/* 데이터 구조 및 알고리즘 */

신현규 강사, 화/목 20:00

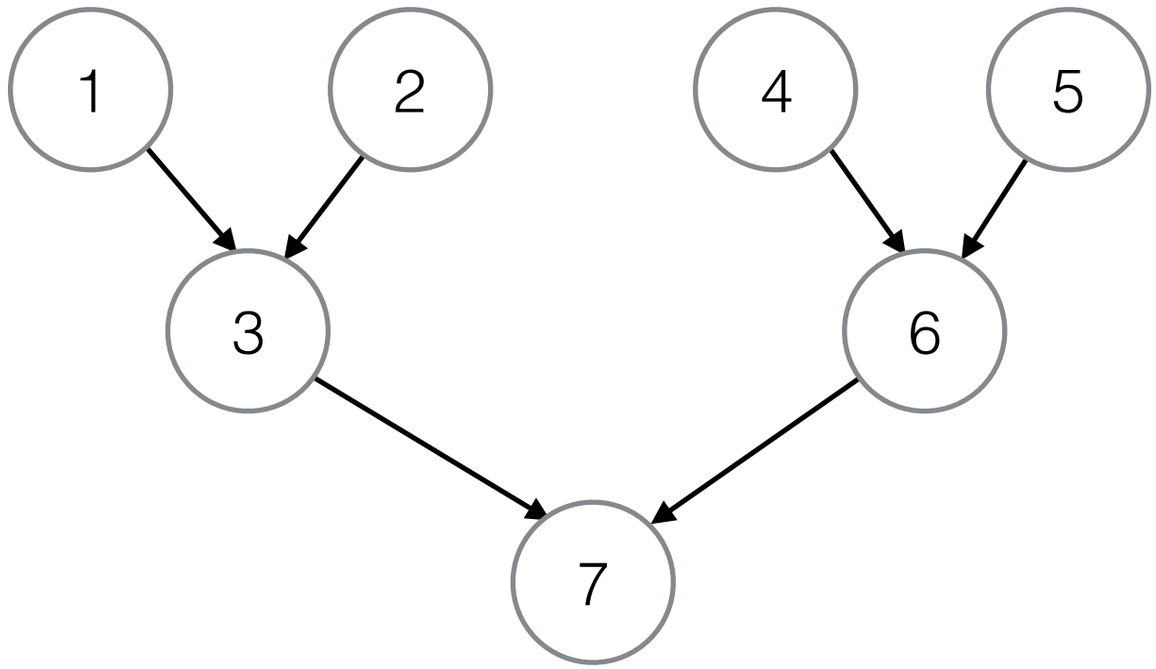
시간복잡도 (1)

/* elice */

퀴즈

1. 다음 문제는 스택으로 해결이 가능할까 ? 큐로는 해결 가능할까 ?

일을 어느 순서대로 처리해야 하는지를 출력하시오.
단, 1 —> 3 일 경우 3보다는 1을 먼저 해야 함.



2. 다음 예제에서 (19 , 7) 은 어느 위치에 삽입되는가 ?

hash function h(x) = (3*x)%8

	0	1	2	3	4	5	6	7
hash Table		(9, 17)	(14, 12)	(1, 8)		(7, 2)		

주차별 커리큘럼

1주차	과정 소개, 배열, 연결리스트, 클래스
2주차	스택, 큐, 해싱
3주차	**시간복잡도**
4주차	트리, 트리순회, 재귀호출
5주차	힙
6주차	그래프 소개, DFS
7주차	그래프 심화, BFS
8주차	강의 요약, 알고리즘 과정 소개

컴퓨터를 이용한 문제 해결 과정

컴퓨터를 이용한 문제 해결 과정

1. 문제를 정확히 이해한다

컴퓨터를 이용한 문제 해결 과정

1. 문제를 정확히 이해한다

2. 문제를 해결하는 알고리즘을 개발한다

컴퓨터를 이용한 문제 해결 과정

1. 문제를 정확히 이해한다
2. 문제를 해결하는 알고리즘을 개발한다
3. 알고리즘이 문제를 해결한다는 것을 증명한다

컴퓨터를 이용한 문제 해결 과정

1. 문제를 정확히 이해한다
2. 문제를 해결하는 알고리즘을 개발한다
3. 알고리즘이 문제를 해결한다는 것을 증명한다
4. 알고리즘이 제한시간 내에 동작한다는 것을 보인다

컴퓨터를 이용한 문제 해결 과정

1. 문제를 정확히 이해한다
2. 문제를 해결하는 알고리즘을 개발한다
3. 알고리즘이 문제를 해결한다는 것을 증명한다
4. 알고리즘이 제한시간 내에 동작한다는 것을 보인다
5. 알고리즘을 코드로 작성한다

컴퓨터를 이용한 문제 해결 과정

1. 문제를 정확히 이해한다
2. 문제를 해결하는 알고리즘을 개발한다
3. 알고리즘이 문제를 해결한다는 것을 증명한다
4. 알고리즘이 제한시간 내에 동작한다는 것을 보인다
5. 알고리즘을 코드로 작성한다
6. 제출 후 만점을 받고 매우 기뻐한다

컴퓨터를 이용한 문제 해결 과정

1. 문제를 정확히 이해한다
2. 문제를 해결하는 알고리즘을 개발한다
3. 알고리즘이 **문제를 해결한다는 것을 증명한다**
4. 알고리즘이 **제한시간 내에 동작한다는 것을 보인다**
5. 알고리즘을 코드로 작성한다
6. 제출 후 만점을 받고 매우 기뻐한다

[연습문제] 최댓값 기계

숫자들 중에서 최댓값을 반환

시스템 입력

```
mc = maxMachine()


mc.addNumber(2)
mc.addNumber(3)
print(mc.getMax())
mc.removeNumber(3)
print(mc.getMax())
```

시스템 출력

```
3
2
```

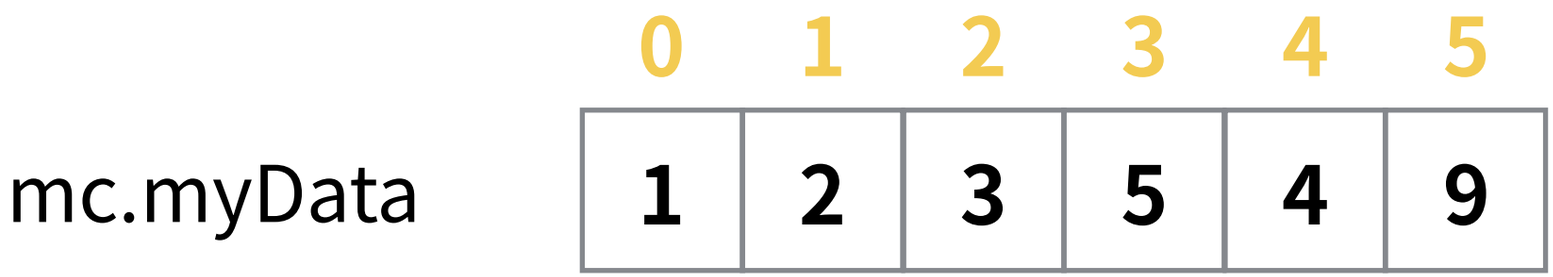
컴퓨터를 이용한 문제 해결 과정

1. 문제를 정확히 이해한다
2. 문제를 해결하는 알고리즘을 개발한다
3. 알고리즘이 **문제를 해결한다는 것을 증명한다**
4. 알고리즘이 **제한시간 내에 동작한다는 것을 보인다**
5. 알고리즘을 코드로 작성한다
6. 제출 후 만점을 받고 매우 기뻐한다

컴퓨터를 이용한 문제 해결 과정

2. 문제를 해결하는 알고리즘을 개발한다

`addNumber()` `removeNumber()` `getMax()`



컴퓨터를 이용한 문제 해결 과정

2. 문제를 해결하는 알고리즘을 개발한다

`addNumber()` `removeNumber()` `getMax()`

```
def addNumber(self, n) :
    self.myData.append(n)
```

컴퓨터를 이용한 문제 해결 과정

2. 문제를 해결하는 알고리즘을 개발한다

`addNumber()` `removeNumber()` `getMax()`

```
def addNumber(self, n) :
    self.myData.append(n)
```

```
def removeNumber(self, n) :
    self.myData.remove(n)
```

컴퓨터를 이용한 문제 해결 과정

2. 문제를 해결하는 알고리즘을 개발한다

`addNumber()`

```
def addNumber(self, n) :
    self.myData.append(n)
```

`removeNumber()`

```
def removeNumber(self, n) :
    self.myData.remove(n)
```

`getMax()`

```
def getMax(self) :
    return max(self.myData)
```

컴퓨터를 이용한 문제 해결 과정

3. 알고리즘이 **문제를 해결한다는 것을 증명한다**

`addNumber()` `removeNumber()` `getMax()`

함수가 하는 일이 곧 증명

컴퓨터를 이용한 문제 해결 과정

4. 알고리즘이 **제한시간** 내에 동작한다는 것을 보인다

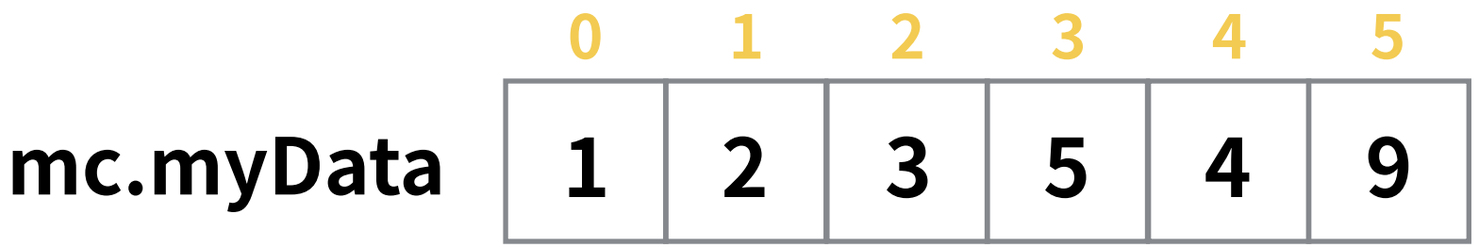
`addNumber()` `removeNumber()` `getMax()`

컴퓨터를 이용한 문제 해결 과정

4. 알고리즘이 **제한시간** 내에 동작한다는 것을 보인다

`addNumber()` `removeNumber()` `getMax()`

대충 몇 개의 명령을 수행하는가 ?



컴퓨터를 이용한 문제 해결 과정

4. 알고리즘이 제한시간 내에 동작한다는 것을 보인다

`addNumber()`

```
def addNumber(self, n) :
    self.myData.append(n)
```

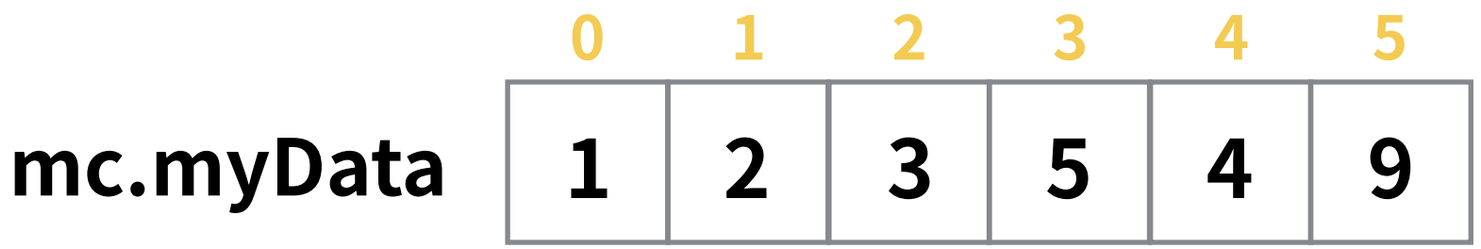
`removeNumber()`

```
def removeNumber(self, n) :
    self.myData.remove(n)
```

`getMax()`

```
def getMax(self) :
    return max(self.myData)
```

대충 몇 개의 명령을 수행하는가 ?



컴퓨터를 이용한 문제 해결 과정

4. 알고리즘이 제한시간 내에 동작한다는 것을 보인다

`addNumber()`

```
def addNumber(self, n) :
    self.myData.append(n)
```

`removeNumber()`

```
def removeNumber(self, n) :
    self.myData.remove(n)
```

`getMax()`

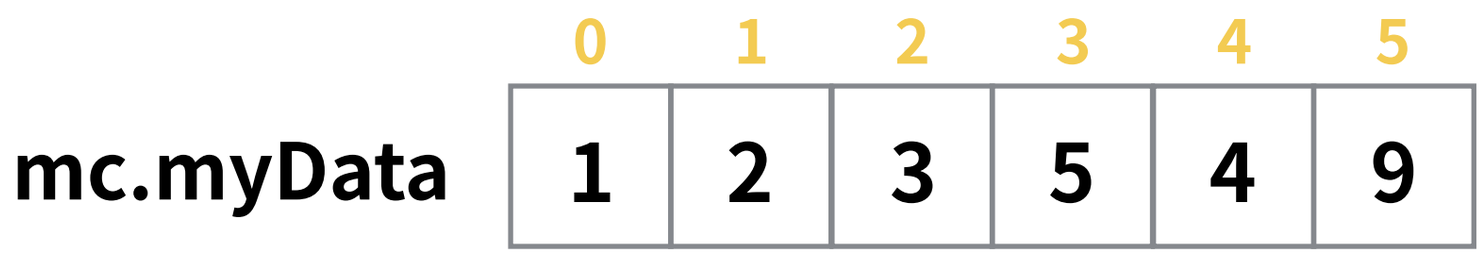
```
def getMax(self) :
    return max(self.myData)
```

대충 몇 개의 명령을 수행하는가 ?

대충 2개

대충 n개

대충 n개



컴퓨터를 이용한 문제 해결 과정

4. 알고리즘이 제한시간 내에 동작한다는 것을 보인다

addNumber()

```
def addNumber(self, n) :
    self.myData.append(n)
```

removeNumber()

```
def removeNumber(self, n) :
    self.myData.remove(n)
```

getMax()

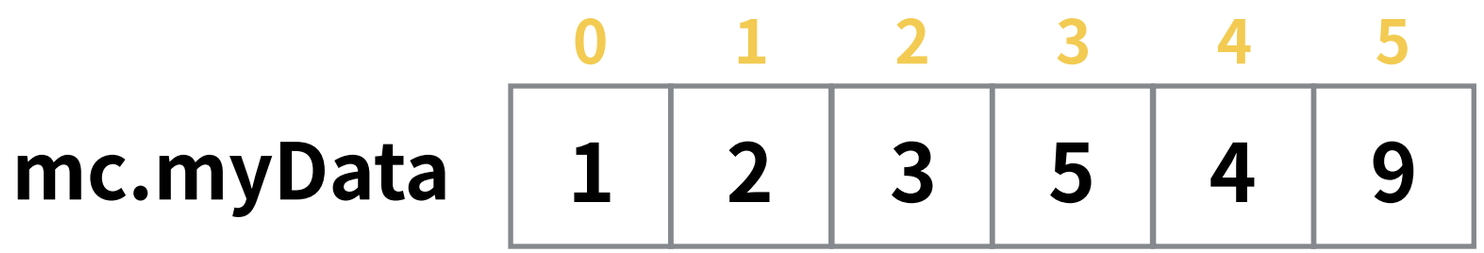
```
def getMax(self) :
    return max(self.myData)
```

대충 몇 개의 명령을 수행하는가 ?

O(1)

O(n)

O(n)



컴퓨터를 이용한 문제 해결 과정

5. 알고리즘을 코드로 작성한다

6. 제출 후 만점을 받고 매우 기뻐한다

[문제 7] 주문 처리하기

Elice씨가 주문을 처리하는 순서를 출력

시스템 입력

```
4
1 3 0
3 3 0
5 3 0
7 3 1
```

시스템 출력

```
1 2 4 3
```

컴퓨터를 이용한 문제 해결 과정

1. 문제를 정확히 이해한다
2. 문제를 해결하는 알고리즘을 개발한다
3. 알고리즘이 **문제를 해결한다는 것을 증명한다**
4. 알고리즘이 **제한시간 내에 동작한다는 것을 보인다**
5. 알고리즘을 코드로 작성한다
6. 제출 후 만점을 받고 매우 기뻐한다

컴퓨터를 이용한 문제 해결 과정

2. 문제를 해결하는 알고리즘을 개발한다

```
4
1 3 0
3 3 0
5 3 0
7 3 1
```

컴퓨터를 이용한 문제 해결 과정

2. 문제를 해결하는 알고리즘을 개발한다

일단 첫번째 일을 해볼까…

```
4
1 3 0
3 3 0
5 3 0
7 3 1
```

컴퓨터를 이용한 문제 해결 과정

2. 문제를 해결하는 알고리즘을 개발한다

일단 첫번째 일을 해볼까… — 시간 4에 끝난다

```
4
1 3 0
3 3 0
5 3 0
7 3 1
```

컴퓨터를 이용한 문제 해결 과정

2. 문제를 해결하는 알고리즘을 개발한다

일단 첫번째 일을 해볼까… — 시간 4에 끝난다

그럼 두 번째 일을 바로 할 수 있네 — 시간 7에 끝난다

4

~~1 3 0~~

3 3 0

5 3 0

7 3 1

컴퓨터를 이용한 문제 해결 과정

2. 문제를 해결하는 알고리즘을 개발한다

일단 첫번째 일을 해볼까… — 시간 4에 끝난다

그럼 두 번째 일을 바로 할 수 있네 — 시간 7에 끝난다

그럼 할 수 있는 일이 두개가 생긴다!

4

~~1 3 0~~

~~3 3 0~~

5 3 0

7 3 1

컴퓨터를 이용한 문제 해결 과정

2. 문제를 해결하는 알고리즘을 개발한다

일단 첫번째 일을 해볼까… — 시간 4에 끝난다

그럼 두 번째 일을 바로 할 수 있네 — 시간 7에 끝난다

그럼 할 수 있는 일이 두개가 생긴다! — VIP부터

4

~~1 3 0~~

~~3 3 0~~

5 3 0

7 3 1

컴퓨터를 이용한 문제 해결 과정

2. 문제를 해결하는 알고리즘을 개발한다

일단 첫번째 일을 해볼까… — 시간 4에 끝난다

그럼 두 번째 일을 바로 할 수 있네 — 시간 7에 끝난다

그럼 할 수 있는 일이 두개가 생긴다! — VIP부터

4

~~1 3 0~~

~~3 3 0~~

~~5 3 0~~

~~7 3 1~~

컴퓨터를 이용한 문제 해결 과정

2. 문제를 해결하는 알고리즘을 개발한다

관찰 1 : Elice씨가 “일을 마치는 시간”이 중요하다

관찰 2 : 일 사이에 우선순위가 있다

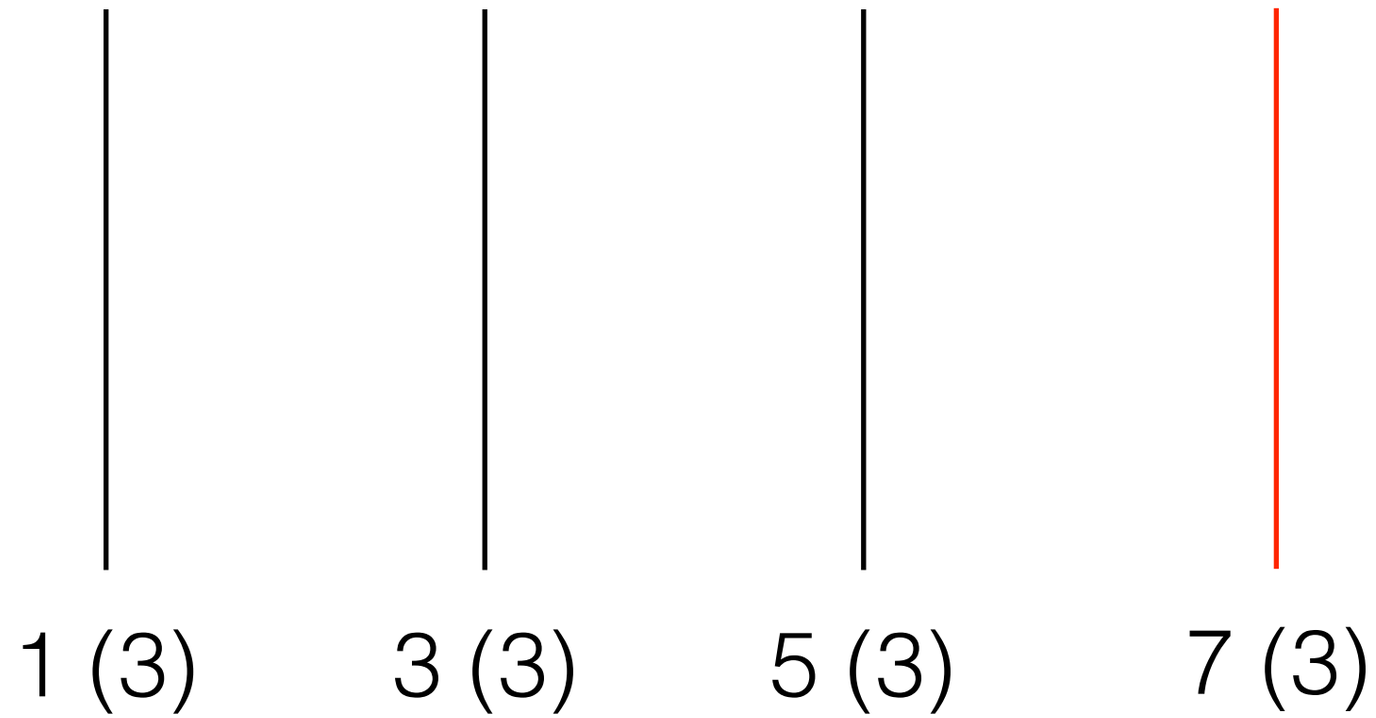
```
4
1 3 0
3 3 0
5 3 0
7 3 1
```

컴퓨터를 이용한 문제 해결 과정

2. 문제를 해결하는 알고리즘을 개발한다

관찰 1 : Elice씨가 “일을 마치는 시간”이 중요하다

관찰 2 : 일 사이에 우선순위가 있다



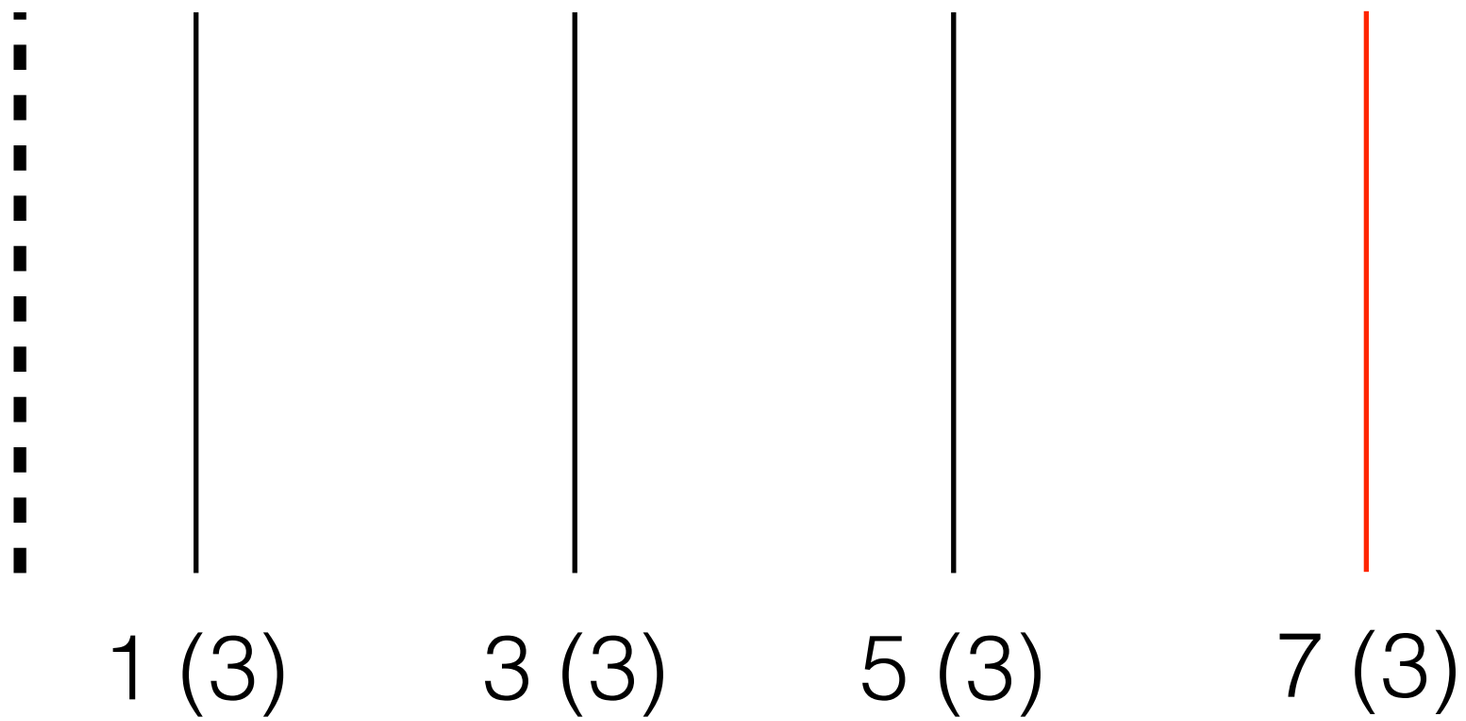

```
4
1 3 0
3 3 0
5 3 0
7 3 1
```

컴퓨터를 이용한 문제 해결 과정

2. 문제를 해결하는 알고리즘을 개발한다

관찰 1 : Elice씨가 “일을 마치는 시간”이 중요하다

관찰 2 : 일 사이에 우선순위가 있다



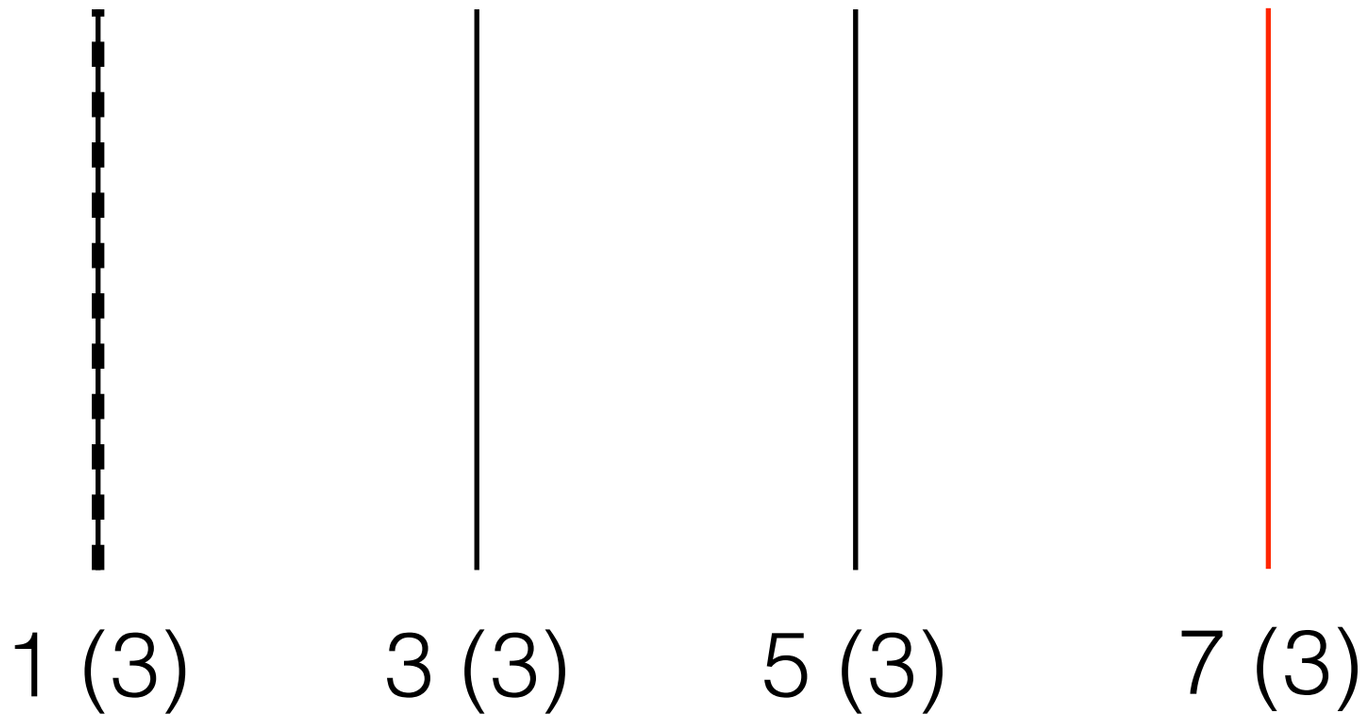

```
4
1 3 0
3 3 0
5 3 0
7 3 1
```

컴퓨터를 이용한 문제 해결 과정

2. 문제를 해결하는 알고리즘을 개발한다

관찰 1 : Elice씨가 “일을 마치는 시간”이 중요하다

관찰 2 : 일 사이에 우선순위가 있다



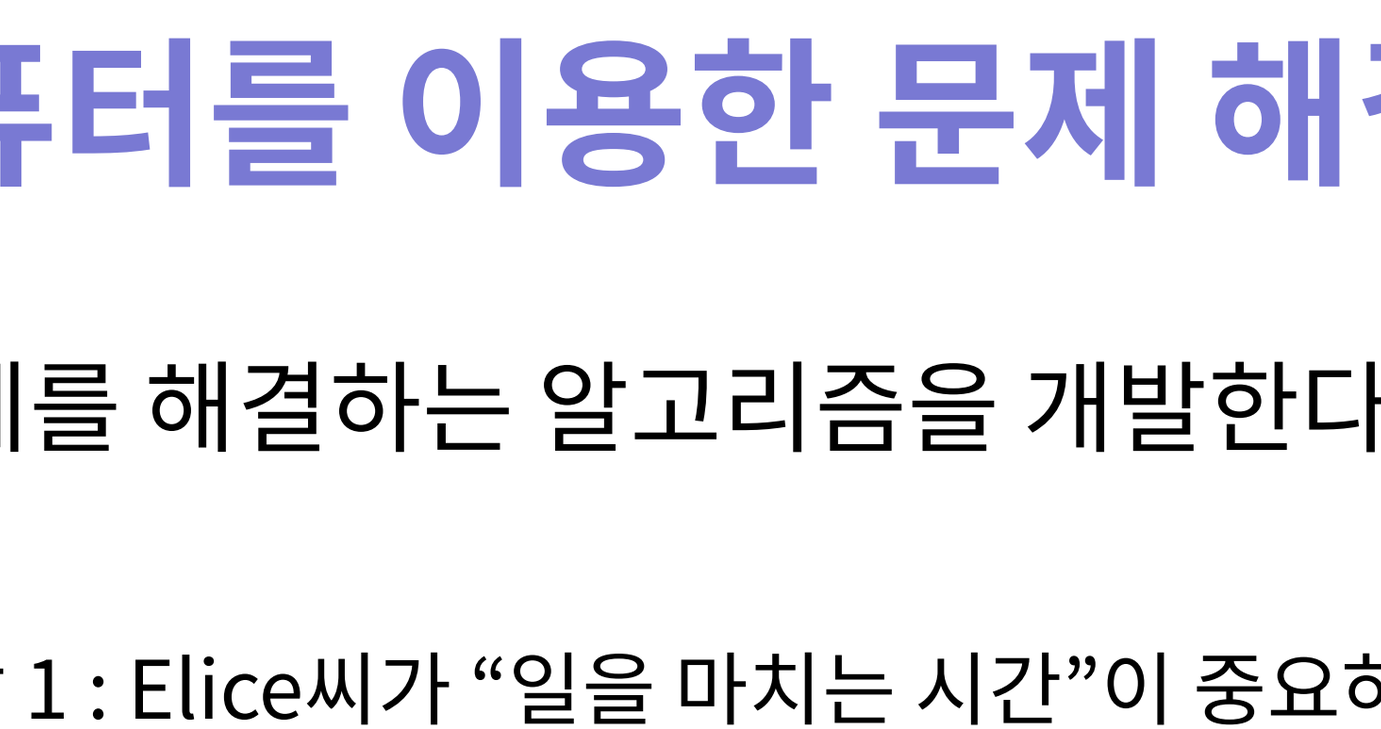

```
4
1 3 0
3 3 0
5 3 0
7 3 1
```

컴퓨터를 이용한 문제 해결 과정

2. 문제를 해결하는 알고리즘을 개발한다

관찰 1 : Elice씨가 “일을 마치는 시간”이 중요하다

관찰 2 : 일 사이에 우선순위가 있다



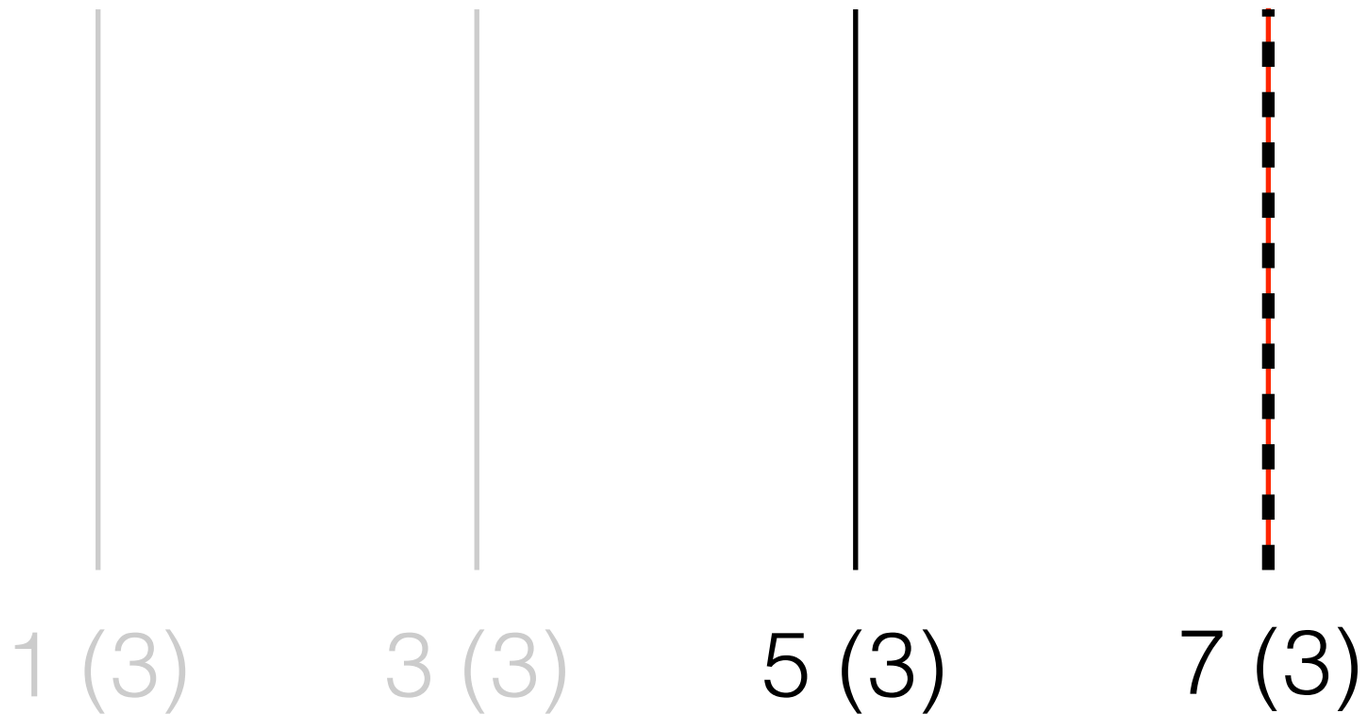

```
4
1 3 0
3 3 0
5 3 0
7 3 1
```

컴퓨터를 이용한 문제 해결 과정

2. 문제를 해결하는 알고리즘을 개발한다

관찰 1 : Elice씨가 “일을 마치는 시간”이 중요하다

관찰 2 : 일 사이에 우선순위가 있다



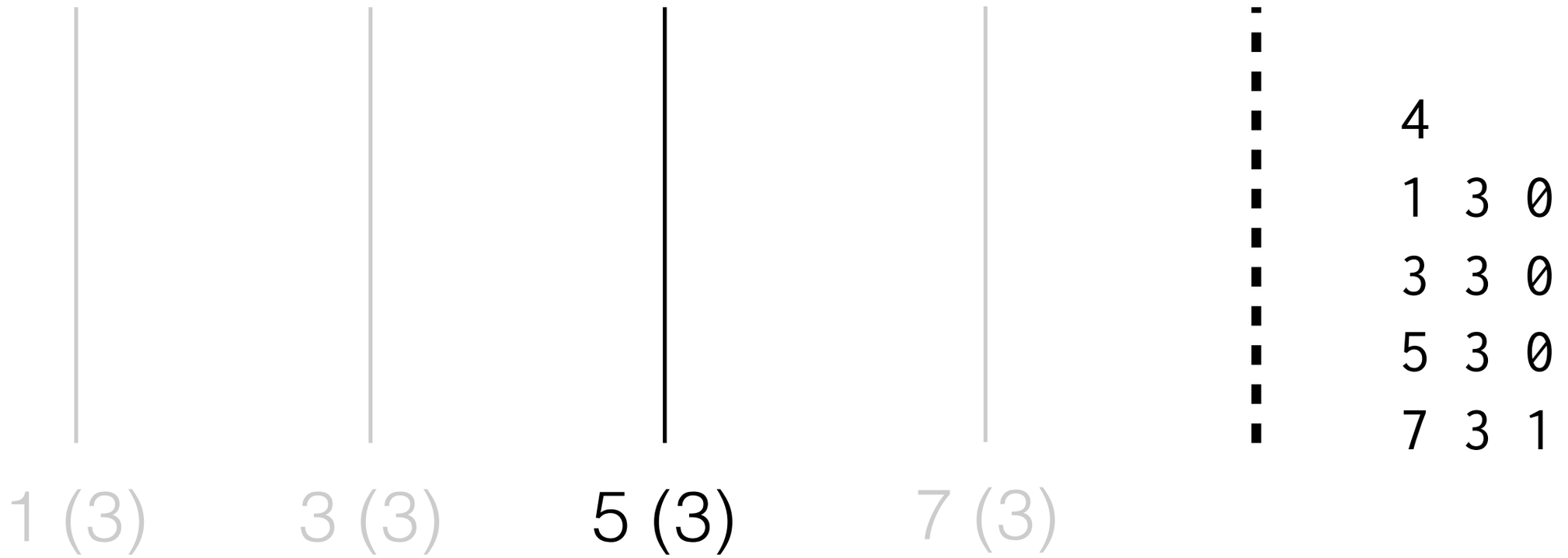

```
4
1 3 0
3 3 0
5 3 0
7 3 1
```

컴퓨터를 이용한 문제 해결 과정

2. 문제를 해결하는 알고리즘을 개발한다

관찰 1 : Elice씨가 “일을 마치는 시간”이 중요하다

관찰 2 : 일 사이에 우선순위가 있다

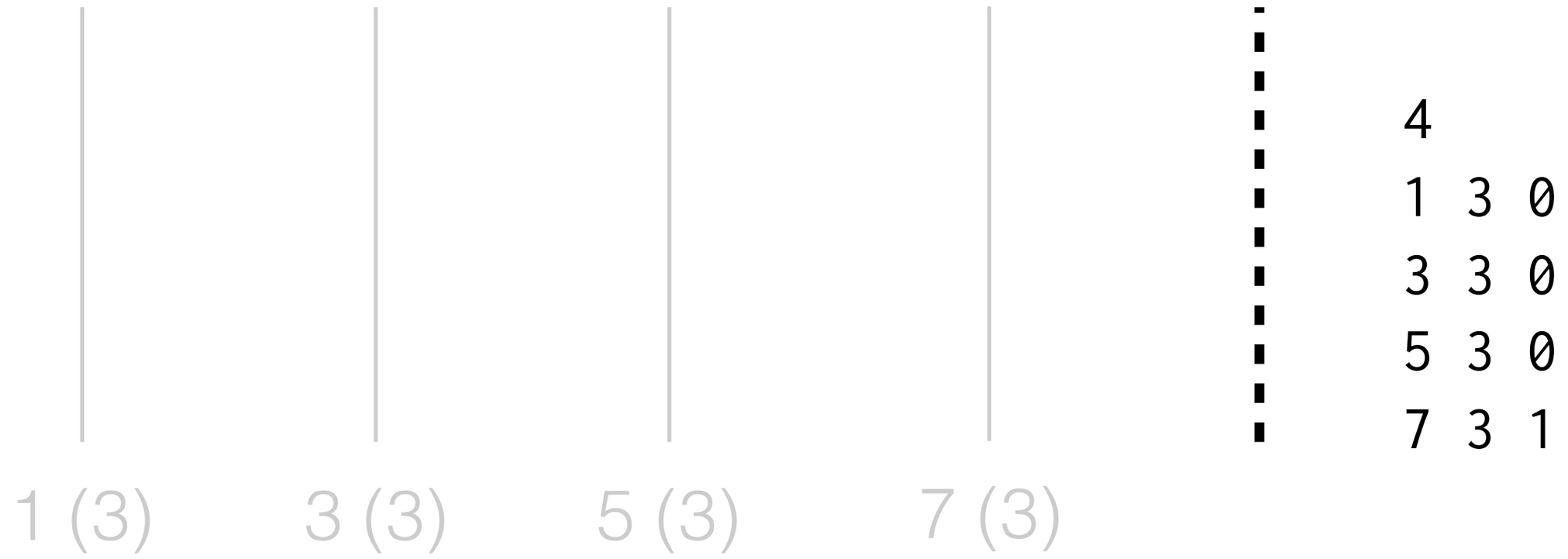


컴퓨터를 이용한 문제 해결 과정

2. 문제를 해결하는 알고리즘을 개발한다

관찰 1 : Elice씨가 “일을 마치는 시간”이 중요하다

관찰 2 : 일 사이에 우선순위가 있다



컴퓨터를 이용한 문제 해결 과정

2. 문제를 해결하는 알고리즘을 개발한다

관찰 1 : Elice씨가 “일을 마치는 시간”이 중요하다

관찰 2 : 일 사이에 우선순위가 있다

관찰 3 : “내가 일을 마치는 시간” 이전에도 일이 있을 수 있다

```
4
1 3 0
3 3 0
5 3 0
7 3 1
```

컴퓨터를 이용한 문제 해결 과정

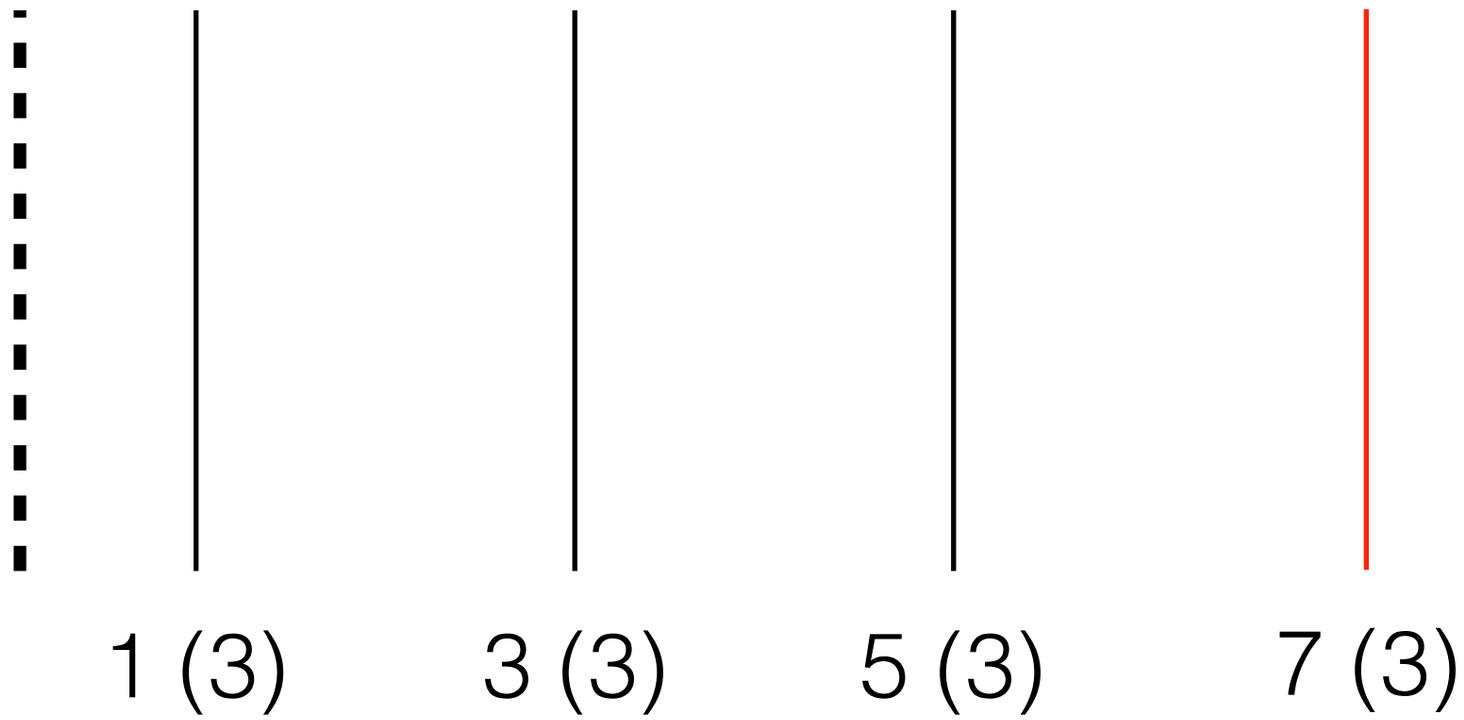
2. 문제를 해결하는 알고리즘을 개발한다

Elice씨가 다음으로 무슨 일을 처리해야 하는가 ?

컴퓨터를 이용한 문제 해결 과정

2. 문제를 해결하는 알고리즘을 개발한다

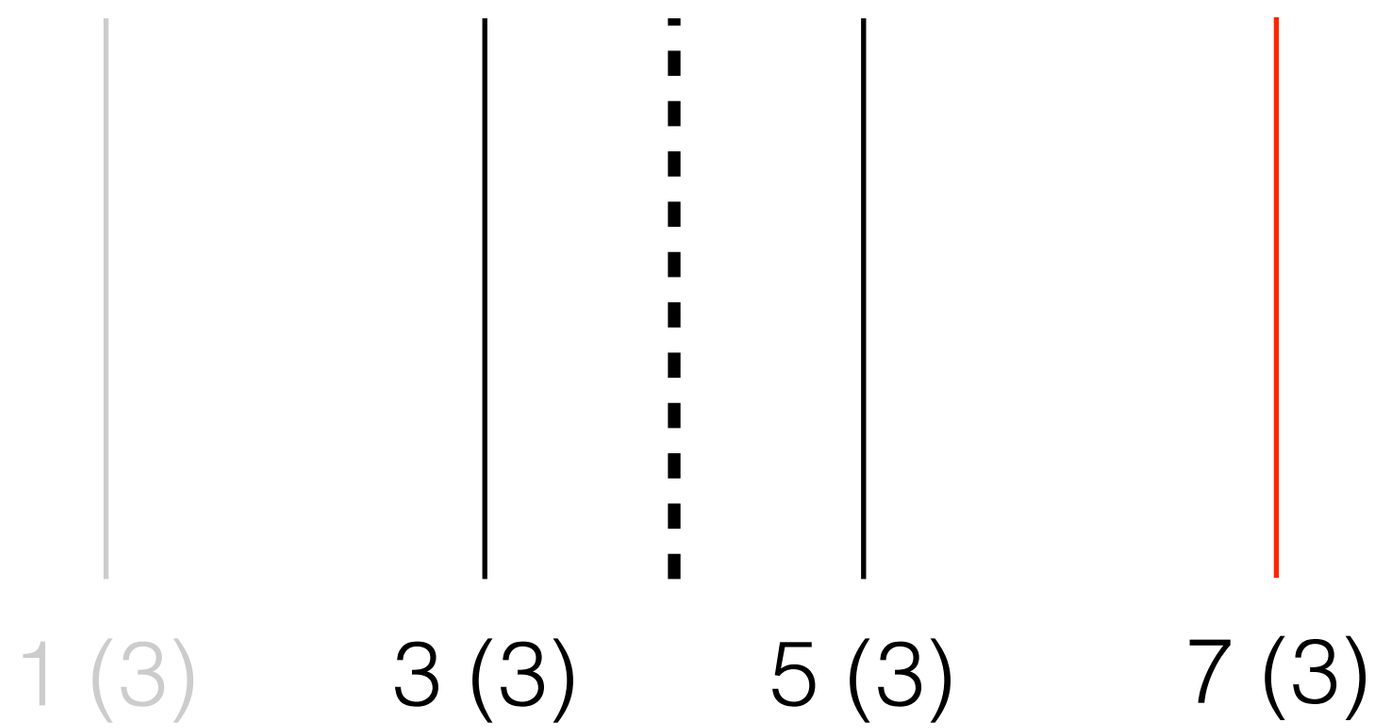
Elice씨가 다음으로 무슨 일을 처리해야 하는가 ?



컴퓨터를 이용한 문제 해결 과정

2. 문제를 해결하는 알고리즘을 개발한다

Elice씨가 다음으로 무슨 일을 처리해야 하는가 ?

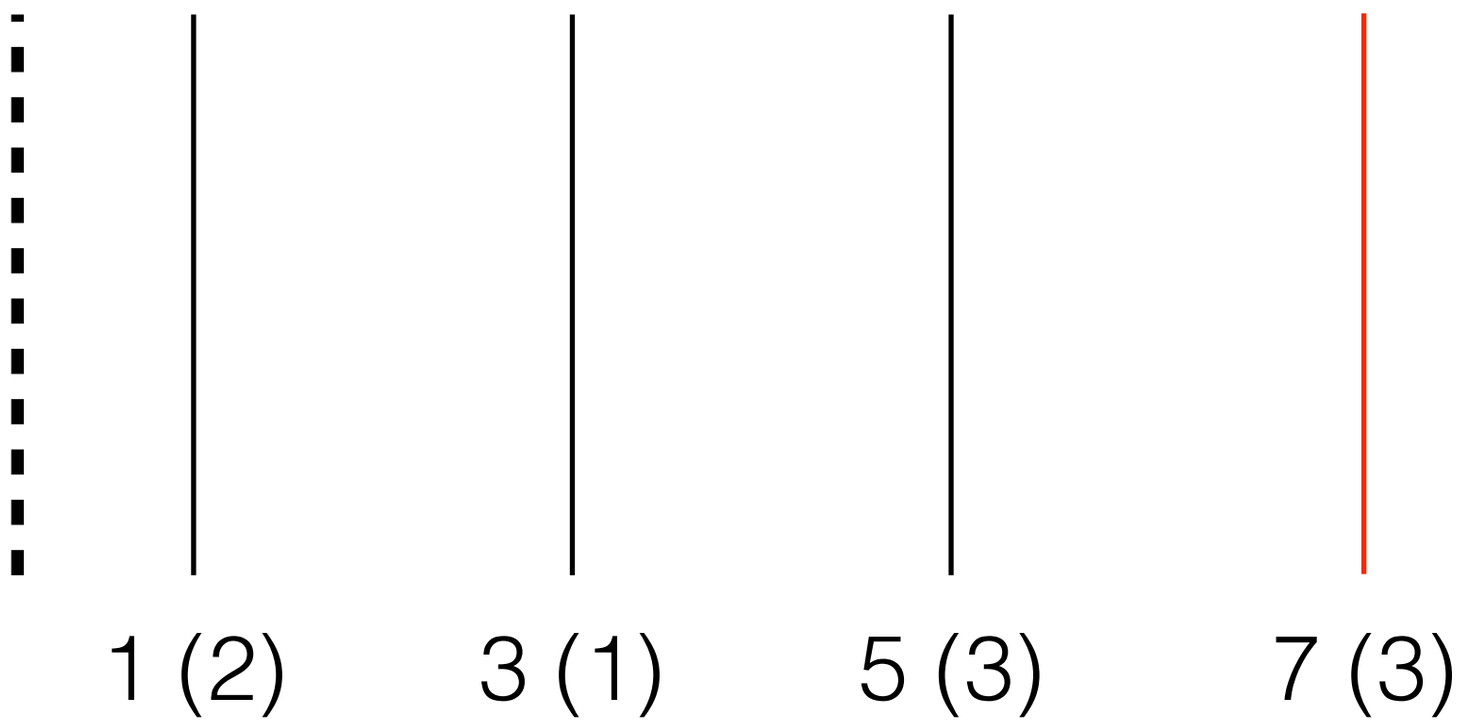


컴퓨터를 이용한 문제 해결 과정

2. 문제를 해결하는 알고리즘을 개발한다

두 가지 경우의 수

a. Elice씨가 “일을 마친 시간” 이전의 모든 일을 처리한 경우

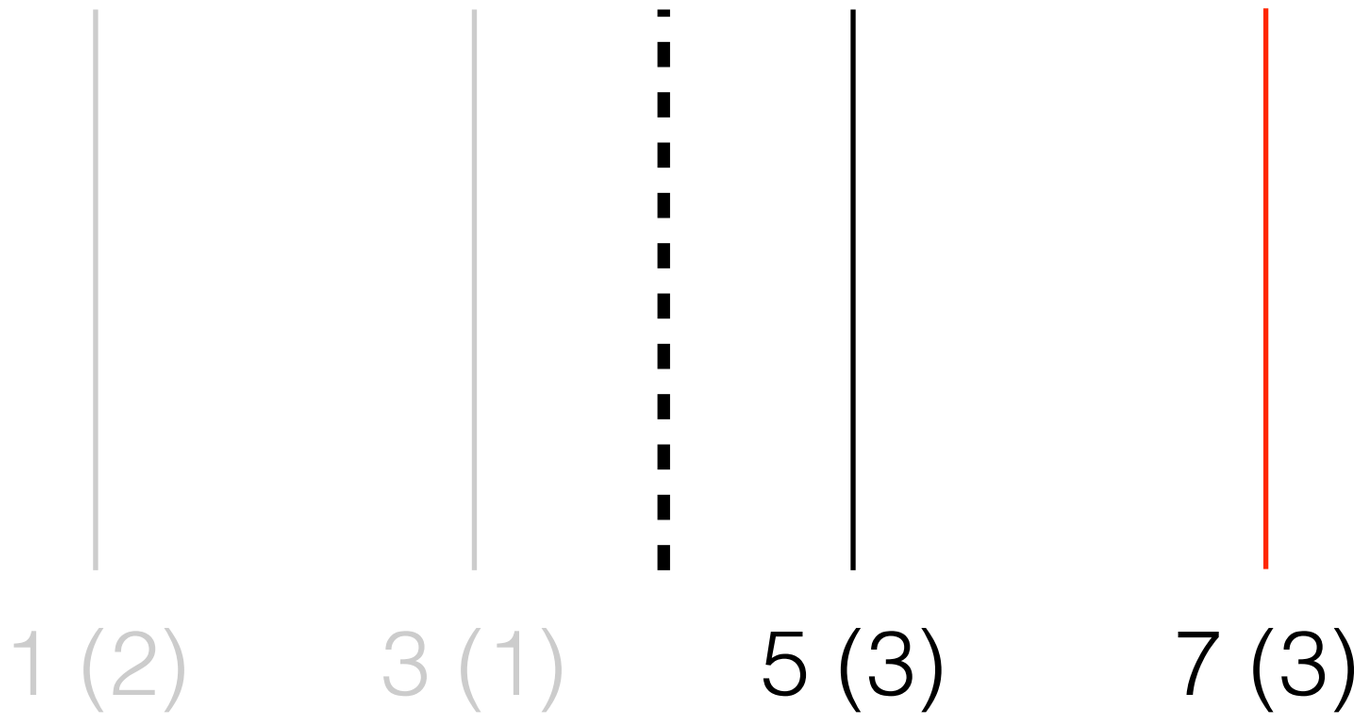


컴퓨터를 이용한 문제 해결 과정

2. 문제를 해결하는 알고리즘을 개발한다

두 가지 경우의 수

a. Elice씨가 “일을 마친 시간” 이전의 모든 일을 처리한 경우



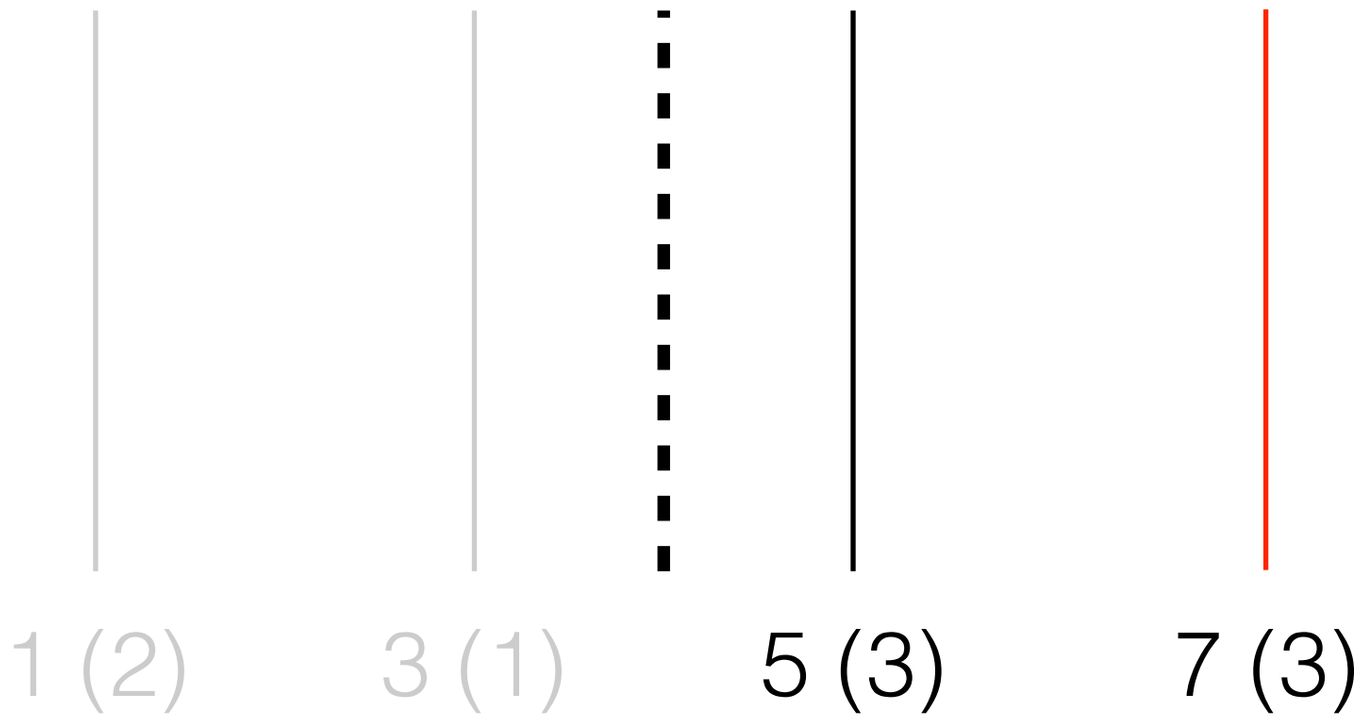
컴퓨터를 이용한 문제 해결 과정

2. 문제를 해결하는 알고리즘을 개발한다

두 가지 경우의 수

a. Elice씨가 “일을 마친 시간” 이전의 모든 일을 처리한 경우

→ 먼저 들어오는 주문을 처리

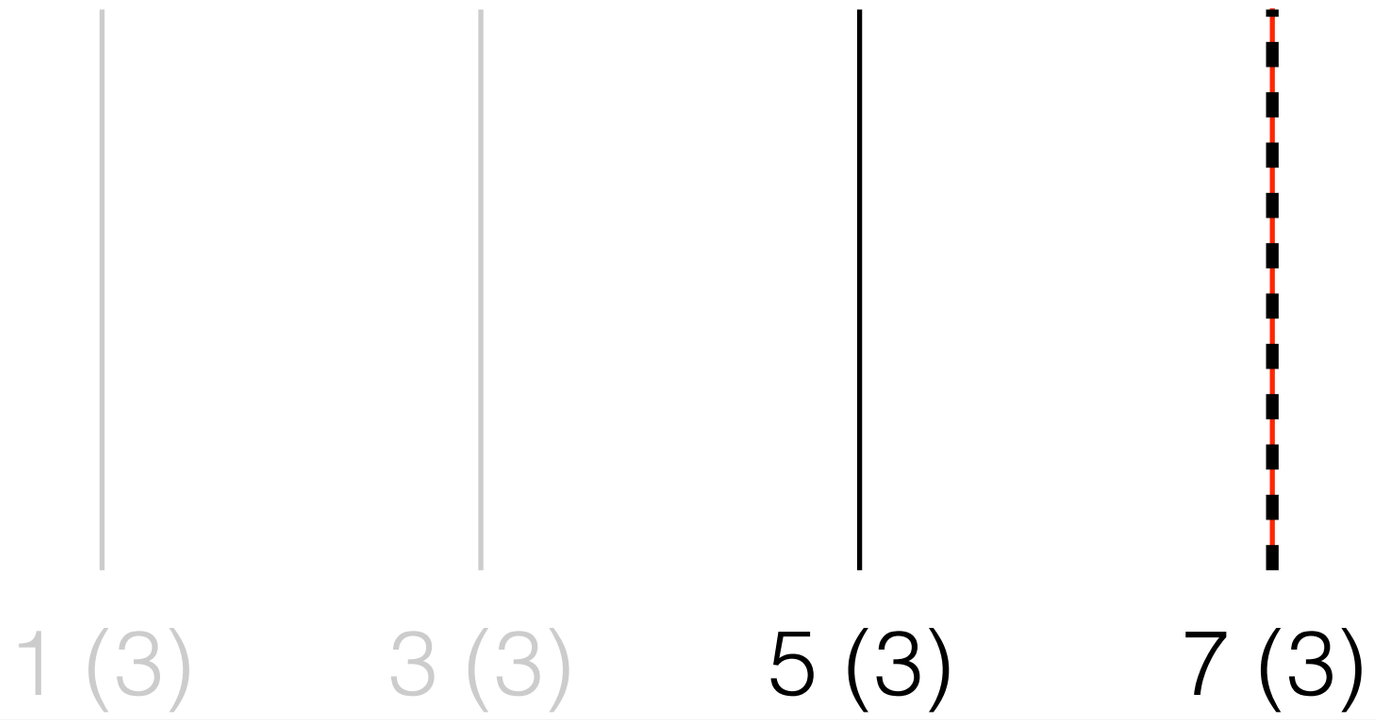


컴퓨터를 이용한 문제 해결 과정

2. 문제를 해결하는 알고리즘을 개발한다

두 가지 경우의 수

b. Elice씨가 “일을 마친 시간” 이전의 모든 일을 처리 못한 경우



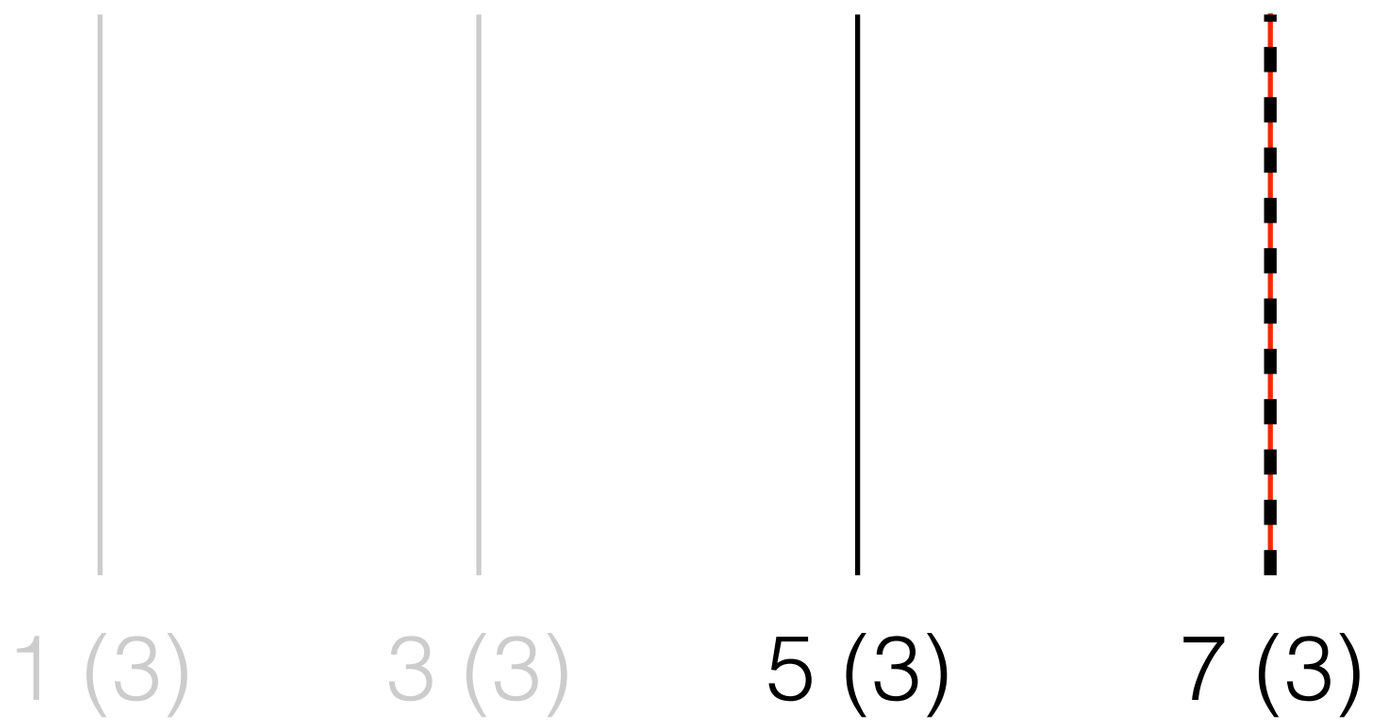
컴퓨터를 이용한 문제 해결 과정

2. 문제를 해결하는 알고리즘을 개발한다

두 가지 경우의 수

b. Elice씨가 “일을 마친 시간” 이전의 모든 일을 처리 못한 경우

→ 우선순위가 높은 일을 처리



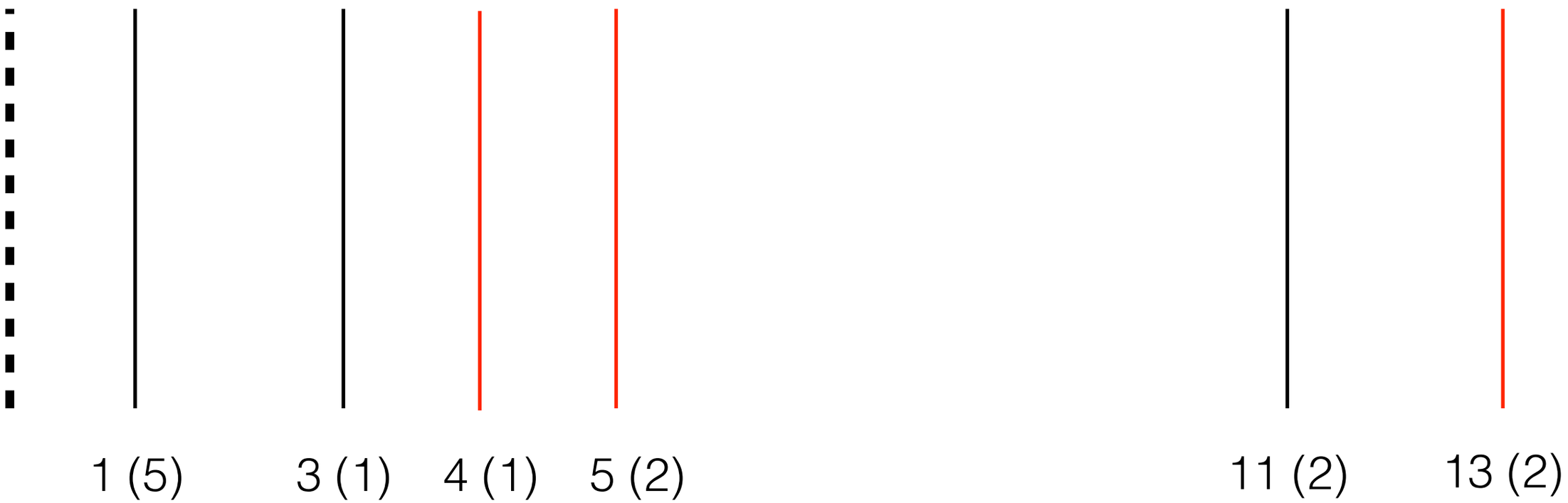
컴퓨터를 이용한 문제 해결 과정

2. 문제를 해결하는 알고리즘을 개발한다

두 가지 경우의 수

a. Elice씨가 “일을 마친 시간” 이전의 모든 일을 처리한 경우

b. Elice씨가 “일을 마친 시간” 이전의 모든 일을 처리 못한 경우



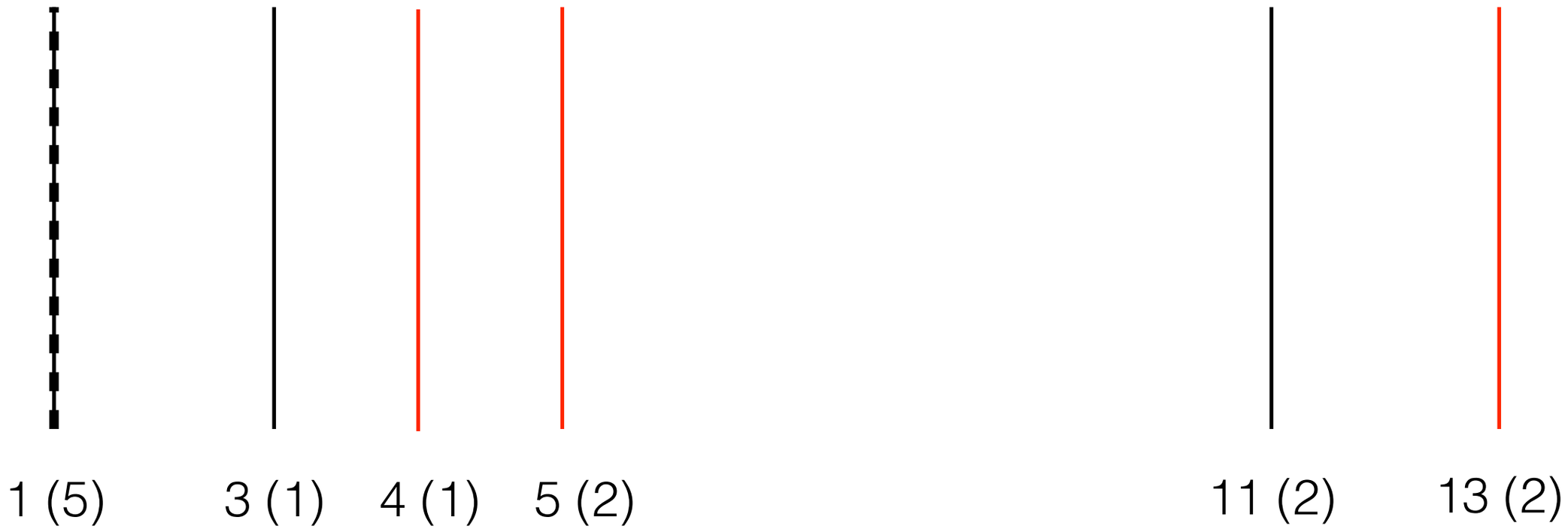
컴퓨터를 이용한 문제 해결 과정

2. 문제를 해결하는 알고리즘을 개발한다

두 가지 경우의 수

a. Elice씨가 “일을 마친 시간” 이전의 모든 일을 처리한 경우

b. Elice씨가 “일을 마친 시간” 이전의 모든 일을 처리 못한 경우



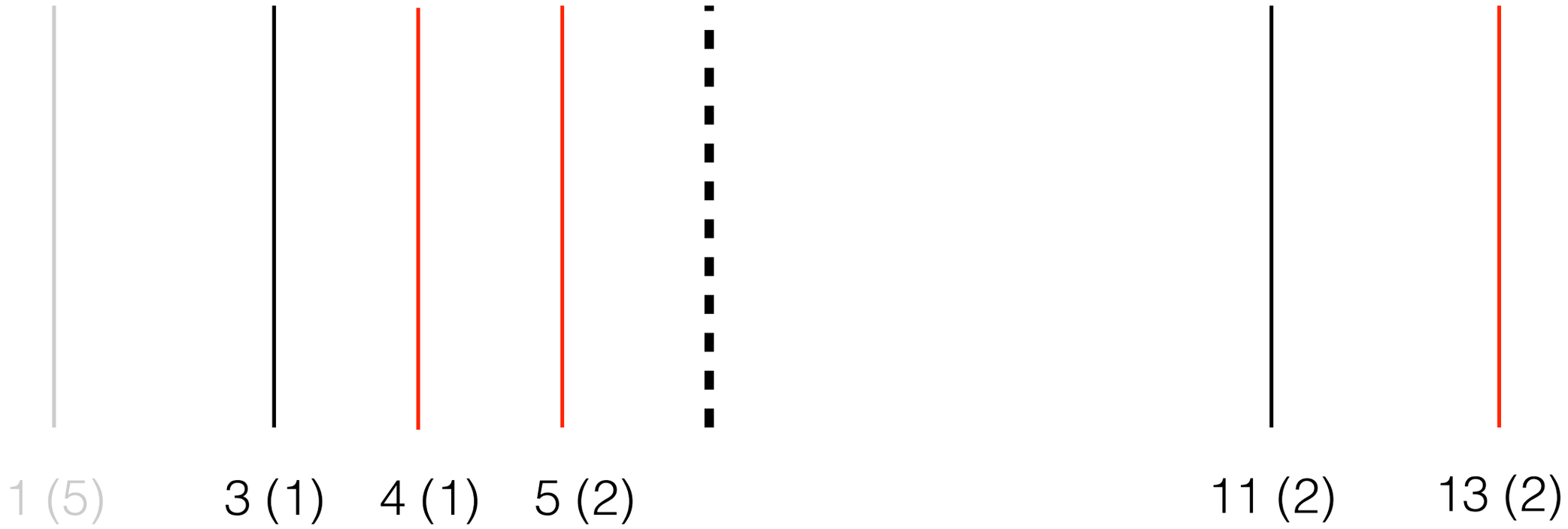
컴퓨터를 이용한 문제 해결 과정

2. 문제를 해결하는 알고리즘을 개발한다

두 가지 경우의 수

a. Elice씨가 “일을 마친 시간” 이전의 모든 일을 처리한 경우

b. Elice씨가 “일을 마친 시간” 이전의 모든 일을 처리 못한 경우



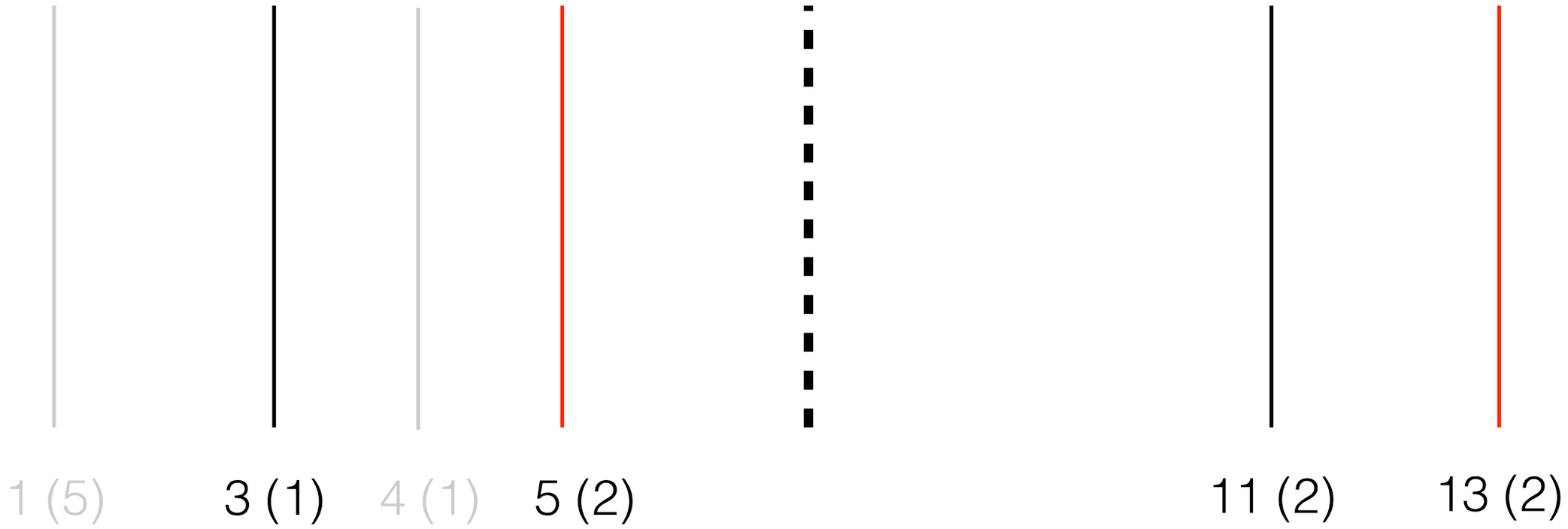
컴퓨터를 이용한 문제 해결 과정

2. 문제를 해결하는 알고리즘을 개발한다

두 가지 경우의 수

a. Elice씨가 “일을 마친 시간” 이전의 모든 일을 처리한 경우

b. Elice씨가 “일을 마친 시간” 이전의 모든 일을 처리 못한 경우



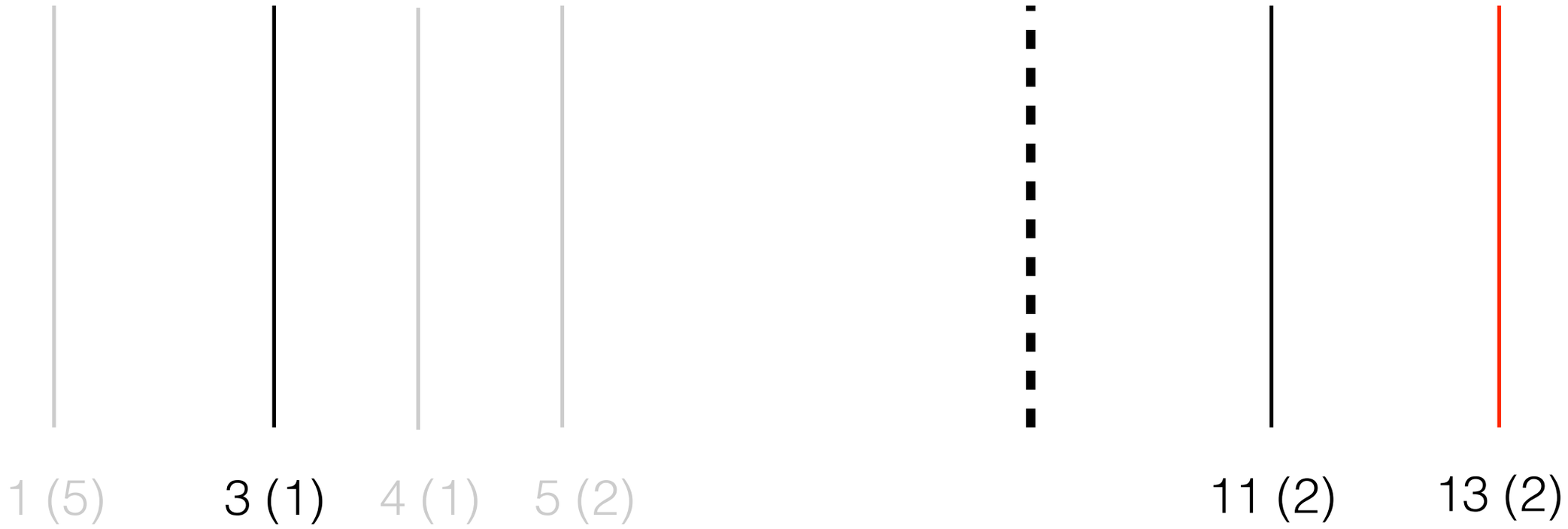
컴퓨터를 이용한 문제 해결 과정

2. 문제를 해결하는 알고리즘을 개발한다

두 가지 경우의 수

a. Elice씨가 “일을 마친 시간” 이전의 모든 일을 처리한 경우

b. Elice씨가 “일을 마친 시간” 이전의 모든 일을 처리 못한 경우



컴퓨터를 이용한 문제 해결 과정

2. 문제를 해결하는 알고리즘을 개발한다

두 가지 경우의 수

a. Elice씨가 “일을 마친 시간” 이전의 모든 일을 처리한 경우

b. Elice씨가 “일을 마친 시간” 이전의 모든 일을 처리 못한 경우

컴퓨터를 이용한 문제 해결 과정

2. 문제를 해결하는 알고리즘을 개발한다

두 가지 경우의 수

a. Elice씨가 “일을 마친 시간” 이전의 모든 일을 처리한 경우

b. Elice씨가 “일을 마친 시간” 이전의 모든 일을 처리 못한 경우

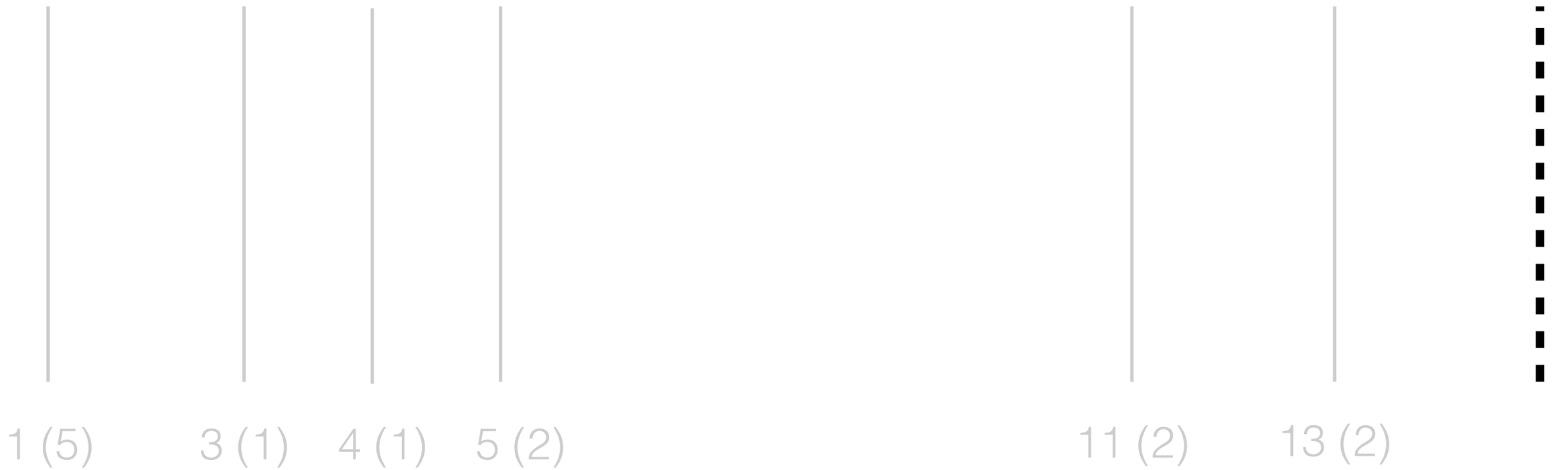
컴퓨터를 이용한 문제 해결 과정

2. 문제를 해결하는 알고리즘을 개발한다

두 가지 경우의 수

a. Elice씨가 “일을 마친 시간” 이전의 모든 일을 처리한 경우

b. Elice씨가 “일을 마친 시간” 이전의 모든 일을 처리 못한 경우



컴퓨터를 이용한 문제 해결 과정

3. 알고리즘이 **문제를 해결한다는 것을 증명한다**

두 가지 경우의 수

a. Elice씨가 “일을 마친 시간” 이전의 모든 일을 처리한 경우

b. Elice씨가 “일을 마친 시간” 이전의 모든 일을 처리 못한 경우

컴퓨터를 이용한 문제 해결 과정

3. 알고리즘이 **문제를 해결한다는 것을 증명한다**

두 가지 경우의 수

a. Elice씨가 “일을 마친 시간” 이전의 모든 일을 처리한 경우

b. Elice씨가 “일을 마친 시간” 이전의 모든 일을 처리 못한 경우

1. 위의 두 가지 경우가 모든 경우이다

2. 각각의 경우를 올바로 처리하였다

컴퓨터를 이용한 문제 해결 과정

3. 알고리즘이 **문제를 해결한다는 것을 증명한다**

두 가지 경우의 수

a. Elice씨가 “일을 마친 시간” 이전의 모든 일을 처리한 경우

b. Elice씨가 “일을 마친 시간” 이전의 모든 일을 처리 못한 경우

1. 위의 두 가지 경우가 모든 경우이다 (Y)

2. 각각의 경우를 올바로 처리하였다 (Y)

컴퓨터를 이용한 문제 해결 과정

4. 알고리즘이 **제한시간 내에 동작한다는 것을 보인다**

이 문제에서는 중요하지 않기때문에 넘어갑니다

컴퓨터를 이용한 문제 해결 과정

5. 알고리즘을 코드로 작성한다

컴퓨터를 이용한 문제 해결 과정

5. 알고리즘을 코드로 작성한다

목적이 먼저, 자료구조는 그 다음!

컴퓨터를 이용한 문제 해결 과정

5. 알고리즘을 코드로 작성한다

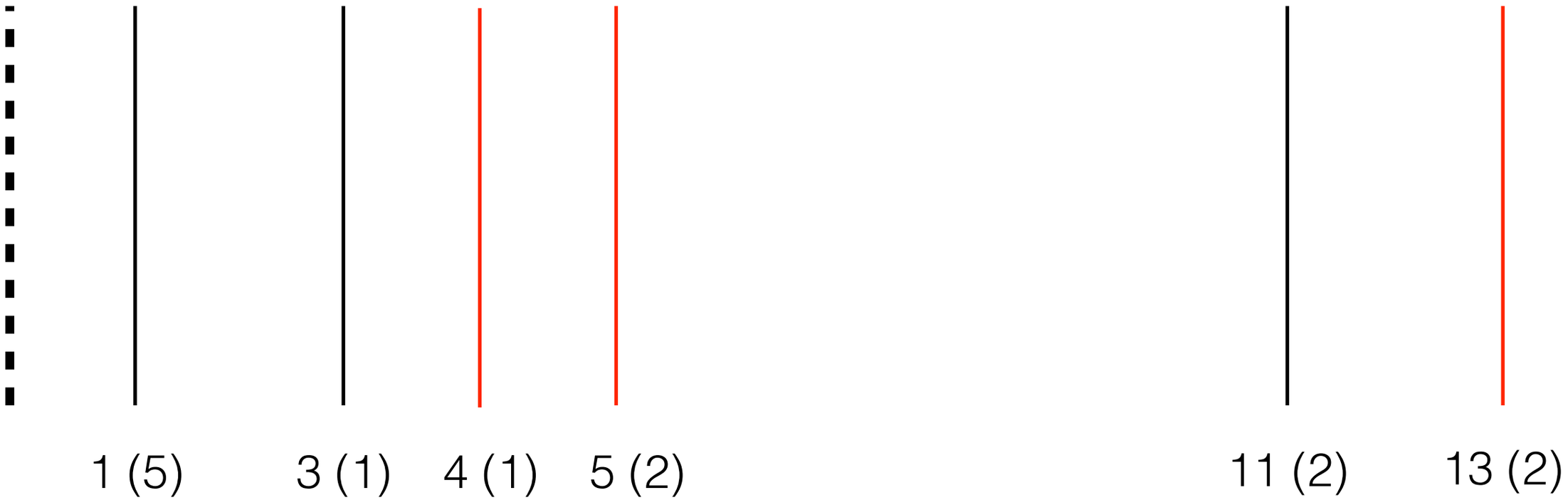
목적이 먼저, 자료구조는 그 다음!

우리의 코드가 무슨 일을 해야하는가 ?

컴퓨터를 이용한 문제 해결 과정

5. 알고리즘을 코드로 작성한다

우리의 코드가 무슨 일을 해야하는가 ?



컴퓨터를 이용한 문제 해결 과정

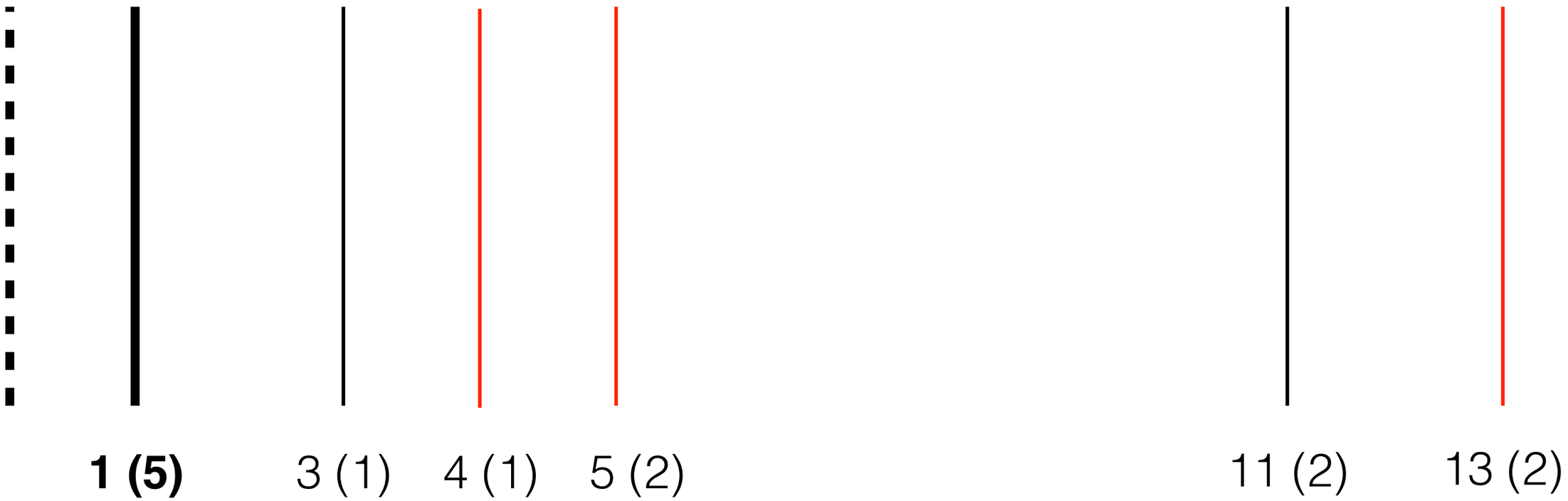
5. 알고리즘을 코드로 작성한다

우리의 코드가 무슨 일을 해야하는가 ?

컴퓨터를 이용한 문제 해결 과정

5. 알고리즘을 코드로 작성한다

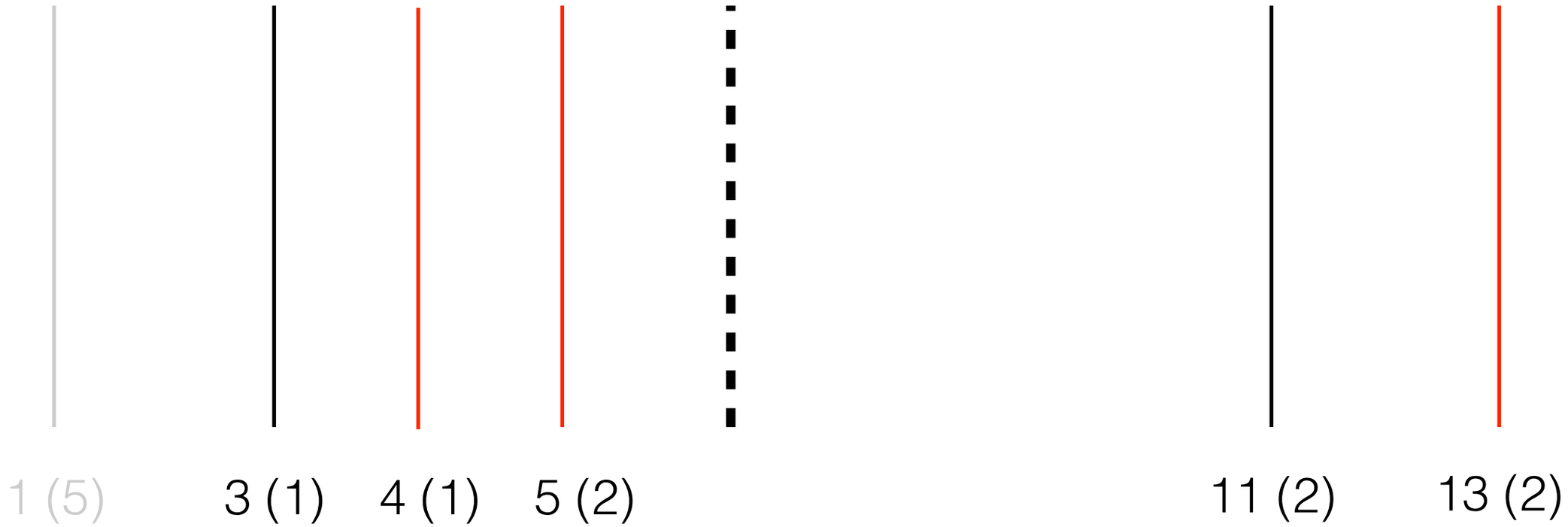
우리의 코드가 무슨 일을 해야하는가 ?



컴퓨터를 이용한 문제 해결 과정

5. 알고리즘을 코드로 작성한다

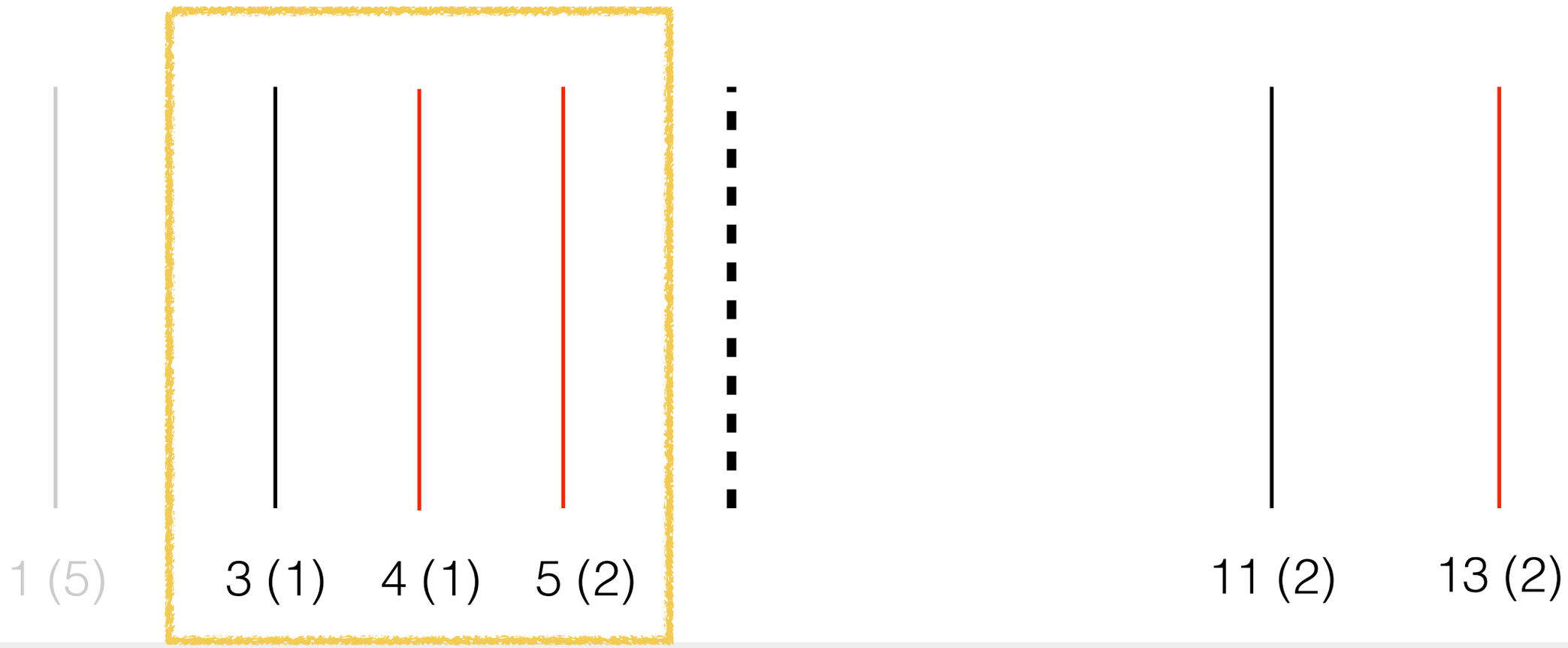
우리의 코드가 무슨 일을 해야하는가 ?



컴퓨터를 이용한 문제 해결 과정

5. 알고리즘을 코드로 작성한다

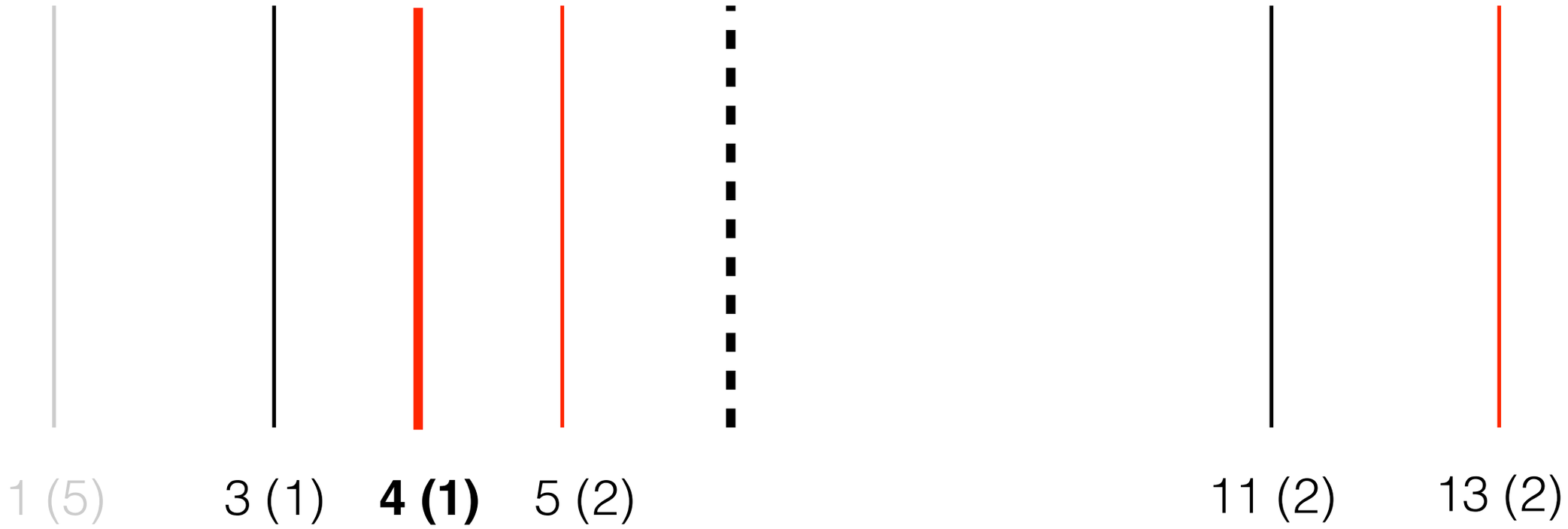
우리의 코드가 무슨 일을 해야하는가 ?



컴퓨터를 이용한 문제 해결 과정

5. 알고리즘을 코드로 작성한다

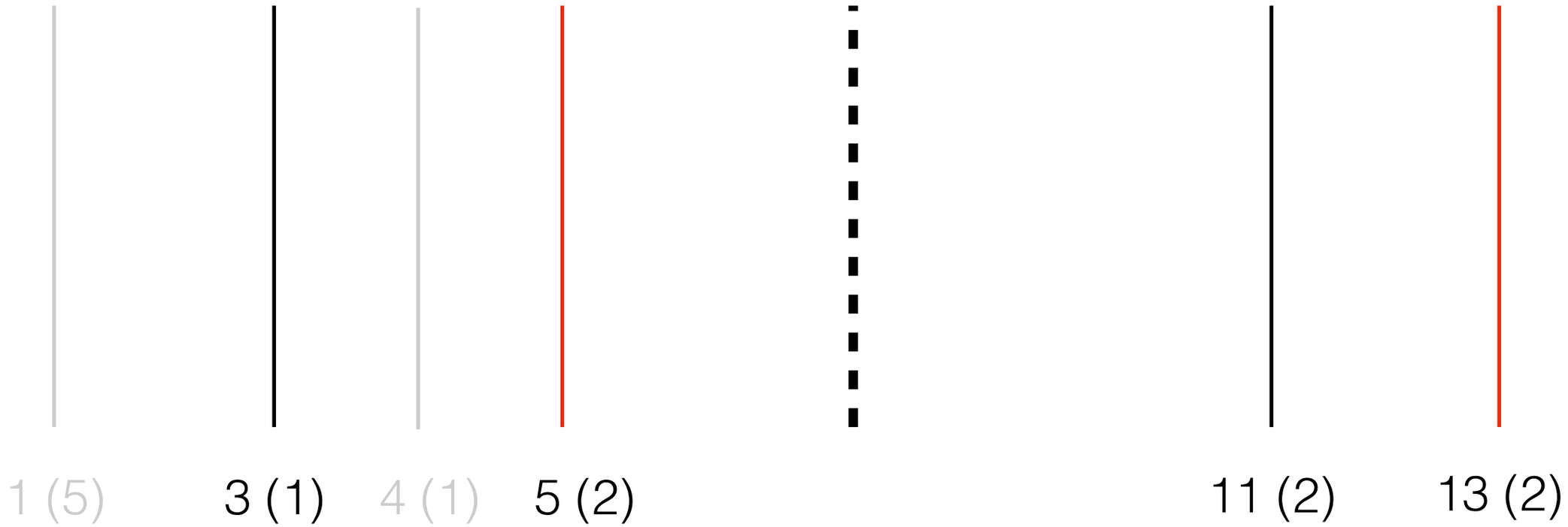
우리의 코드가 무슨 일을 해야하는가 ?



컴퓨터를 이용한 문제 해결 과정

5. 알고리즘을 코드로 작성한다

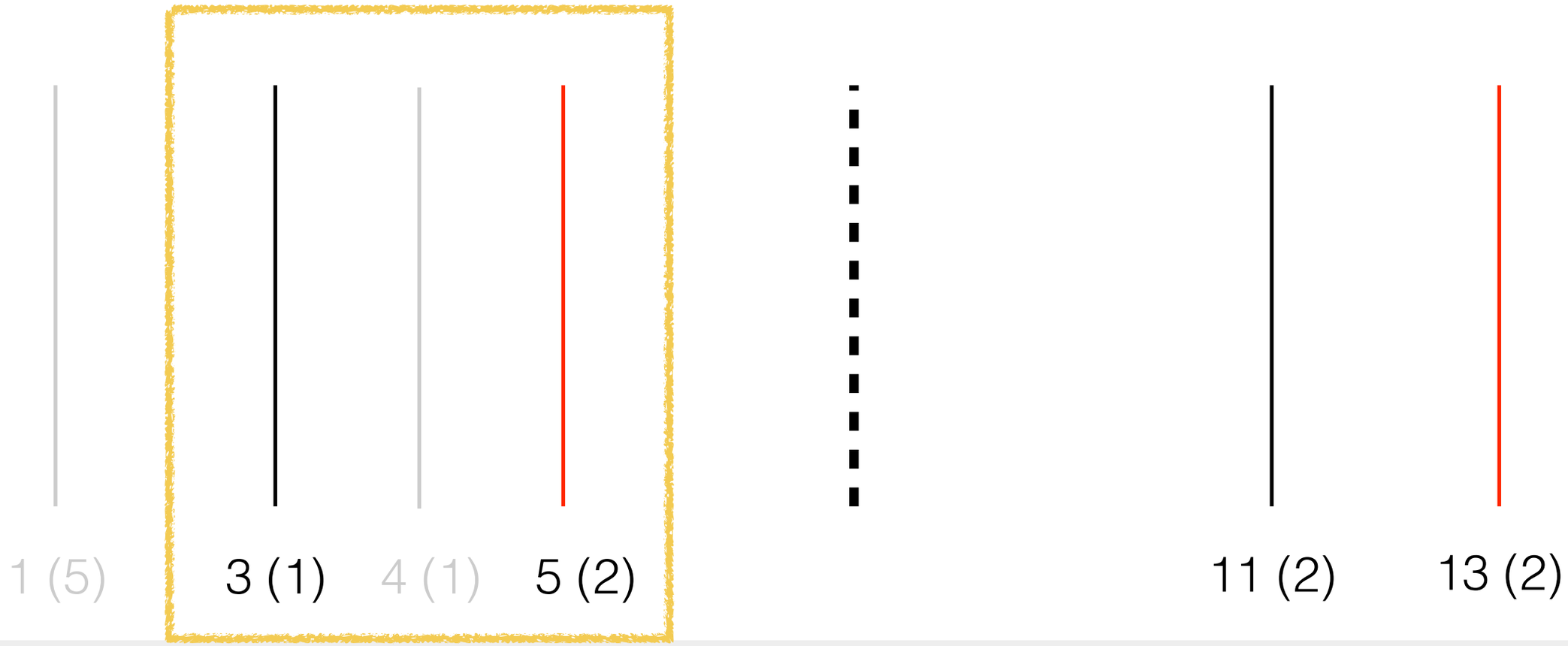
우리의 코드가 무슨 일을 해야하는가 ?



컴퓨터를 이용한 문제 해결 과정

5. 알고리즘을 코드로 작성한다

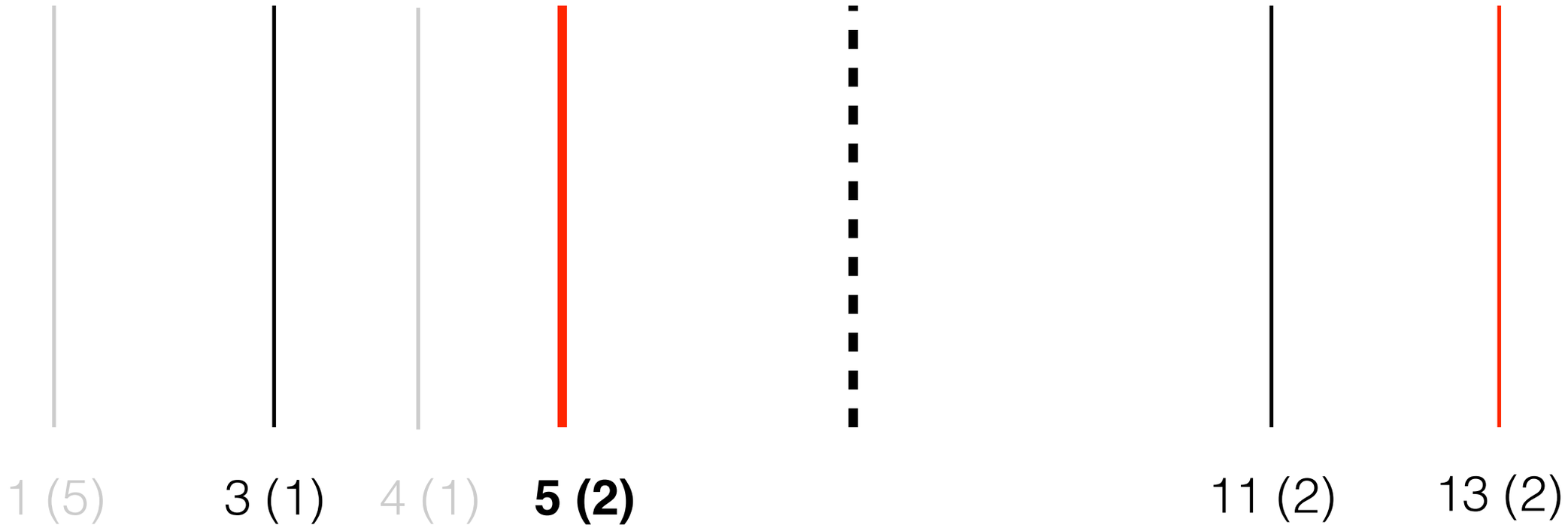
우리의 코드가 무슨 일을 해야하는가 ?



컴퓨터를 이용한 문제 해결 과정

5. 알고리즘을 코드로 작성한다

우리의 코드가 무슨 일을 해야하는가 ?



컴퓨터를 이용한 문제 해결 과정

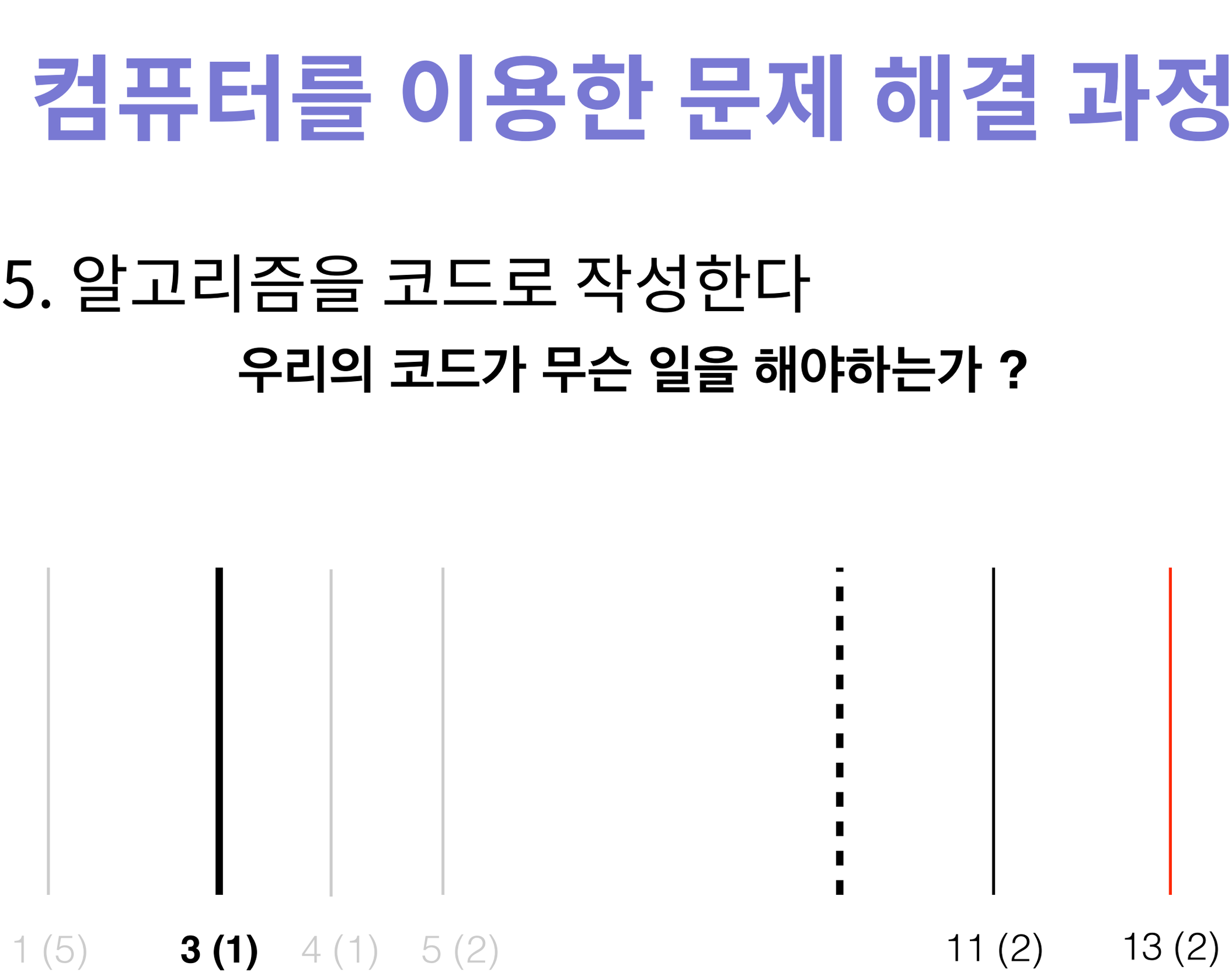
5. 알고리즘을 코드로 작성한다

우리의 코드가 무슨 일을 해야하는가 ?

컴퓨터를 이용한 문제 해결 과정

5. 알고리즘을 코드로 작성한다

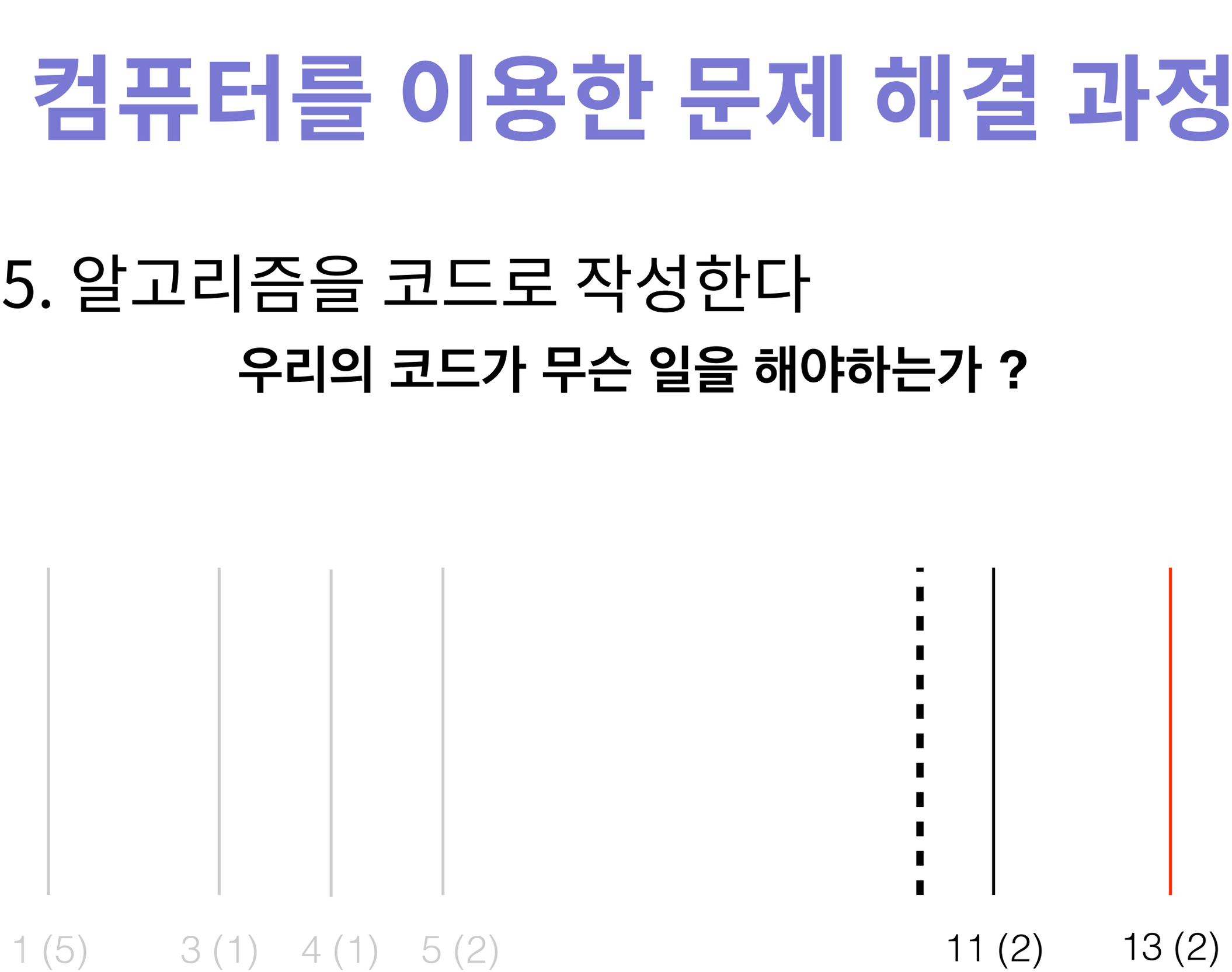
우리의 코드가 무슨 일을 해야하는가 ?



컴퓨터를 이용한 문제 해결 과정

5. 알고리즘을 코드로 작성한다

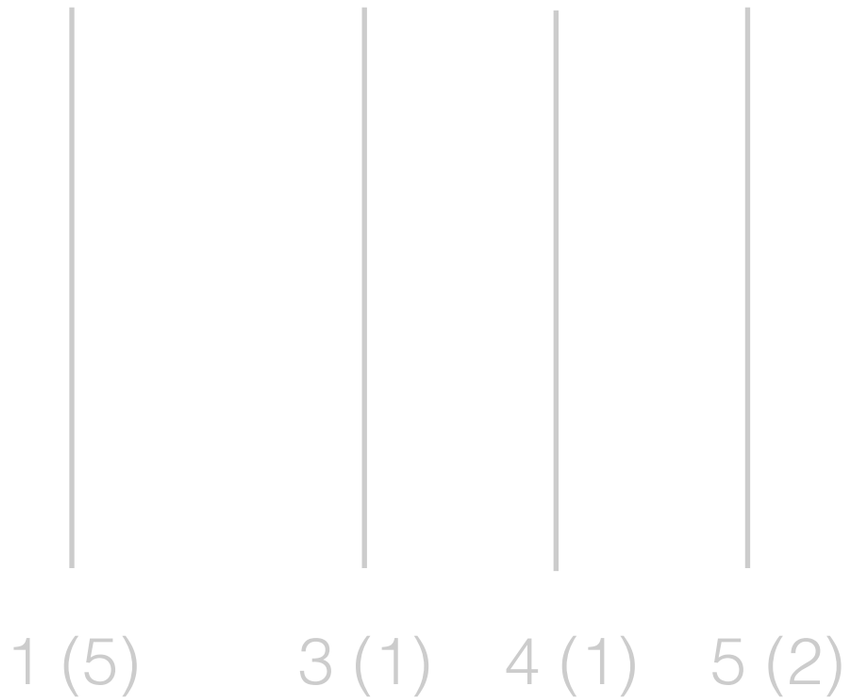
우리의 코드가 무슨 일을 해야하는가 ?

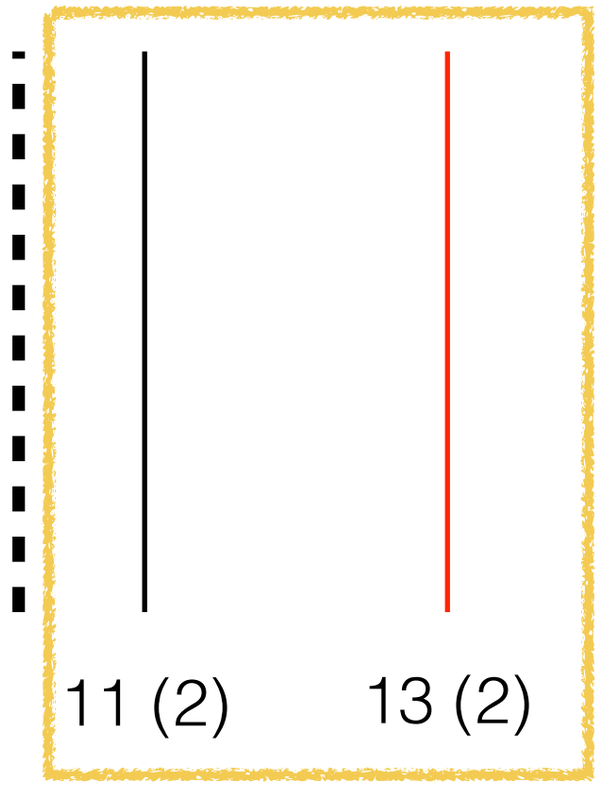


컴퓨터를 이용한 문제 해결 과정

5. 알고리즘을 코드로 작성한다

우리의 코드가 무슨 일을 해야하는가 ?

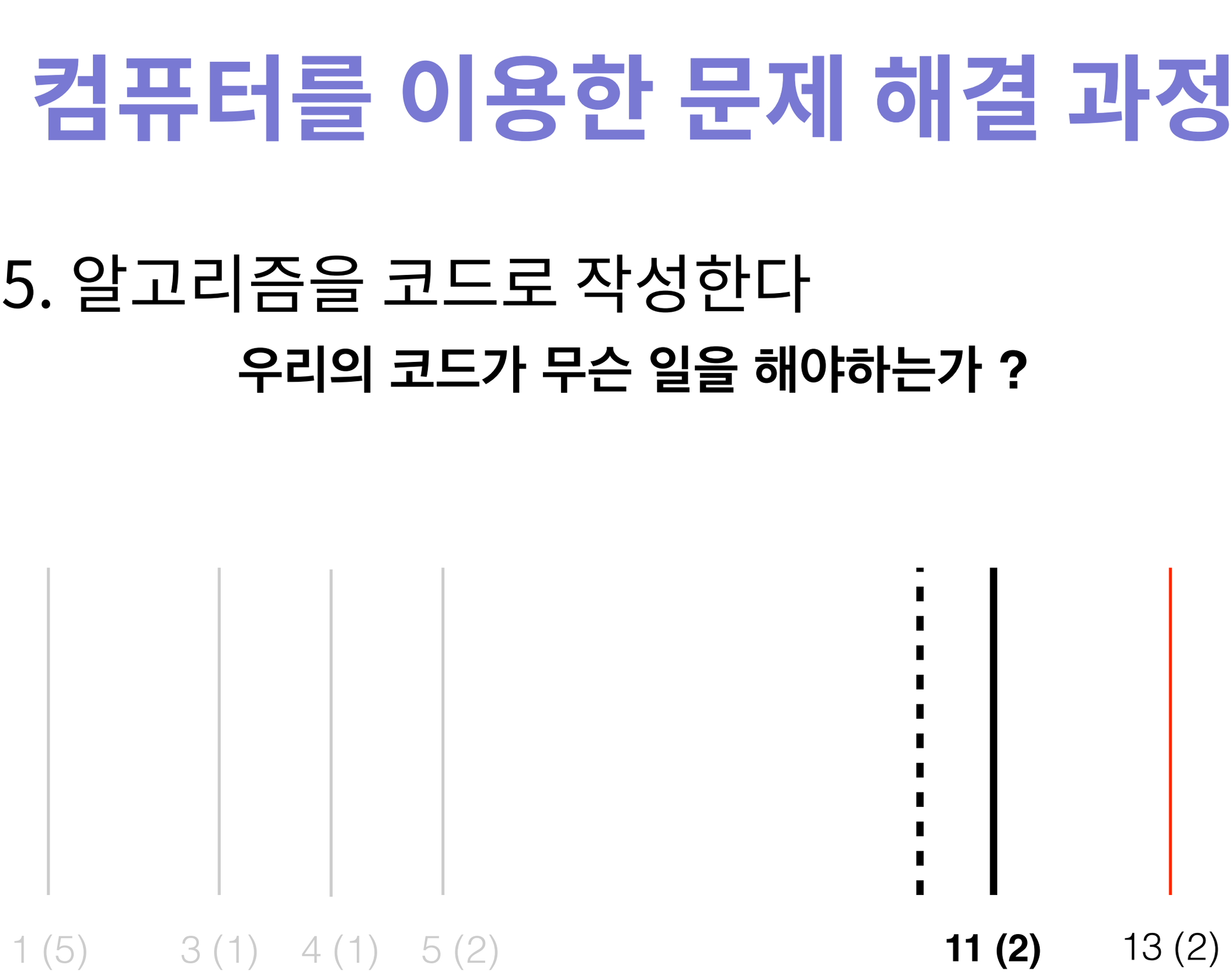




컴퓨터를 이용한 문제 해결 과정

5. 알고리즘을 코드로 작성한다

우리의 코드가 무슨 일을 해야하는가 ?



컴퓨터를 이용한 문제 해결 과정

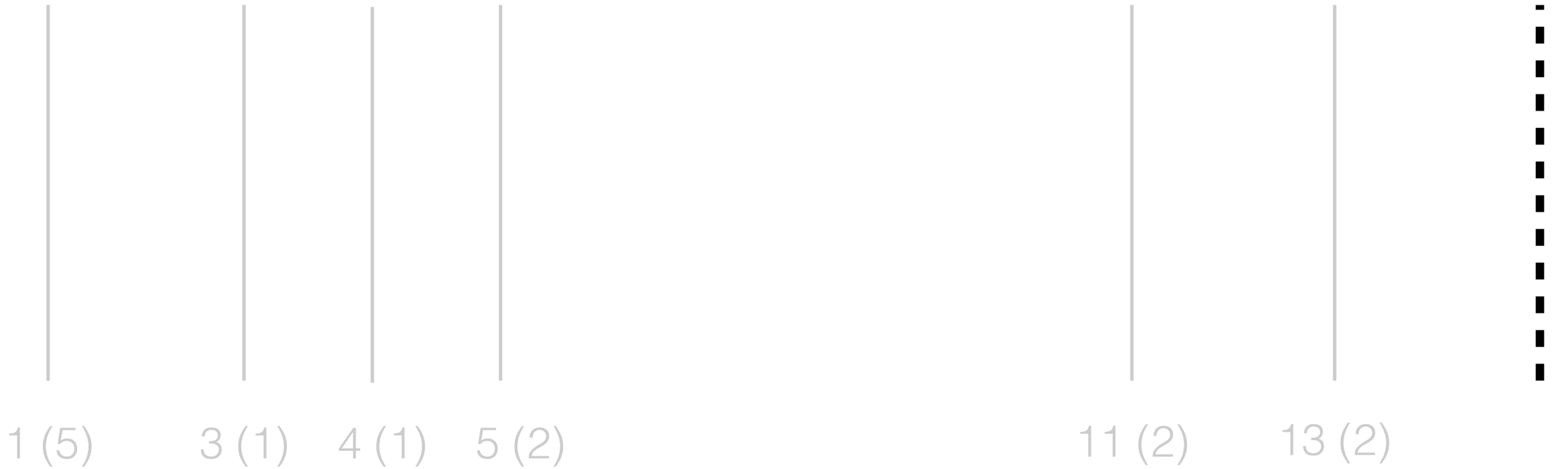
5. 알고리즘을 코드로 작성한다

우리의 코드가 무슨 일을 해야하는가 ?

컴퓨터를 이용한 문제 해결 과정

5. 알고리즘을 코드로 작성한다

우리의 코드가 무슨 일을 해야하는가 ?



컴퓨터를 이용한 문제 해결 과정

5. 알고리즘을 코드로 작성한다

우리의 코드가 무슨 일을 해야하는가 ?

컴퓨터를 이용한 문제 해결 과정

5. 알고리즘을 코드로 작성한다

우리의 코드가 무슨 일을 해야하는가 ?

a. 고려해야 하는 일의 후보를 정한다

컴퓨터를 이용한 문제 해결 과정

5. 알고리즘을 코드로 작성한다

우리의 코드가 무슨 일을 해야하는가 ?

a. 고려해야 하는 일의 후보를 정한다

b. 일의 후보 중에서 우선순위가 가장 높은 일을 찾는다

컴퓨터를 이용한 문제 해결 과정

5. 알고리즘을 코드로 작성한다

우리의 코드가 무슨 일을 해야하는가 ?

a. 고려해야 하는 일의 후보를 정한다
b. 일의 후보 중에서 우선순위가 가장 높은 일을 찾는다
c. 그 일을 처리하고 다시 a. 로 돌아간다

컴퓨터를 이용한 문제 해결 과정

5. 알고리즘을 코드로 작성한다

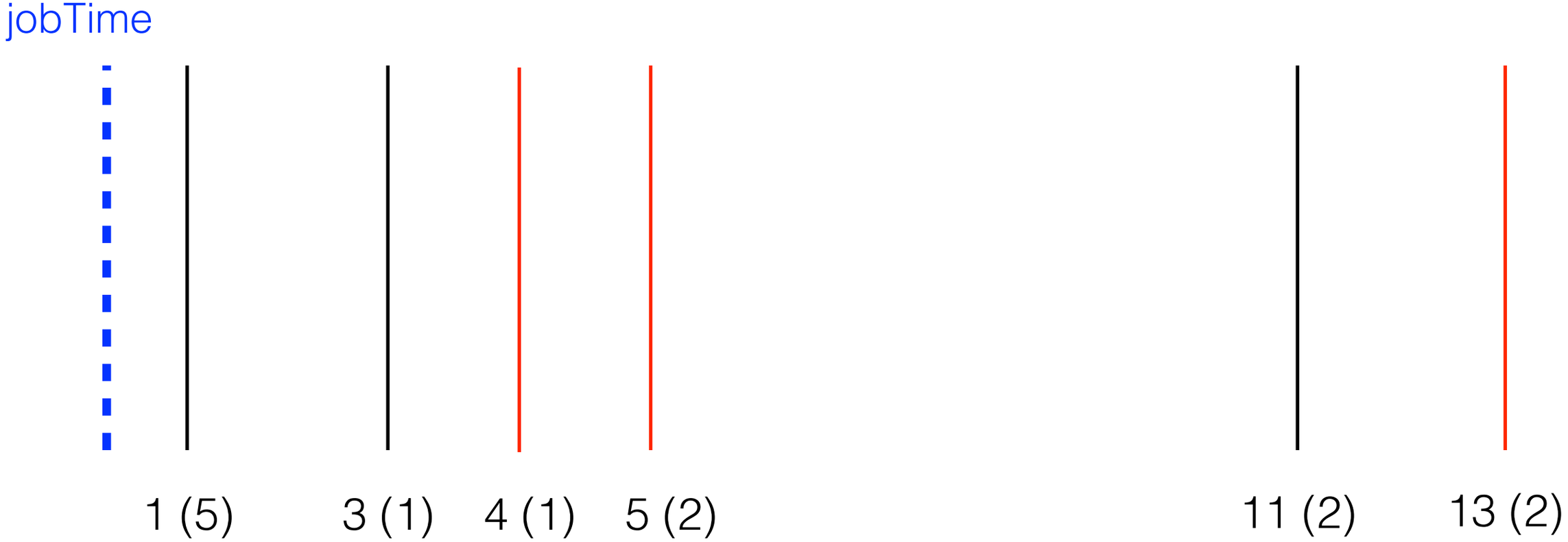
우리의 코드가 무슨 일을 해야하는가 ?

```
while (we still have a job) :
    candidates = findCandidates(orders)
    to_do = findTopPriority(candidates)
    processJob(to_do)
```

컴퓨터를 이용한 문제 해결 과정

5. 알고리즘을 코드로 작성한다

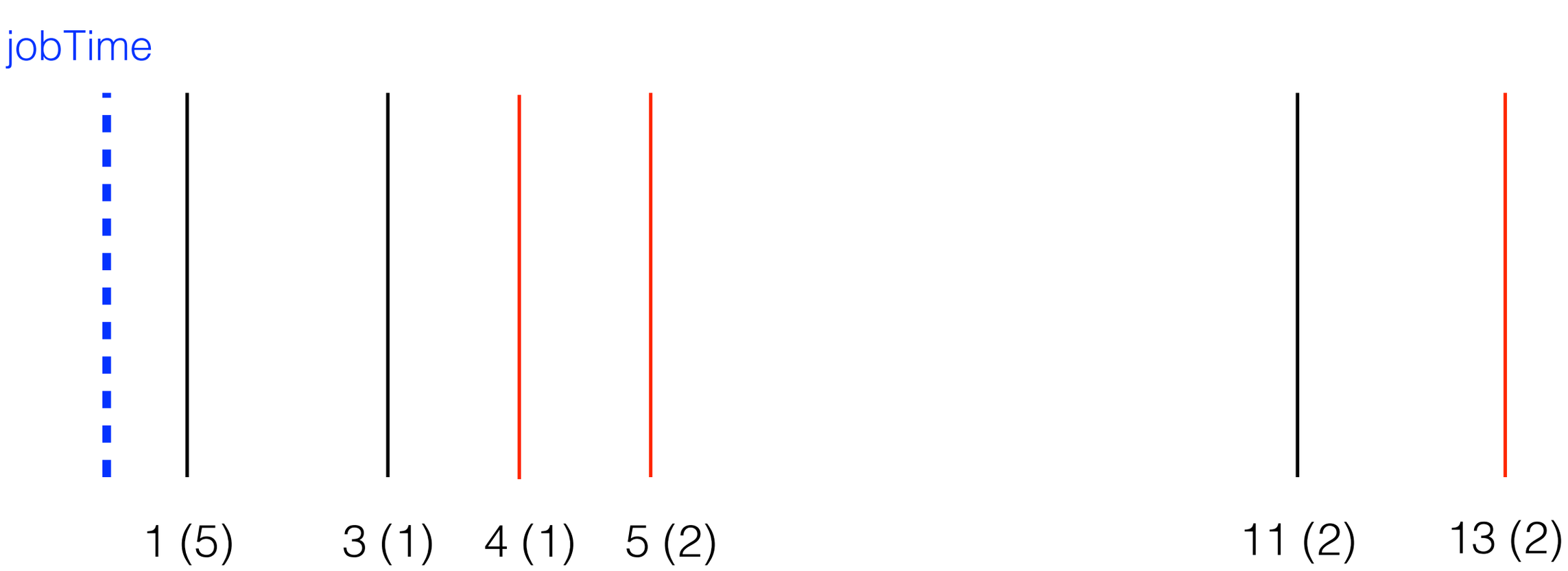
통찰 : 일반 주문, VIP 주문은 반드시 순서대로 처리되어야 한다



컴퓨터를 이용한 문제 해결 과정

5. 알고리즘을 코드로 작성한다

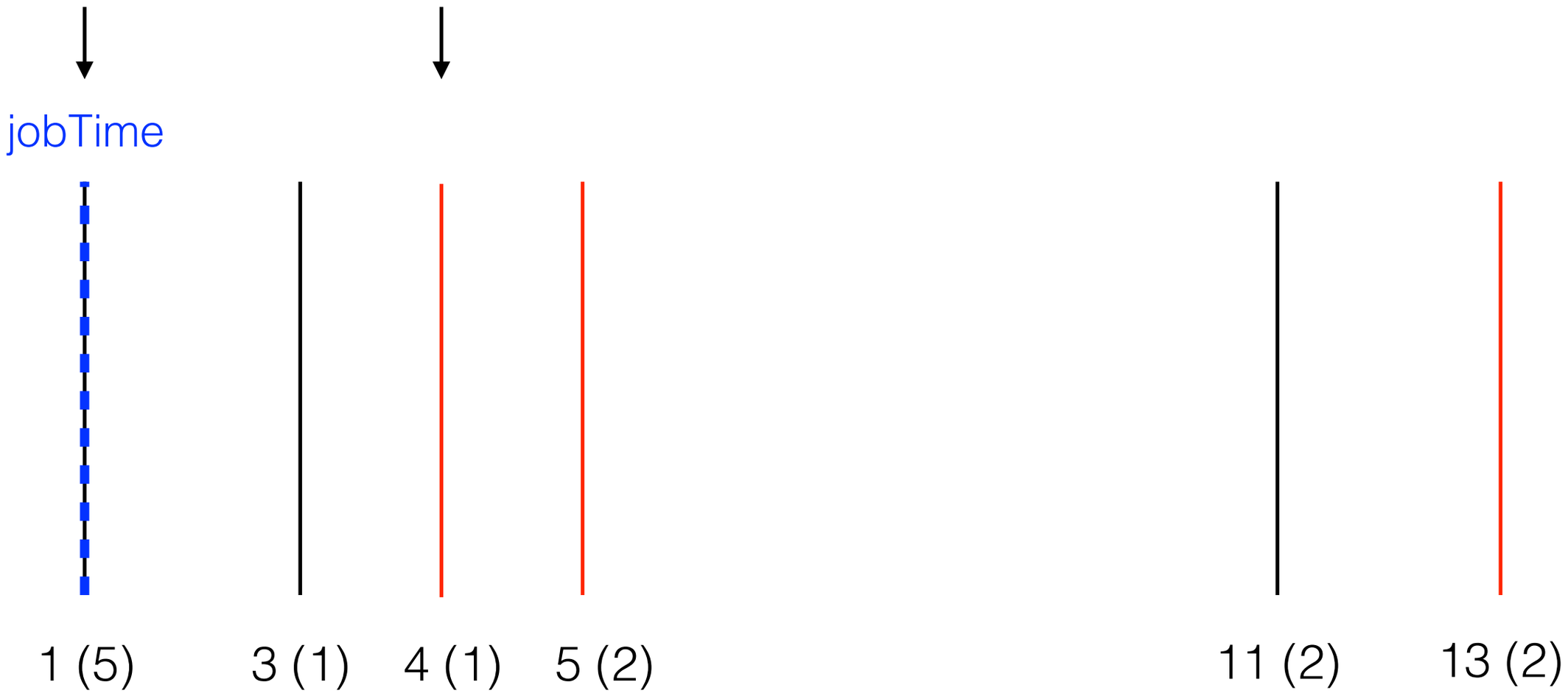
통찰 : 일반 주문, VIP 주문은 반드시 순서대로 처리되어야 한다



컴퓨터를 이용한 문제 해결 과정

5. 알고리즘을 코드로 작성한다

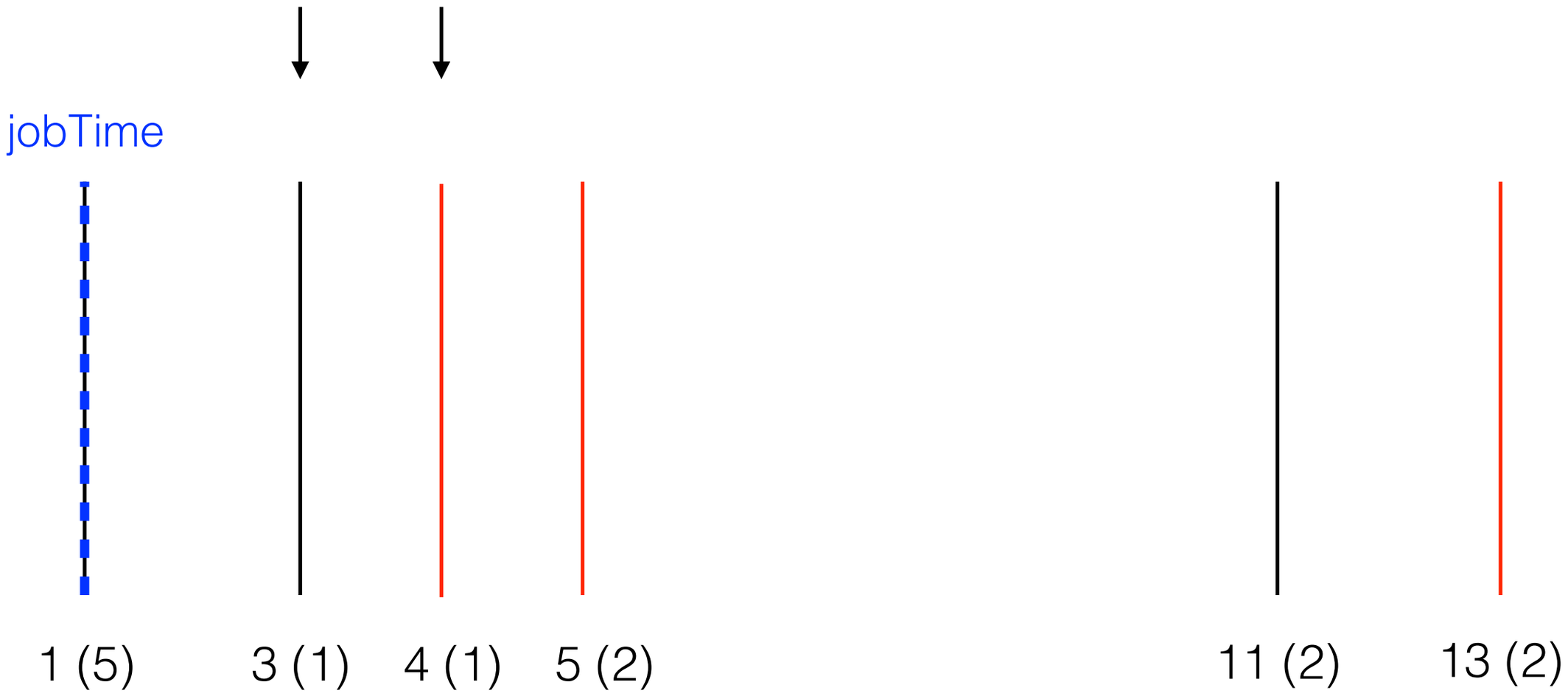
통찰 : 일반 주문, VIP 주문은 반드시 순서대로 처리되어야 한다



컴퓨터를 이용한 문제 해결 과정

5. 알고리즘을 코드로 작성한다

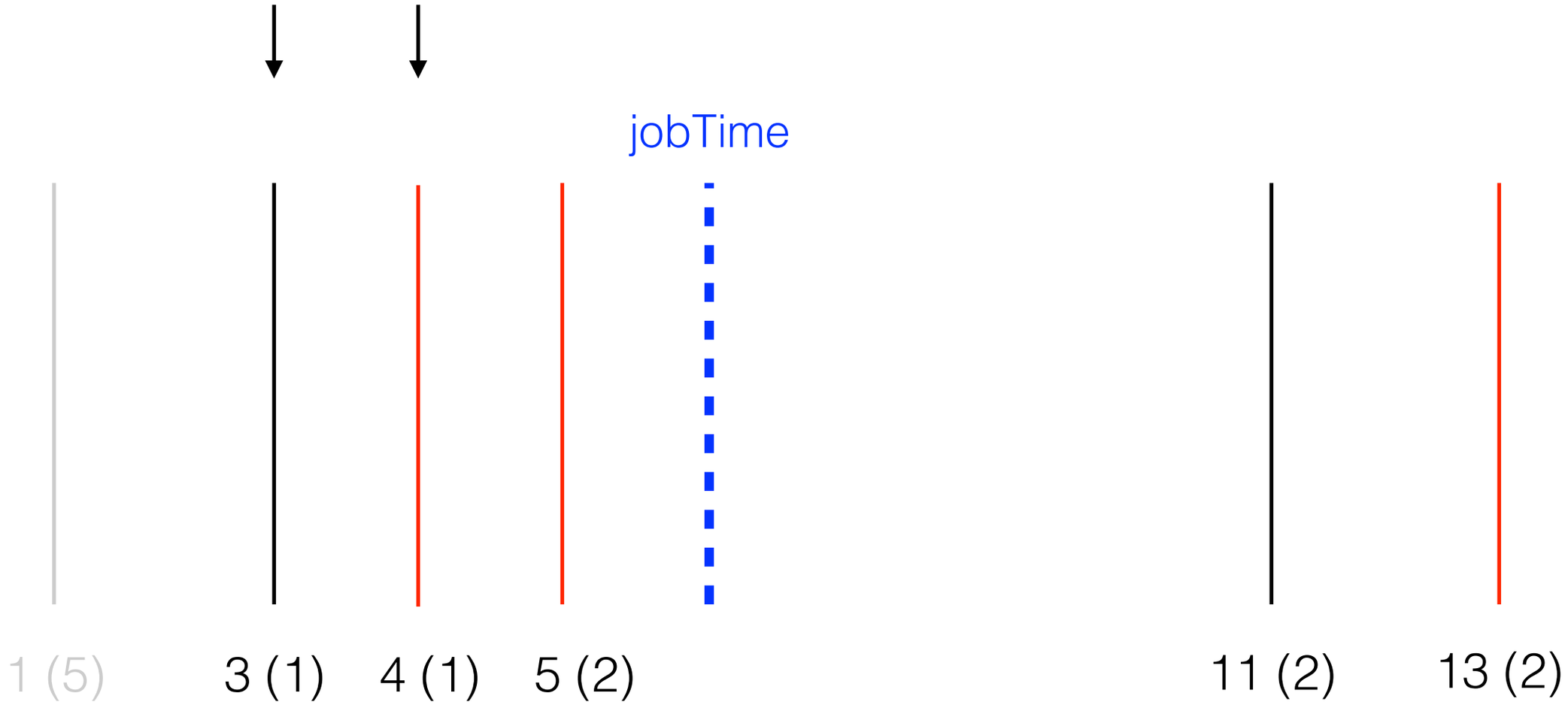
통찰 : 일반 주문, VIP 주문은 반드시 순서대로 처리되어야 한다



컴퓨터를 이용한 문제 해결 과정

5. 알고리즘을 코드로 작성한다

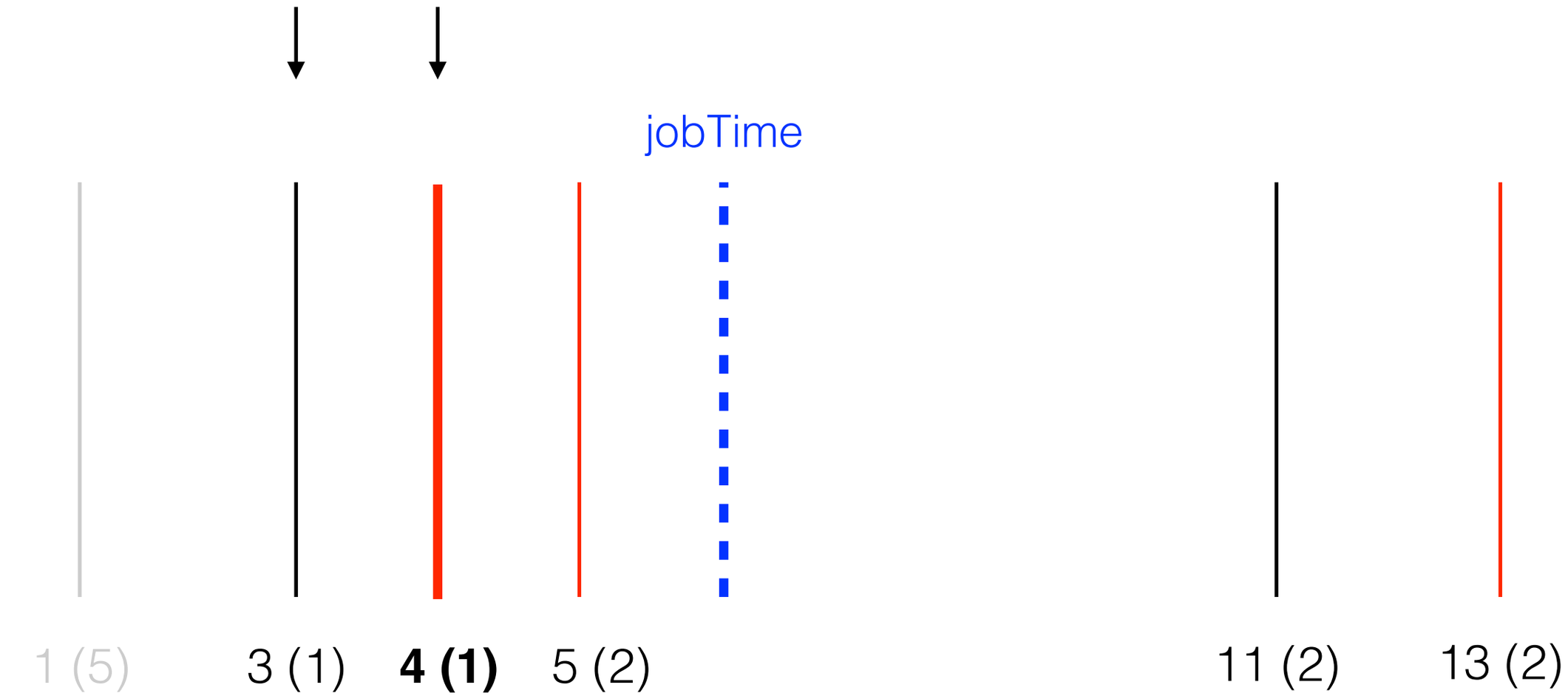
통찰 : 일반 주문, VIP 주문은 반드시 순서대로 처리되어야 한다



컴퓨터를 이용한 문제 해결 과정

5. 알고리즘을 코드로 작성한다

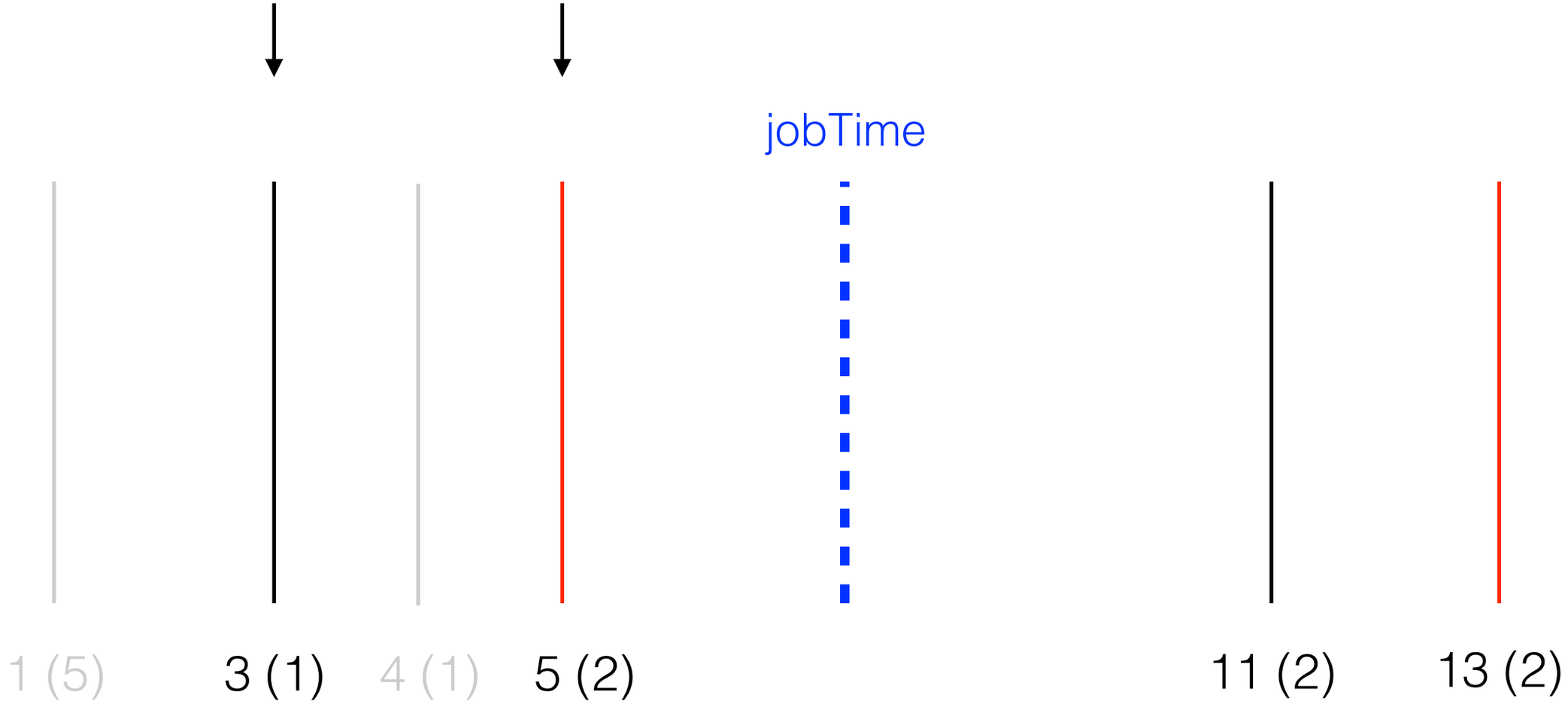
통찰 : 일반 주문, VIP 주문은 반드시 순서대로 처리되어야 한다



컴퓨터를 이용한 문제 해결 과정

5. 알고리즘을 코드로 작성한다

통찰 : 일반 주문, VIP 주문은 반드시 순서대로 처리되어야 한다



컴퓨터를 이용한 문제 해결 과정

5. 알고리즘을 코드로 작성한다

통찰 : 일반 주문, VIP 주문은 반드시 순서대로 처리되어야 한다

컴퓨터를 이용한 문제 해결 과정

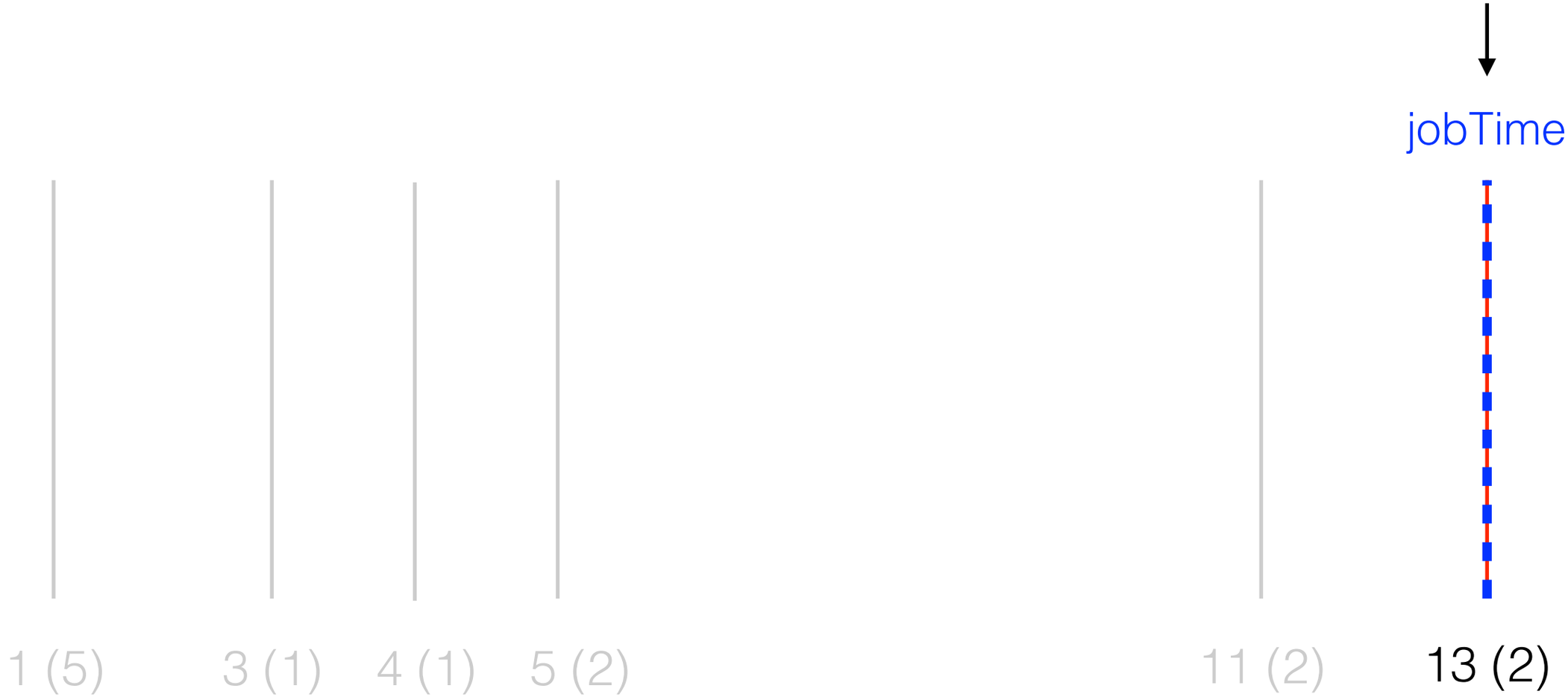
5. 알고리즘을 코드로 작성한다

통찰 : 일반 주문, VIP 주문은 반드시 순서대로 처리되어야 한다

컴퓨터를 이용한 문제 해결 과정

5. 알고리즘을 코드로 작성한다

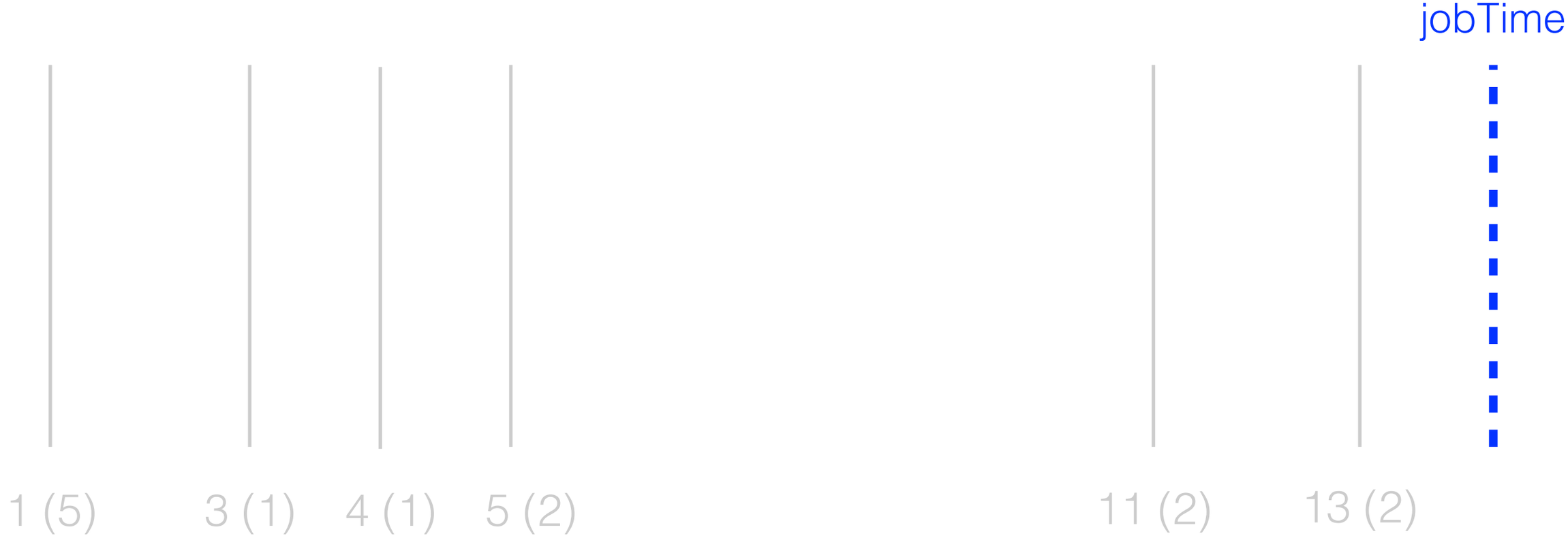
통찰 : 일반 주문, VIP 주문은 반드시 순서대로 처리되어야 한다



컴퓨터를 이용한 문제 해결 과정

5. 알고리즘을 코드로 작성한다

통찰 : 일반 주문, VIP 주문은 반드시 순서대로 처리되어야 한다



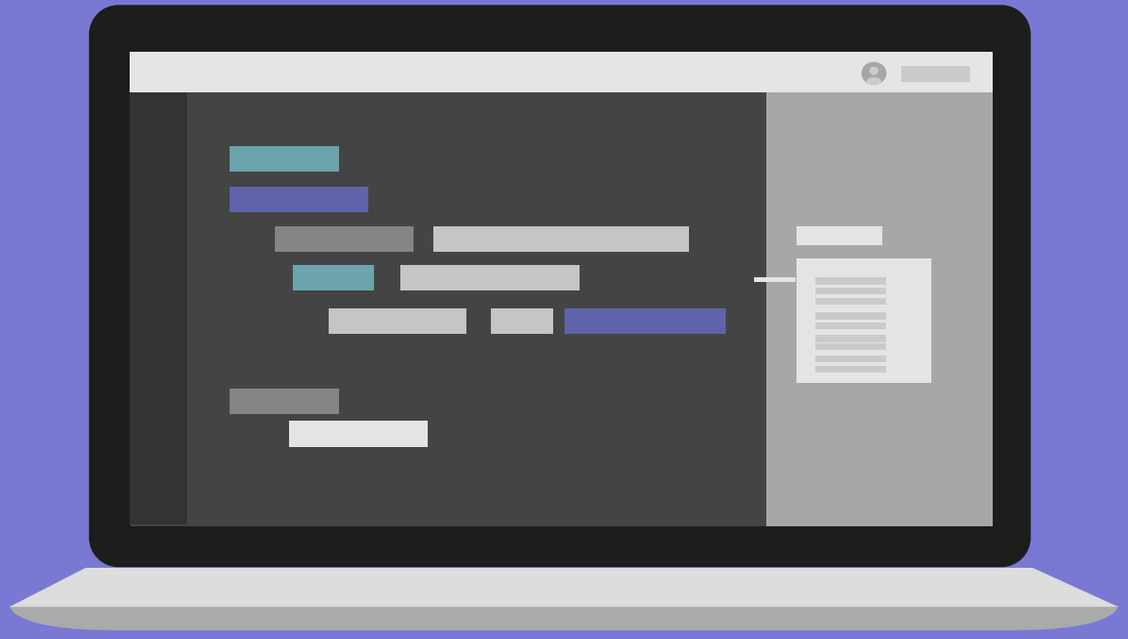
컴퓨터를 이용한 문제 해결 과정

5. 알고리즘을 코드로 작성한다

통찰 : 일반 주문, VIP 주문은 반드시 순서대로 처리되어야 한다

따라서 큐를 쓰자

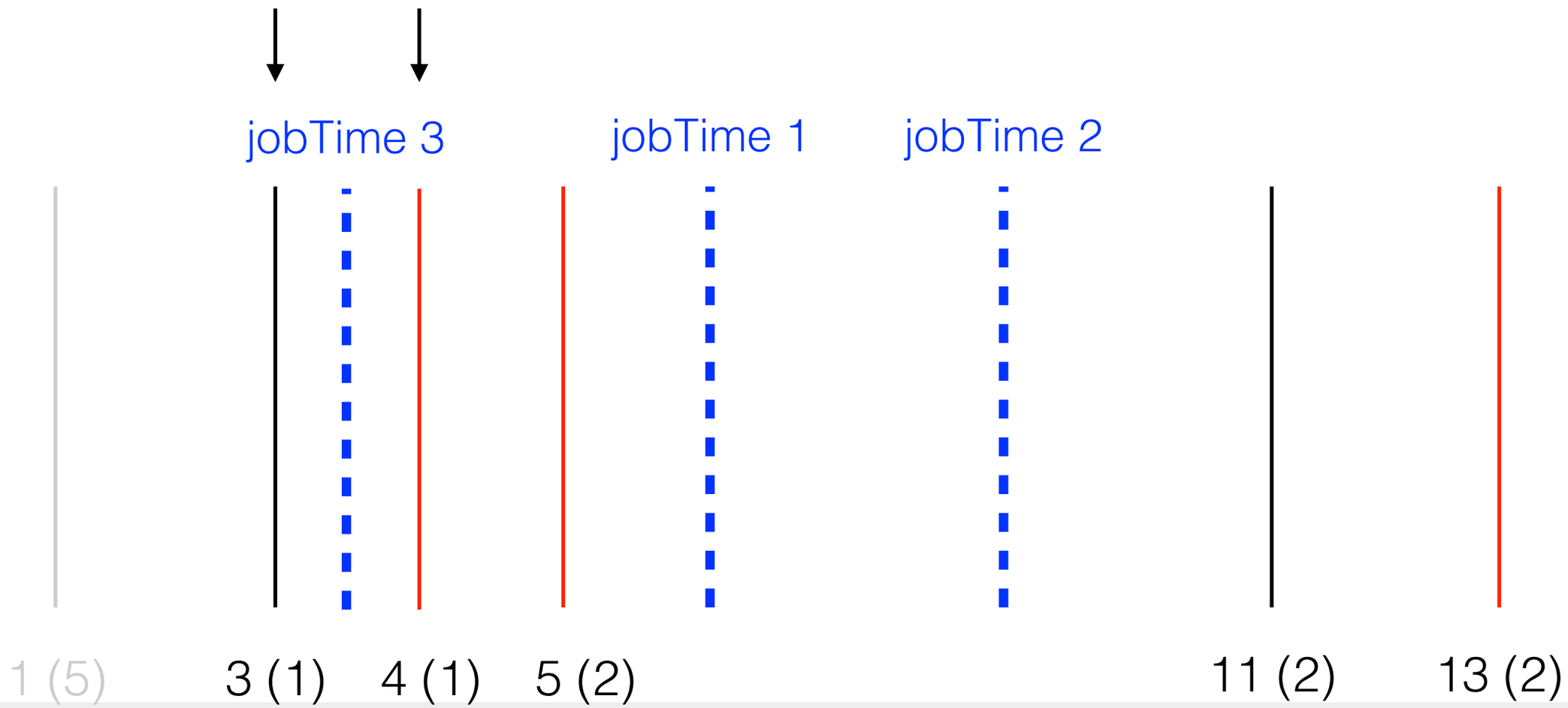
[문제 7] 주문 처리하기



[문제 8, 9] 주문 처리하기 2, M

1. 어느 큐를 선택할 것인가 ?

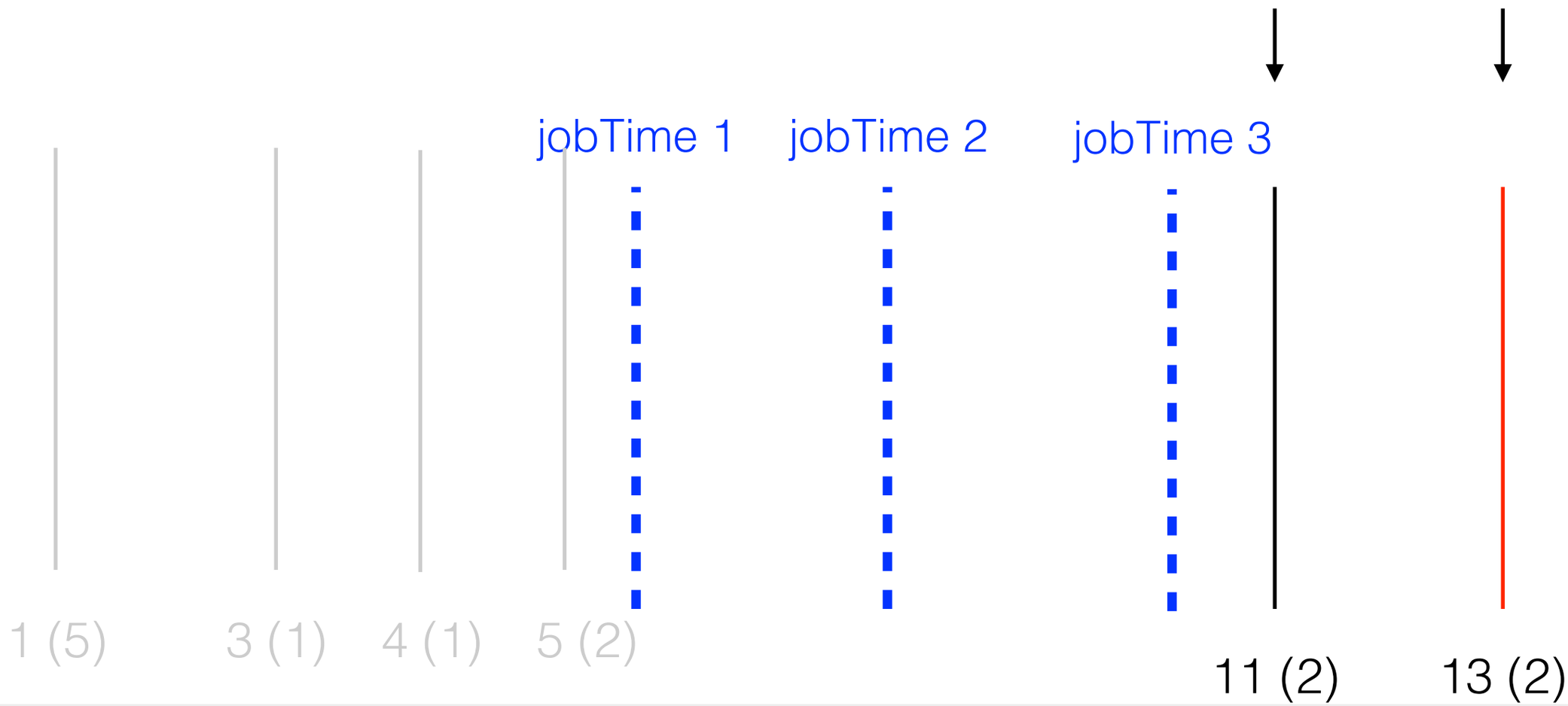
2. 어느 사람에게 일을 시킬 것인가 ?



[문제 8, 9] 주문 처리하기 2, M

1. 어느 큐를 선택할 것인가 ?

2. 어느 사람에게 일을 시킬 것인가 ?



시간복잡도 (Time Complexity)

알고리즘이 **대략** 몇개의 명령을 수행하는가?

프로그램의 수행 시간을 유추할 수 있음

시간복잡도 (Time Complexity)

알고리즘이 **대략** 몇개의 명령을 수행하는가?

프로그램의 수행 시간을 유추할 수 있음

```
sum = 0

for i in range(n) :
   sum = sum + i
```

시간복잡도 (Time Complexity)

알고리즘이 **대략** 몇개의 명령을 수행하는가?

프로그램의 수행 시간을 유추할 수 있음

```
sum = 0

for i in range(n) :
   sum = sum + i
```

n 개

시간복잡도 (Time Complexity)

알고리즘이 **대략** 몇개의 명령을 수행하는가?

프로그램의 수행 시간을 유추할 수 있음

```
sum = 0

for i in range(n) :
   sum = sum + i
```

O(n)

시간복잡도 (Time Complexity)

알고리즘이 **대략** 몇개의 명령을 수행하는가?

프로그램의 수행 시간을 유추할 수 있음

```
sum = 0

for i in range(n) :
   for j in range(n) :
      sum = sum + i + j
```

시간복잡도 (Time Complexity)

알고리즘이 **대략** 몇개의 명령을 수행하는가?

프로그램의 수행 시간을 유추할 수 있음

```
sum = 0

for i in range(n) :
   for j in range(n) :
      sum = sum + i + j
```

O(n²)

시간복잡도 (Time Complexity)

알고리즘이 **대략** 몇개의 명령을 수행하는가?

프로그램의 수행 시간을 유추할 수 있음

```
sum = 0

for i in range(n) :
   for j in range(i) :
      sum = sum + i + j
```

시간복잡도 (Time Complexity)

알고리즘이 **대략** 몇개의 명령을 수행하는가?

프로그램의 수행 시간을 유추할 수 있음

```
sum = 0

for i in range(n) :
   for j in range(i) :
      sum = sum + i + j
```

$O(n^2)$

시간복잡도 (Time Complexity)

알고리즘이 **대략** 몇개의 명령을 수행하는가?

프로그램의 수행 시간을 유추할 수 있음

```
def findNumber(myList, target) :

   for v in myList :
      if v == target :
         return True

   return False
```

시간복잡도 (Time Complexity)

알고리즘이 **대략** 몇개의 명령을 수행하는가?

프로그램의 수행 시간을 유추할 수 있음

```
def findNumber(myList, target) :

  for v in myList :
    if v == target :
      return True

  return False
```

O(n)

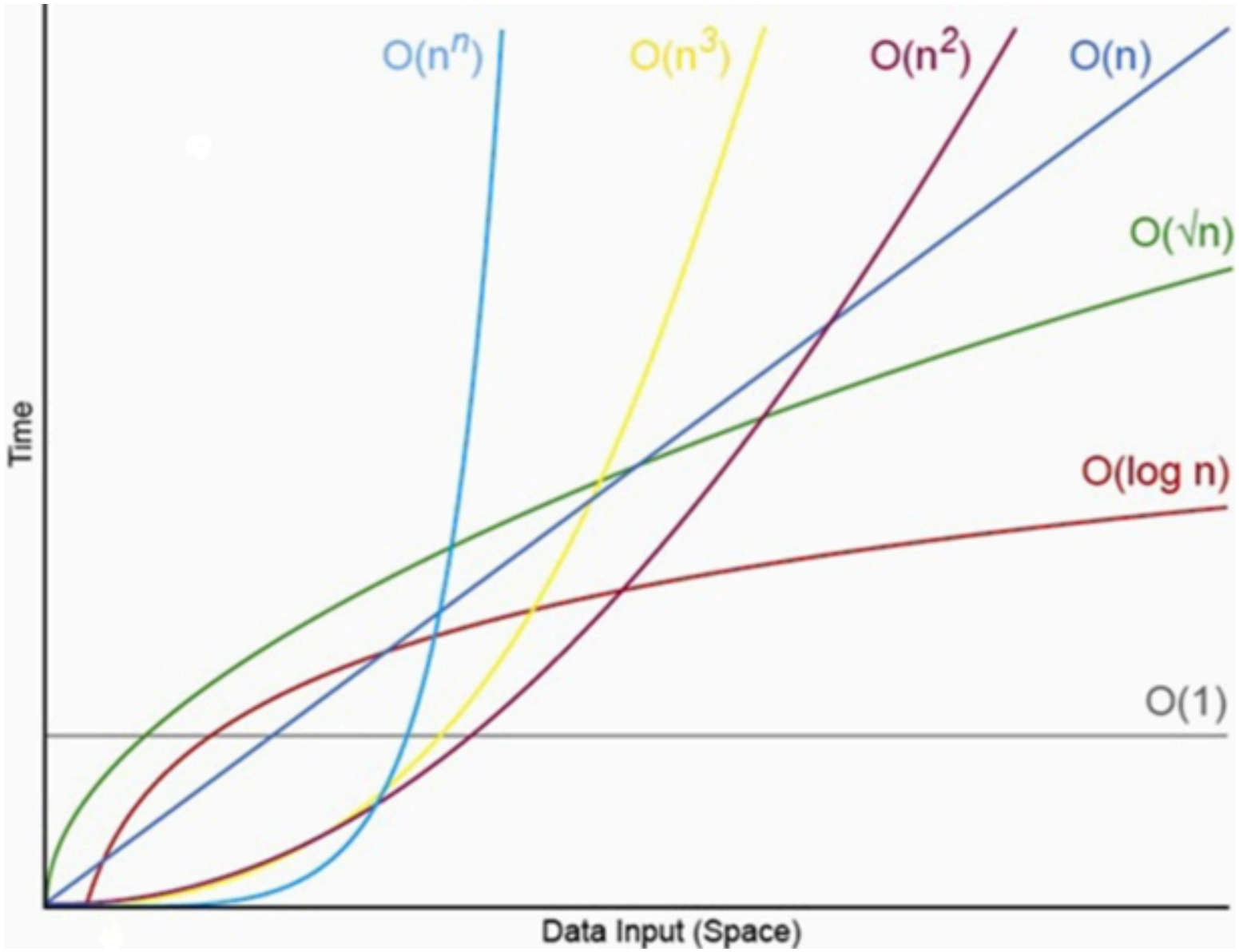
시간복잡도 (Time Complexity)

알고리즘이 **대략** 몇개의 명령을 수행하는가?

프로그램의 수행 시간을 유추할 수 있음

Big-O 표기 : 최악의 경우에 수행하는 명령 수

시간복잡도 (Time Complexity)



https://apelbaum.wordpress.com/2011/05/05/big-o/

시간복잡도와 실제 수행 시간

많은 명령을 수행한다 = 오래 걸린다

시간복잡도와 실제 수행 시간

많은 명령을 수행한다 = 오래 걸린다

몇 개의 명령을 수행해야 1초가 걸리는가 ?

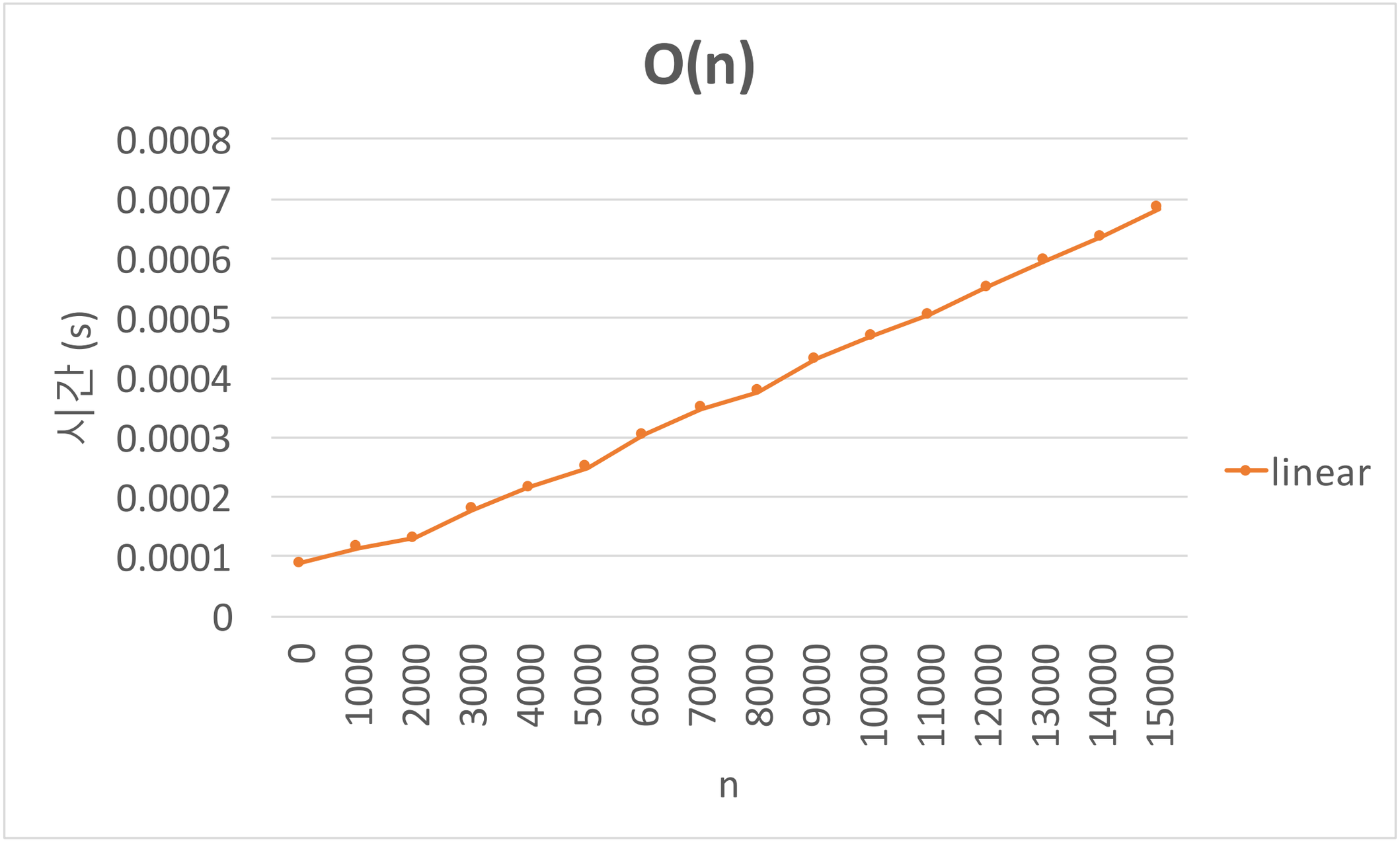
실험

아래 코드에 대하여 Elice에서 수행 시간을 측정

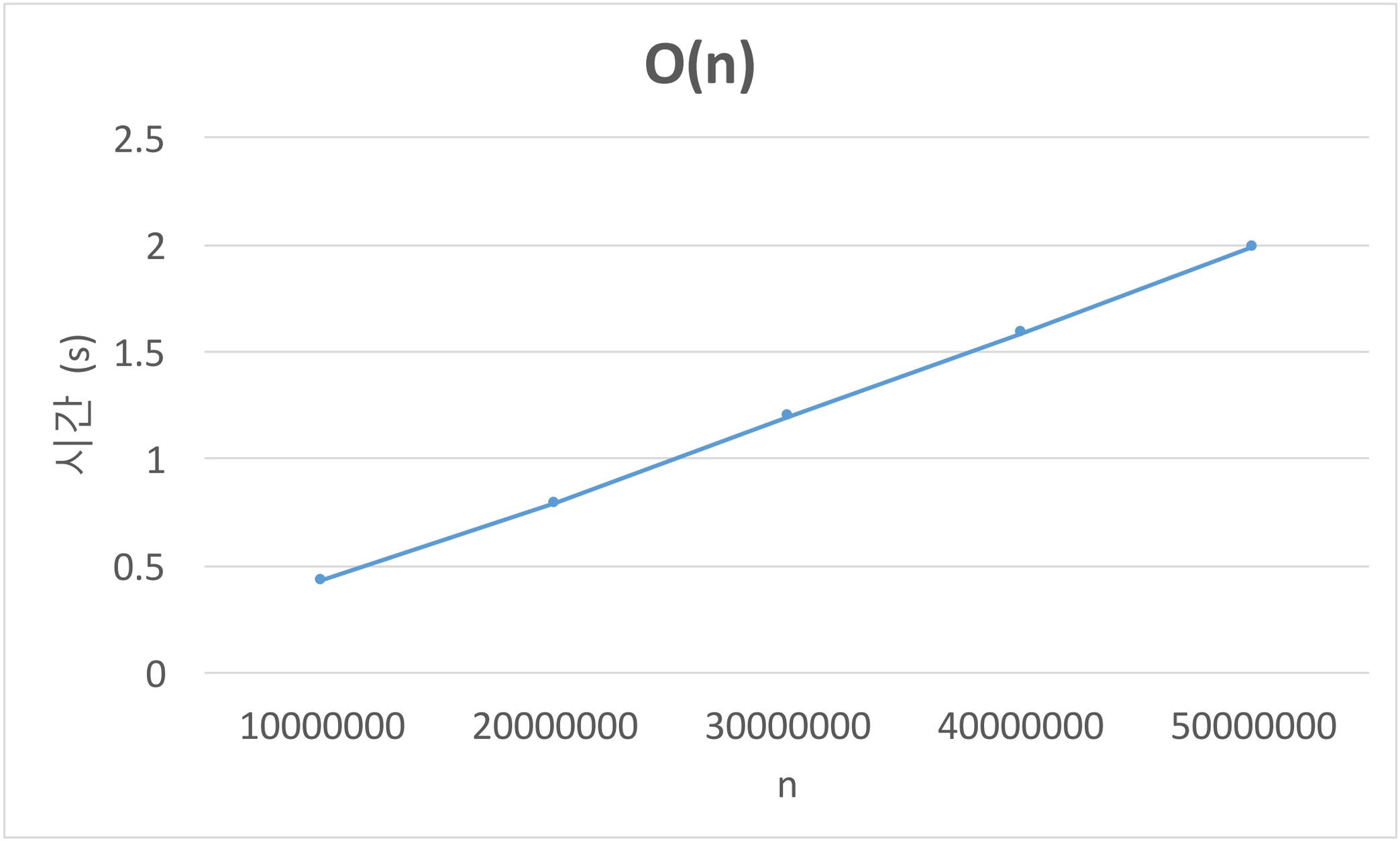
```
for i in range(n) :
  sum = sum + 1
```

```
for i in range(n) :
  for j in range(n) :
    sum = sum + 1
```

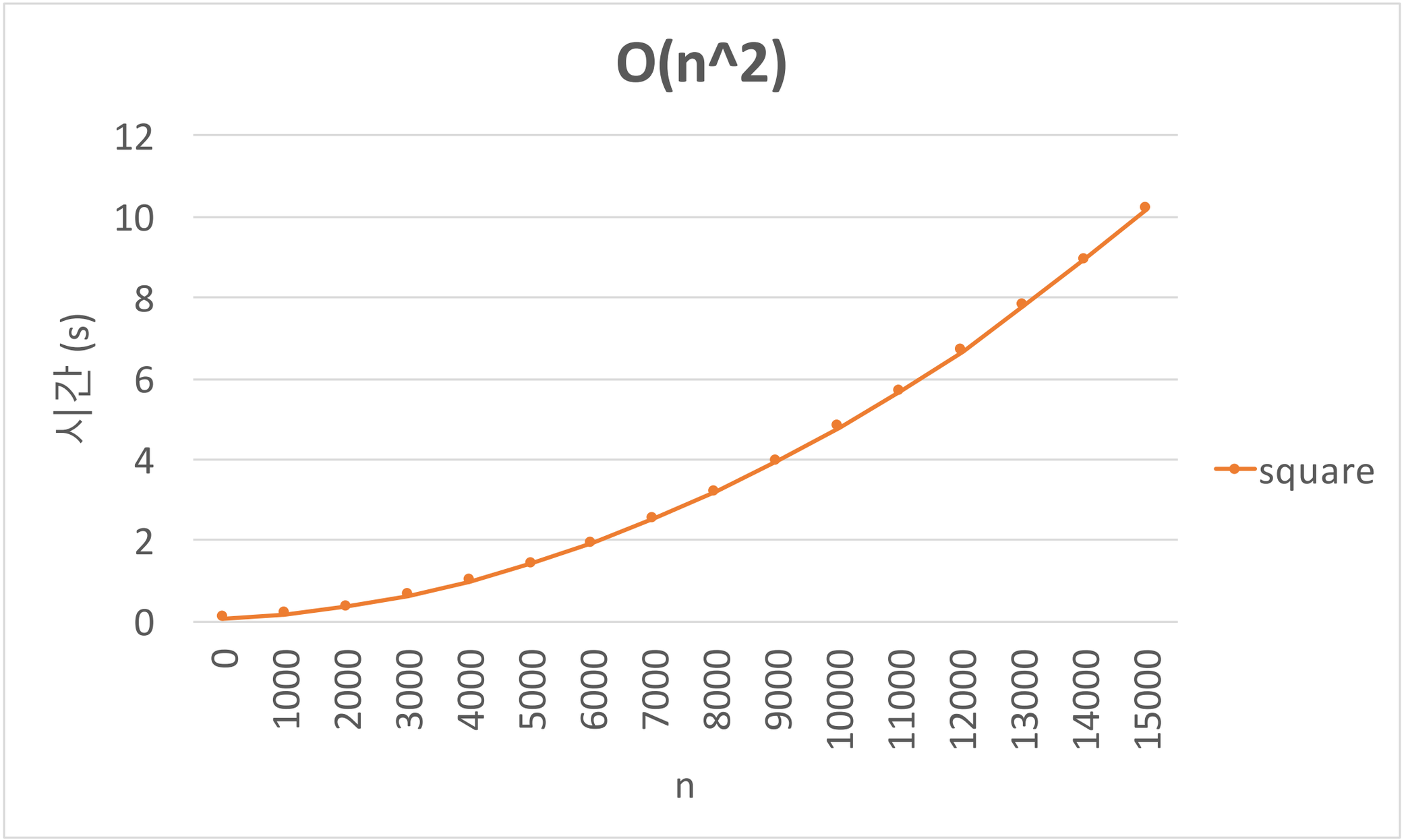
결과 : O(n)


O(n)
0.0008
0.0007
0.0006
0.0005
0.0004
0.0003
0.0002
0.0001
0
시간 (s)
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
11000
12000
13000
14000
15000
n
linear

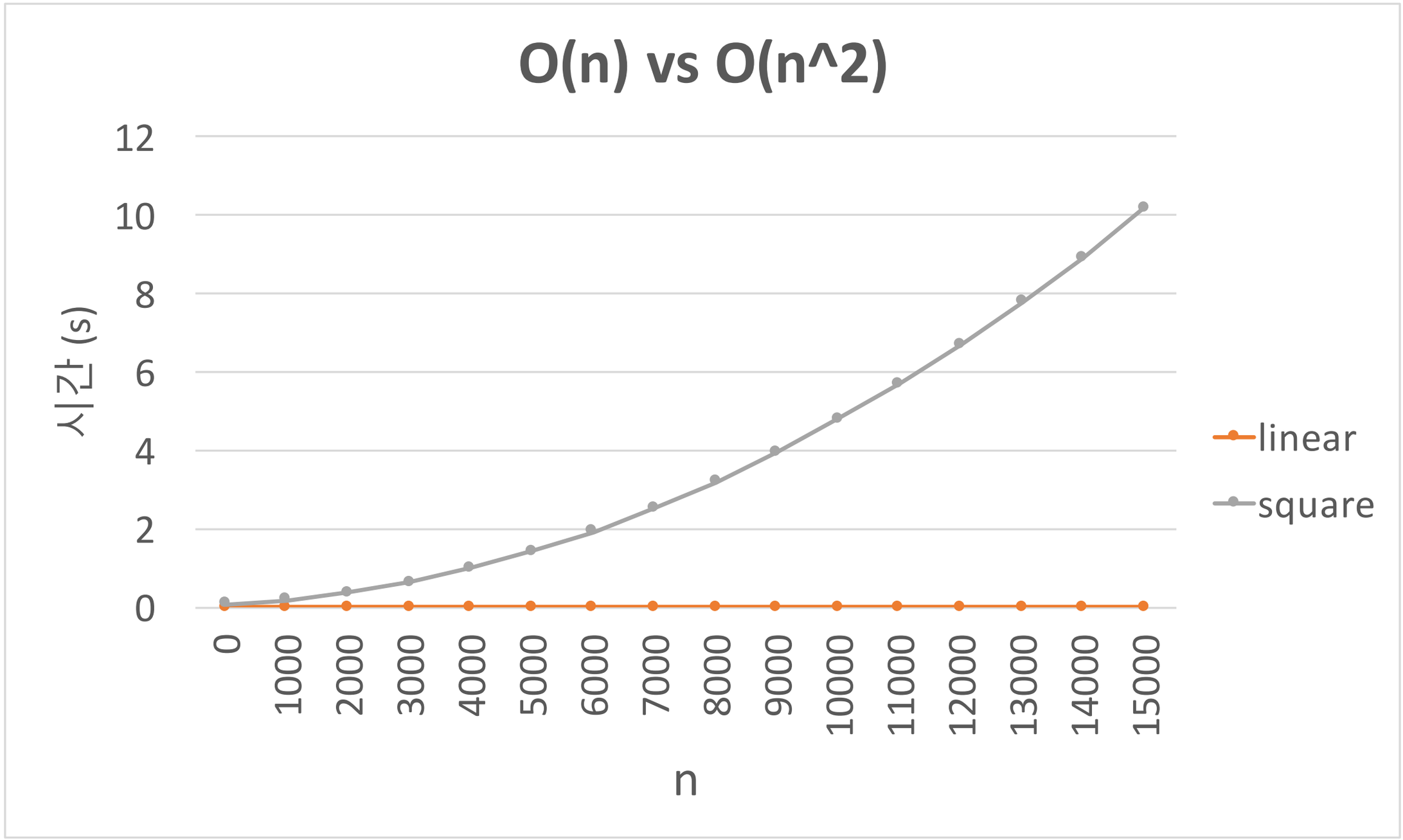
결과 : O(n)


O(n)
2.5
2
1.5
1
0.5
0
시간 (s)
10000000
20000000
30000000
40000000
50000000
n

결과 : $O(n^2)$


O(n^2)
12
10
8
6
4
2
0
시간 (s)
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
11000
12000
13000
14000
15000
n
square

결과 : O(n) vs O(n^2)


O(n) vs O(n^2)
시간 (s)
12
10
8
6
4
2
0
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
11000
12000
13000
14000
15000
n
linear
square

결론

대략 2500만개의 명령을

수행하면 1초가 걸린다

내 알고리즘이 **최악의 경우**에 2500만개를

수행하는지 고민해보자

[맛보기 문제 2] 두 번째 최댓값

행렬이 주어질 때 (합, 최댓값, 두 번째 최댓값) 을 구하라

(단, $1 \leq n \leq 1{,}000$, 제한시간 1초)

입력의 예

```
3
1 2 3
2 3 4
4 4 2
```

출력의 예

```
(25, 4, 3)
```

컴퓨터를 이용한 문제 해결 과정

1. 문제를 정확히 이해한다
2. 문제를 해결하는 알고리즘을 개발한다
3. 알고리즘이 **문제를 해결한다는 것을 증명한다**
4. 알고리즘이 **제한시간 내에 동작한다는 것을 보인다**
5. 알고리즘을 코드로 작성한다
6. 제출 후 만점을 받고 매우 기뻐한다

2. 알고리즘 개발

우선 최댓값을 찾은 후, 최댓값을 모두 지운다.

다시 최댓값을 찾으면, 이는 두 번째 최댓값이다.

1 2 3

2 3 4

4 4 2

2. 알고리즘 개발

우선 최댓값을 찾은 후, 최댓값을 모두 지운다.

다시 최댓값을 찾으면, 이는 두 번째 최댓값이다.

1	2	3
2	3	-1
-1	-1	2

2. 알고리즘 개발

우선 최댓값을 찾은 후, 최댓값을 모두 지운다.

다시 최댓값을 찾으면, 이는 두 번째 최댓값이다.

1	2	**3**
2	**3**	-1
-1	-1	2

2. 알고리즘 개발

우선 최댓값을 찾은 후, 최댓값을 모두 지운다.

다시 최댓값을 찾으면, 이는 두 번째 최댓값이다.

1	2	**3**	
2	**3**	-1	**O(3n²)**
-1	-1	2	

2. 알고리즘 개발

우선 최댓값을 찾은 후, 최댓값을 모두 지운다.

다시 최댓값을 찾으면, 이는 두 번째 최댓값이다.

1	2	**3**	
2	**3**	-1	**O(n²)**
-1	-1	2	

3. 풀이 증명

두 번째 최댓값 = “최댓값을 제외한 수” 중에서의 최댓값

1	2	**3**	
2	**3**	-1	**O(n²)**
-1	-1	2	

4. 제한시간 내에 동작하는지 판단

n의 최댓값은 1000

$O(n^2)$

4. 제한시간 내에 동작하는지 판단

n의 최댓값은 1000

$n^2 \leq 1{,}000{,}000$

$O(n^2)$

4. 제한시간 내에 동작하는지 판단

n의 최댓값은 1000

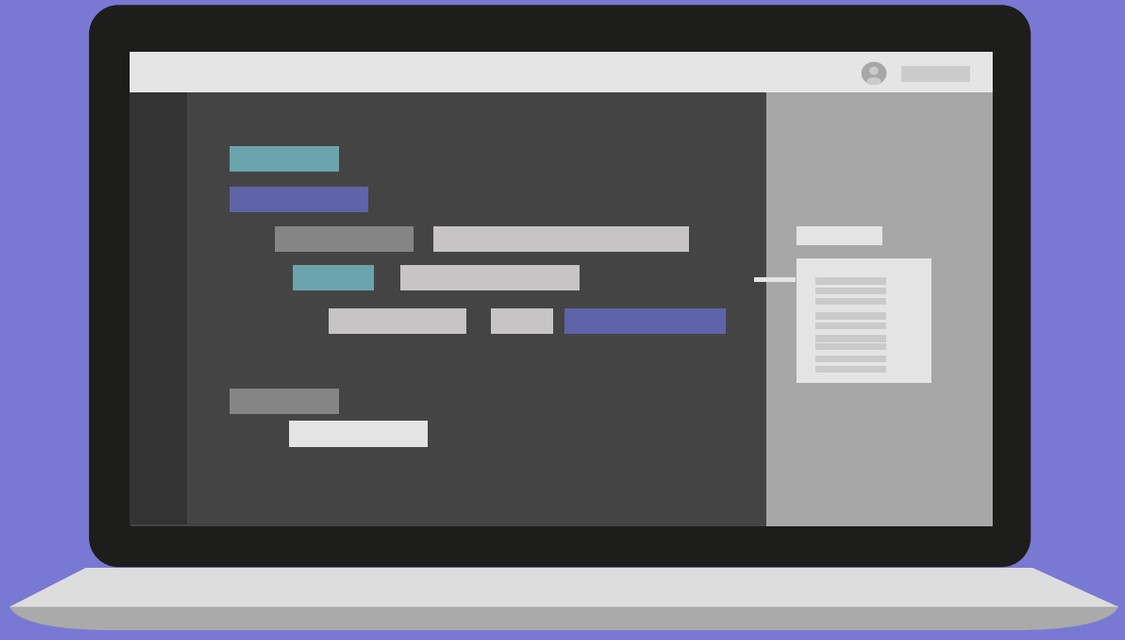
$n^2 \leq 1{,}000{,}000$

2500만번보다 적으므로 1초 내에 나온다

$O(n^2)$

[문제 1] 두 번째 최댓값

이 문제는 숙제로 해보세요 :)



[맛보기 문제 3] 소수 판정

숫자가 주어질 때, 소수인지 판정하라

(단, $1 \leq n \leq 1{,}000{,}000{,}000$ // 제한시간 1초)

입력의 예	출력의 예
7	True
4	False

[맛보기 문제 3] 소수 판정

알고리즘 개발

2 ~ n-1의 모든 수로 나누어본다.

[맛보기 문제 3] 소수 판정

알고리즘 개발

2 ~ n-1의 모든 수로 나누어본다.

풀이 증명

가능한 모든 수로 나누어보기 때문에 옳은 풀이이다

[맛보기 문제 3] 소수 판정

알고리즘 개발

2 ~ n-1의 모든 수로 나누어본다.

풀이 증명

가능한 모든 수로 나누어보기 때문에 옳은 풀이이다

시간복잡도

O(n). 2500만보다 더 많은 명령을 수행하므로 **1초에 안됨**

[맛보기 문제 3] 소수 판정

더 나은 알고리즘 개발

$2 \leq i \leq \sqrt{n}$ 의 모든 수로 나누어본다.

[맛보기 문제 3] 소수 판정

풀이 증명

증명해야 하는 명제 : **우리 알고리즘이 소수 판정을 옳게 한다**

[맛보기 문제 3] 소수 판정

풀이 증명

증명해야 하는 명제 : **우리 알고리즘이 소수 판정을 옳게 한다**

Case 1. n이 소수인 경우

n이 소수인 경우, 2 ~ $\sqrt{n}$ 의 모든 숫자로도 나누어 떨어지지 않는다.

따라서 우리 알고리즘은 True를 반환한다.

[맛보기 문제 3] 소수 판정

풀이 증명

증명해야 하는 명제 : **우리 알고리즘이 소수 판정을 옳게 한다**

Case 2. n이 소수가 아닌 경우

[맛보기 문제 3] 소수 판정

풀이 증명

증명해야 하는 명제 : **우리 알고리즘이 소수 판정을 옳게 한다**

Case 2. n이 소수가 아닌 경우

n은 소수가 아니므로 약수가 존재한다.

이 약수들 중에서 $\sqrt{n}$보다 작거나 같은 약수가 **반드시 존재한다.**

[맛보기 문제 3] 소수 판정

풀이 증명

증명해야 하는 명제 : **우리 알고리즘이 소수 판정을 옳게 한다**

Case 2. n이 소수가 아닌 경우

관찰 1. a가 n의 약수면, (n / a)는 자연수이다.

관찰 2. 만약 $a \geq \sqrt{n}$ 이면, $(n / a) \leq \sqrt{n}$ 이다.

[맛보기 문제 3] 소수 판정

풀이 증명

증명해야 하는 명제 : **우리 알고리즘이 소수 판정을 옳게 한다**

Case 2. n이 소수가 아닌 경우

만약 모든 약수가 $\sqrt{n}$보다 크다고 가정하자. 이 약수를 a라고 하자.

[맛보기 문제 3] 소수 판정

풀이 증명

증명해야 하는 명제 : **우리 알고리즘이 소수 판정을 옳게 한다**

Case 2. n이 소수가 아닌 경우

만약 모든 약수가 $\sqrt{n}$보다 크다고 가정하자. 이 약수를 a라고 하자.

그러면 (관찰 2)에 의하여 $(n / a) \leq \sqrt{n}$ 이고, 이 또한 n의 약수이다.

[맛보기 문제 3] 소수 판정

풀이 증명

증명해야 하는 명제 : **우리 알고리즘이 소수 판정을 옳게 한다**

Case 2. n이 소수가 아닌 경우

만약 모든 약수가 $\sqrt{n}$보다 크다고 가정하자. 이 약수를 a라고 하자.

그러면 (관찰 2)에 의하여 $(n / a) \leq \sqrt{n}$ 이고, 이 또한 n의 약수이다.

따라서 $\sqrt{n}$보다 작거나 같은 약수가 적어도 하나 존재한다.

[맛보기 문제 3] 소수 판정

풀이 증명

증명해야 하는 명제 : **우리 알고리즘이 소수 판정을 옳게 한다**

Case 2. n이 소수가 아닌 경우

우리 알고리즘은 $2 \leq i \leq \sqrt{n}$ 의 수가 n으로
나누어 떨어지는지 테스트한다.

해당 범위에는 적어도 하나의 약수가 반드시 존재하므로,
False가 반환된다.

[맛보기 문제 3] 소수 판정

시간복잡도

O(√n). n ≤ 1,000,000,000 이므로, √n ≤ 31,622

[맛보기 문제 3] 소수 판정

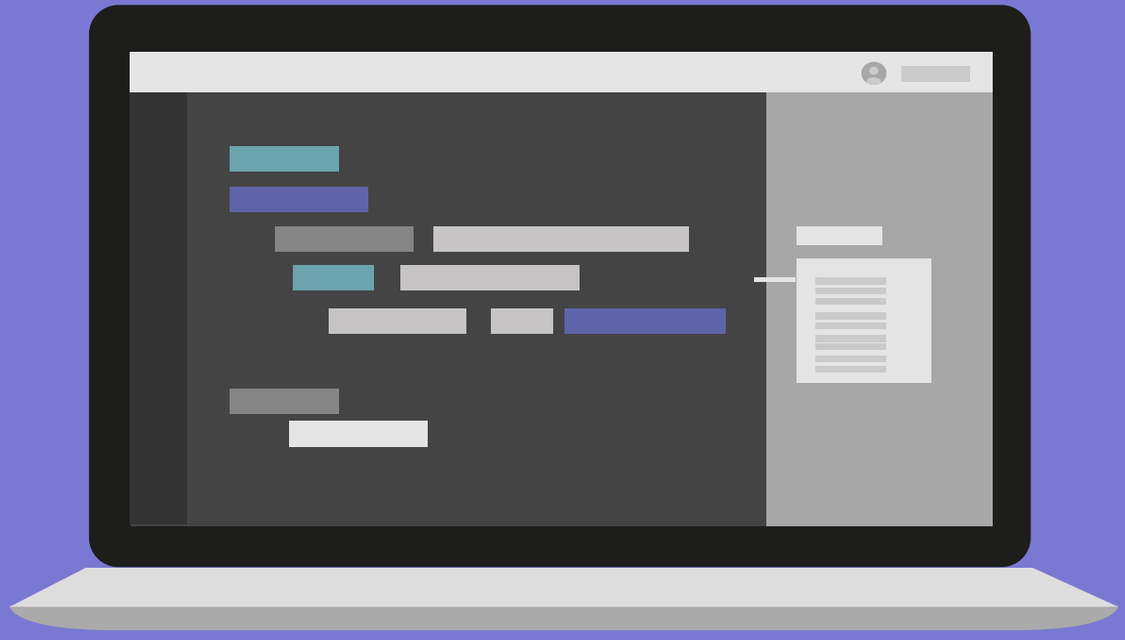
시간복잡도

O(√n). n ≤ 1,000,000,000 이므로, √n ≤ 31,622

2500만번보다 적은 횟수의 명령이므로

1초 내에 결과가 나온다

[문제 2] 소수 판정



[문제 3] 범위 소수 판정

범위가 주어질 때, 소수가 몇개인지 출력하라

(단, 1 ≤ a, b ≤ 100,000 // 제한시간 1초)

입력의 예

```
1  7
```

```
4  17
```

출력의 예

```
4
```

```
5
```

2. 알고리즘 개발

에라토스테네스의 체를 사용

		2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39

2. 알고리즘 개발

에라토스테네스의 체를 사용

2	3	4	5	6	7	8	9		
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39

2. 알고리즘 개발

에라토스테네스의 체를 사용

	2	3	4	5	6	7	8	9	
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39

2. 알고리즘 개발

에라토스테네스의 체를 사용

	2	**3**	4	5	6	7	8	9	
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39

2. 알고리즘 개발

에라토스테네스의 체를 사용

	2	**3**	4	5	6	7	8	9	
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39

2. 알고리즘 개발

에라토스테네스의 체를 사용

2	**3**	4	**5**	6	7	8	9		
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39

2. 알고리즘 개발

에라토스테네스의 체를 사용

2	**3**	4	**5**	6	7	8	9		
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39

2. 알고리즘 개발

에라토스테네스의 체를 사용

2	**3**	4	**5**	6	**7**	8	9		
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39

감사합니다!

신현규

E-mail : hyungyu.sh@kaist.ac.kr

Kakao : yougatup

/* elice */

문의 및 연락처

academy.elice.io

contact@elice.io

facebook.com/elice.io

blog.naver.com/elicer