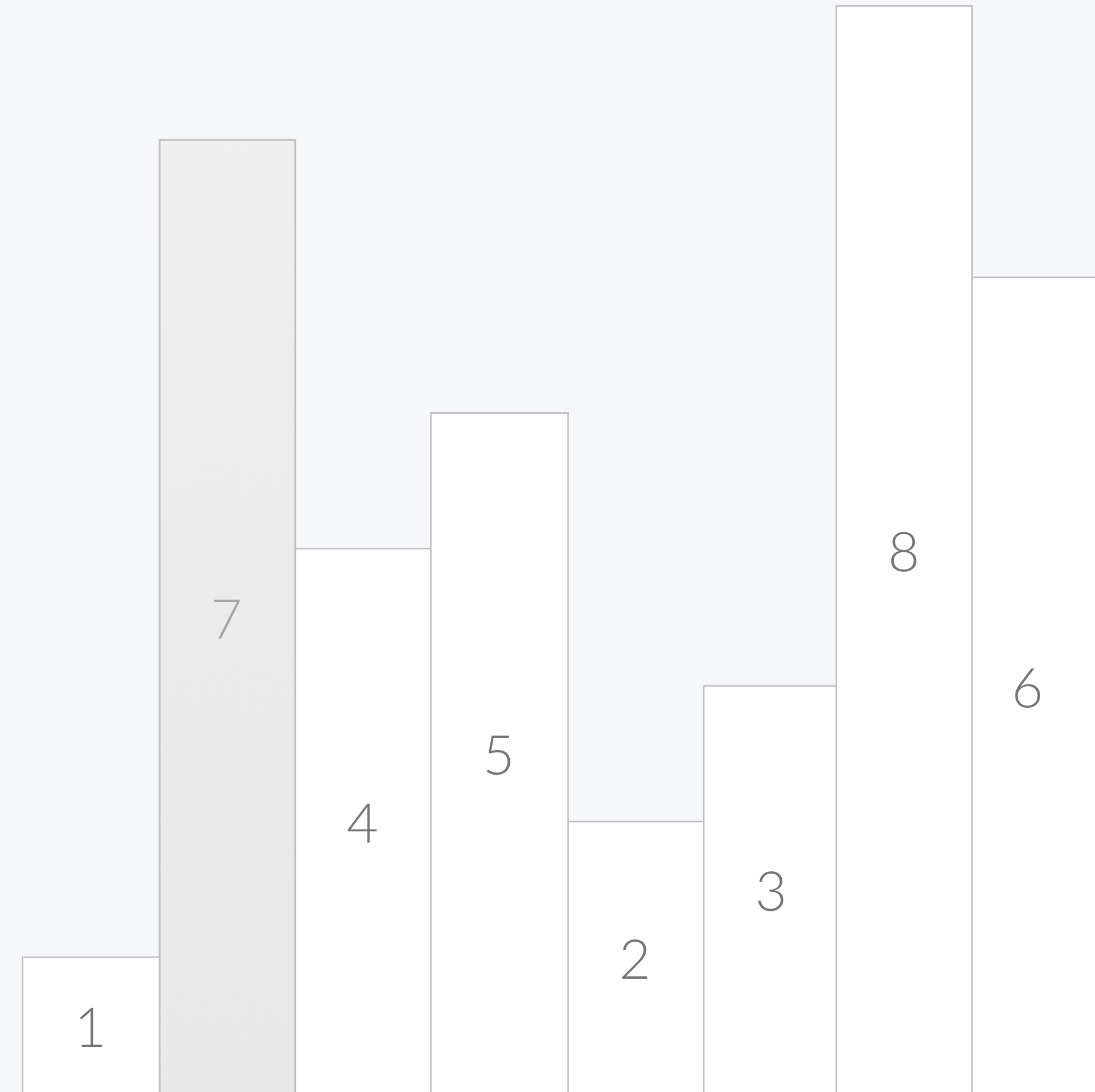


# 오아시스 재결합

72

<https://www.acmicpc.net/problem/3015>

- 스택: 1
- 1은 7보다 키가 작다
- 7의 뒤에 있는 사람은 절대로 1을 볼 수 없다
- 정답에 1을 더해주고 (1과 7)
- 1을 스택에서 제거하고 7을 스택에 추가
- 스택: 7



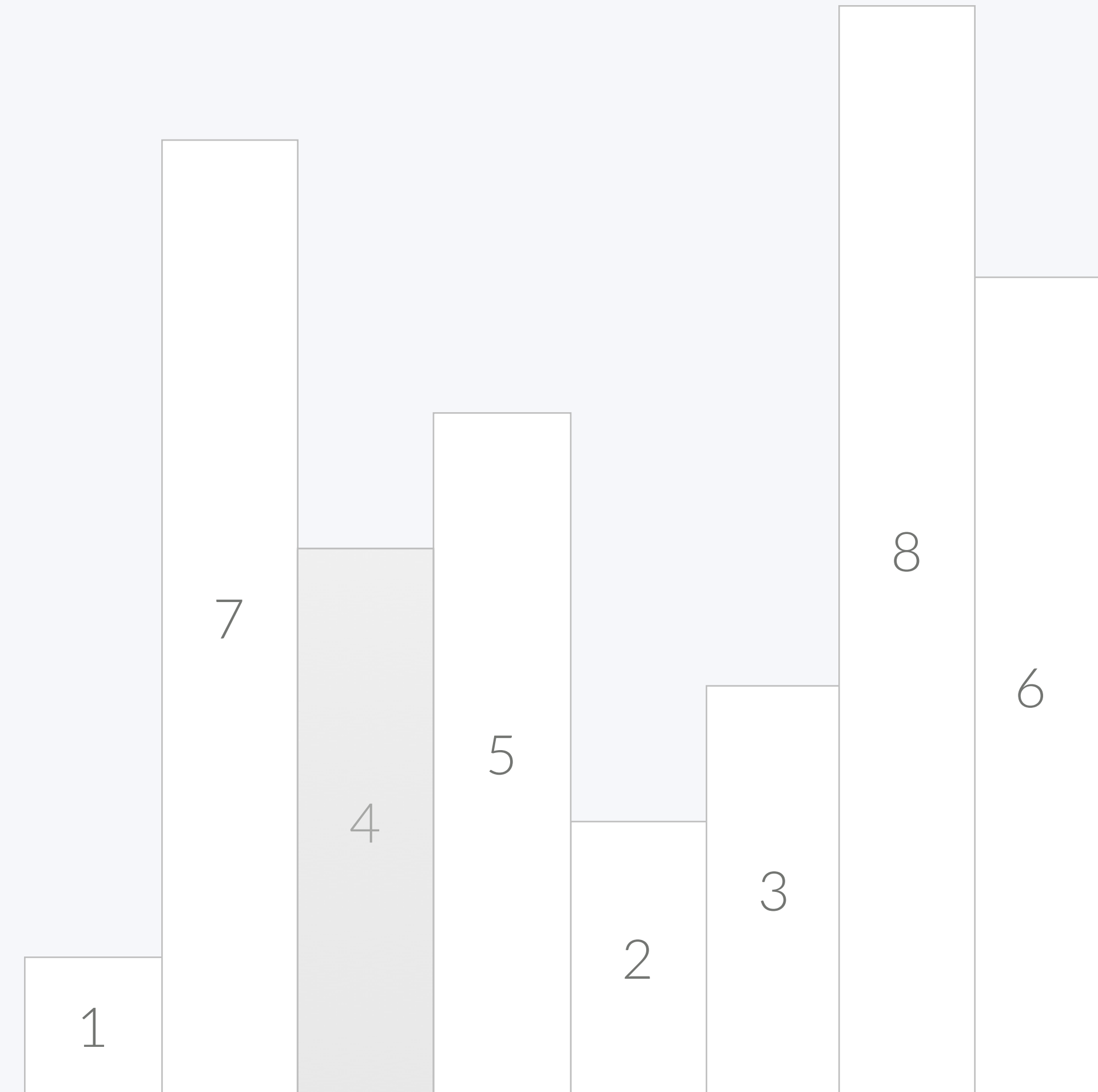


# 오아시스 재결합

73

<https://www.acmicpc.net/problem/3015>

- 스택: 7
- 7은 4보다 키가 크다
- 따라서, 스택에는 아무 일도 일어나지 않는다
- 스택이 비어있지 않기 때문에
- 정답에 1을 추가한다 (7과 4)
- 4를 스택에 추가한다
- 스택: 7 4

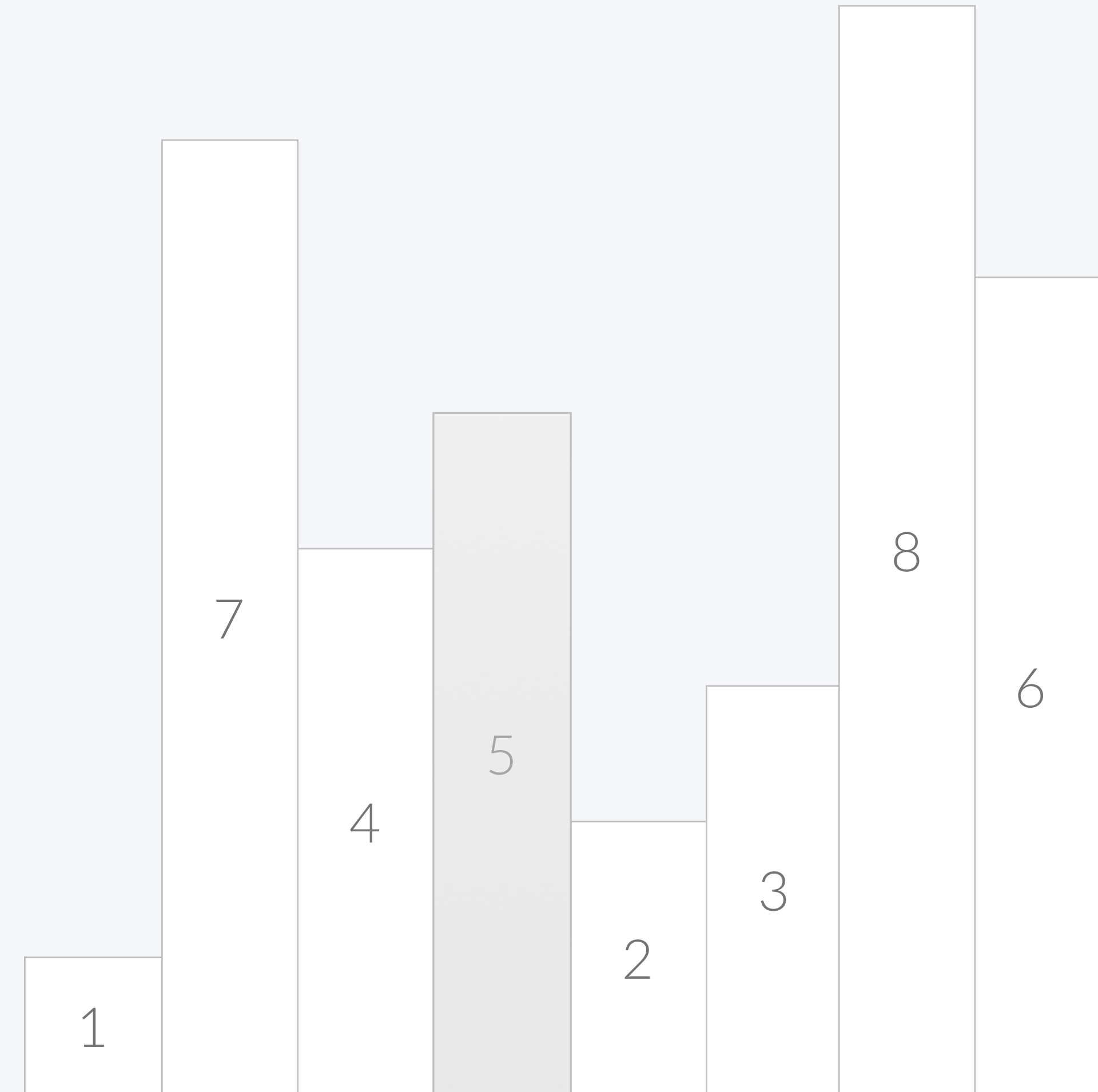


# 오아시스 재결합

74

<https://www.acmicpc.net/problem/3015>

- 스택: 7 4
- 4는 5보다 키가 작다
- 4를 스택에서 제거하고
- 정답에 1을 더한다 (4와 5)
- 스택: 7
- 7은 5보다 키가 크다
- 스택이 비어있지 않기 때문에
- 정답에 1을 더하고 (7과 5)
- 5를 스택에 추가한다

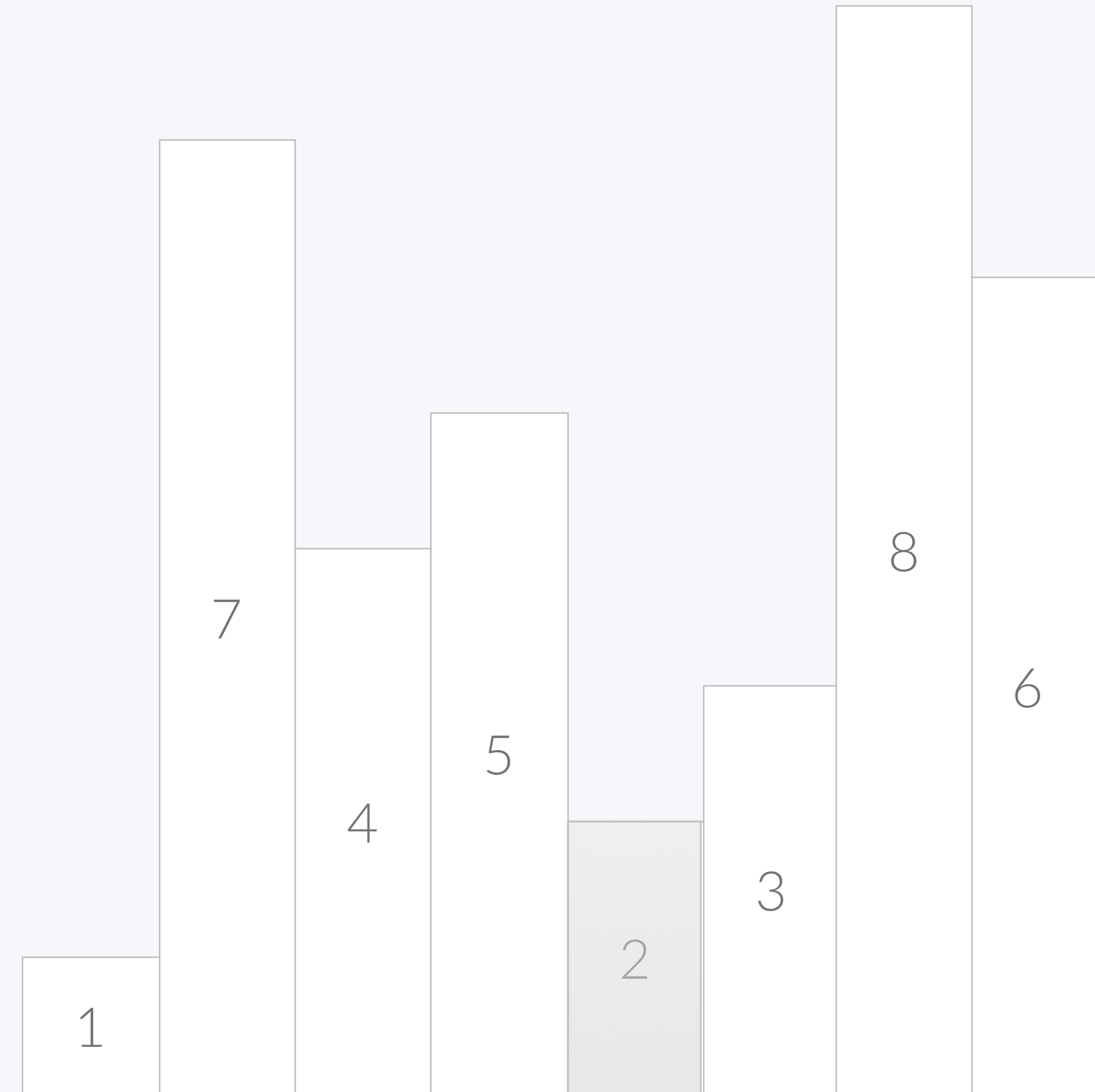


# 오아시스 재결합

75

<https://www.acmicpc.net/problem/3015>

- 스택: 7 5
- 2는 5보다 키가 작기 때문에 스택에 추가한다
- 스택이 비어있지 않기 때문에
- 정답에 1을 더한다 (5와 2)
- 스택: 7 5 2

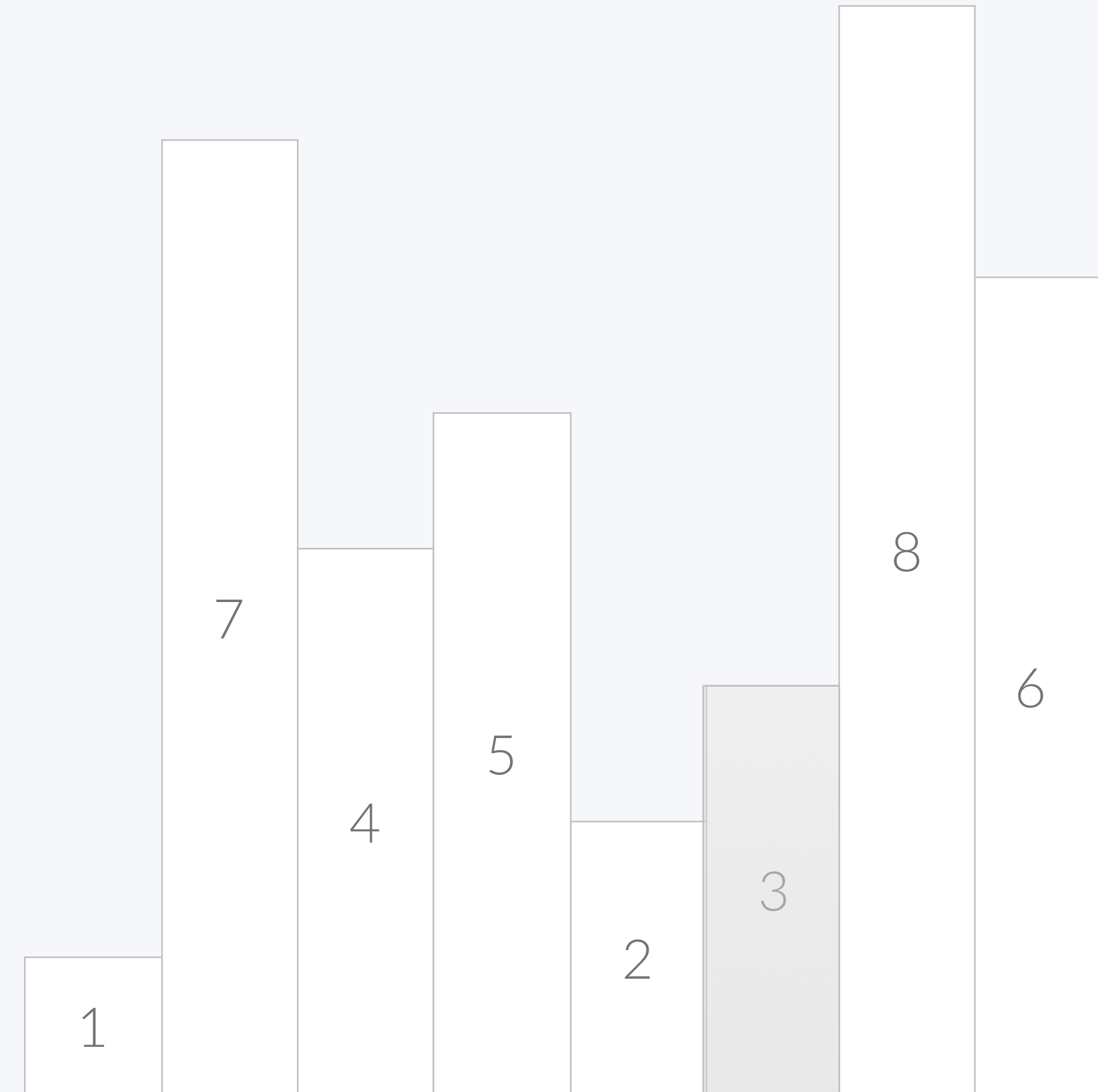


# 오아시스 재결합

76

<https://www.acmicpc.net/problem/3015>

- 스택: 7 5 2
- 2는 3보다 키가 작다.
- 3의 뒤에 있는 사람은 2를 절대로 볼 수가 없다
- 정답에 1을 더해준다 (2와 3)
- 스택: 7 5
- 5는 3보다 키가 크다.
- 3을 스택에 추가하고, 정답에 1을 더한다 (5와 3)
- 스택: 7 5 3

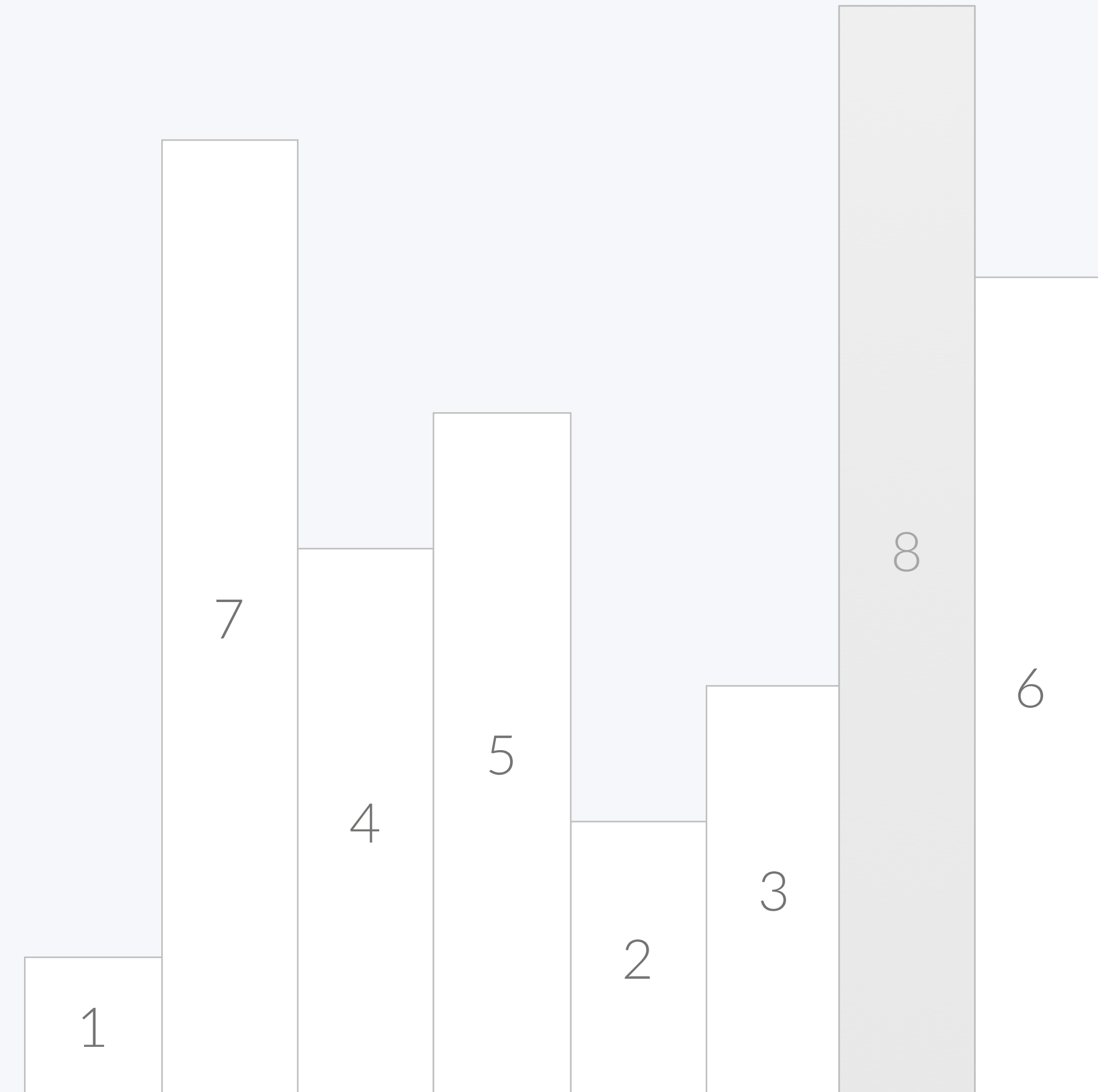


# 오아시스 재결합

77

<https://www.acmicpc.net/problem/3015>

- 스택: 7 5 3
- 8은 3보다 키가 크다
- 8의 뒤에 있는 사람을 3를 볼 수 없다
- 3를 스택에서 빼고 정답에 1을 더한다 (3와 8)
- 스택: 7 5
- 8은 5보다 키가 크다
- 8의 뒤에 있는 사람은 5를 볼 수 없다
- 5를 스택에서 빼고 정답에 1을 더한다 (5와 8)
- 스택: 7

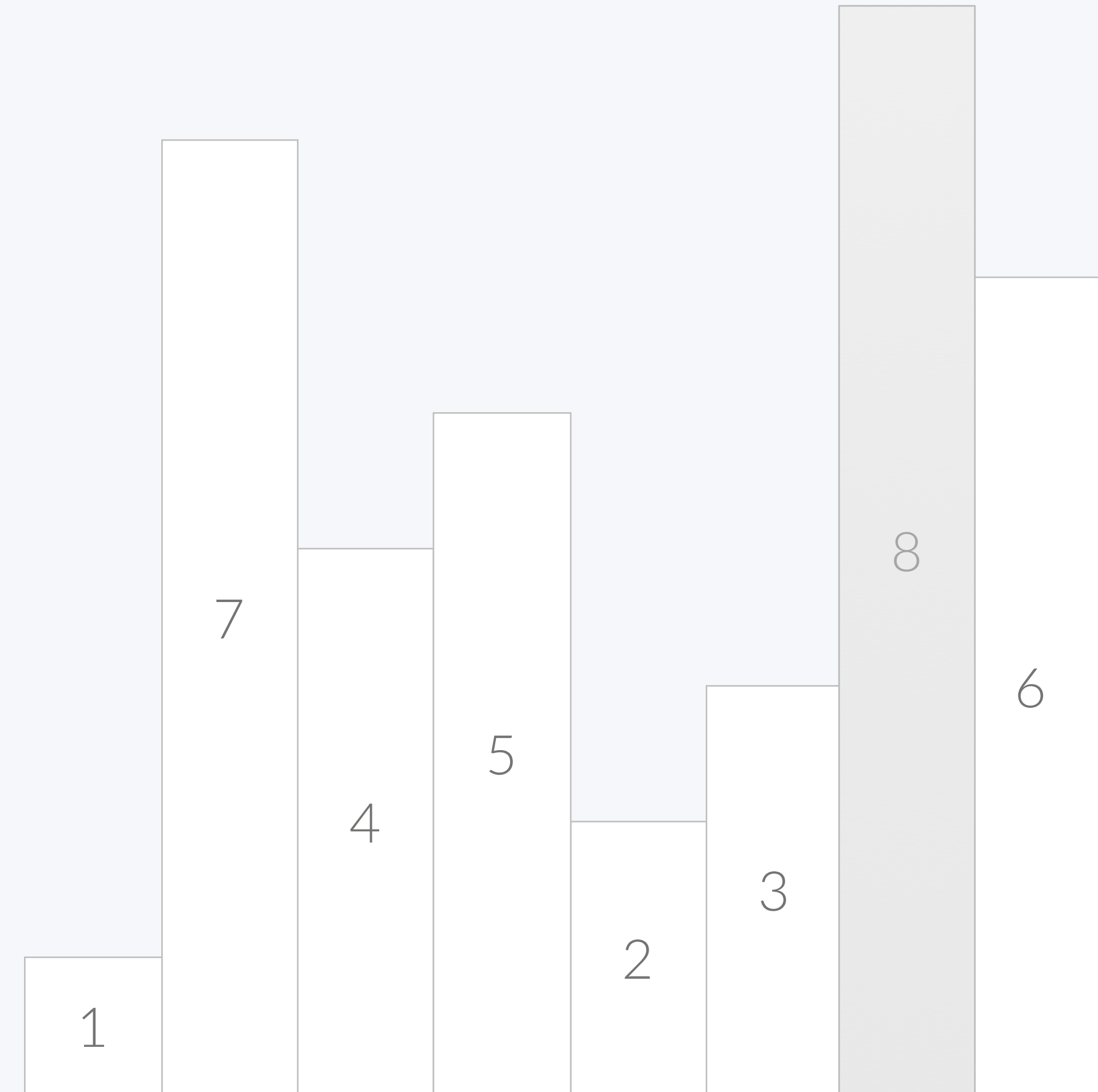


# 오아시스 재결합

78

<https://www.acmicpc.net/problem/3015>

- 스택: 7
- 8은 7보다 키가 크다
- 8의 뒤에 있는 사람을 7를 볼 수 없다
- 7를 스택에서 빼고 정답에 1을 더한다 (7와 8)
- 스택:
- 스택이 비어있으니 이제 8을 넣는다
- 스택: 8

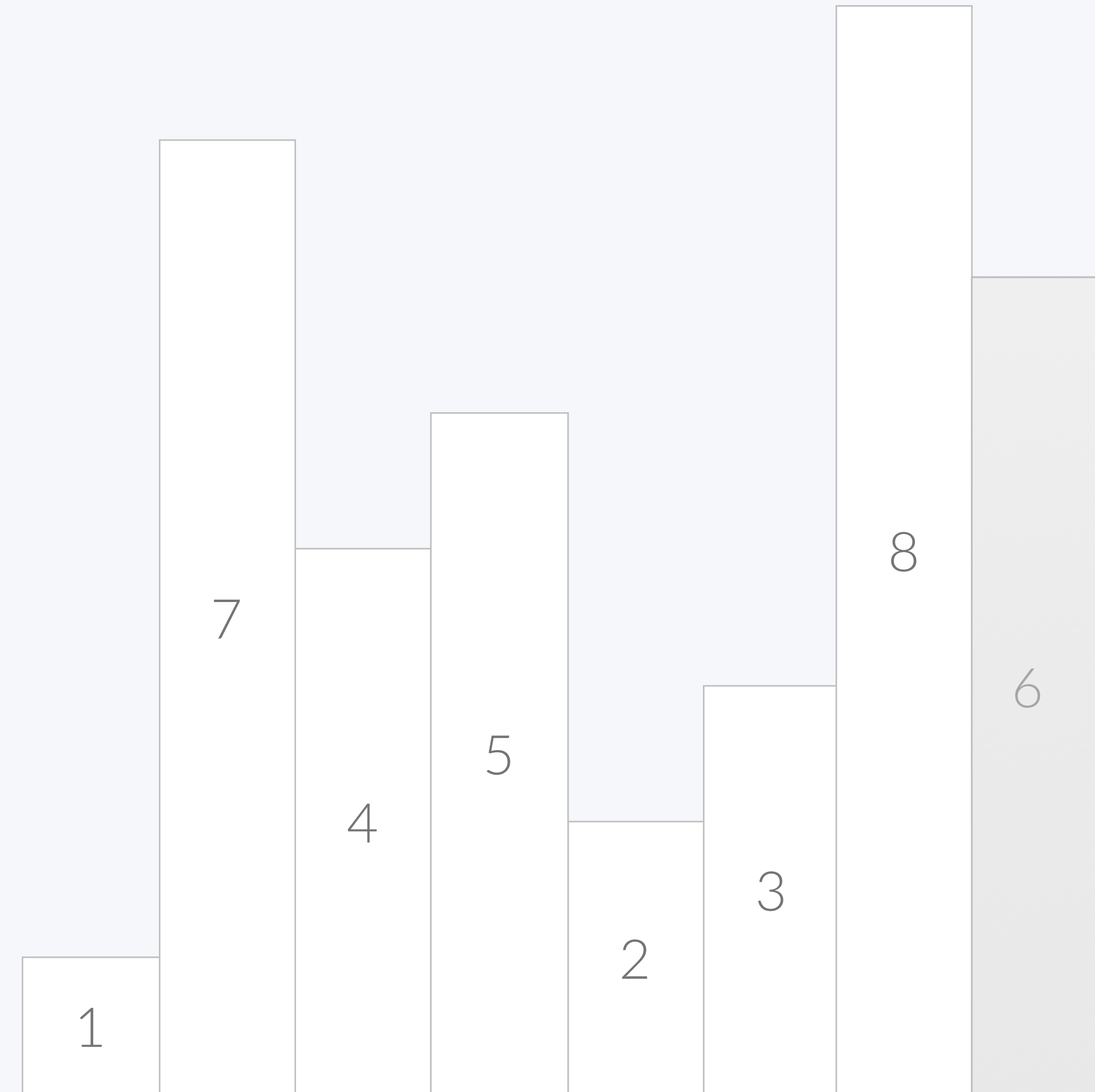


# 오아시스 재결합

79

<https://www.acmicpc.net/problem/3015>

- 스택: 8
- 6은 8보다 키가 작다
- 스택이 비어있지 않기 때문에
- 정답에 1을 더하고 (8과 6)
- 스택에 6을 추가한다
- 스택: 8 6



# 오아시스 재결합

<https://www.acmicpc.net/problem/3015>

```
for (int i=0; i<n; i++) {  
    while (!s.empty()) {  
        if (s.top() < a[i]) {  
            ans += 1;  
            s.pop();  
        } else {  
            break;  
        }  
    }  
    if (!s.empty()) ans += 1;  
    s.push(a[i]);  
}
```



# 오아시스 재결합

81

<https://www.acmicpc.net/problem/3015>

- 이제 원래 문제를 풀어야 한다 2
- 키가 같은 사람이 들어오기 때문에
- 스택에 키와, 그 키를 가진 사람의 수를 넣어서 문제를 푼다

1 1 3 3 2 2 4 4



1 1 3 3  
 $O(N^2)$

# 오아시스 재결합

82

<https://www.acmicpc.net/problem/3015>

```
for (int i=0; i<n; i++) {
    pair<int,int> p = {a[i], 1};
    while (!s.empty()) {
        if (s.top().first < a[i]) {
            ans += (long long)s.top().second;
            if (s.top().first == a[i]) {
                p.second += s.top().second;
            }
            s.pop();
        } else {
            break;
        }
    }
    if (!s.empty()) ans += 1LL;
    s.push(p);
}
```

# 오아시스 재결합

<https://www.acmicpc.net/problem/3015>

- C/C++: <https://gist.github.com/Baekjoon/f080cf3009603815aa62>
- Java: <https://gist.github.com/Baekjoon/6557d554cf1400002a86>

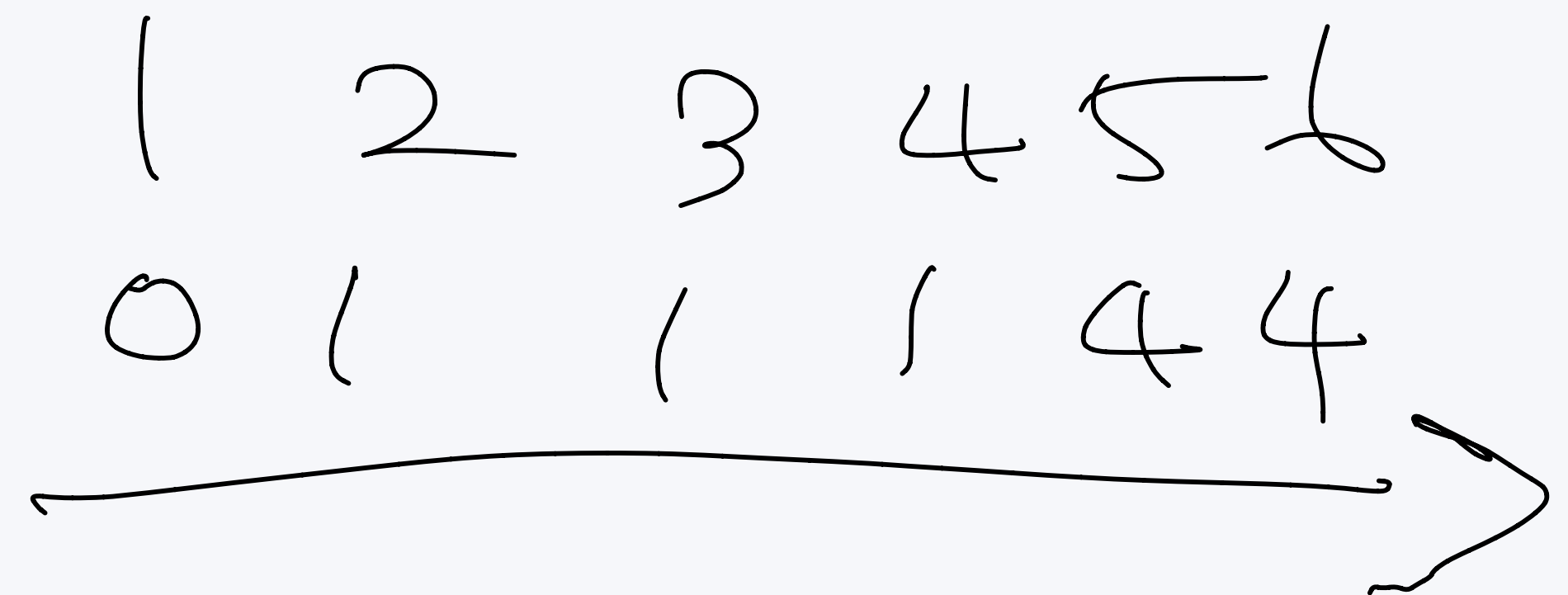
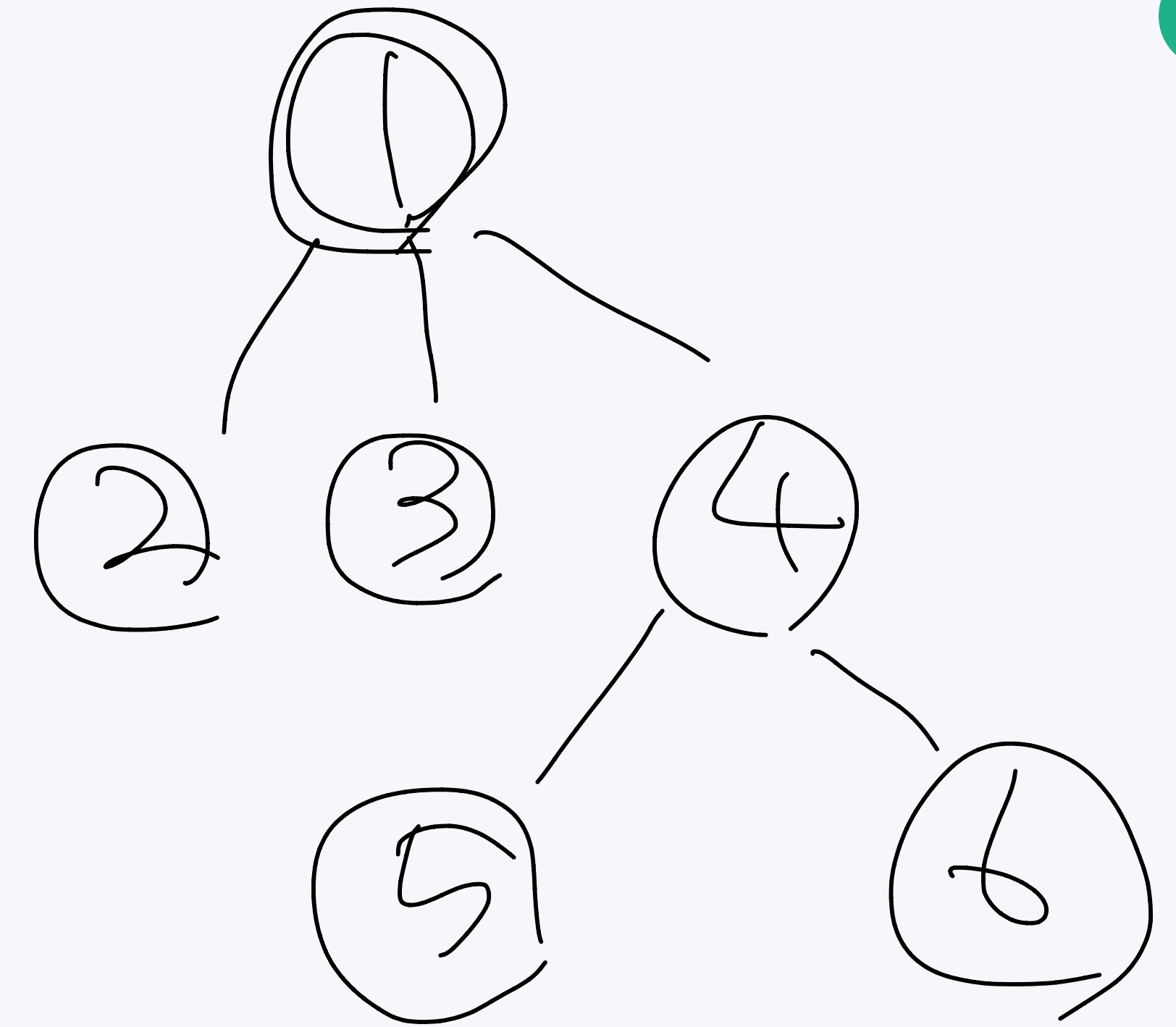
# 유니온 파인드

Disjoint-set  
알고리즘을 적게 사용하는

# 유니온 파인드

Union Find

- 상호 배타적 집합(Disjoint-set)이라고도 한다.
- 2가지 연산으로 이루어져 있다.
  1. Find: x가 어떤 집합에 포함되어 있는지 찾는 연산
  2. Union: x와 y가 포함되어 있는 집합을 합치는 연산
- 구현은 간단한 트리를 이용해서 한다.
- $\text{parent}[i] = i$ 의 parent가 저장되어 있음




# 유니온 파인드

Union Find

86

$\text{Parent}[i] = i$



- 가장 처음에는  $\text{parent}[i] = i$ 로 초기화 한다.

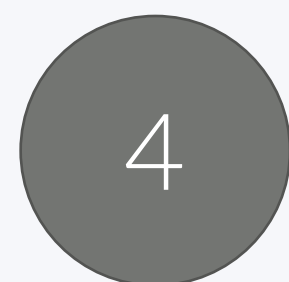
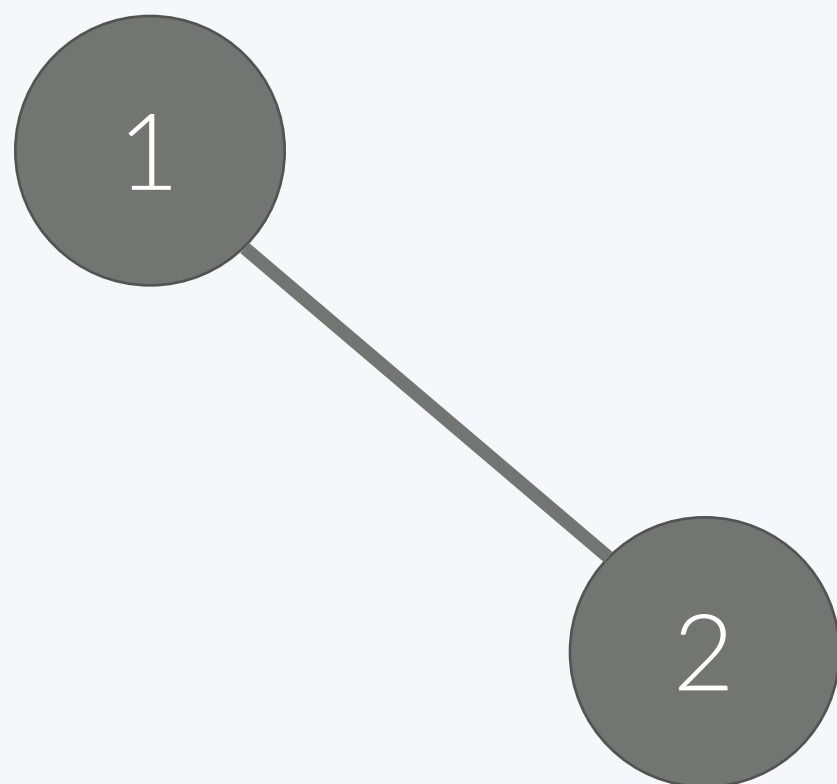


i	1	2	3	4	5	6	7	8
P[i]	1	2	3	4	5	6	7	8

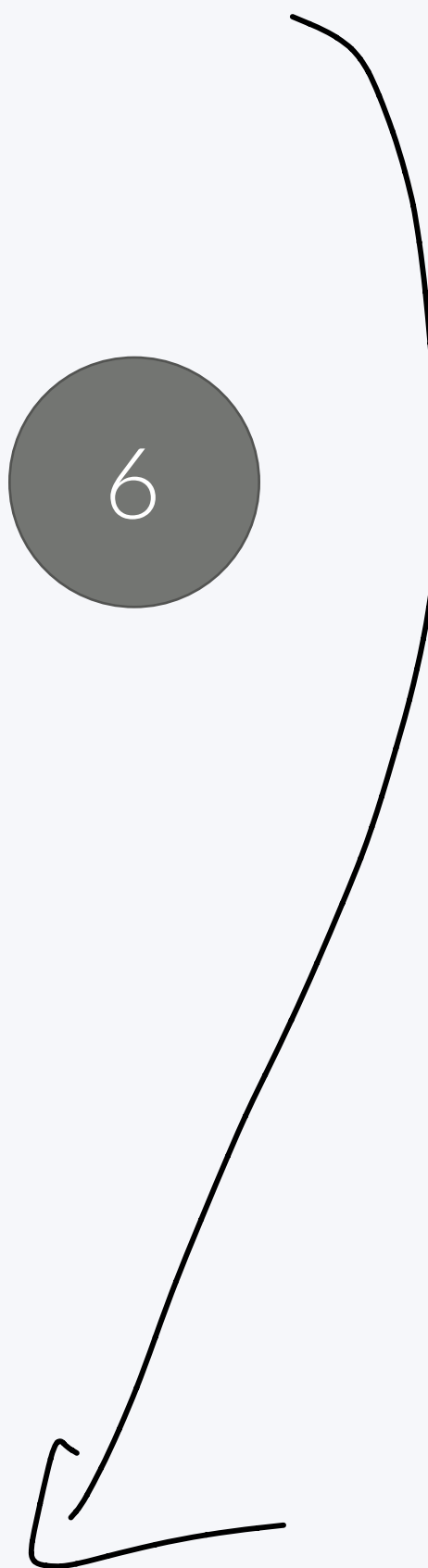
# 유니온 파인드

Union Find

- Union (1, 2)



Union (4, 5)



i	1	2	3	4	5	6	7	8
P[i]	1	1	3	4	5	6	7	8

# 유니온 파인드

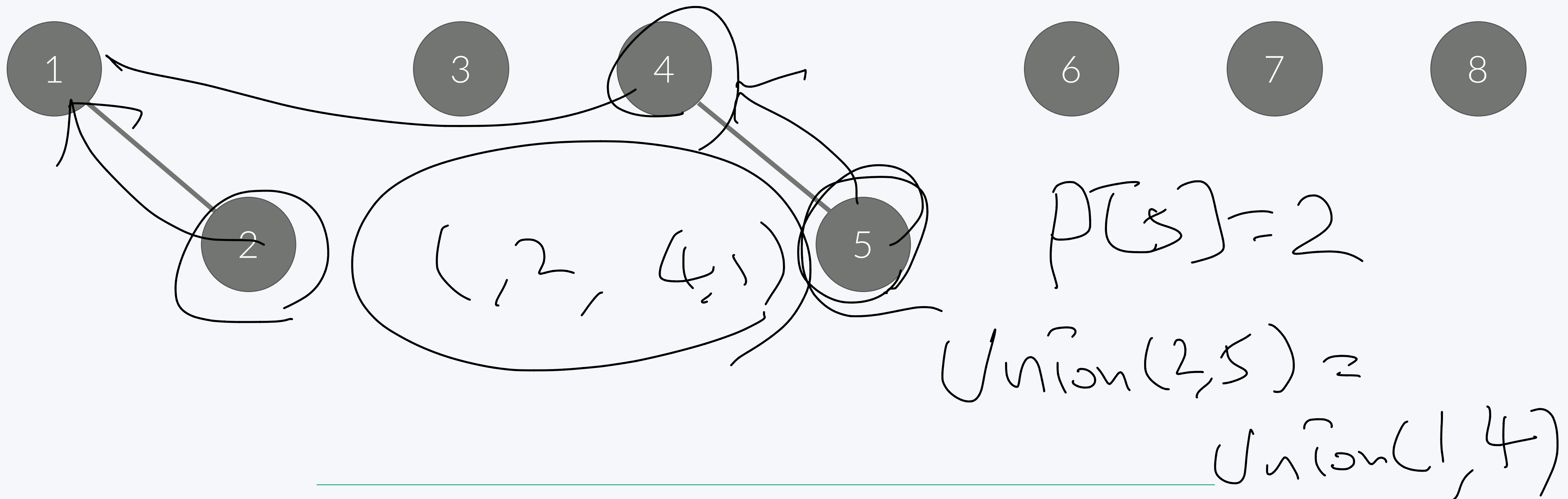
Union Find

- Union (4, 5)

Union (X, Y)

Parent[Y] = X

88



i	1	2	3	4	5	6	7	8
P[i]	1	1	3	4	4	6	7	8

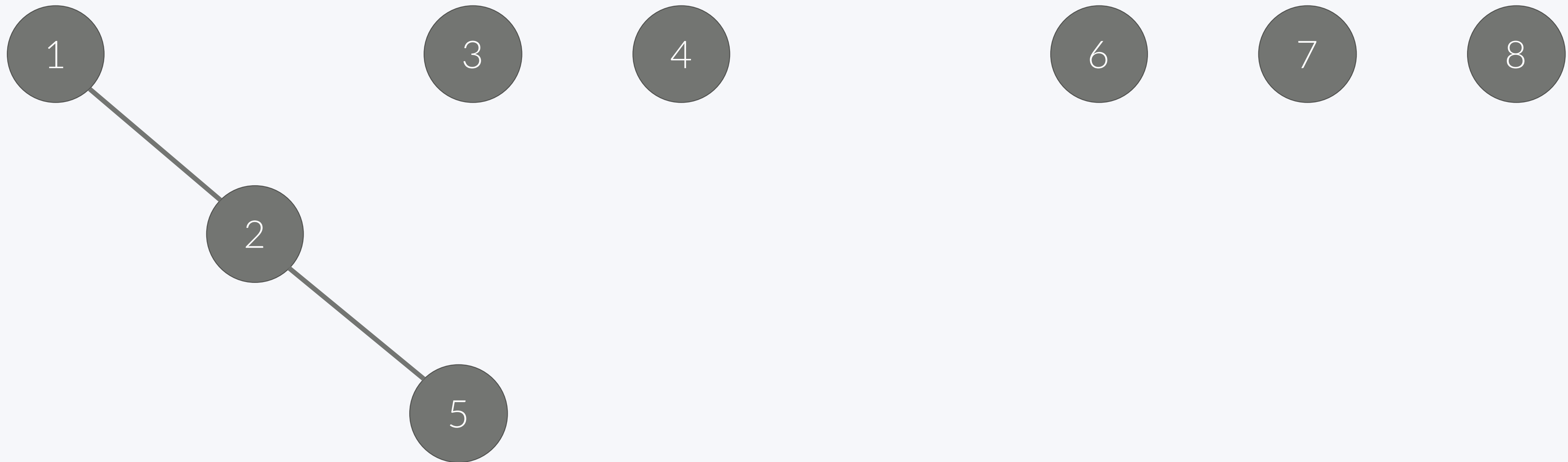


# 유니온 파인드

Union Find

89

- Union (2, 5)



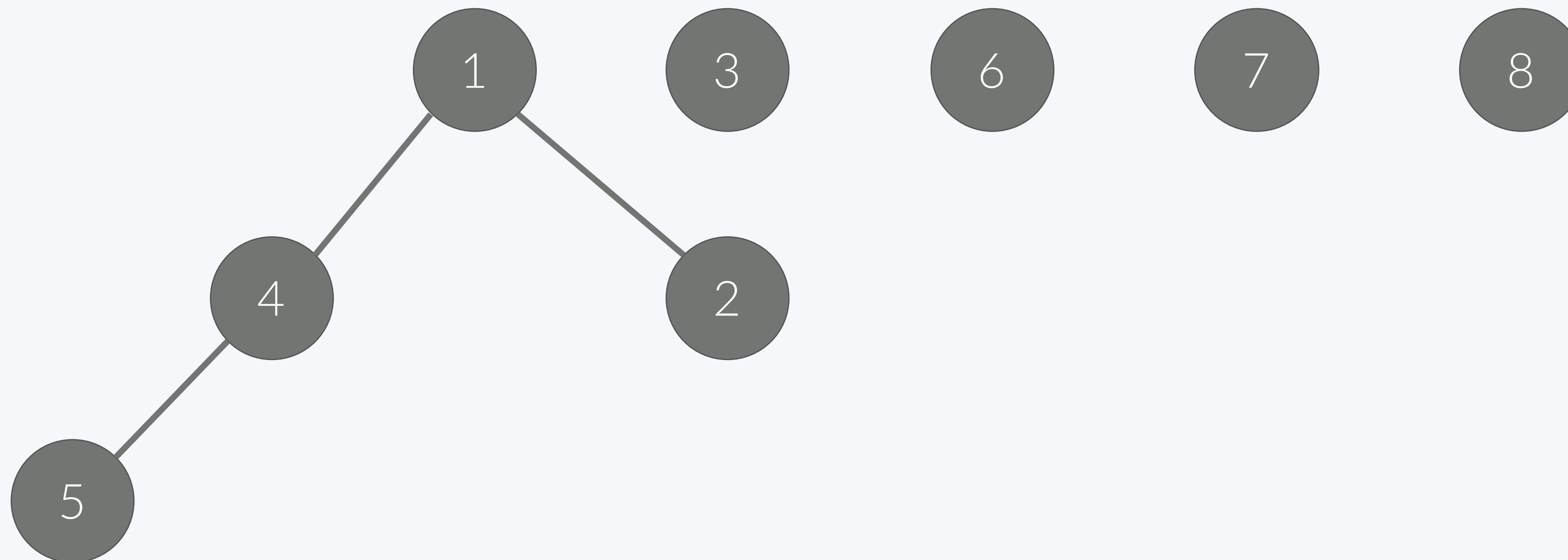
i	1	2	3	4	5	6	7	8
P[i]	1	1	3	4	2	6	7	8

# 유니온 파인드

90

Union Find

- Union(2, 5)
- Find(2) = 1
- Find(5) = 4
- Union(1, 4)

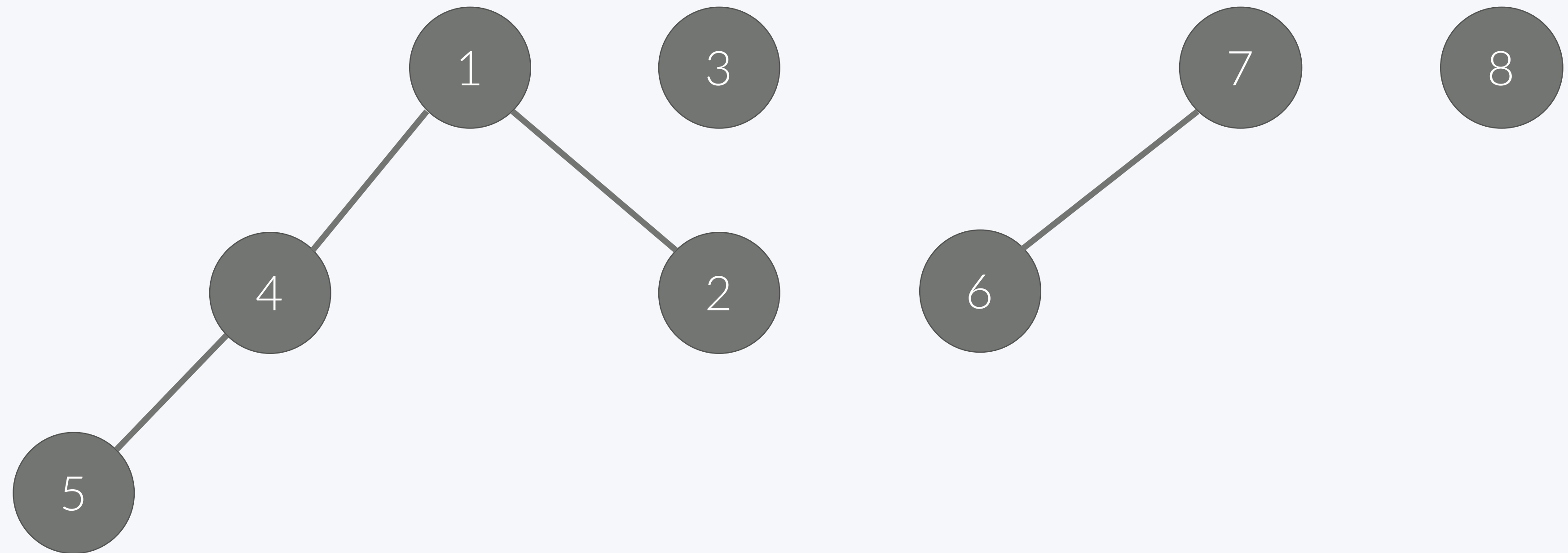


i	1	2	3	4	5	6	7	8
P[i]	1	1	3	1	4	6	7	8

# 유니온 파인드

Union Find

- Union(7, 6)



i	1	2	3	4	5	6	7	8
P[i]	1	1	3	1	4	7	7	8

# 유니온 파인드

## Union Find

- Find의 재귀 호출 구현

```
int Find(int x) {  
    if (x == parent[x]) {  
        return x;  
    } else {  
        return Find(parent[x]);  
    }  
}
```

Handwritten notes:  $\frac{2}{1}$  and a horizontal line.

# 유니온 파인드

## Union Find

- Union의 구현
- $\text{Union}(x, y) \Rightarrow y$ 의 parent를  $x$ 로 설정한다.

```
int Union(int x, int y) {  
    x = Find(x);  
    y = Find(y);  
    parent[y] = x;  
}
```

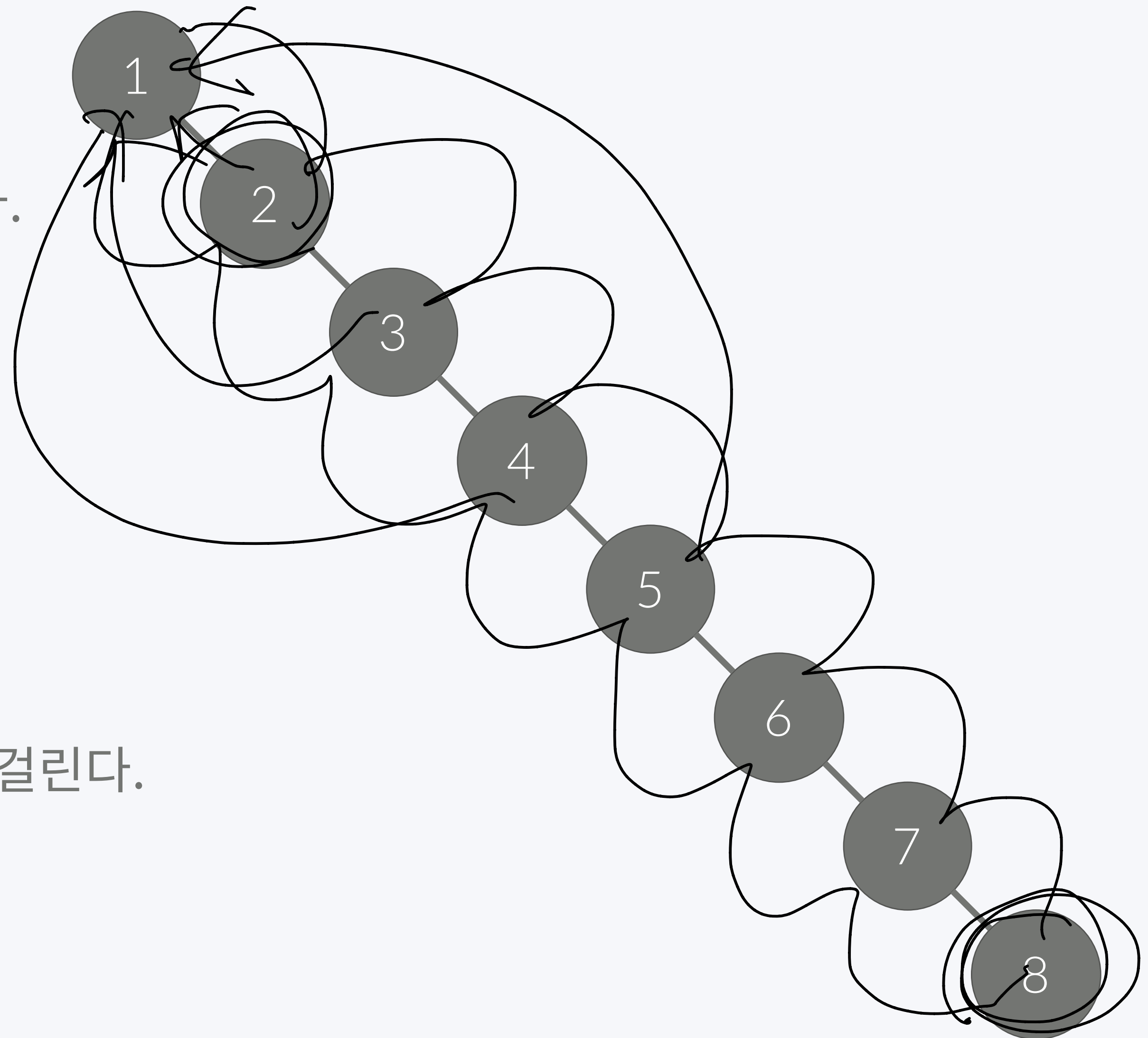
# 유니온 파인드

## Union Find

- Union의 구현
- $\text{Union}(x, y) \Rightarrow y$ 의 parent를  $x$ 로 설정한다.

```
int union(int x, int y) {  
    x = find(x);  
    y = find(y);  
    p[y] = x;  
}
```

- 오른쪽 그림과 같은 문제가 생길 수 있다.
- 이렇게 되면 find의 시간 복잡도가  $O(N)$ 이 걸린다.

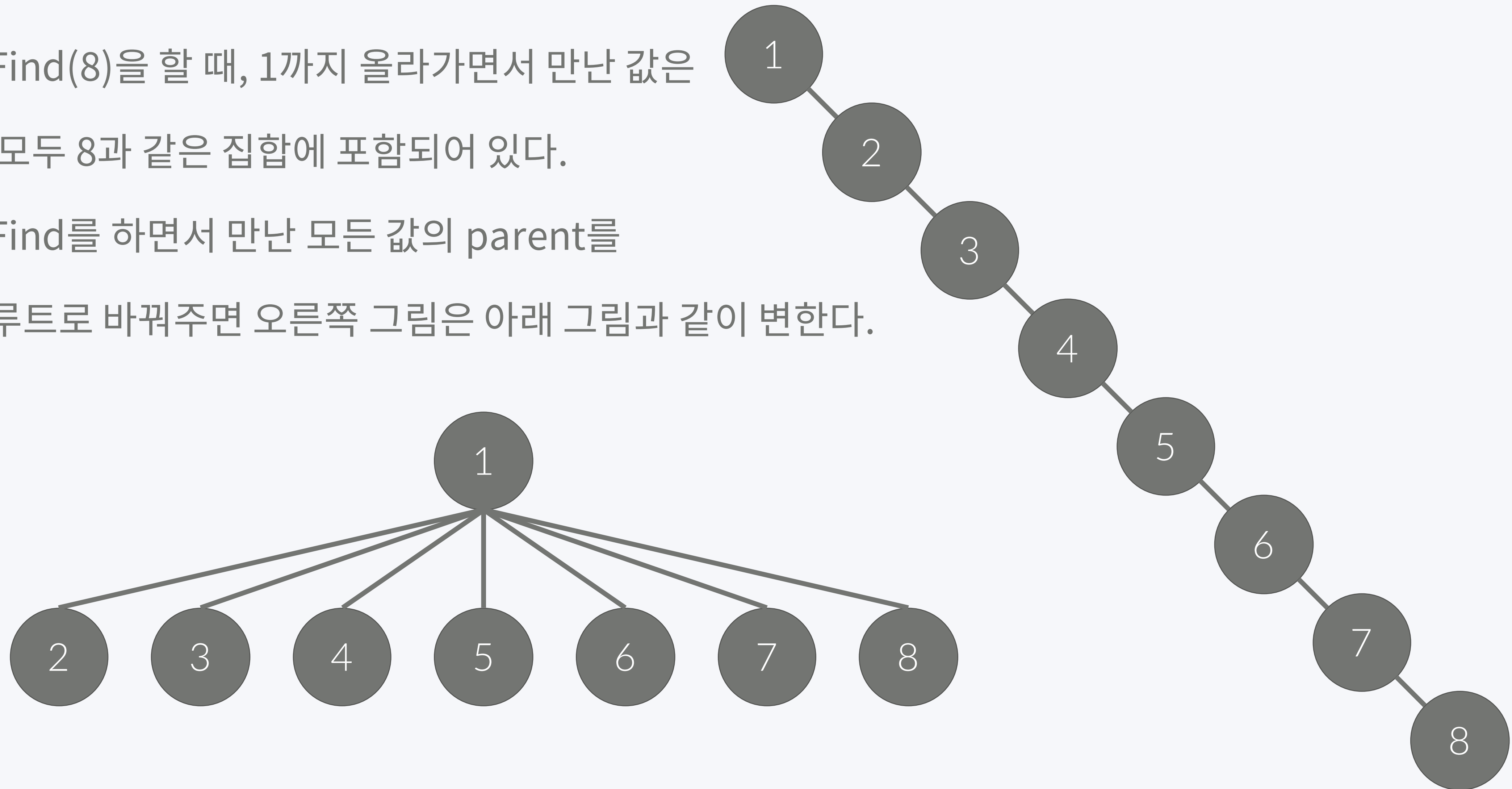


# 유니온 파인드

Union Find

95

- Find(8)을 할 때, 1까지 올라가면서 만난 값은
- 모두 8과 같은 집합에 포함되어 있다.
- Find를 하면서 만난 모든 값의 parent를
- 루트로 바꿔주면 오른쪽 그림은 아래 그림과 같이 변한다.



# 유니온 파인드

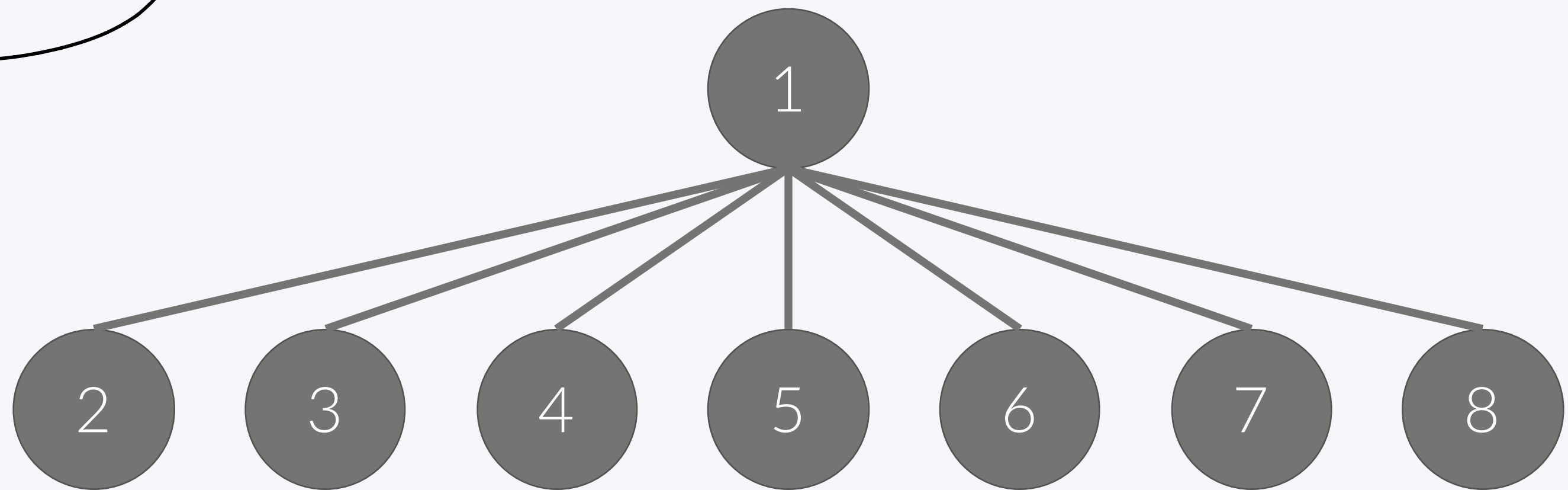
## Union Find

- 이런 방식을 경로 압축이라고 한다.

```
int Find(int x) {  
    if (x == parent[x]) {  
        return x;  
    } else {  
        int y = Find(parent[x]);  
        parent[x] = y;  
        return y;  
    }  
}
```

Handwritten annotations on the code:

- A bracket groups the recursive call `Find(parent[x])` and the assignment `parent[x] = y`.
- An arrow points from the `return y` statement to the `return x` statement in the `if` block.
- The variable `y` is circled in the `return y` statement.
- The variable `x` is circled in the `parent[x] = y` statement.



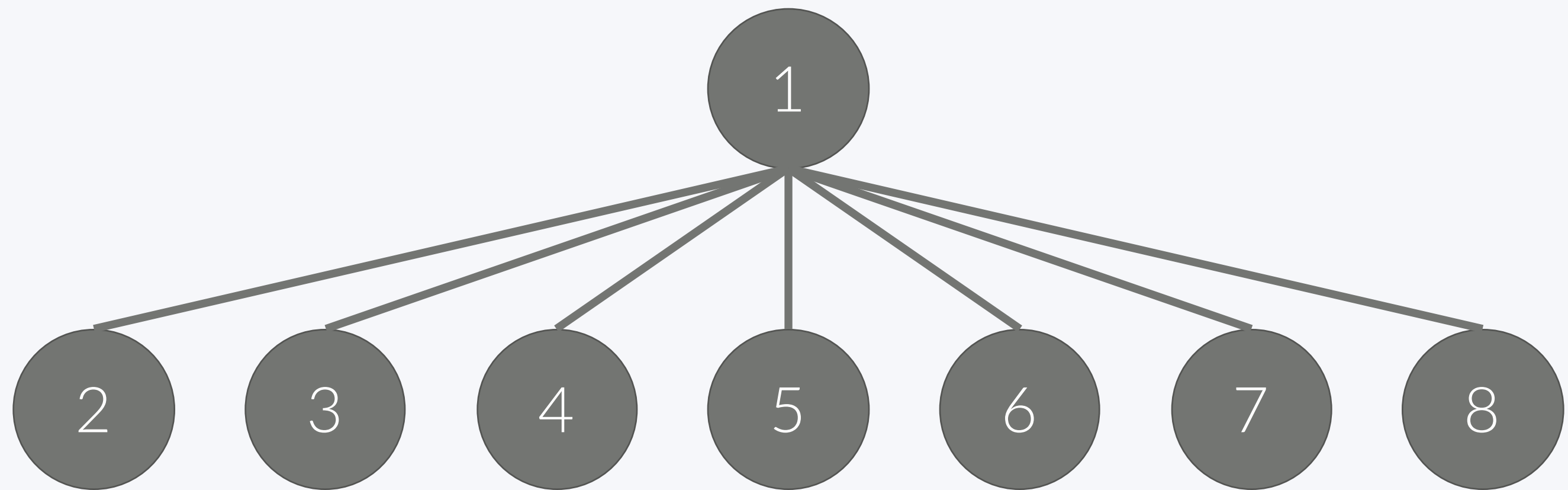


# 유니온 파인드

## Union Find

- 아래와 같이 간단하게도 구현할 수 있다.

```
int find(int x) {  
    if (x == p[x]) {  
        return x;  
    } else {  
        return p[x] = find(p[x]);  
    }  
}
```



# 집합의 표현

<https://www.acmicpc.net/problem/1717>

- 초기에  $\{0\}, \{1\}, \{2\}, \dots \{n\}$  이 각각  $n+1$ 개의 집합을 이루고 있다
- 여기에 합집합 연산과, 두 원소가 같은 집합에 포함되어 있는지를 확인하는 연산을 수행
- Disjoint-set 자료구조를 구현하는 문제

# 집합의 표현

<https://www.acmicpc.net/problem/1717>

- C/C++: <https://gist.github.com/Baekjoon/d7b93169d5320566a979b74cc74e7976>

# 바이러스

<https://www.acmicpc.net/problem/2606>

- 유니온 파인드로 푸는 문제

# 바이러스

101

<https://www.acmicpc.net/problem/2606>

- C/C++: <https://gist.github.com/Baekjoon/47e0a140ecad408e5ec2412399e60dbe>

# 비트마스크

---

# 비트마스크

Bitmask

103

- 비트(bit) 연산을 사용해서 부분 집합을 표현할 수 있다.

# 비트 연산

bitwise operation

104

- $\&$  (and),  $|$  (or),  $\sim$  (not),  $\wedge$  (xor)

A	B	$\sim A$	$A \& B$	$A   B$	$A \wedge B$
0	0	1	0	0	0
0	1	1	0	1	1
1	0	0	0	1	1
1	1	0	1	1	0



# 비트 연산

bitwise operation

- 두 수 A와 B를 비트 연산 하는 경우에는 가장 뒤의 자리부터 하나씩 연산을 수행하면 된다.
- A = 27, B = 83인 경우
- $A = 11011_2, B = 1010011_2$
- $A \& B = 19, A \mid B = 91, A \wedge B = 73$

0	0	1	1	0	1	1		0	0	1	1	0	1	1		0	0	1	1	0	1	1	
&	1	0	1	0	0	1	1		1	0	1	0	0	1	1	^	1	0	1	0	0	1	1
-----								-----								-----							
0	0	1	0	0	1	1		1	0	1	1	0	1	1		1	0	0	1	0	0	0	

# 비트 연산

106

bitwise operation

- not 연산의 경우에는 자료형에 따라 결과가 달라진다.
- $A = 83 = 1010011_2$
- $\sim A = 10101100_2$  (8비트 자료형인 경우)
- $\sim A = 11111111\ 11111111\ 11111111\ 10101100_2$  (32비트 자료형인 경우)
- 또, unsigned, signed에 따라서 보여지는 값은 다르다.

# 비트 연산

bitwise operation

- shift left (<<) 와 shift right (>>) 연산이 있다.
- $A \ll B$  (A를 왼쪽으로 B비트만큼 민다.)
- $1 \ll 0 = 1$
- $1 \ll 1 = 2 \ (10_2)$
- $1 \ll 2 = 4 \ (100_2)$
- $1 \ll 3 = 8 \ (1000_2)$
- $1 \ll 4 = 16 \ (10000_2)$
- $3 \ll 3 = 24 \ (11000_2)$
- $5 \ll 10 = 5120 \ (101000000000000_2)$

# 비트 연산

bitwise operation

- shift left ( $\ll$ ) 와 shift right ( $\gg$ ) 연산이 있다.
- $A \gg B$  (A를 오른쪽으로 B비트만큼 민다.)
- $1 \gg 0 = 1$
- $1 \gg 1 = 0$  ( $0_2$ )
- $10 \gg 1 = 5$  ( $101_2$ )
- $10 \gg 2 = 2$  ( $10_2$ )
- $10 \gg 3 = 1$  ( $1_2$ )
- $30 \gg 1 = 15$  ( $1111_2$ )
- $1024 \gg 10 = 1$  ( $1_2$ )

# 비트 연산

bitwise operation

- $A \ll B$ 는  $A \times 2^B$ 와 같다.
- $A \gg B$ 는  $A / 2^B$ 와 같다.
- $(A + B) / 2$ 는  $(A+B) \gg 1$ 로 쓸 수 있다.
- 어떤 수가 홀수 인지 판별하는  $\text{if } (N \% 2 == 1)$  은  $\text{if } (N \& 1)$ 로 줄여 쓸 수 있다.

# 비트마스크

Bitmask

110

- 정수로 집합을 나타낼 수 있다.
- $\{1, 3, 4, 5, 9\} = 570 = 2^1 + 2^3 + 2^4 + 2^5 + 2^9$

# 비트마스크

111

Bitmask

- $\{1, 3, 4, 5, 9\} = 570$
- 0이 포함되어 있는지 검사
  - $570 \ \& \ 2^0 = 570 \ \& \ (1 \ll 0) = 0$
- 1이 포함되어 있는지 검사
  - $570 \ \& \ 2^1 = 570 \ \& \ (1 \ll 1) = 2$
- 2이 포함되어 있는지 검사
  - $570 \ \& \ 2^2 = 570 \ \& \ (1 \ll 2) = 0$
- 3이 포함되어 있는지 검사
  - $570 \ \& \ 2^3 = 570 \ \& \ (1 \ll 3) = 8$

# 비트마스크

112

Bitmask

- $\{1, 3, 4, 5, 9\} = 570$
- 1 추가하기
  - $570 \mid 2^1 = 570 + (1 \ll 1) = 570 \ (1000111010_2)$
- 2 추가하기
  - $570 \mid 2^2 = 570 \mid (1 \ll 2) = 574 \ (1000111110_2)$
- 3 추가하기
  - $574 \mid 2^3 = 570 + (1 \ll 3) = 570 \ (1000111010_2)$
- 4 추가하기
  - $574 \mid 2^4 = 570 \mid (1 \ll 4) = 570 \ (1000111010_2)$



# 비트마스크

113

Bitmask

- $\{1, 3, 4, 5, 9\} = 570$
- 1 제거하기
  - $570 \ \& \sim 2^1 = 570 \ \& \sim (1 \ll 1) = 568 \ (1000111000_2)$
- 2 제거하기
  - $570 \ \& \sim 2^2 = 570 \ \& \sim (1 \ll 2) = 570 \ (1000111010_2)$
- 3 제거하기
  - $562 \ \& \sim 2^3 = 562 \ \& \sim (1 \ll 3) = 562 \ (1000110010_2)$
- 4 제거하기
  - $562 \ \& \sim 2^4 = 562 \ \& \sim (1 \ll 4) = 546 \ (1000101010_2)$

# 비트마스크

Bitmask

114

- 전체 집합
  - $(1 \ll N) - 1$
- 공집합
  - 0

# 비트마스크

115

## Bitmask

- 현재 집합이 S일때
- i를 추가
  - $S \mid (1 \ll i)$
- i를 검사
  - $S \& (1 \ll i)$
- i를 제거
  - $S \& \sim(1 \ll i)$
- i를 토글 (0을 1로, 1을 0으로)
  - $S \wedge (1 \ll i)$

# 집합

116

<https://www.acmicpc.net/problem/11723>

- 비트마스크를 연습해보는 문제

# 집합

117

<https://www.acmicpc.net/problem/11723>

- C++: <https://gist.github.com/Baekjoon/3503aaa55c03cdde9df51b1bd5155486>

# 비트마스크

Bitmask

118

- 물론 배열을 사용하는 것이 더욱 편리하지만, 비트마스크를 사용하는 이유는
- 집합을 배열의 인덱스로 표현할 수 있기 때문이다.
- 상태 다이나믹을 할 때 자주 사용하게 된다.

# bitset

bitset

119

- 비트마스크는 STL의 `bitset`을 이용해서 더 쉽게 나타낼 수 있다.

한

---

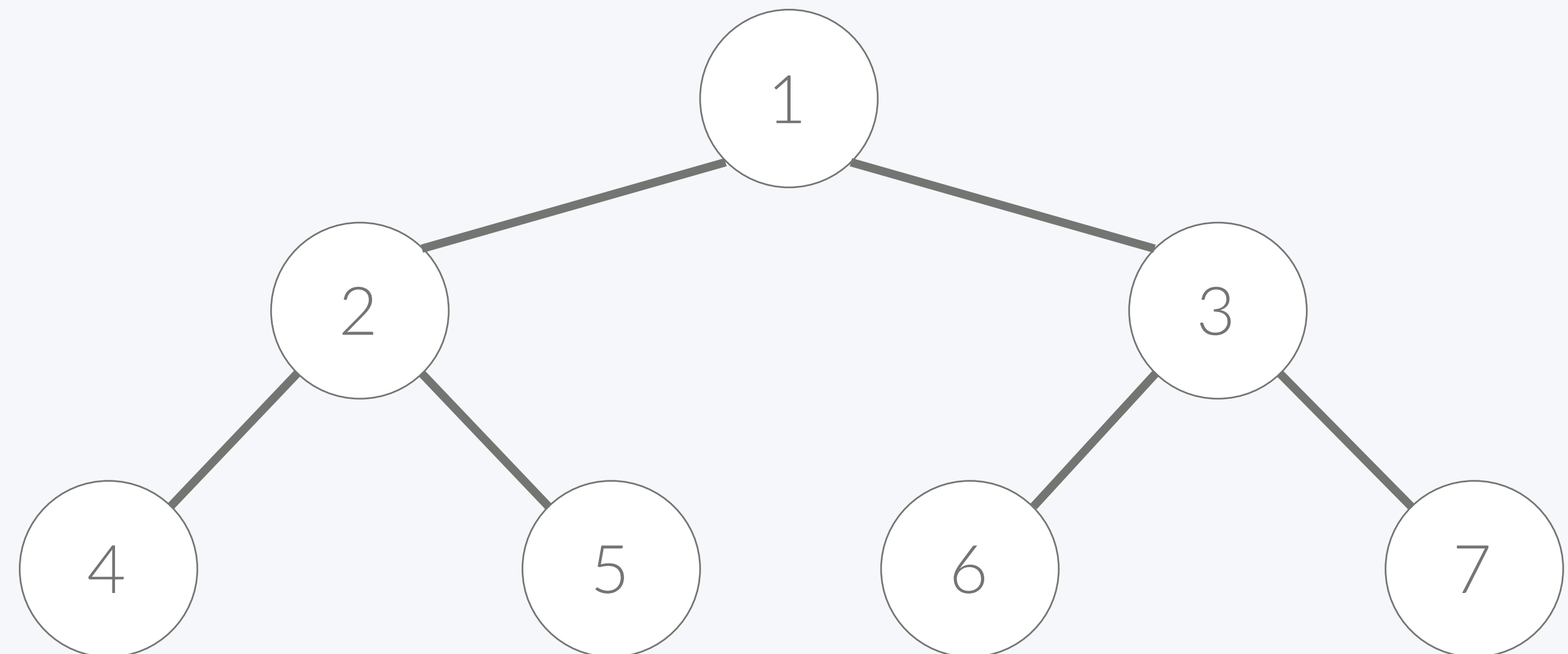


# Perfect Binary Tree

121

## Perfect Binary Tree

- 리프 노드를 제외한 노드의 자식의 수: 2
- 리프 노드의 자식의 수: 0
- 모든 리프 노드의 depth가 같아야 함
- 높이가  $h$ 인 트리의 노드 개수  $= 2^h - 1$

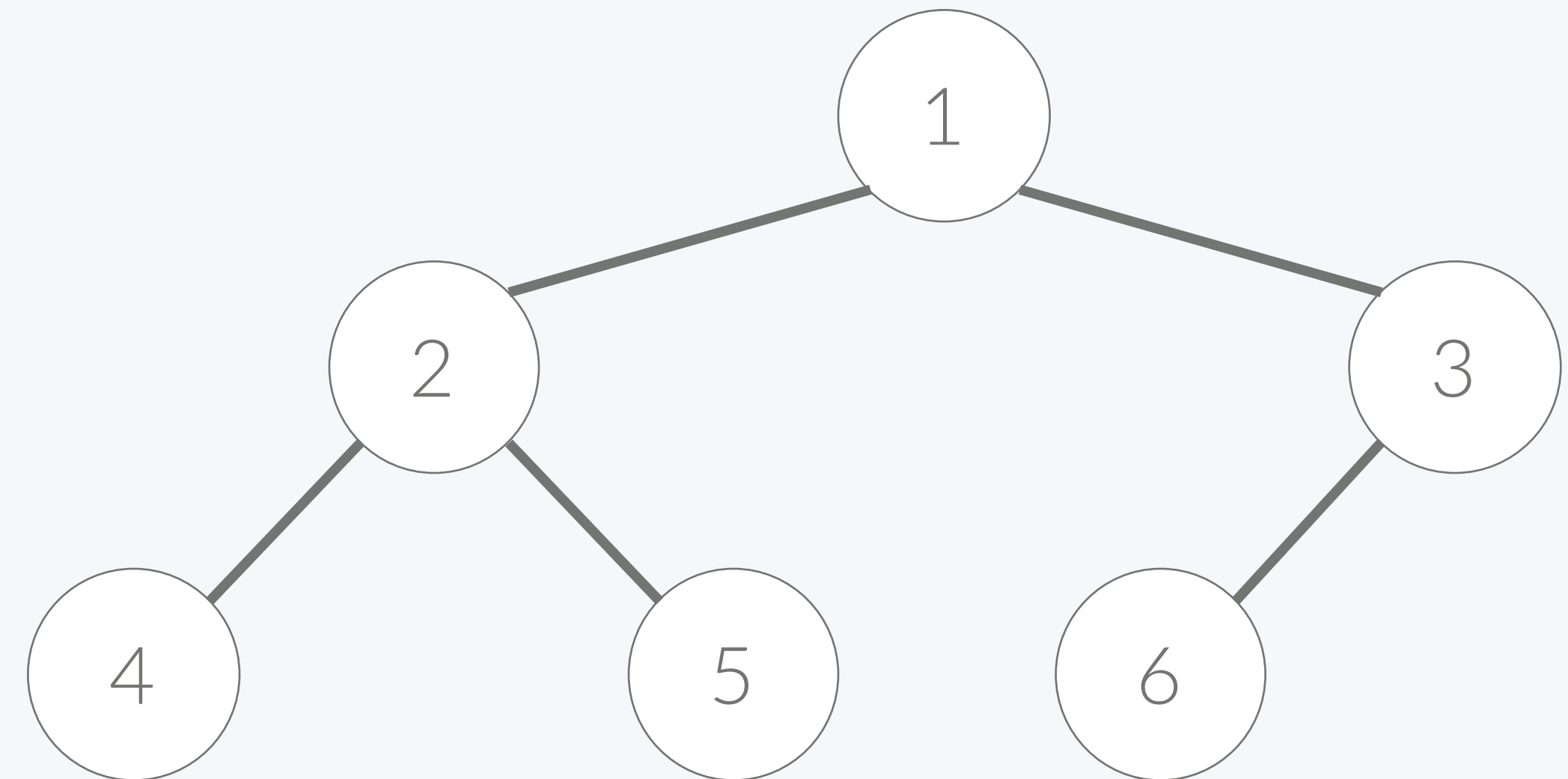
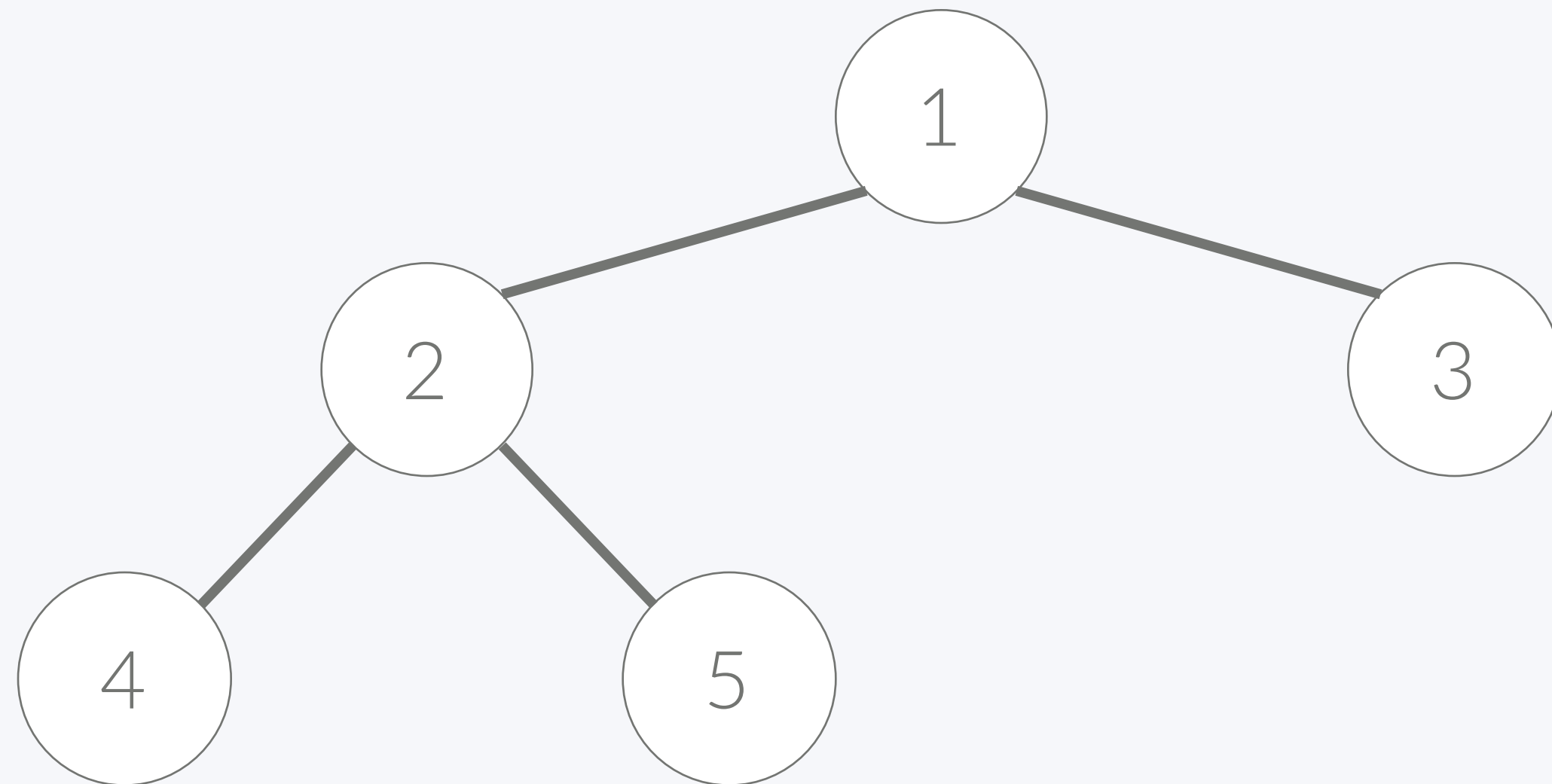


# Complete Binary Tree

122

## Complete Binary Tree

- 리프 노드를 제외한 노드의 자식의 수: 2
- 리프 노드의 자식의 수: 0
- 마지막 레벨에는 노드가 일부는 없을 수도 있음
- 오른쪽에서부터 몇 개가 사라진 형태

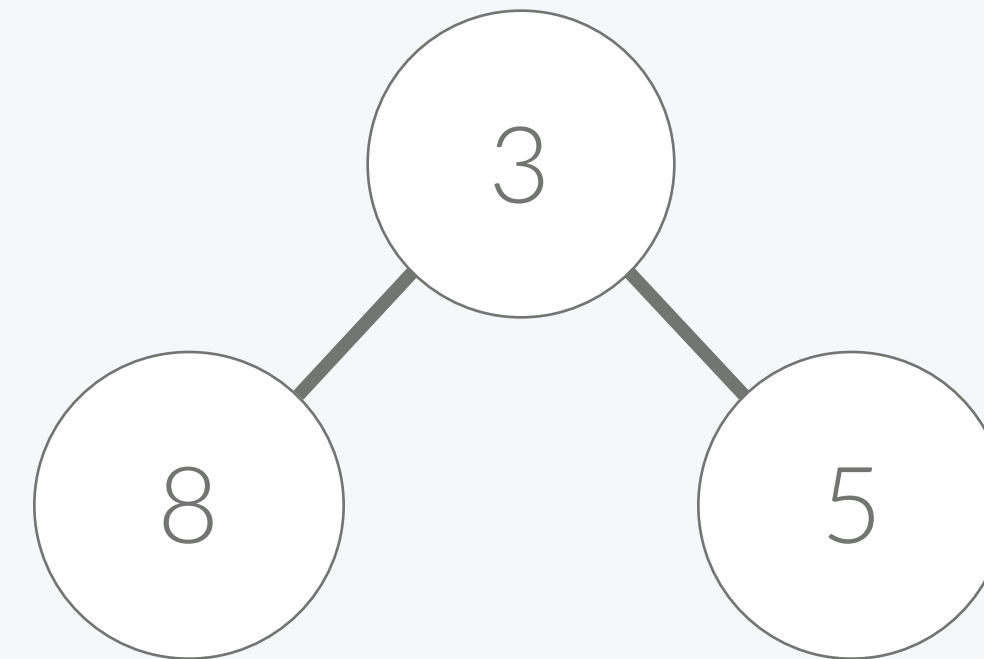
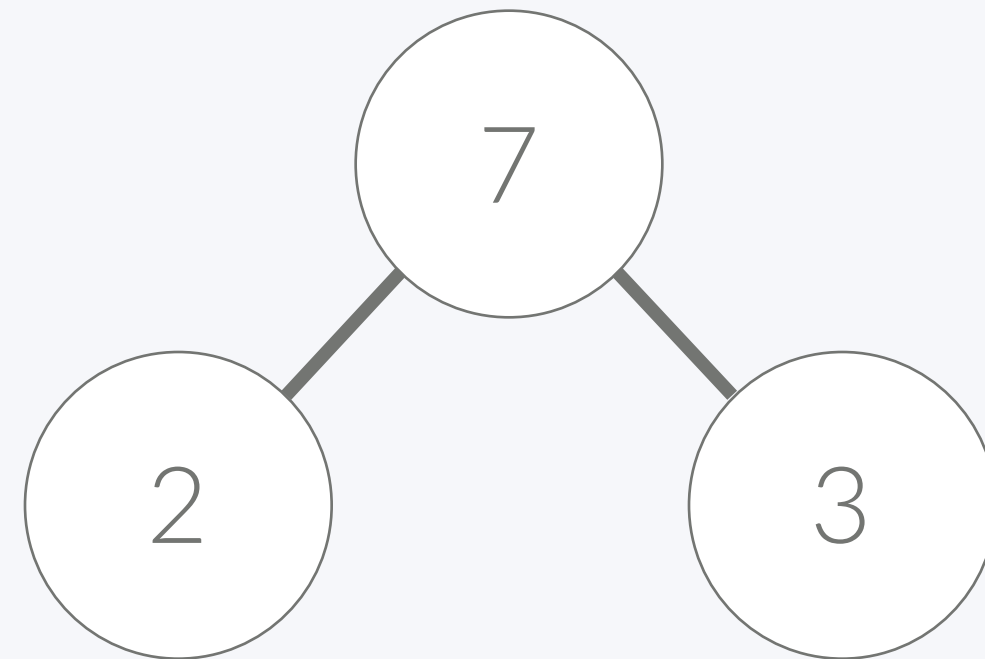


# 힙

## Heap

123

- Max-heap
  - 부모 노드는 자식 노드에 들어있는 값보다 크다
- Min-heap
  - 부모 노드는 자식 노드에 들어있는 값보다 작다.

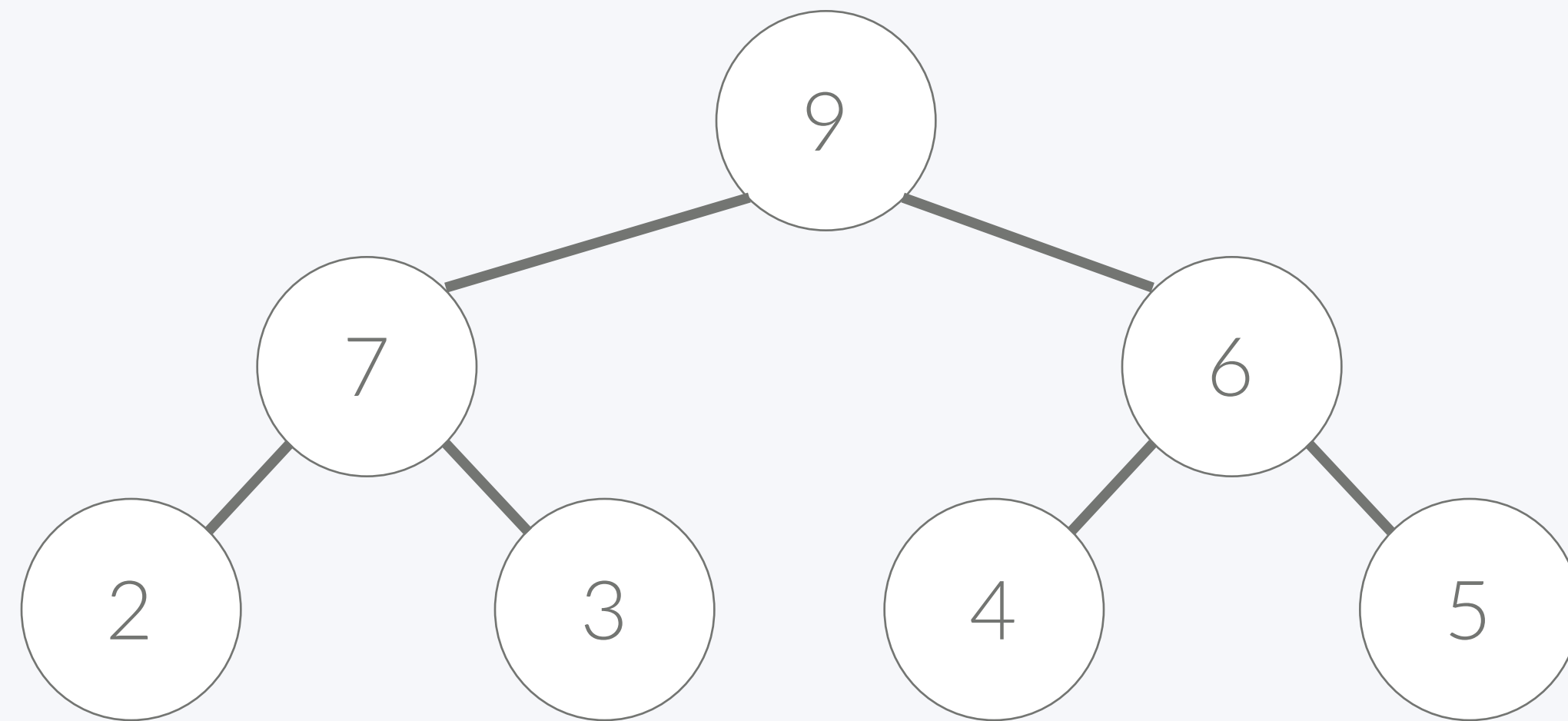


# 최대 힙

Max-Heap

124

- Max-heap 에서 가장 큰 값은 루트에 들어가 있다.
- N개가 Heap에 들어가있으면 높이는  $\lg N$ 이 된다.

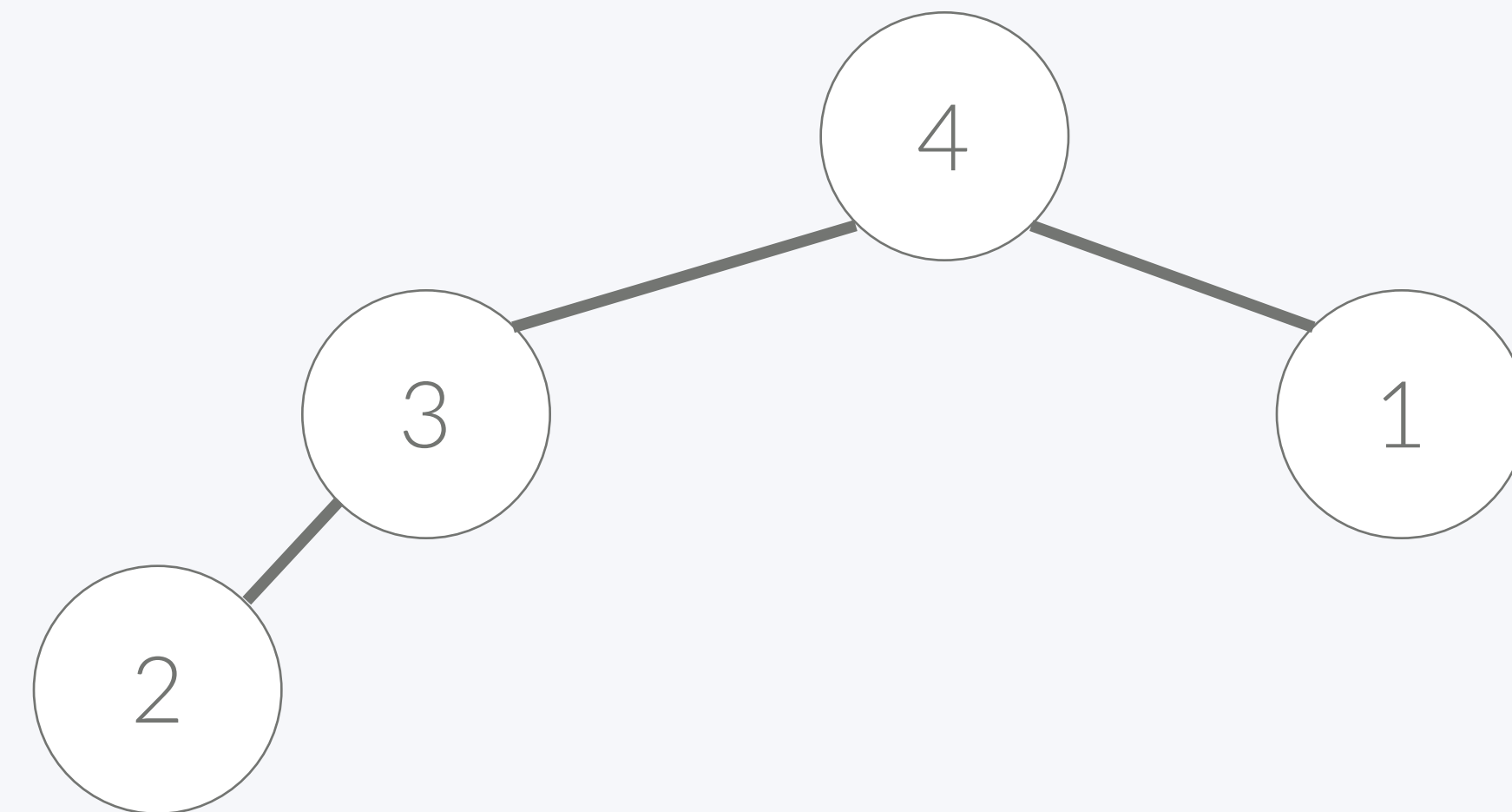


# 최대 힙 삽입

Max-Heap

125

- 가장 마지막 위치에 새로운 수를 넣는다.
- 그 수와 parent를 계속해서 비교해가면서
- 루트 < 자식 이면 두 수를 바꿔준다.
- 이런 Max-Heap 에 5를 넣어보자

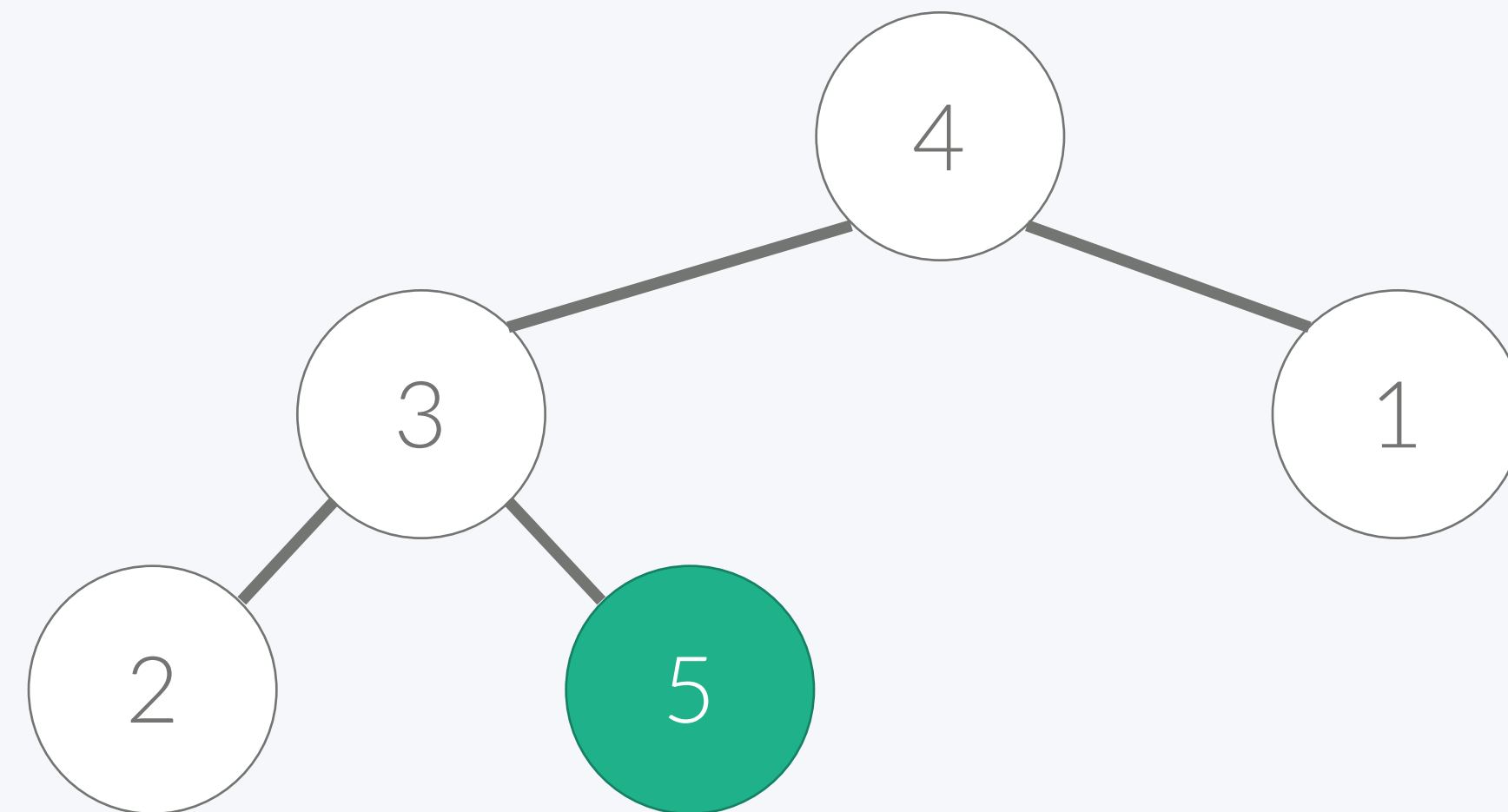


# 최대 힙 삽입

Max-Heap

- 5의 위치

126

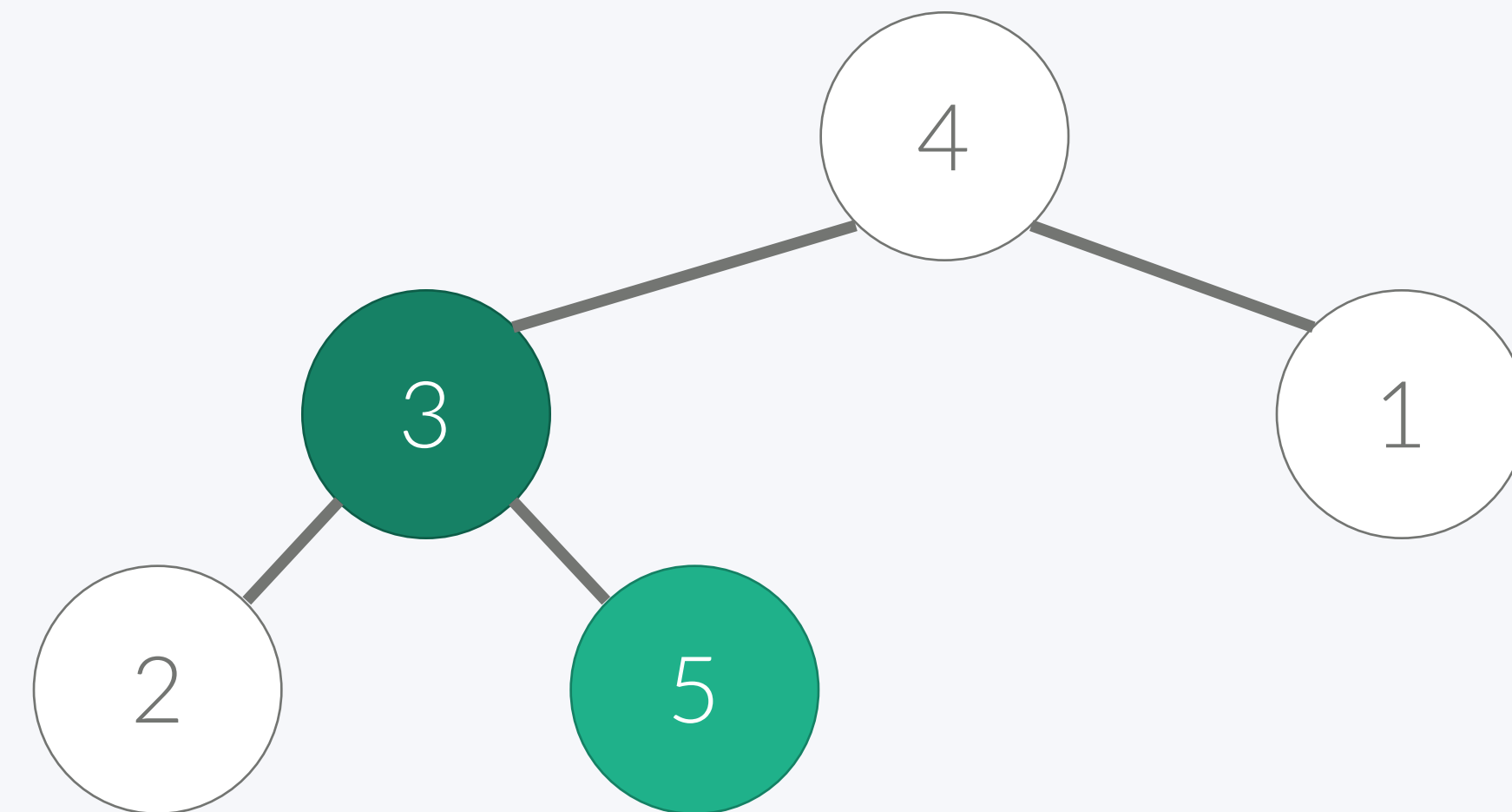


# 최대 힙 삽입

127

Max-Heap

- 5와 parent를 비교한다.
- 이 경우에  $3 < 5$  이기 때문에 Max-heap의 성질을 만족하지 않는다
- 따라서 두 수를 swap

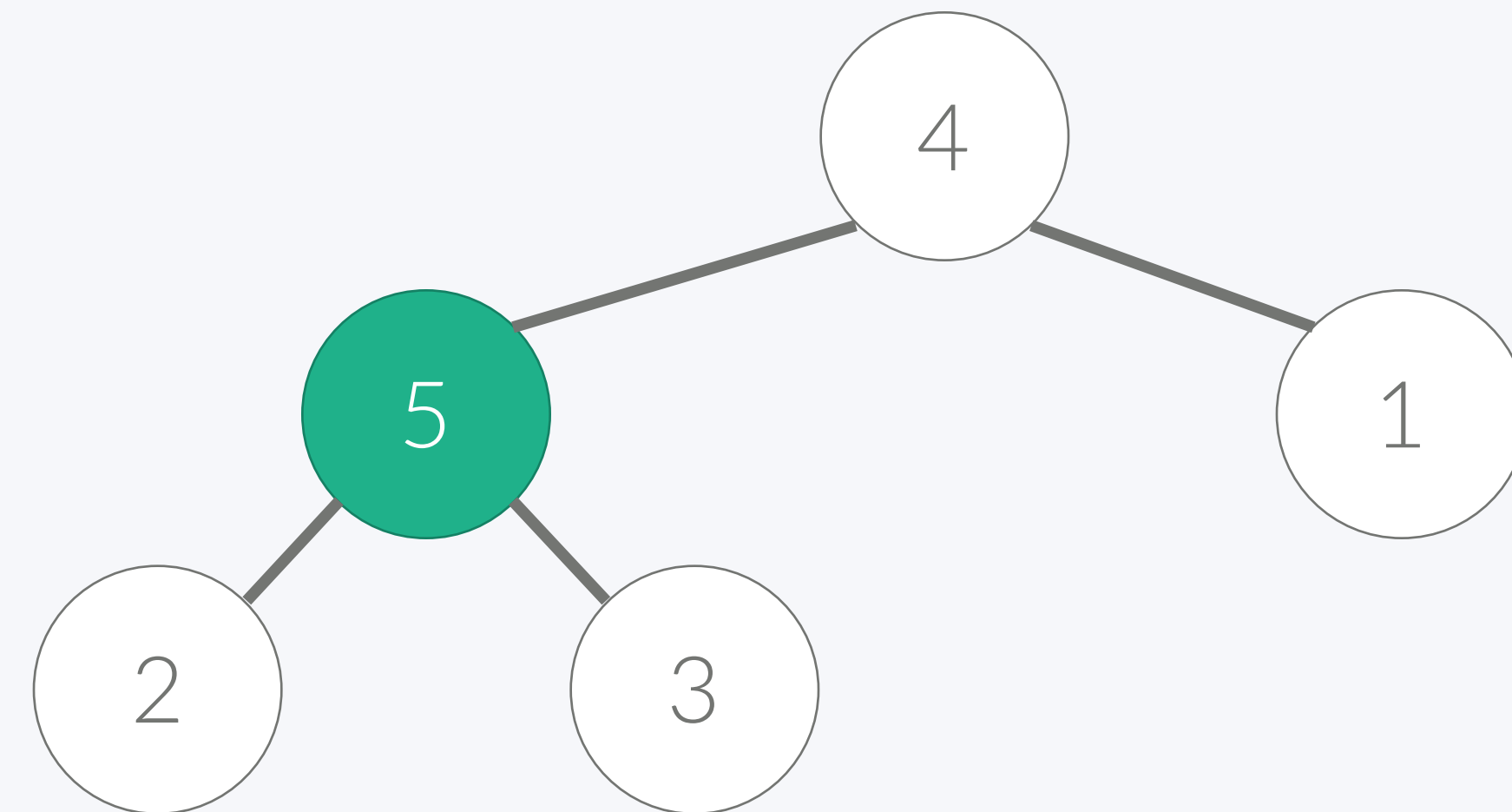


# 최대 힙 삽입

128

Max-Heap

- 5와 parent를 비교한다.
- 이 경우에  $3 < 5$  이기 때문에 Max-heap의 성질을 만족하지 않는다
- 따라서 두 수를 swap



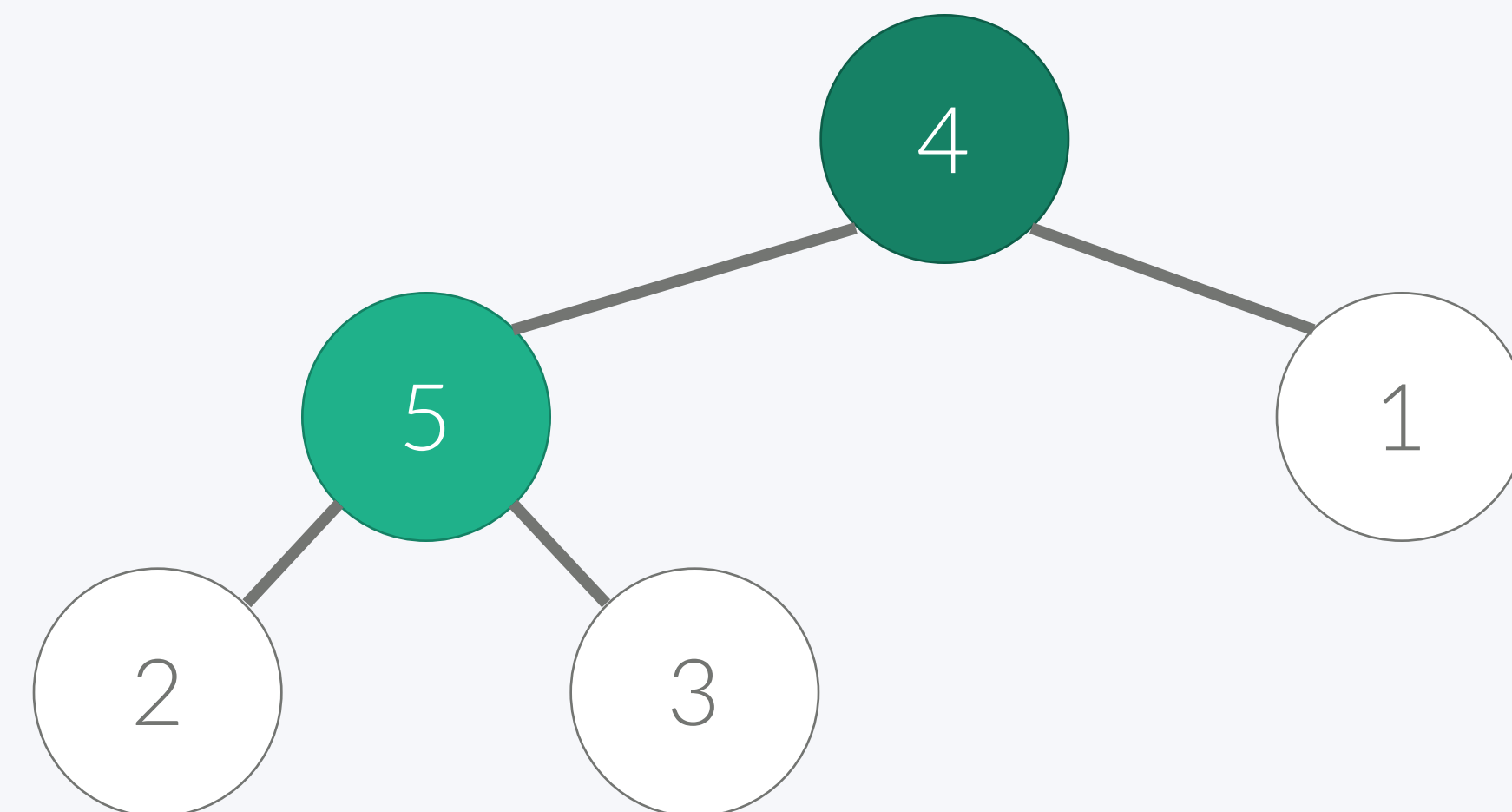


# 최대 힙 삽입

129

Max-Heap

- 5와 parent를 비교한다.
- 이 경우에  $4 < 5$  이기 때문에 Max-heap의 성질을 만족하지 않는다
- 따라서 두 수를 swap

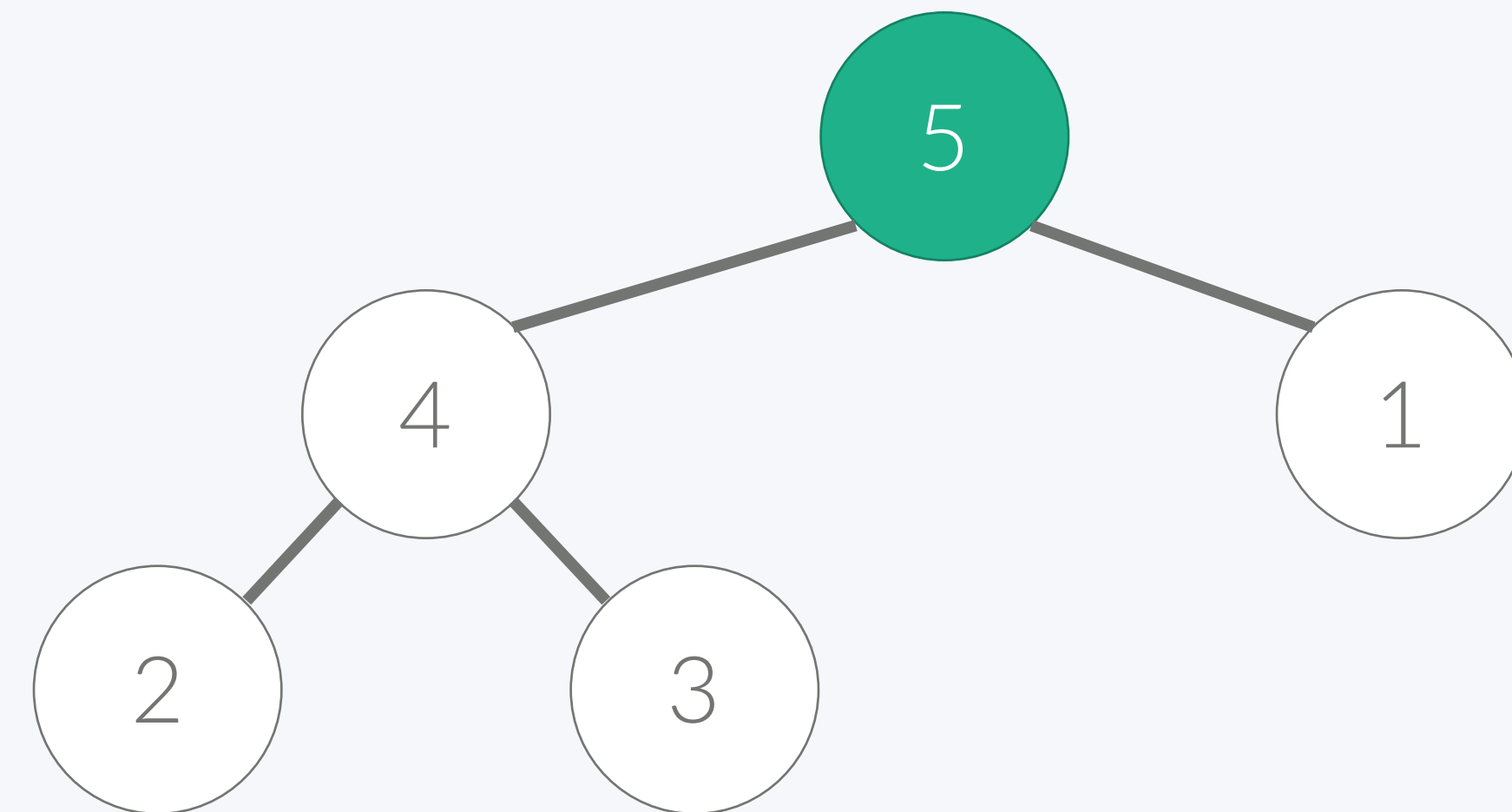


# 최대 힙 삽입

130

Max-Heap

- 5와 parent를 비교한다.
- 이 경우에  $4 < 5$  이기 때문에 Max-heap의 성질을 만족하지 않는다
- 따라서 두 수를 swap

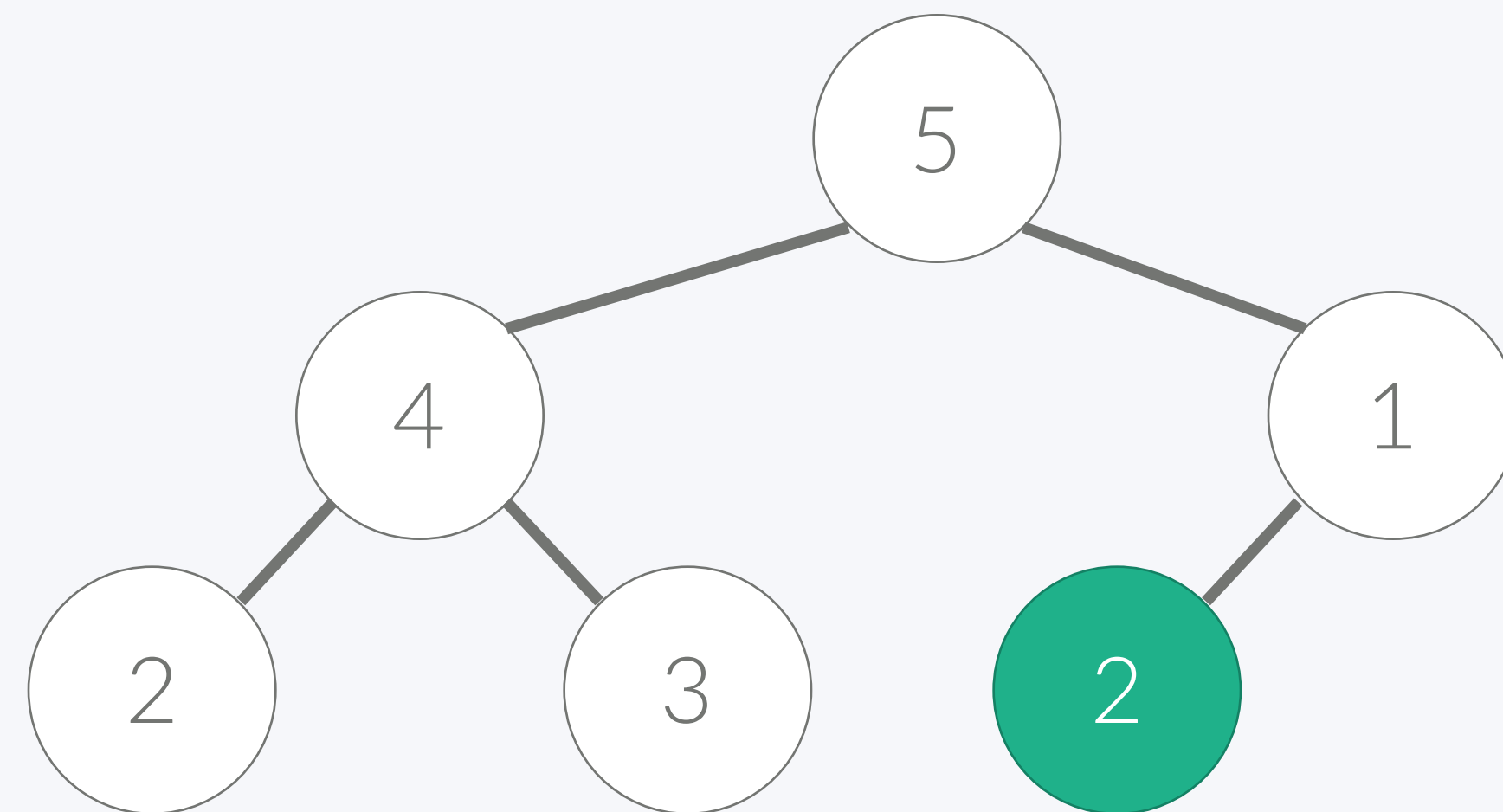


# 최대 힙 삽입

Max-Heap

- 2를 넣는다

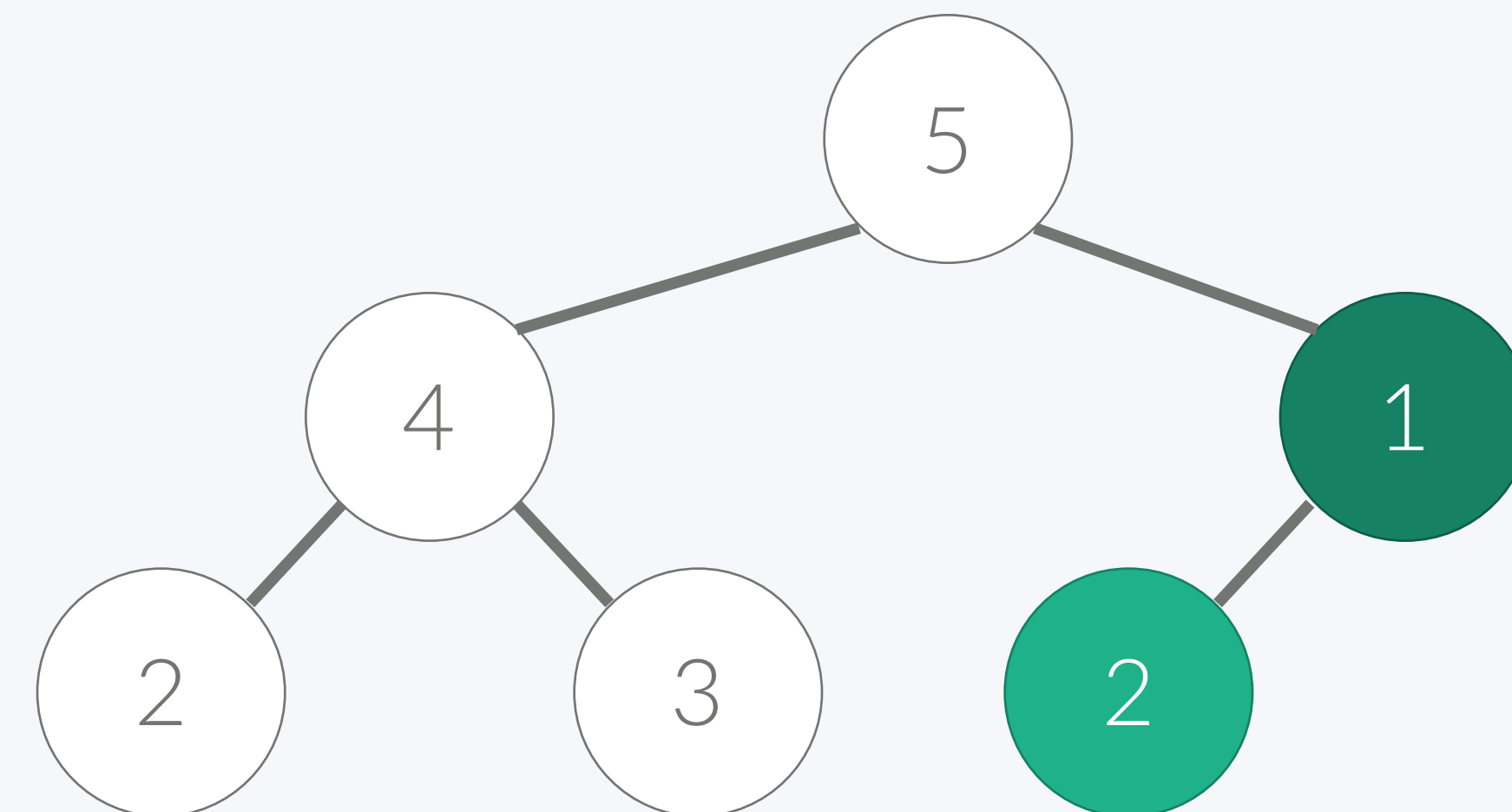
131



# 최대 힙 삽입

Max-Heap

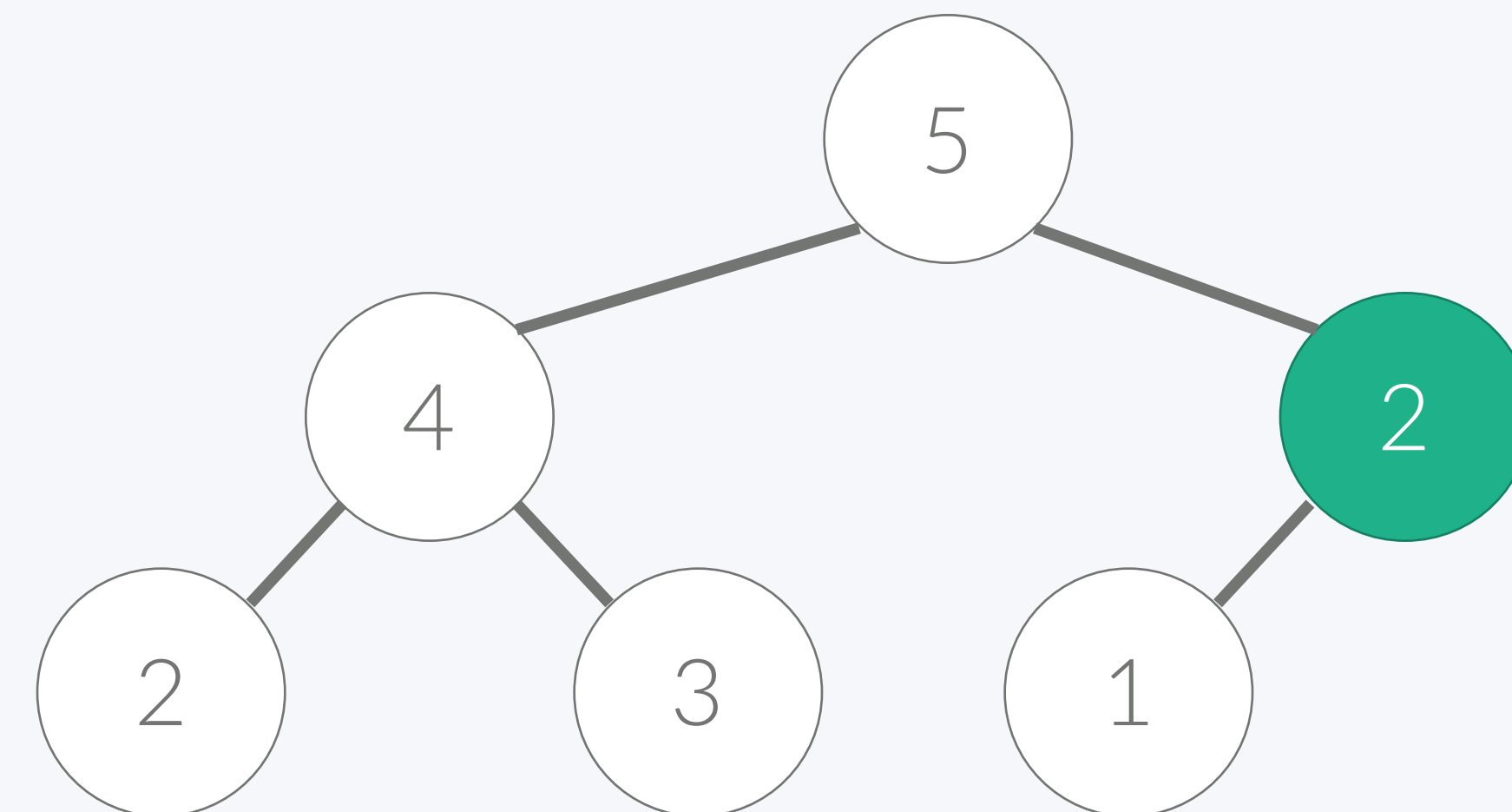
- 2와 2의 parent 1을 비교
- $1 < 2$  이기 때문에 swap



# 최대 힙 삽입

Max-Heap

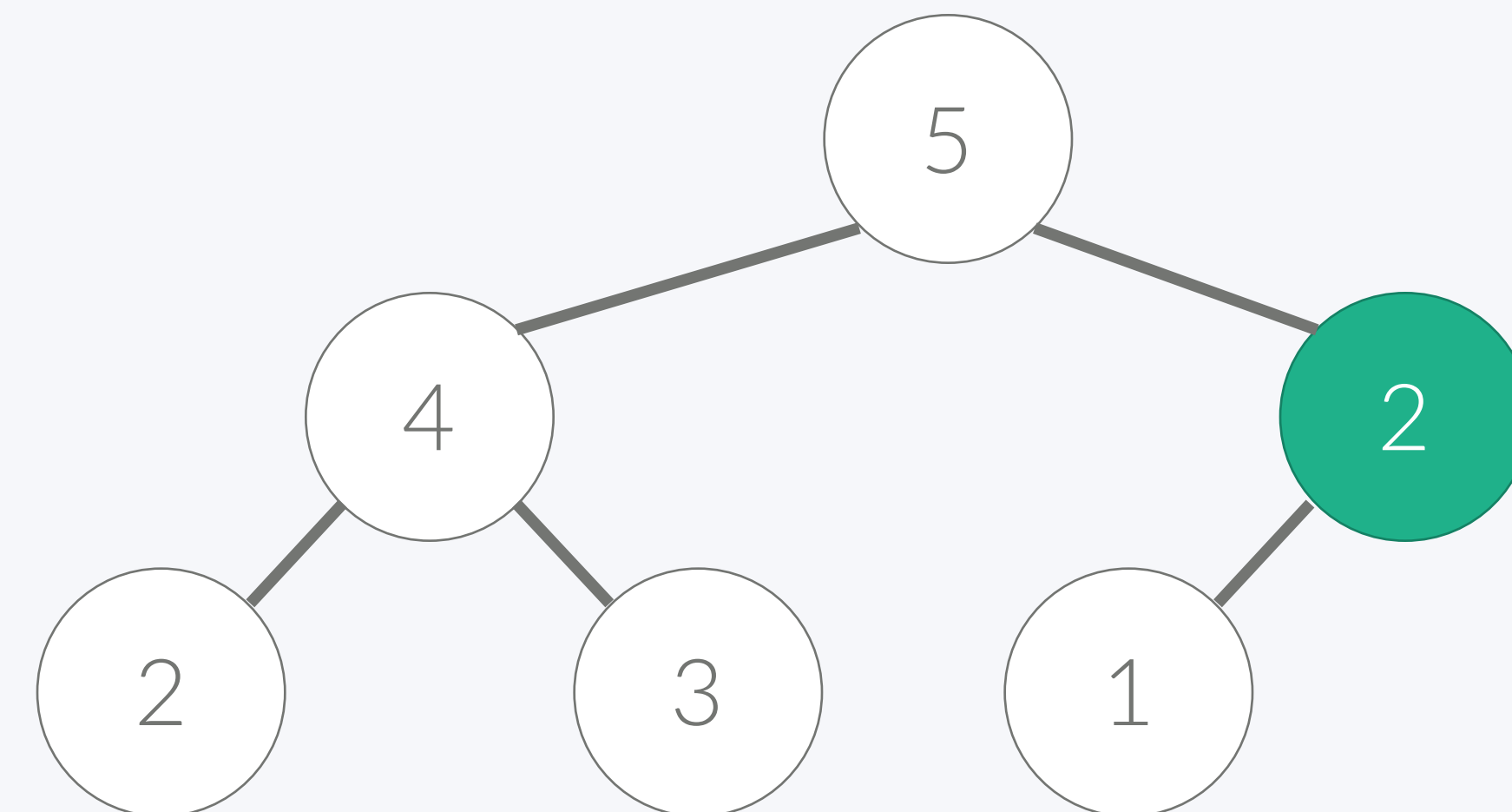
- 2와 2의 parent 1을 비교
- $1 < 2$  이기 때문에 swap



# 최대 힙 삽입

Max-Heap

- 2와 2의 parent 5을 비교
- $5 > 1$  이기 때문에 여기서 종료

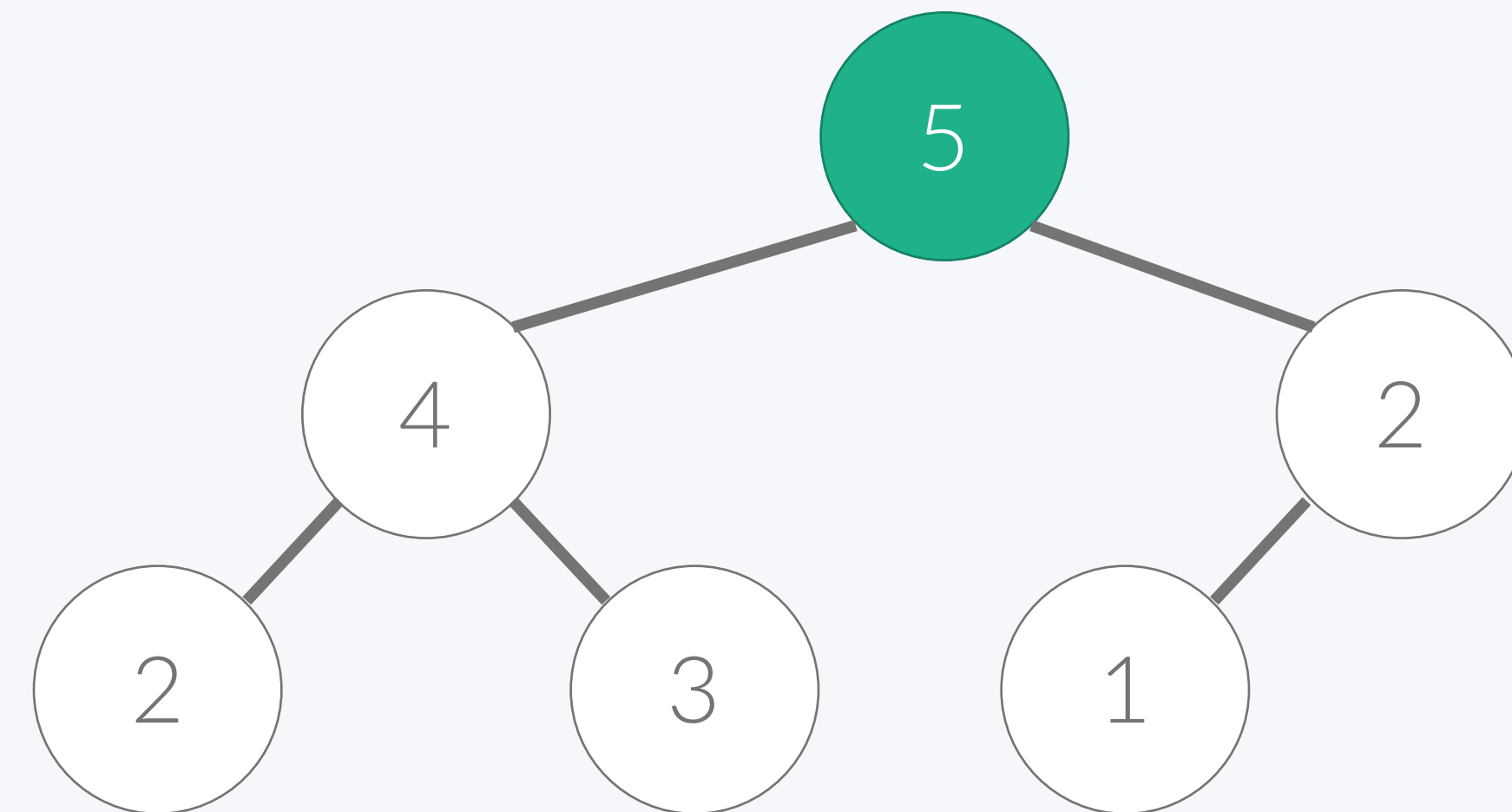


# 최대 힙 제거

Max-Heap

135

- 루트를 가장 마지막에 있는 값으로 바꿈

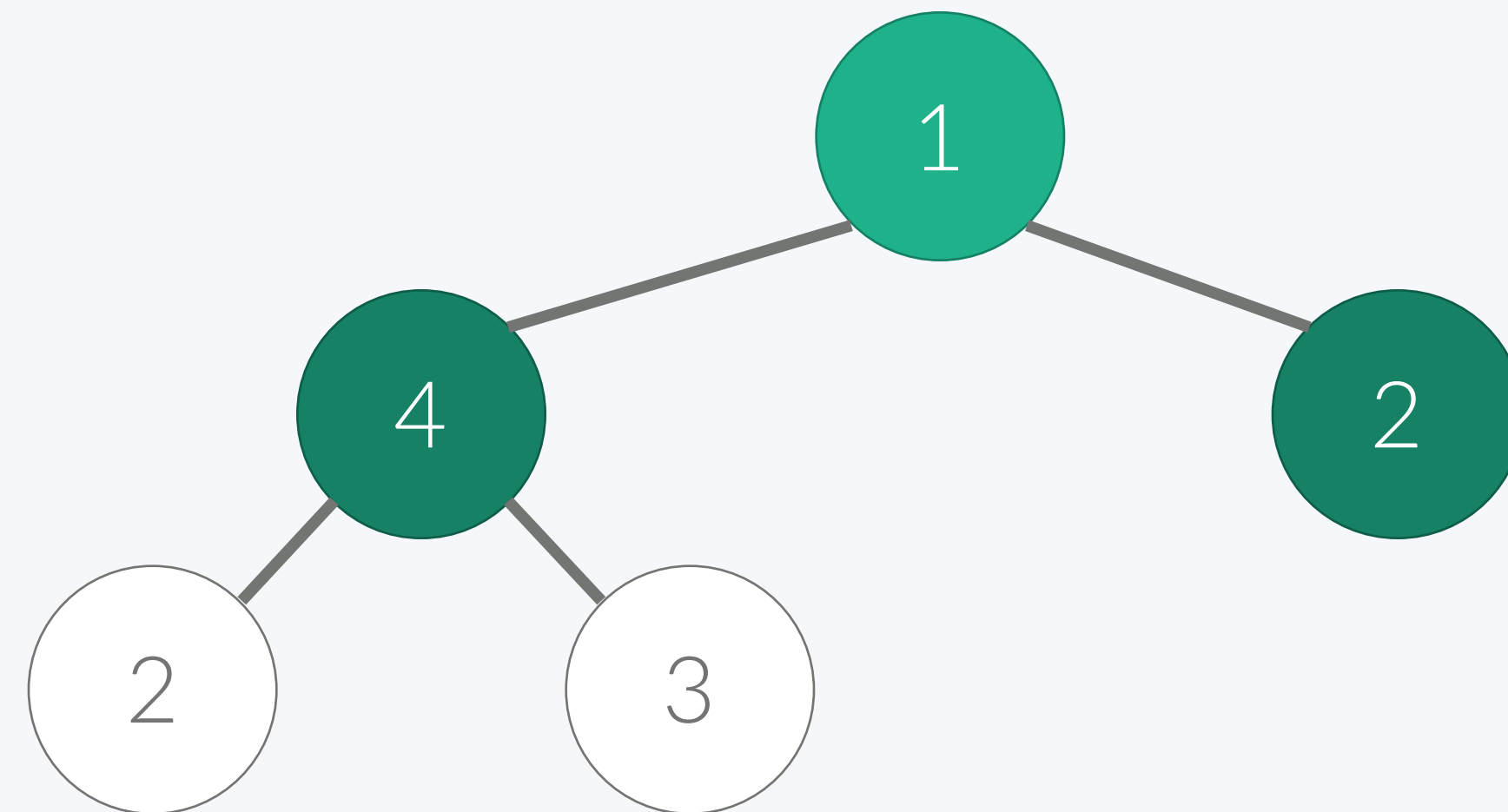


# 최대 힙 제거

Max-Heap

136

- children과 비교하면서 아래로 내려감
- Max-heap이기 때문에
- 루트 > children 을 만족하려면
- 4와 1을 바꿔야 함



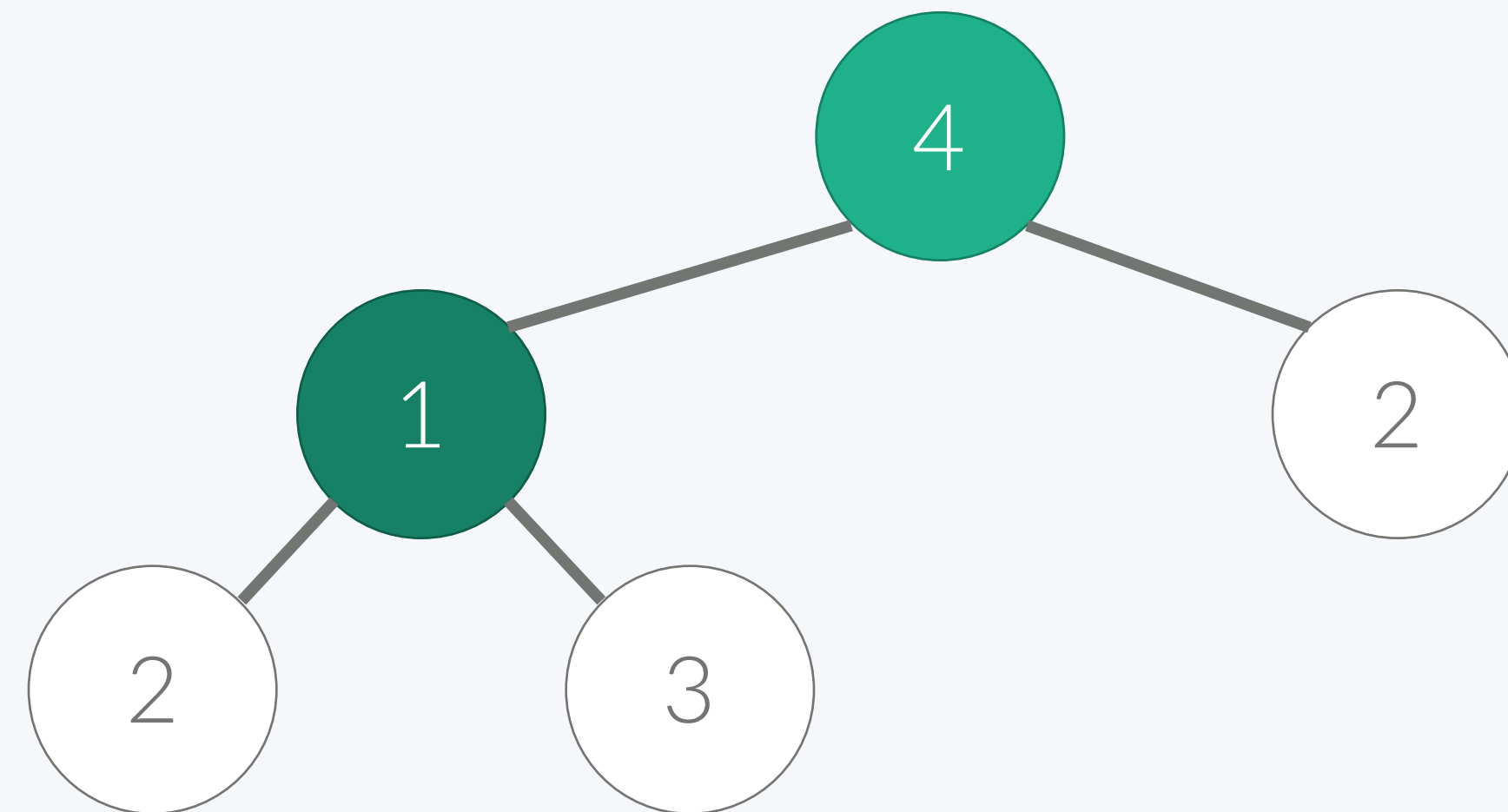


# 최대 힙 제거

Max-Heap

137

- children과 비교하면서 아래로 내려감
- Max-heap이기 때문에
- 루트 > children 을 만족하려면
- 4와 1을 바꿔야 함

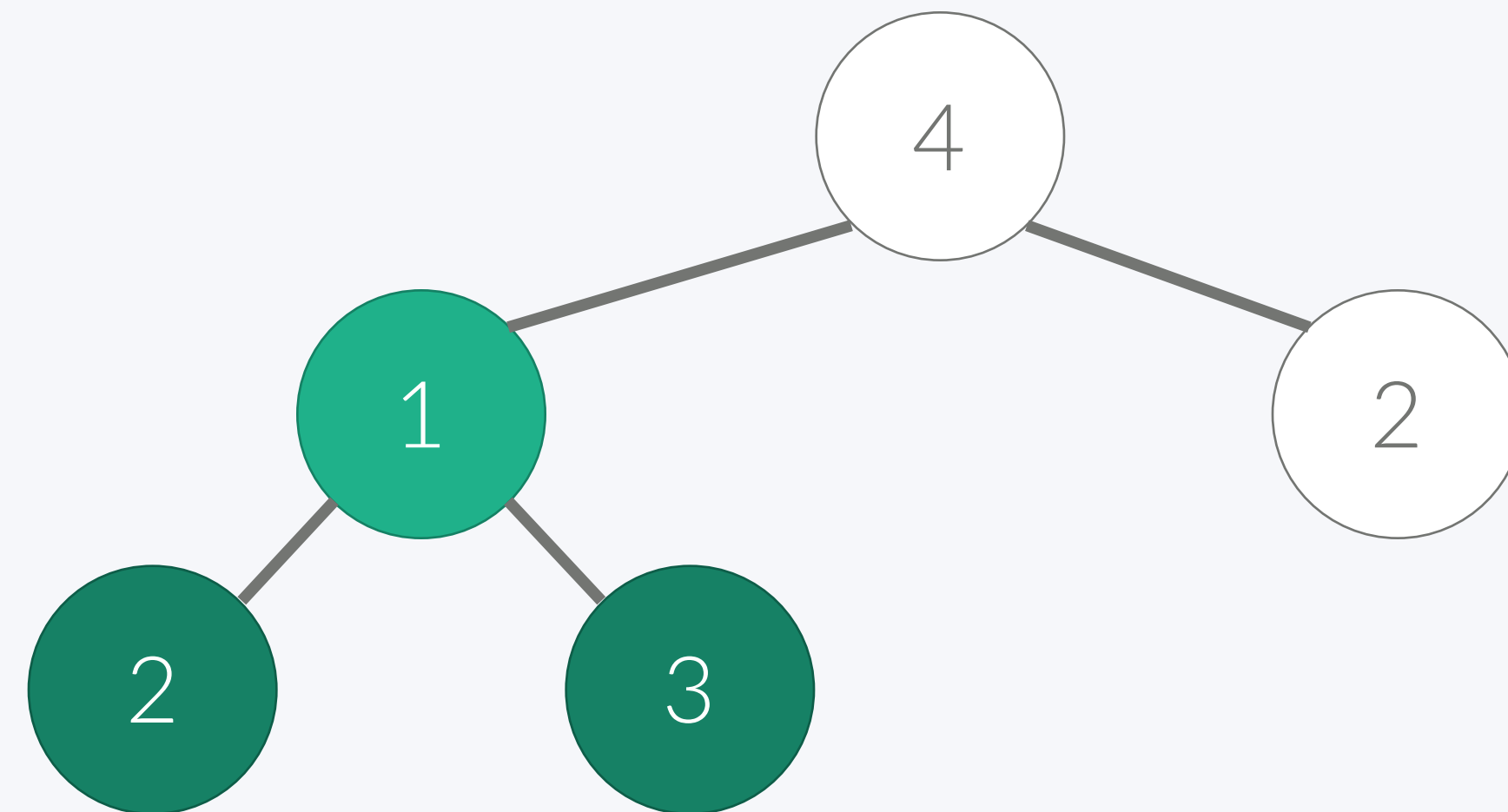


# 최대 힙 제거

Max-Heap

138

- children과 비교하면서 아래로 내려감

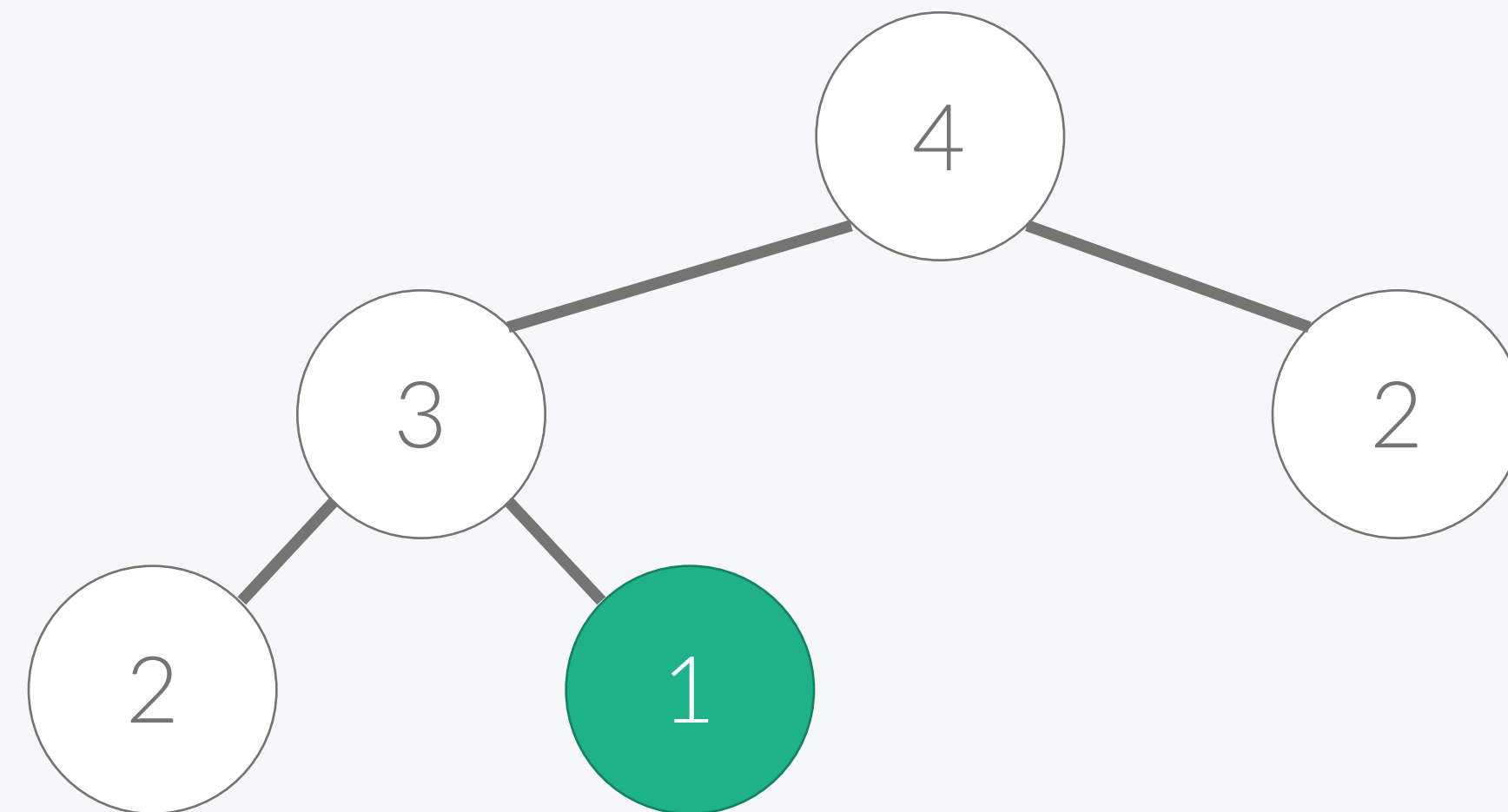


# 최대 힙 제거

Max-Heap

139

- children과 비교하면서 아래로 내려감

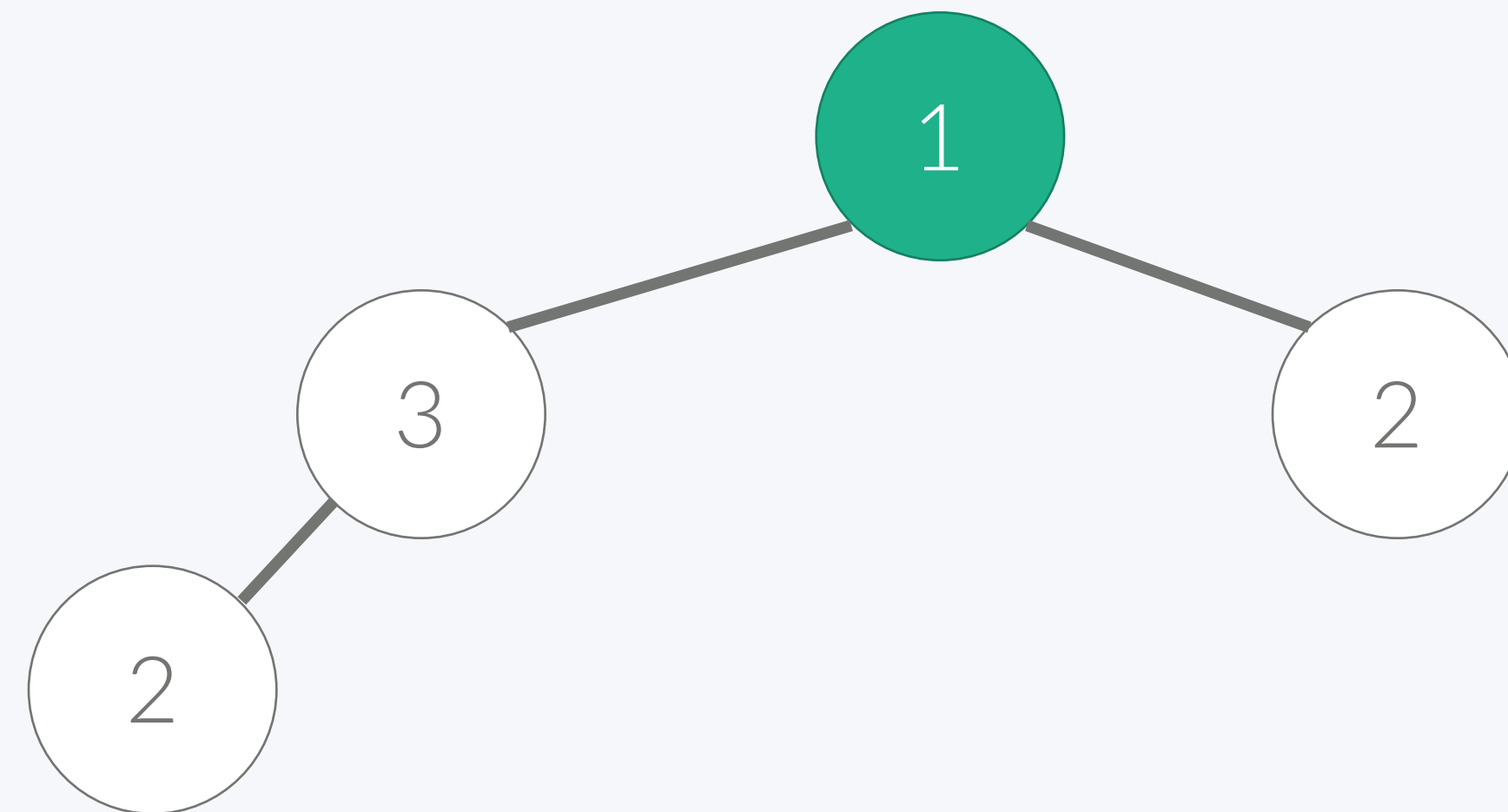


# 최대 힙 제거

Max-Heap

140

- 한 번 더 제거

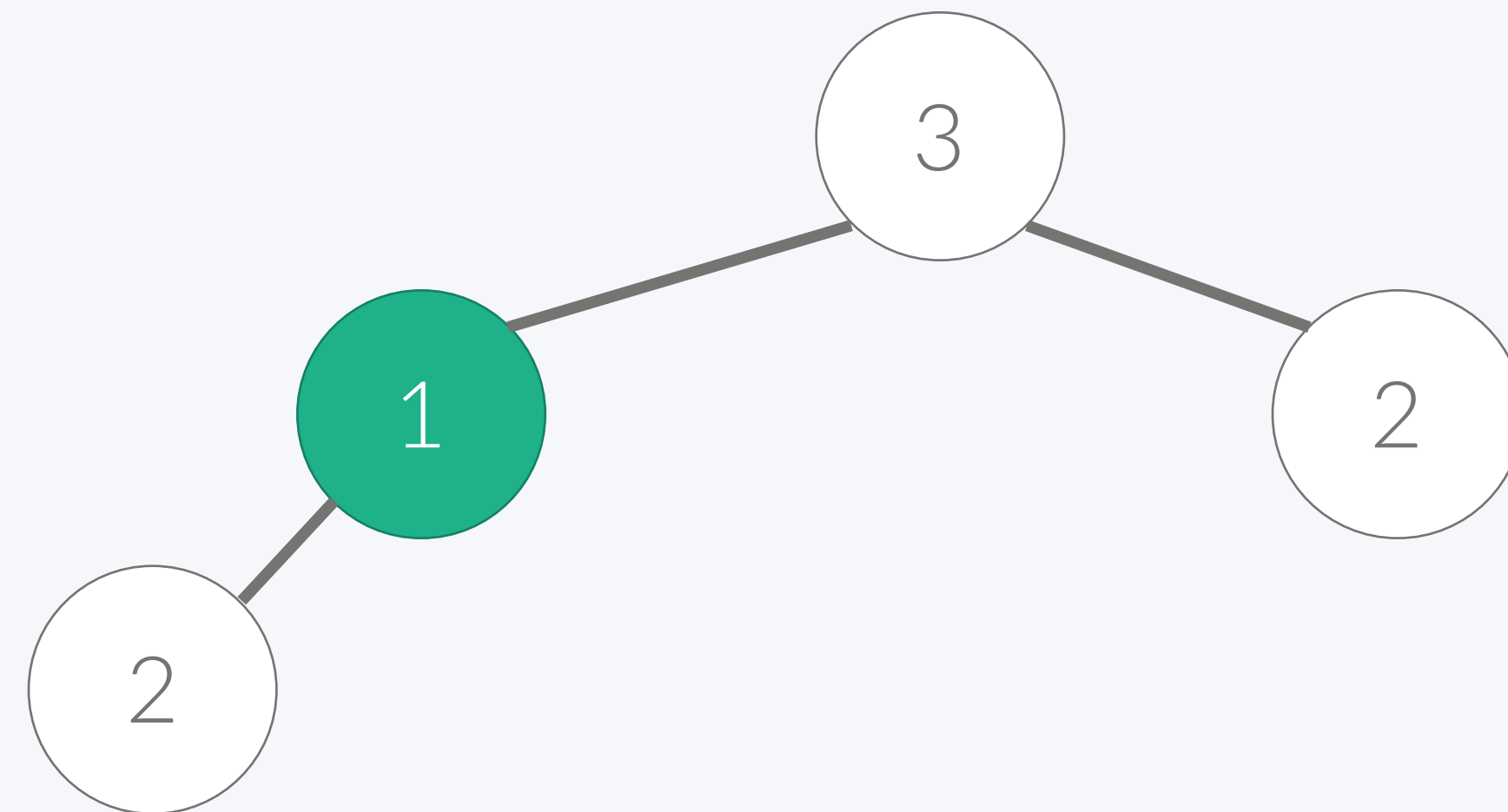


# 최대 힙 제거

Max-Heap

141

- children과 비교하면서 아래로 내려감

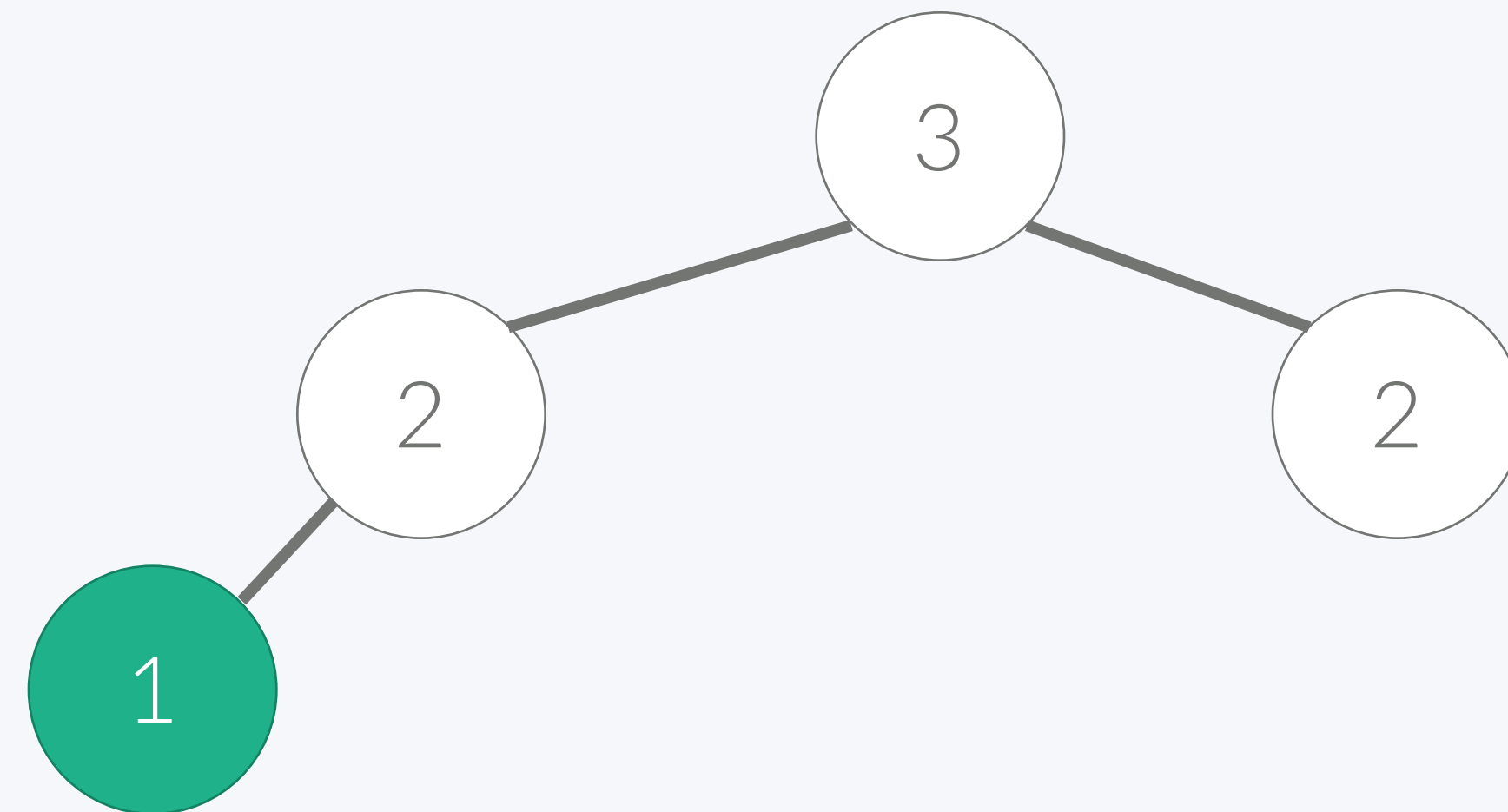


# 최대 힙 제거

Max-Heap

142

- children과 비교하면서 아래로 내려감



# 최대 힙

Max-Heap

143

- C/C++: <https://gist.github.com/Baekjoon/1ac8d329d982e564a3ea>
- Java: <https://gist.github.com/Baekjoon/87a8137d783fcbf45aa7>

# 최대 힙

144

<https://www.acmicpc.net/problem/11279>

- C/C++: <https://gist.github.com/Baekjoon/02af142497091c742baa>
- Java: <https://gist.github.com/Baekjoon/5b1349b9159e7548bb54>



# 최소 힙

145

<https://www.acmicpc.net/problem/1927>

- C/C++: <https://gist.github.com/Baekjoon/92f4fd7261b10723227d>
- Java: <https://gist.github.com/Baekjoon/215d09220124cf3cb2af>

# 이진 검색 트리

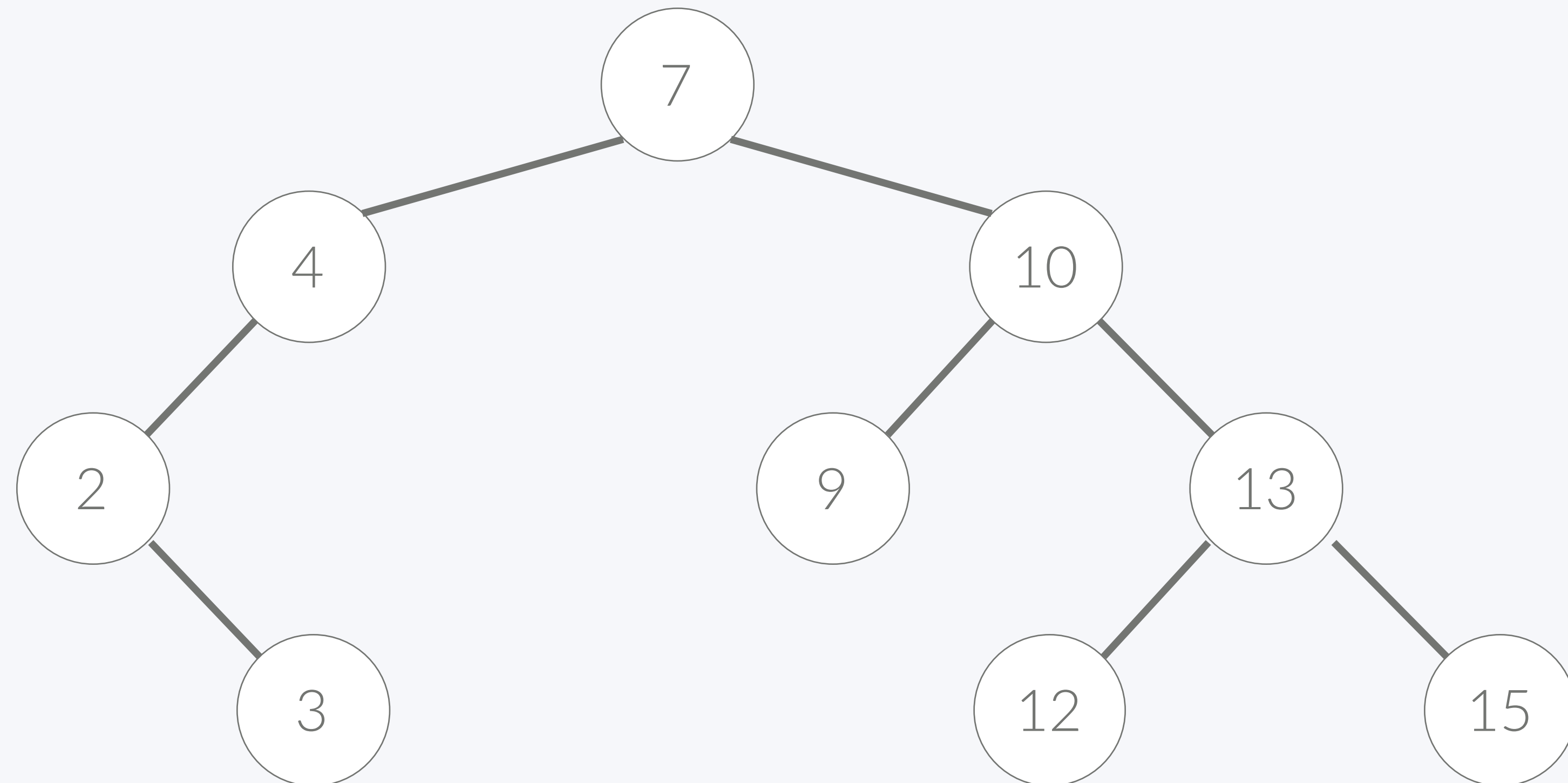
---

# 이진 검색 트리

Binary Search Tree

147

- 이진 트리
- 현재 노드의 왼쪽 서브 트리에는 항상 현재 노드의 값보다 작은 값이 들어있고
- 현재 노드의 오른쪽 서브 트리에는 항상 현재 노드의 값보다 큰 값이 들어있다



# 이진 검색 트리

Binary Search Tree

148

- 이진 검색의 원리를 이용해서 트리를 만들었다
- 어떤 자료의 삽입/삭제/검색이 모두  $O(\lg N)$ 이 걸린다
- 트리가 균형이 맞지 않으면  $O(N)$ 이 걸리기 때문에
- 균형이 맞춰져있는 BST를 사용해야 한다

# 이진 검색 트리

Binary Search Tree

149

- 균형이 맞춰져 있는 BST는
- AVL-Tree
- Red-black Tree
- Splay Tree
- Treap 이 있다

# 이진 검색 트리

Binary Search Tree

150

- STL을 사용하는 경우에는 set을 사용하면 된다.

# 회사에 있는 사람

151

<https://www.acmicpc.net/problem/7785>

- 회사 모든 사람의 출입카드 시스템 로그를 가지고 있다.
- 로그는 어떤 사람이 들어갔는지, 나갔는지가 기록되어 있다.
- 로그가 주어졌을 때, 회사에 있는 모든 사람을 구하는 문제

# 회사에 있는 사람

152

<https://www.acmicpc.net/problem/7785>

- C++: <https://gist.github.com/Baekjoon/9ea0da3cf46c394dc1a5>



# 듣보잡

153

<https://www.acmicpc.net/problem/1764>

- 듣도 못한 사람과 보도 못한 사람의 명단이 주어졌을 때
- 듣도 보도 못한 사람의 명단을 구하는 문제

# 등보잡

<https://www.acmicpc.net/problem/7785>

- map 사용: <https://gist.github.com/Baekjoon/eff5f116fbbb07231675>
- set 사용: <https://gist.github.com/Baekjoon/2f5caeca4fd25b8c4021>
- 머지 소트 사용: <https://gist.github.com/Baekjoon/385f7d8129d46dd353d7>
- set\_intersection 사용: <https://gist.github.com/Baekjoon/130329076d21f97f76cc>