>>>>>>>>>>> 제어 구조의 설계 원리를 중심으로 배우는 >>>>>>>>>>>

프로그래밍의 정석

파이썬

도경구 지음

생능출판

CHAPTER 7

표채워풀기

표채워풀기

동적계획법

Dynamic Programming

피보나찌 수열

조합 계산

1까지 줄이는 최소 스텝

하노이의 탑

CHAPTER 7

표채워풀기

피보나치 수열

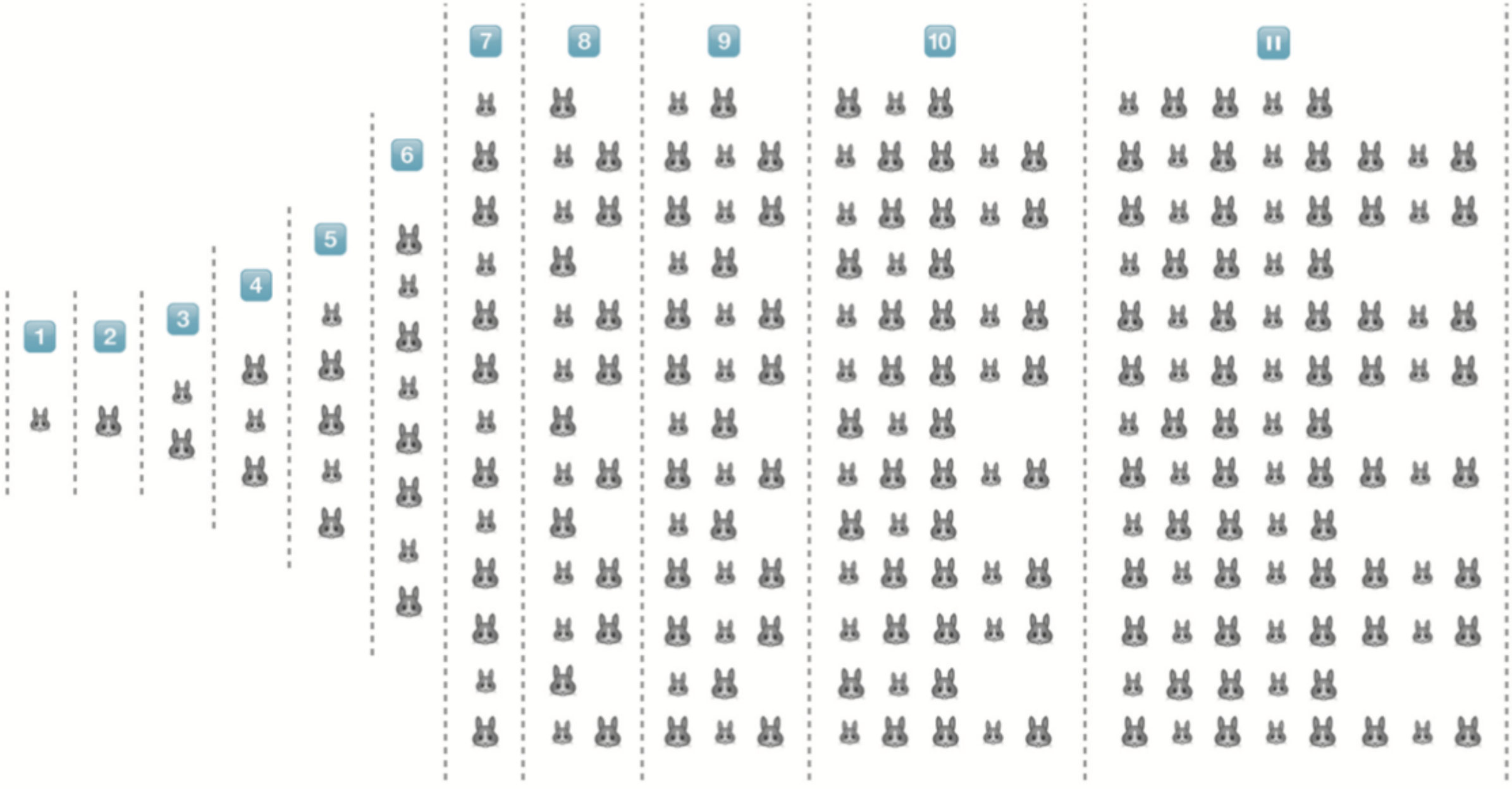
레오나르도 피보나치(Leonardo Fibonacci, 1170~1250년)

토끼가 달나라에 가면 다음과 같은 규칙으로 번식한다고 하자.

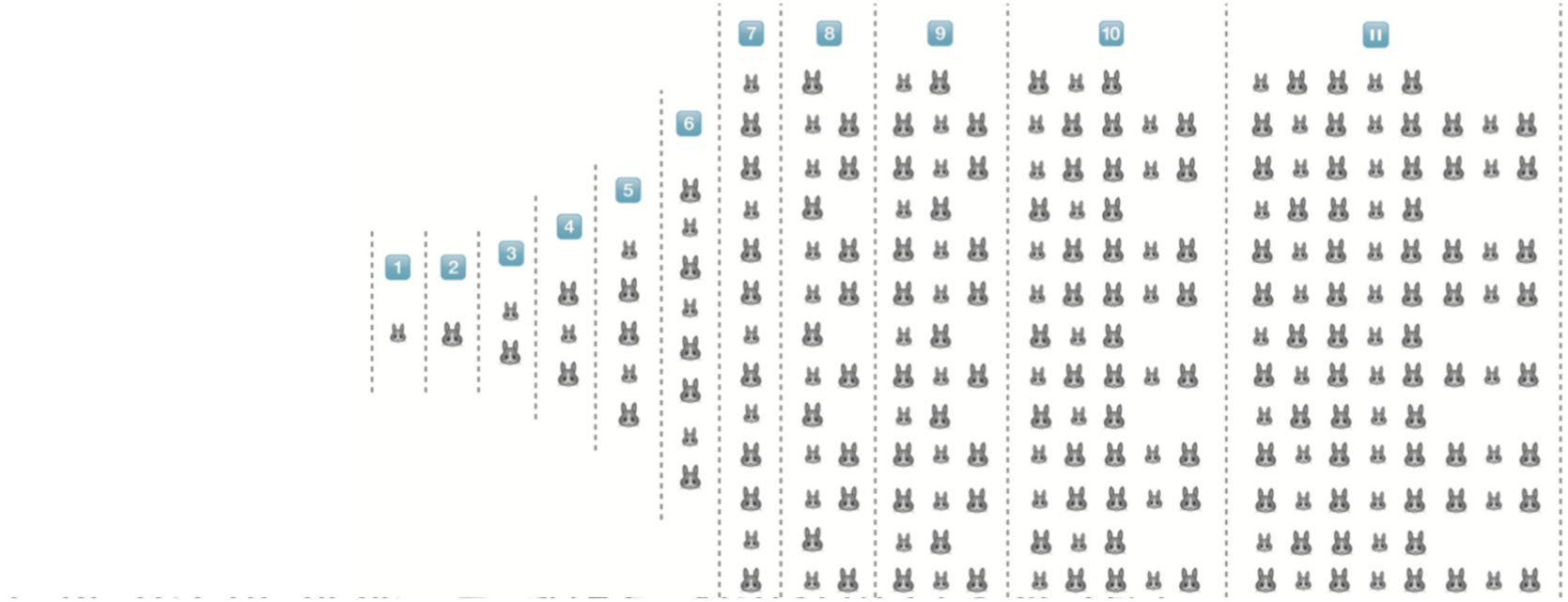
- 어른 토끼 암수 1쌍은 매달 아기 토끼 암수 1쌍씩 낳는다.
- 아기 토끼는 태어난 지 1달이 지나면 어른 토끼가 되어 번식 가능해진다.
- 토끼는 늙지도 않고 죽지도 않는다.

아기 토끼 암수 1쌍이 달나라에 정착하여 번식을 시작하면, 1년 뒤에 몇 쌍이 될까?

아기 토끼 암수 1쌍이 달나라에 정착하여 번식을 시작하면, 1년 뒤에 몇 쌍이 될까?



아기 토끼 암수 1쌍이 달나라에 정착하여 번식을 시작하면, 1년 뒤에 몇 쌍이 될까?



(단위: 쌍)

월	i	0	1	2	3	4	5	6	7	8	9	10	11	12
아기 토끼	$\mathtt{bunny}_i$	0	1	0	1	1	2	3	5	8	13	21	34	55
어른 토끼	$\mathtt{rabby}_i$	0	0	1	1	2	3	5	8	13	21	34	55	89
총	$\mathtt{fib}_i$	0	1	1	2	3	5	8	13	21	34	55	89	144

(단위: 쌍)

월	i	0	1	2	3	4	5	6	7	8	9	10	11	12
아기 토끼	$\mathtt{bunny}_i$	0	1	0	1	1	2	3	5	8	13	21	34	55
어른 토끼	$\mathtt{rabby}_i$	0	0	1	1	2	3	5	8	13	21	34	55	89
총	$\mathtt{fib}_i$	0	1	1	2	3	5	8	13	21	34	55	89	144

$$\mathtt{fib}_i = \mathtt{rabby}_i + \mathtt{bunny}_i$$

$$\mathtt{rabby}_i = \mathtt{fib}_{i-1} \qquad \mathtt{bunny}_i = \mathtt{fib}_{i-2}$$

$$\mathtt{fib}_i = \mathtt{fib}_{i-1} + \mathtt{fib}_{i-2}$$

5년(60개월) 후에 토끼 가족의 규모는 얼마나 될까?

재귀 알고리즘

$$fib(n) = \begin{cases} fib(n-1) + fib(n-2) & \text{if } n > 1 \\ 1 & \text{if } n = 1 \\ 0 & \text{if } n = 0 \end{cases}$$

code : 7-1.py

```
1  def fib(n):
2      if n > 1:
3          return fib(n - 1) + fib(n - 2)
4      else:
5          return n
```

code : 7-1.py

```
1 def fib(n):
2     if n > 1:
3         return fib(n - 1) + fib(n - 2)
4     else:
5         return n
```

```
   fib(5)
=>
```

code : 7-1.py

```
1  def fib(n):
2      if n > 1:
3          return fib(n - 1) + fib(n - 2)
4      else:
5          return n
```

```
   fib(5)
=> fib(4) + fib(3)
=>
```

code : 7-1.py

```
1  def fib(n):
2      if n > 1:
3          return fib(n - 1) + fib(n - 2)
4      else:
5          return n
```

```
   fib(5)
=> fib(4) + fib(3)
=> (fib(3) + fib(2)) + fib(3)
=>
```

code : 7-1.py

```
1  def fib(n):
2      if n > 1:
3          return fib(n - 1) + fib(n - 2)
4      else:
5          return n
```

```
   fib(5)
=> fib(4) + fib(3)
=> (fib(3) + fib(2)) + fib(3)
=> ((fib(2) + fib(1)) + fib(2)) + fib(3)
=>
```

code : 7-1.py

```
1  def fib(n):
2      if n > 1:
3          return fib(n - 1) + fib(n - 2)
4      else:
5          return n
```

```
   fib(5)
=> fib(4) + fib(3)
=> (fib(3) + fib(2)) + fib(3)
=> ((fib(2) + fib(1)) + fib(2)) + fib(3)
=> (((fib(1) + fib(0)) + fib(1)) + fib(2)) + fib(3)
=>
```

code : 7-1.py

```
1  def fib(n):
2      if n > 1:
3          return fib(n - 1) + fib(n - 2)
4      else:
5          return n
```

```
   fib(5)
=> fib(4) + fib(3)
=> (fib(3) + fib(2)) + fib(3)
=> ((fib(2) + fib(1)) + fib(2)) + fib(3)
=> (((fib(1) + fib(0)) + fib(1)) + fib(2)) + fib(3)
=> (((1 + fib(0)) + fib(1)) + fib(2)) + fib(3)
=>
```

code : 7-1.py

```
1  def fib(n):
2      if n > 1:
3          return fib(n - 1) + fib(n - 2)
4      else:
5          return n
```

```
   fib(5)
=> fib(4) + fib(3)
=> (fib(3) + fib(2)) + fib(3)
=> ((fib(2) + fib(1)) + fib(2)) + fib(3)
=> (((fib(1) + fib(0)) + fib(1)) + fib(2)) + fib(3)
=> (((1 + fib(0)) + fib(1)) + fib(2)) + fib(3)
=> (((1 + 0) + fib(1)) + fib(2)) + fib(3)
=>
```

code : 7-1.py

```
1 def fib(n):
2     if n > 1:
3         return fib(n - 1) + fib(n - 2)
4     else:
5         return n
```

```
   fib(5)
=> fib(4) + fib(3)
=> (fib(3) + fib(2)) + fib(3)
=> ((fib(2) + fib(1)) + fib(2)) + fib(3)
=> (((fib(1) + fib(0)) + fib(1)) + fib(2)) + fib(3)
=> (((1 + fib(0)) + fib(1)) + fib(2)) + fib(3)
=> (((1 + 0) + fib(1)) + fib(2)) + fib(3)
=> ((1 + fib(1)) + fib(2)) + fib(3)
=>
```

code : 7-1.py

```
1  def fib(n):
2      if n > 1:
3          return fib(n - 1) + fib(n - 2)
4      else:
5          return n
```

```
   fib(5)
=> fib(4) + fib(3)
=> (fib(3) + fib(2)) + fib(3)
=> ((fib(2) + fib(1)) + fib(2)) + fib(3)
=> (((fib(1) + fib(0)) + fib(1)) + fib(2)) + fib(3)
=> (((1 + fib(0)) + fib(1)) + fib(2)) + fib(3)
=> (((1 + 0) + fib(1)) + fib(2)) + fib(3)
=> ((1 + fib(1)) + fib(2)) + fib(3)
=> ((1 + 1) + fib(2)) + fib(3)
=>
```

code : 7-1.py

```
1  def fib(n):
2      if n > 1:
3          return fib(n - 1) + fib(n - 2)
4      else:
5          return n
```

```
   fib(5)
=> fib(4) + fib(3)
=> (fib(3) + fib(2)) + fib(3)
=> ((fib(2) + fib(1)) + fib(2)) + fib(3)
=> (((fib(1) + fib(0)) + fib(1)) + fib(2)) + fib(3)
=> (((1 + fib(0)) + fib(1)) + fib(2)) + fib(3)
=> (((1 + 0) + fib(1)) + fib(2)) + fib(3)
=> ((1 + fib(1)) + fib(2)) + fib(3)
=> ((1 + 1) + fib(2)) + fib(3)
=> (2 + fib(2)) + fib(3)
=>
```

code : 7-1.py

```
1  def fib(n):
2      if n > 1:
3          return fib(n - 1) + fib(n - 2)
4      else:
5          return n
```

```
   fib(5)
=> fib(4) + fib(3)
=> (fib(3) + fib(2)) + fib(3)
=> ((fib(2) + fib(1)) + fib(2)) + fib(3)
=> (((fib(1) + fib(0)) + fib(1)) + fib(2)) + fib(3)
=> (((1 + fib(0)) + fib(1)) + fib(2)) + fib(3)
=> (((1 + 0) + fib(1)) + fib(2)) + fib(3)
=> ((1 + fib(1)) + fib(2)) + fib(3)
=> ((1 + 1) + fib(2)) + fib(3)
=> (2 + fib(2)) + fib(3)
=> (2 + (fib(1) + fib(0))) + fib(3)
=>
```

code : 7-1.py

```
1 def fib(n):
2     if n > 1:
3         return fib(n - 1) + fib(n - 2)
4     else:
5         return n
```

```
   fib(5)
=> fib(4) + fib(3)
=> (fib(3) + fib(2)) + fib(3)
=> ((fib(2) + fib(1)) + fib(2)) + fib(3)
=> (((fib(1) + fib(0)) + fib(1)) + fib(2)) + fib(3)
=> (((1 + fib(0)) + fib(1)) + fib(2)) + fib(3)
=> (((1 + 0) + fib(1)) + fib(2)) + fib(3)
=> ((1 + fib(1)) + fib(2)) + fib(3)
=> ((1 + 1) + fib(2)) + fib(3)
=> (2 + fib(2)) + fib(3)
=> (2 + (fib(1) + fib(0))) + fib(3)
=> (2 + (1 + fib(0))) + fib(3)
=>
```

code : 7-1.py

```
1  def fib(n):
2      if n > 1:
3          return fib(n - 1) + fib(n - 2)
4      else:
5          return n
```

```
   fib(5)
=> fib(4) + fib(3)
=> (fib(3) + fib(2)) + fib(3)
=> ((fib(2) + fib(1)) + fib(2)) + fib(3)
=> (((fib(1) + fib(0)) + fib(1)) + fib(2)) + fib(3)
=> (((1 + fib(0)) + fib(1)) + fib(2)) + fib(3)
=> (((1 + 0) + fib(1)) + fib(2)) + fib(3)
=> ((1 + fib(1)) + fib(2)) + fib(3)
=> ((1 + 1) + fib(2)) + fib(3)
=> (2 + fib(2)) + fib(3)
=> (2 + (fib(1) + fib(0))) + fib(3)
=> (2 + (1 + fib(0))) + fib(3)
=> (2 + (1 + 0)) + fib(3)
=>
```

code : 7-1.py

```
1  def fib(n):
2      if n > 1:
3          return fib(n - 1) + fib(n - 2)
4      else:
5          return n
```

```
   fib(5)
=> fib(4) + fib(3)
=> (fib(3) + fib(2)) + fib(3)
=> ((fib(2) + fib(1)) + fib(2)) + fib(3)
=> (((fib(1) + fib(0)) + fib(1)) + fib(2)) + fib(3)
=> (((1 + fib(0)) + fib(1)) + fib(2)) + fib(3)
=> (((1 + 0) + fib(1)) + fib(2)) + fib(3)
=> ((1 + fib(1)) + fib(2)) + fib(3)
=> ((1 + 1) + fib(2)) + fib(3)
=> (2 + fib(2)) + fib(3)
=> (2 + (fib(1) + fib(0))) + fib(3)
=> (2 + (1 + fib(0))) + fib(3)
=> (2 + (1 + 0)) + fib(3)
=> (2 + 1) + fib(3)
=>
```

code : 7-1.py

```
1  def fib(n):
2      if n > 1:
3          return fib(n - 1) + fib(n - 2)
4      else:
5          return n
```

```
   fib(5)
=> fib(4) + fib(3)
=> (fib(3) + fib(2)) + fib(3)
=> ((fib(2) + fib(1)) + fib(2)) + fib(3)
=> (((fib(1) + fib(0)) + fib(1)) + fib(2)) + fib(3)
=> (((1 + fib(0)) + fib(1)) + fib(2)) + fib(3)
=> (((1 + 0) + fib(1)) + fib(2)) + fib(3)
=> ((1 + fib(1)) + fib(2)) + fib(3)
=> ((1 + 1) + fib(2)) + fib(3)
=> (2 + fib(2)) + fib(3)
=> (2 + (fib(1) + fib(0))) + fib(3)
=> (2 + (1 + fib(0))) + fib(3)
=> (2 + (1 + 0)) + fib(3)
=> (2 + 1) + fib(3)
=> 3 + fib(3)
=>
```

code : 7-1.py

```
1  def fib(n):
2      if n > 1:
3          return fib(n - 1) + fib(n - 2)
4      else:
5          return n
```

```
   fib(5)
=> fib(4) + fib(3)
=> (fib(3) + fib(2)) + fib(3)
=> ((fib(2) + fib(1)) + fib(2)) + fib(3)
=> (((fib(1) + fib(0)) + fib(1)) + fib(2)) + fib(3)
=> (((1 + fib(0)) + fib(1)) + fib(2)) + fib(3)
=> (((1 + 0) + fib(1)) + fib(2)) + fib(3)
=> ((1 + fib(1)) + fib(2)) + fib(3)
=> ((1 + 1) + fib(2)) + fib(3)
=> (2 + fib(2)) + fib(3)
=> (2 + (fib(1) + fib(0))) + fib(3)
=> (2 + (1 + fib(0))) + fib(3)
=> (2 + (1 + 0)) + fib(3)
=> (2 + 1) + fib(3)
=> 3 + fib(3)
=> 3 + (fib(2) + fib(1))
=>
```

code : 7-1.py

```
1 def fib(n):
2     if n > 1:
3         return fib(n - 1) + fib(n - 2)
4     else:
5         return n
```

```
   fib(5)
=> fib(4) + fib(3)
=> (fib(3) + fib(2)) + fib(3)
=> ((fib(2) + fib(1)) + fib(2)) + fib(3)
=> (((fib(1) + fib(0)) + fib(1)) + fib(2)) + fib(3)
=> (((1 + fib(0)) + fib(1)) + fib(2)) + fib(3)
=> (((1 + 0) + fib(1)) + fib(2)) + fib(3)
=> ((1 + fib(1)) + fib(2)) + fib(3)
=> ((1 + 1) + fib(2)) + fib(3)
=> (2 + fib(2)) + fib(3)
=> (2 + (fib(1) + fib(0))) + fib(3)
=> (2 + (1 + fib(0))) + fib(3)
=> (2 + (1 + 0)) + fib(3)
=> (2 + 1) + fib(3)
=> 3 + fib(3)
=> 3 + (fib(2) + fib(1))
=> 3 + ((fib(1) + fib(0)) + fib(1))
=>
```

code : 7-1.py

```
1  def fib(n):
2      if n > 1:
3          return fib(n - 1) + fib(n - 2)
4      else:
5          return n
```

```
   fib(5)
=> fib(4) + fib(3)
=> (fib(3) + fib(2)) + fib(3)
=> ((fib(2) + fib(1)) + fib(2)) + fib(3)
=> (((fib(1) + fib(0)) + fib(1)) + fib(2)) + fib(3)
=> (((1 + fib(0)) + fib(1)) + fib(2)) + fib(3)
=> (((1 + 0) + fib(1)) + fib(2)) + fib(3)
=> ((1 + fib(1)) + fib(2)) + fib(3)
=> ((1 + 1) + fib(2)) + fib(3)
=> (2 + fib(2)) + fib(3)
=> (2 + (fib(1) + fib(0))) + fib(3)
=> (2 + (1 + fib(0))) + fib(3)
=> (2 + (1 + 0)) + fib(3)
=> (2 + 1) + fib(3)
=> 3 + fib(3)
=> 3 + (fib(2) + fib(1))
=> 3 + ((fib(1) + fib(0)) + fib(1))
=> 3 + ((1 + fib(0)) + fib(1))
=>
```

code : 7-1.py

```
1  def fib(n):
2      if n > 1:
3          return fib(n - 1) + fib(n - 2)
4      else:
5          return n
```

```
   fib(5)
=> fib(4) + fib(3)
=> (fib(3) + fib(2)) + fib(3)
=> ((fib(2) + fib(1)) + fib(2)) + fib(3)
=> (((fib(1) + fib(0)) + fib(1)) + fib(2)) + fib(3)
=> (((1 + fib(0)) + fib(1)) + fib(2)) + fib(3)
=> (((1 + 0) + fib(1)) + fib(2)) + fib(3)
=> ((1 + fib(1)) + fib(2)) + fib(3)
=> ((1 + 1) + fib(2)) + fib(3)
=> (2 + fib(2)) + fib(3)
=> (2 + (fib(1) + fib(0))) + fib(3)
=> (2 + (1 + fib(0))) + fib(3)
=> (2 + (1 + 0)) + fib(3)
=> (2 + 1) + fib(3)
=> 3 + fib(3)
=> 3 + (fib(2) + fib(1))
=> 3 + ((fib(1) + fib(0)) + fib(1))
=> 3 + ((1 + fib(0)) + fib(1))
=> 3 + ((1 + 0) + fib(1))
=>
```

code : 7-1.py

```
1  def fib(n):
2      if n > 1:
3          return fib(n - 1) + fib(n - 2)
4      else:
5          return n
```

```
   fib(5)
=> fib(4) + fib(3)
=> (fib(3) + fib(2)) + fib(3)
=> ((fib(2) + fib(1)) + fib(2)) + fib(3)
=> (((fib(1) + fib(0)) + fib(1)) + fib(2)) + fib(3)
=> (((1 + fib(0)) + fib(1)) + fib(2)) + fib(3)
=> (((1 + 0) + fib(1)) + fib(2)) + fib(3)
=> ((1 + fib(1)) + fib(2)) + fib(3)
=> ((1 + 1) + fib(2)) + fib(3)
=> (2 + fib(2)) + fib(3)
=> (2 + (fib(1) + fib(0))) + fib(3)
=> (2 + (1 + fib(0))) + fib(3)
=> (2 + (1 + 0)) + fib(3)
=> (2 + 1) + fib(3)
=> 3 + fib(3)
=> 3 + (fib(2) + fib(1))
=> 3 + ((fib(1) + fib(0)) + fib(1))
=> 3 + ((1 + fib(0)) + fib(1))
=> 3 + ((1 + 0) + fib(1))
=> 3 + (1 + fib(1))
=>
```

code : 7-1.py

```
1  def fib(n):
2      if n > 1:
3          return fib(n - 1) + fib(n - 2)
4      else:
5          return n
```

```
   fib(5)
=> fib(4) + fib(3)
=> (fib(3) + fib(2)) + fib(3)
=> ((fib(2) + fib(1)) + fib(2)) + fib(3)
=> (((fib(1) + fib(0)) + fib(1)) + fib(2)) + fib(3)
=> (((1 + fib(0)) + fib(1)) + fib(2)) + fib(3)
=> (((1 + 0) + fib(1)) + fib(2)) + fib(3)
=> ((1 + fib(1)) + fib(2)) + fib(3)
=> ((1 + 1) + fib(2)) + fib(3)
=> (2 + fib(2)) + fib(3)
=> (2 + (fib(1) + fib(0))) + fib(3)
=> (2 + (1 + fib(0))) + fib(3)
=> (2 + (1 + 0)) + fib(3)
=> (2 + 1) + fib(3)
=> 3 + fib(3)
=> 3 + (fib(2) + fib(1))
=> 3 + ((fib(1) + fib(0)) + fib(1))
=> 3 + ((1 + fib(0)) + fib(1))
=> 3 + ((1 + 0) + fib(1))
=> 3 + (1 + fib(1))
=> 3 + (1 + 1)
=>
```

code : 7-1.py

```
1  def fib(n):
2      if n > 1:
3          return fib(n - 1) + fib(n - 2)
4      else:
5          return n
```

```
   fib(5)
=> fib(4) + fib(3)
=> (fib(3) + fib(2)) + fib(3)
=> ((fib(2) + fib(1)) + fib(2)) + fib(3)
=> (((fib(1) + fib(0)) + fib(1)) + fib(2)) + fib(3)
=> (((1 + fib(0)) + fib(1)) + fib(2)) + fib(3)
=> (((1 + 0) + fib(1)) + fib(2)) + fib(3)
=> ((1 + fib(1)) + fib(2)) + fib(3)
=> ((1 + 1) + fib(2)) + fib(3)
=> (2 + fib(2)) + fib(3)
=> (2 + (fib(1) + fib(0))) + fib(3)
=> (2 + (1 + fib(0))) + fib(3)
=> (2 + (1 + 0)) + fib(3)
=> (2 + 1) + fib(3)
=> 3 + fib(3)
=> 3 + (fib(2) + fib(1))
=> 3 + ((fib(1) + fib(0)) + fib(1))
=> 3 + ((1 + fib(0)) + fib(1))
=> 3 + ((1 + 0) + fib(1))
=> 3 + (1 + fib(1))
=> 3 + (1 + 1)
=> 3 + 2
=>
```

code : 7-1.py

```
1 def fib(n):
2     if n > 1:
3         return fib(n - 1) + fib(n - 2)
4     else:
5         return n
```

```
   fib(5)
=> fib(4) + fib(3)
=> (fib(3) + fib(2)) + fib(3)
=> ((fib(2) + fib(1)) + fib(2)) + fib(3)
=> (((fib(1) + fib(0)) + fib(1)) + fib(2)) + fib(3)
=> (((1 + fib(0)) + fib(1)) + fib(2)) + fib(3)
=> (((1 + 0) + fib(1)) + fib(2)) + fib(3)
=> ((1 + fib(1)) + fib(2)) + fib(3)
=> ((1 + 1) + fib(2)) + fib(3)
=> (2 + fib(2)) + fib(3)
=> (2 + (fib(1) + fib(0))) + fib(3)
=> (2 + (1 + fib(0))) + fib(3)
=> (2 + (1 + 0)) + fib(3)
=> (2 + 1) + fib(3)
=> 3 + fib(3)
=> 3 + (fib(2) + fib(1))
=> 3 + ((fib(1) + fib(0)) + fib(1))
=> 3 + ((1 + fib(0)) + fib(1))
=> 3 + ((1 + 0) + fib(1))
=> 3 + (1 + fib(1))
=> 3 + (1 + 1)
=> 3 + 2
=> 5
```

code : 7-1.py

```
1 def fib(n):
2     if n > 1:
3         return fib(n - 1) + fib(n - 2)
4     else:
5         return n
```

code : 7-2.py

```
1 def run_fib(n):
2     from time import perf_counter
3     start = perf_counter()
4     answer = fib(n)
5     finish = perf_counter()
6     print("fib(", n, ") => ", answer, sep="")
7     print(round(finish-start,4), "seconds")
```

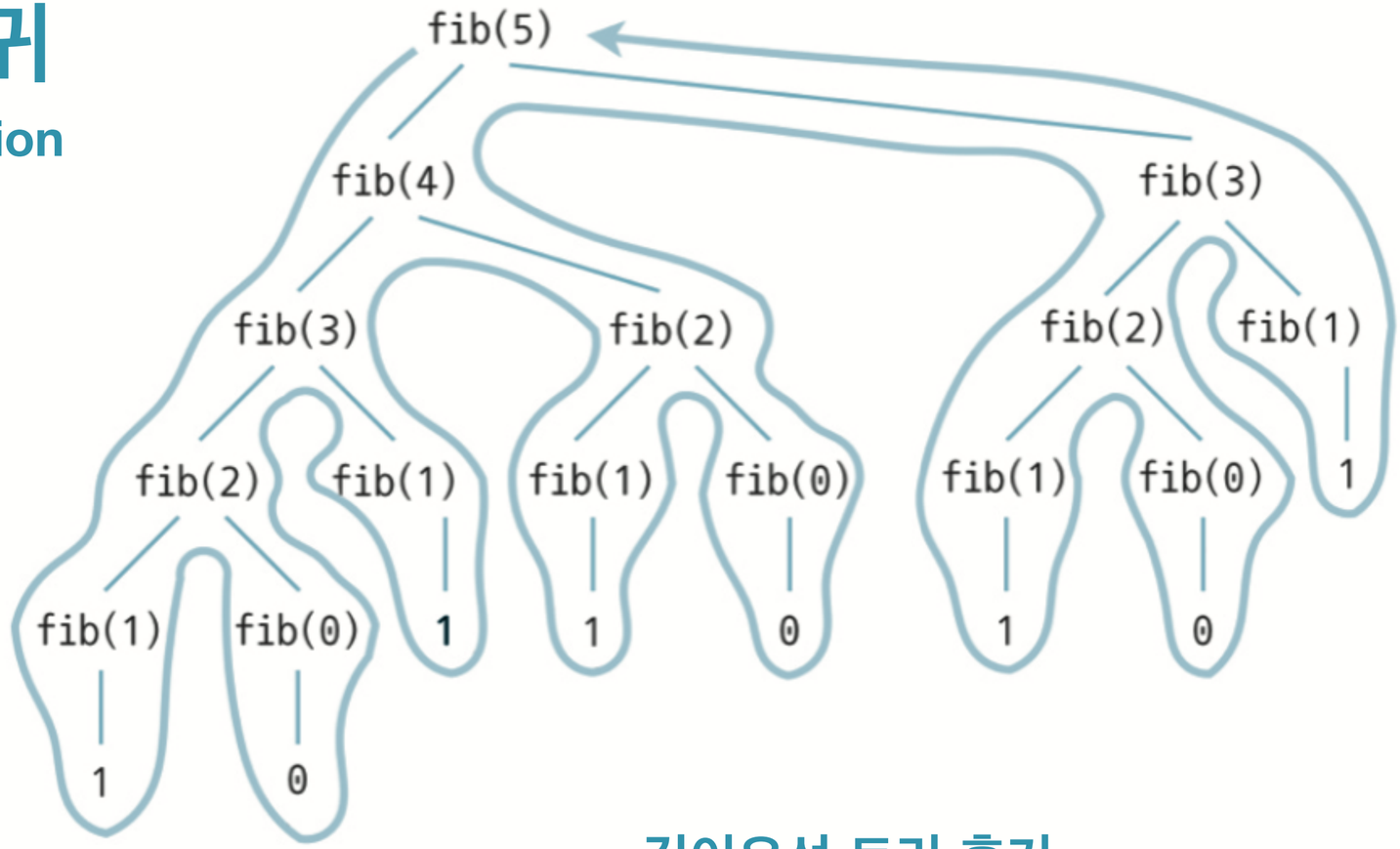
```
>>> run_fib(12)
fib(12) => 144
0.00006 seconds
>>> run_fib(24)
fib(24) => 46368
0.01905 seconds
>>> run_fib(36)
fib(36) => 14930352
5.77897 seconds
>>> run_fib(48)
fib(48) => 4807526976
1889.83474 seconds
```

```
>>> run_fib(60)
```

재귀 알고리즘의 계산 복잡도

트리 재귀

Tree Recursion



깊이우선 트리 훑기

Depth-first Tree Traversal

표채워풀기 알고리즘

Dynamic Programming

$$fib(n) = \begin{cases} fib(n-1) + fib(n-2) & \text{if } n > 1 \\ 1 & \text{if } n = 1 \\ 0 & \text{if } n = 0 \end{cases}$$

```
fib(0) = 0
fib(1) = 1
fib(2) =
```

표채워풀기 알고리즘

Dynamic Programming

$$fib(n) = \begin{cases} fib(n-1) + fib(n-2) & \text{if } n > 1 \\ 1 & \text{if } n = 1 \\ 0 & \text{if } n = 0 \end{cases}$$

```
fib(0) = 0
fib(1) = 1
fib(2) = fib(1) + fib(0) = 1 + 0 = 1
fib(3) =
```

표채워풀기 알고리즘

Dynamic Programming

$$fib(n) = \begin{cases} fib(n-1) + fib(n-2) & \text{if } n > 1 \\ 1 & \text{if } n = 1 \\ 0 & \text{if } n = 0 \end{cases}$$

```
fib(0) = 0
fib(1) = 1
fib(2) = fib(1) + fib(0) = 1 + 0 = 1
fib(3) = fib(2) + fib(1) = 1 + 1 = 2
fib(4) =
```

표채워풀기 알고리즘

Dynamic Programming

$$fib(n) = \begin{cases} fib(n-1) + fib(n-2) & \text{if } n > 1 \\ 1 & \text{if } n = 1 \\ 0 & \text{if } n = 0 \end{cases}$$

```
fib(0) = 0
fib(1) = 1
fib(2) = fib(1) + fib(0) = 1 + 0 = 1
fib(3) = fib(2) + fib(1) = 1 + 1 = 2
fib(4) = fib(3) + fib(2) = 2 + 1 = 3
fib(5) =
```

표채워풀기 알고리즘

Dynamic Programming

$$fib(n) = \begin{cases} fib(n-1) + fib(n-2) & \text{if } n > 1 \\ 1 & \text{if } n = 1 \\ 0 & \text{if } n = 0 \end{cases}$$

```
fib(0) = 0
fib(1) = 1
fib(2) = fib(1) + fib(0) = 1 + 0 = 1
fib(3) = fib(2) + fib(1) = 1 + 1 = 2
fib(4) = fib(3) + fib(2) = 2 + 1 = 3
fib(5) = fib(4) + fib(3) = 3 + 2 = 5
fib(6) =
```

표채워풀기 알고리즘

Dynamic Programming

$$fib(n) = \begin{cases} fib(n-1) + fib(n-2) & \text{if } n > 1 \\ 1 & \text{if } n = 1 \\ 0 & \text{if } n = 0 \end{cases}$$

```
fib(0) = 0
fib(1) = 1
fib(2) = fib(1) + fib(0) = 1 + 0 = 1
fib(3) = fib(2) + fib(1) = 1 + 1 = 2
fib(4) = fib(3) + fib(2) = 2 + 1 = 3
fib(5) = fib(4) + fib(3) = 3 + 2 = 5
fib(6) = fib(5) + fib(4) = 5 + 3 = 8
fib(7) =
```

표채워풀기 알고리즘

Dynamic Programming

$$fib(n) = \begin{cases} fib(n-1) + fib(n-2) & \text{if } n > 1 \\ 1 & \text{if } n = 1 \\ 0 & \text{if } n = 0 \end{cases}$$

```
fib(0) = 0
fib(1) = 1
fib(2) = fib(1) + fib(0) = 1 + 0 = 1
fib(3) = fib(2) + fib(1) = 1 + 1 = 2
fib(4) = fib(3) + fib(2) = 2 + 1 = 3
fib(5) = fib(4) + fib(3) = 3 + 2 = 5
fib(6) = fib(5) + fib(4) = 5 + 3 = 8
fib(7) = fib(6) + fib(5) = 8 + 5 = 13
fib(8) =
```

표채워풀기 알고리즘

Dynamic Programming

$$fib(n) = \begin{cases} fib(n-1) + fib(n-2) & \text{if } n > 1 \\ 1 & \text{if } n = 1 \\ 0 & \text{if } n = 0 \end{cases}$$

```
fib(0) = 0
fib(1) = 1
fib(2) = fib(1) + fib(0) = 1 + 0 = 1
fib(3) = fib(2) + fib(1) = 1 + 1 = 2
fib(4) = fib(3) + fib(2) = 2 + 1 = 3
fib(5) = fib(4) + fib(3) = 3 + 2 = 5
fib(6) = fib(5) + fib(4) = 5 + 3 = 8
fib(7) = fib(6) + fib(5) = 8 + 5 = 13
fib(8) = fib(7) + fib(6) = 13 + 8 = 21
fib(9) =
```

표채워풀기 알고리즘

Dynamic Programming

$$fib(n) = \begin{cases} fib(n-1) + fib(n-2) & \text{if } n > 1 \\ 1 & \text{if } n = 1 \\ 0 & \text{if } n = 0 \end{cases}$$

```
fib(0) = 0
fib(1) = 1
fib(2) = fib(1) + fib(0) = 1 + 0 = 1
fib(3) = fib(2) + fib(1) = 1 + 1 = 2
fib(4) = fib(3) + fib(2) = 2 + 1 = 3
fib(5) = fib(4) + fib(3) = 3 + 2 = 5
fib(6) = fib(5) + fib(4) = 5 + 3 = 8
fib(7) = fib(6) + fib(5) = 8 + 5 = 13
fib(8) = fib(7) + fib(6) = 13 + 8 = 21
fib(9) = fib(8) + fib(7) = 21 + 13 = 34
fib(10) =
```

표채워풀기 알고리즘

Dynamic Programming

$$fib(n) = \begin{cases} fib(n-1) + fib(n-2) & \text{if } n > 1 \\ 1 & \text{if } n = 1 \\ 0 & \text{if } n = 0 \end{cases}$$

```
fib(0) = 0
fib(1) = 1
fib(2) = fib(1) + fib(0) = 1 + 0 = 1
fib(3) = fib(2) + fib(1) = 1 + 1 = 2
fib(4) = fib(3) + fib(2) = 2 + 1 = 3
fib(5) = fib(4) + fib(3) = 3 + 2 = 5
fib(6) = fib(5) + fib(4) = 5 + 3 = 8
fib(7) = fib(6) + fib(5) = 8 + 5 = 13
fib(8) = fib(7) + fib(6) = 13 + 8 = 21
fib(9) = fib(8) + fib(7) = 21 + 13 = 34
fib(10) = fib(9) + fib(8) = 34 + 21 = 55
fib(11) =
```

표채워풀기 알고리즘

Dynamic Programming

$$fib(n) = \begin{cases} fib(n-1) + fib(n-2) & \text{if } n > 1 \\ 1 & \text{if } n = 1 \\ 0 & \text{if } n = 0 \end{cases}$$

```
fib(0) = 0
fib(1) = 1
fib(2) = fib(1) + fib(0) = 1 + 0 = 1
fib(3) = fib(2) + fib(1) = 1 + 1 = 2
fib(4) = fib(3) + fib(2) = 2 + 1 = 3
fib(5) = fib(4) + fib(3) = 3 + 2 = 5
fib(6) = fib(5) + fib(4) = 5 + 3 = 8
fib(7) = fib(6) + fib(5) = 8 + 5 = 13
fib(8) = fib(7) + fib(6) = 13 + 8 = 21
fib(9) = fib(8) + fib(7) = 21 + 13 = 34
fib(10) = fib(9) + fib(8) = 34 + 21 = 55
fib(11) = fib(10) + fib(9) = 55 + 34 = 89
fib(12) =
```

표채워풀기 알고리즘

Dynamic Programming

$$fib(n) = \begin{cases} fib(n-1) + fib(n-2) & \text{if } n > 1 \\ 1 & \text{if } n = 1 \\ 0 & \text{if } n = 0 \end{cases}$$

```
fib(0) = 0
fib(1) = 1
fib(2) = fib(1) + fib(0) = 1 + 0 = 1
fib(3) = fib(2) + fib(1) = 1 + 1 = 2
fib(4) = fib(3) + fib(2) = 2 + 1 = 3
fib(5) = fib(4) + fib(3) = 3 + 2 = 5
fib(6) = fib(5) + fib(4) = 5 + 3 = 8
fib(7) = fib(6) + fib(5) = 8 + 5 = 13
fib(8) = fib(7) + fib(6) = 13 + 8 = 21
fib(9) = fib(8) + fib(7) = 21 + 13 = 34
fib(10) = fib(9) + fib(8) = 34 + 21 = 55
fib(11) = fib(10) + fib(9) = 55 + 34 = 89
fib(12) = fib(11) + fib(10) = 89 + 55 = 144
```

표채워풀기 알고리즘

Dynamic Programming

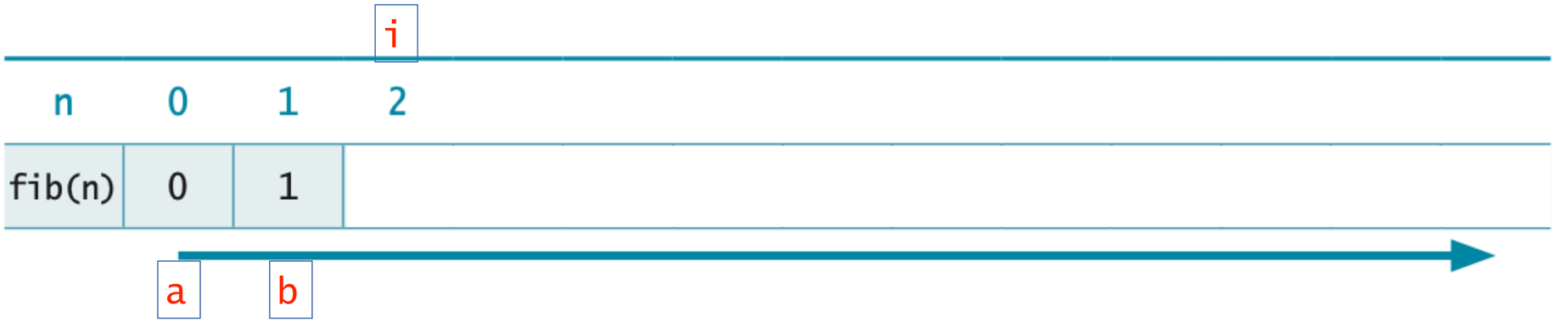
$$fib(n) = \begin{cases} fib(n-1) + fib(n-2) & \text{if } n > 1 \\ 1 & \text{if } n = 1 \\ 0 & \text{if } n = 0 \end{cases}$$

```
fib(0) = 0
fib(1) = 1
fib(2) = fib(1) + fib(0) = 1 + 0 = 1
fib(3) = fib(2) + fib(1) = 1 + 1 = 2
fib(4) = fib(3) + fib(2) = 2 + 1 = 3
fib(5) = fib(4) + fib(3) = 3 + 2 = 5
fib(6) = fib(5) + fib(4) = 5 + 3 = 8
fib(7) = fib(6) + fib(5) = 8 + 5 = 13
fib(8) = fib(7) + fib(6) = 13 + 8 = 21
fib(9) = fib(8) + fib(7) = 21 + 13 = 34
fib(10) = fib(9) + fib(8) = 34 + 21 = 55
fib(11) = fib(10) + fib(9) = 55 + 34 = 89
fib(12) = fib(11) + fib(10) = 89 + 55 = 144
```

n	0	1	2	3	4	5	6	7	8	9	10	11	12
fib(n)	0	1	1	2	3	5	8	13	21	34	55	89	144

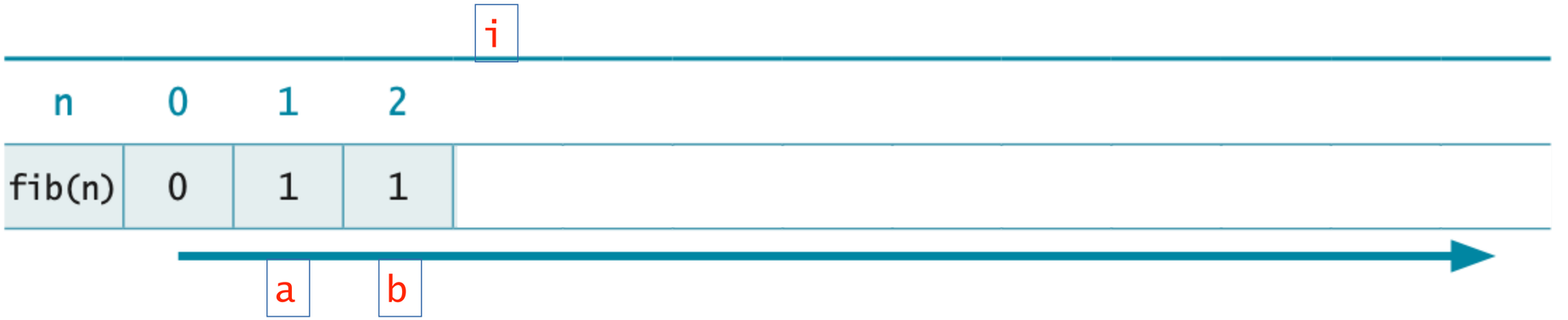
code : 7-3.py

```
1  def fib(n):
2      if n > 1:
3          i = 2
4          a, b = 0, 1
5          while i <= n:
6              a, b = b, a + b
7              i += 1
8          return b
9      else:
10         return n
```



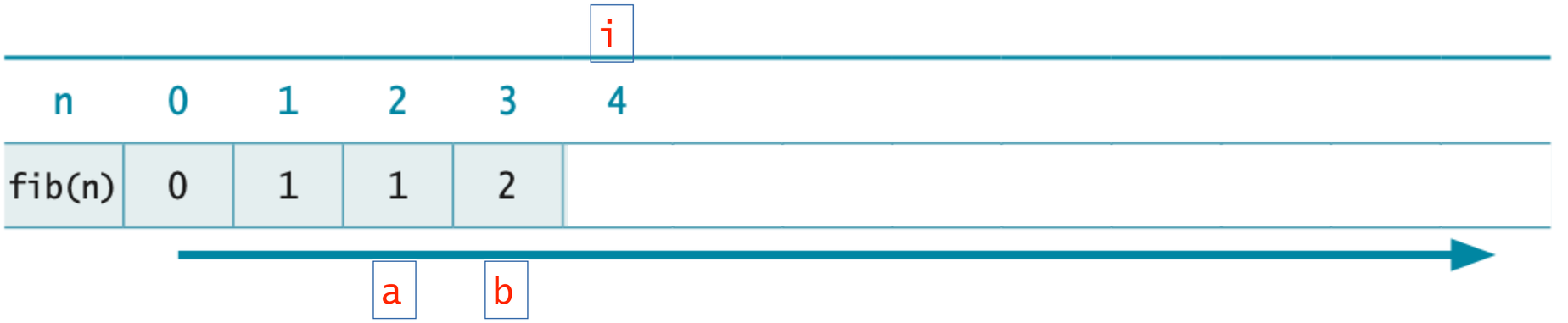
code : 7-3.py

```
1  def fib(n):
2      if n > 1:
3          i = 2
4          a, b = 0, 1
5          while i <= n:
6              a, b = b, a + b
7              i += 1
8          return b
9      else:
10         return n
```



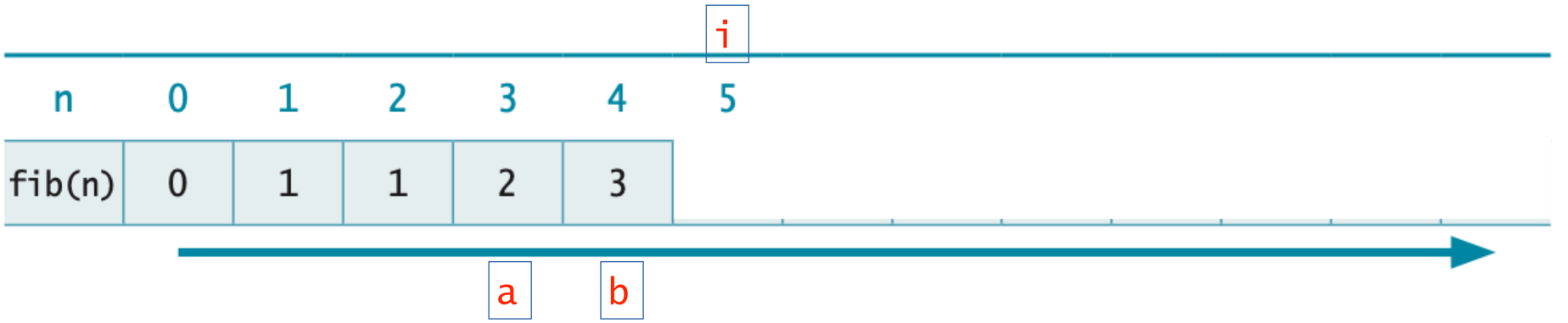
code : 7-3.py

```
1  def fib(n):
2      if n > 1:
3          i = 2
4          a, b = 0, 1
5          while i <= n:
6              a, b = b, a + b
7              i += 1
8          return b
9      else:
10         return n
```



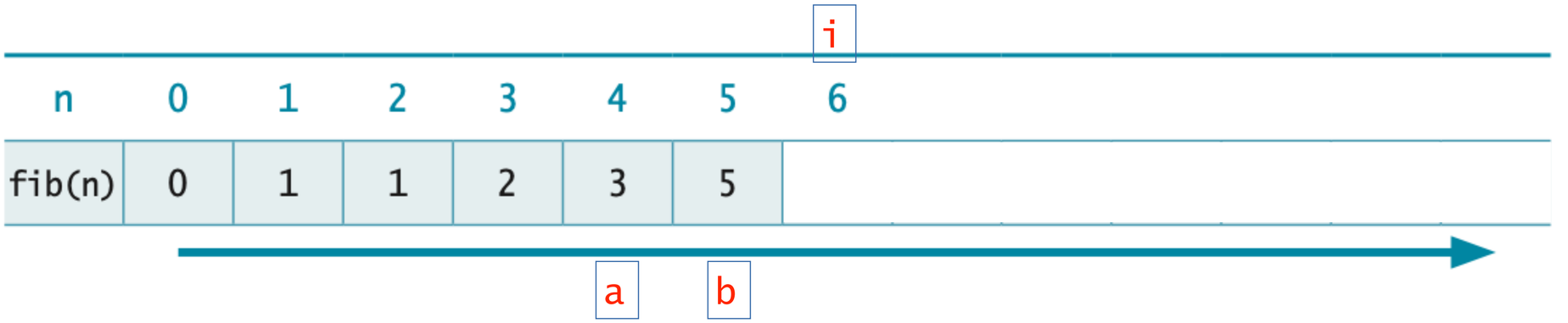
code : 7-3.py

```
1  def fib(n):
2      if n > 1:
3          i = 2
4          a, b = 0, 1
5          while i <= n:
6              a, b = b, a + b
7              i += 1
8          return b
9      else:
10         return n
```



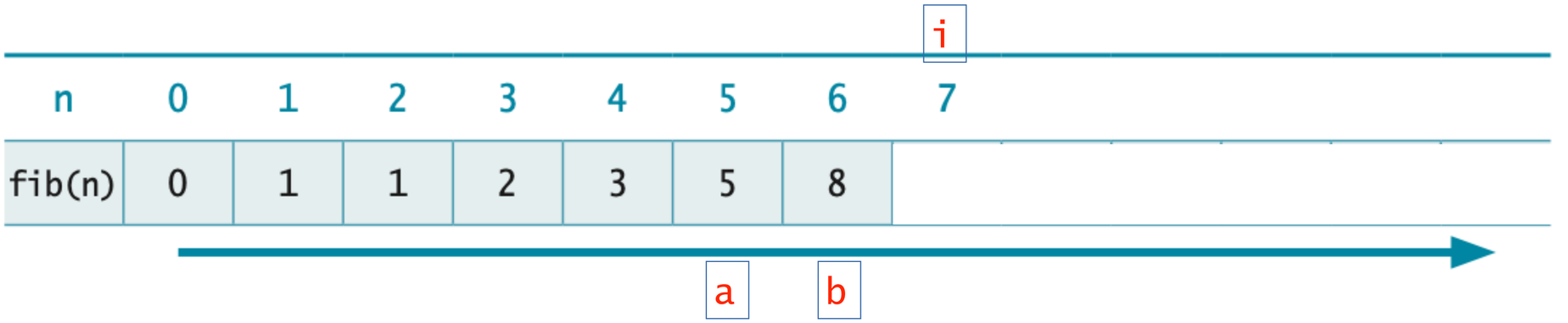
code : 7-3.py

```
1  def fib(n):
2      if n > 1:
3          i = 2
4          a, b = 0, 1
5          while i <= n:
6              a, b = b, a + b
7              i += 1
8          return b
9      else:
10         return n
```



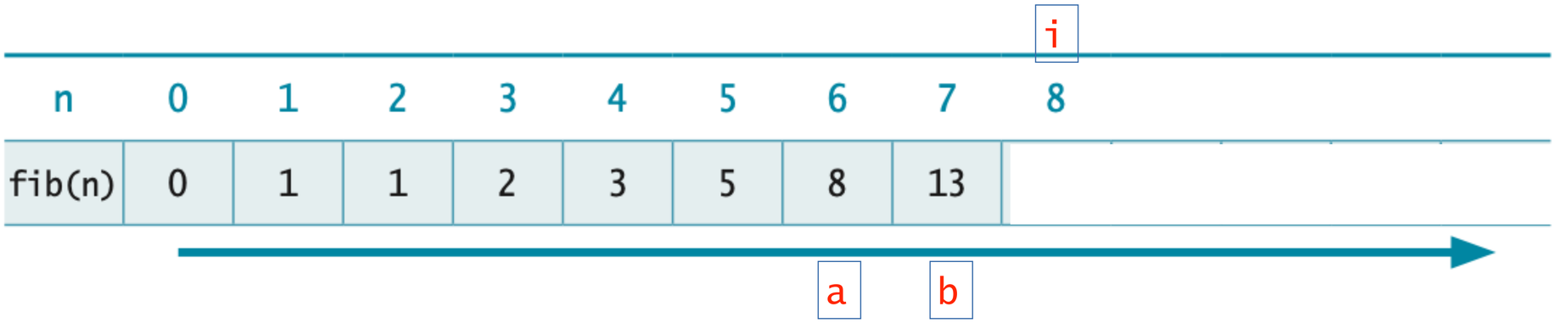
code : 7-3.py

```
1  def fib(n):
2      if n > 1:
3          i = 2
4          a, b = 0, 1
5          while i <= n:
6              a, b = b, a + b
7              i += 1
8          return b
9      else:
10         return n
```



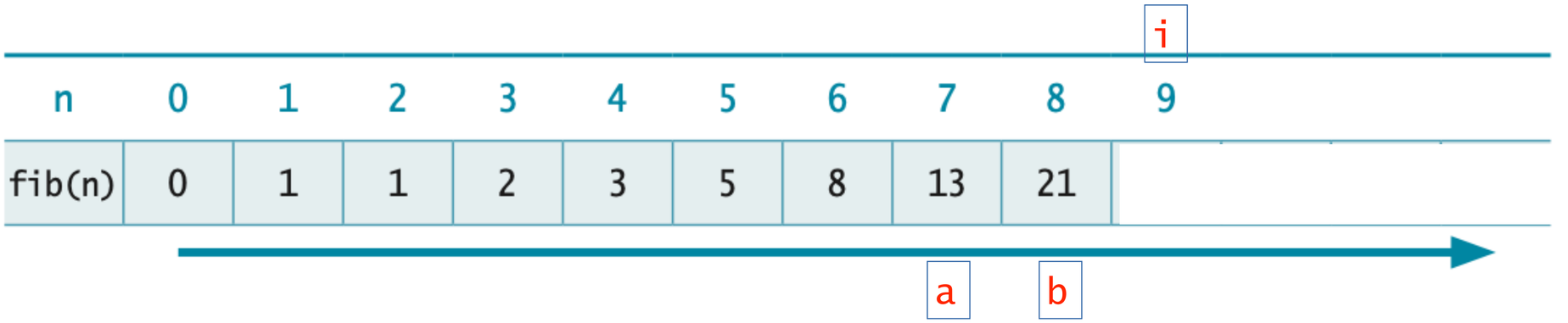
code : 7-3.py

```
1  def fib(n):
2      if n > 1:
3          i = 2
4          a, b = 0, 1
5          while i <= n:
6              a, b = b, a + b
7              i += 1
8          return b
9      else:
10         return n
```



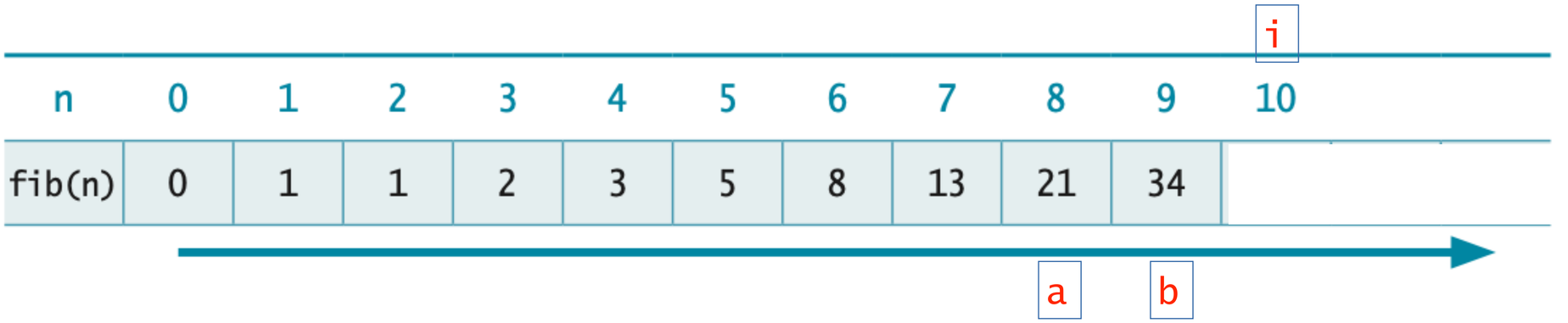
code : 7-3.py

```
1  def fib(n):
2      if n > 1:
3          i = 2
4          a, b = 0, 1
5          while i <= n:
6              a, b = b, a + b
7              i += 1
8          return b
9      else:
10         return n
```



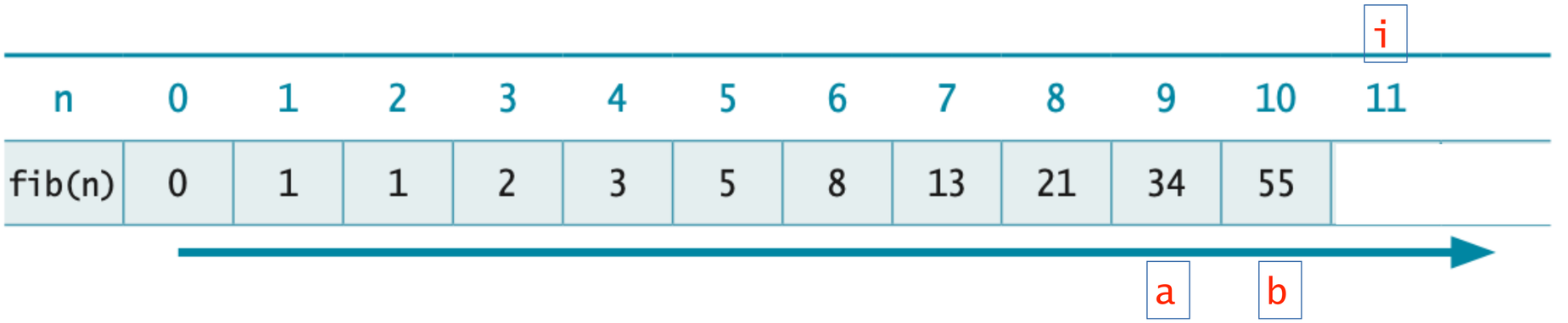
code : 7-3.py

```
1  def fib(n):
2      if n > 1:
3          i = 2
4          a, b = 0, 1
5          while i <= n:
6              a, b = b, a + b
7              i += 1
8          return b
9      else:
10         return n
```



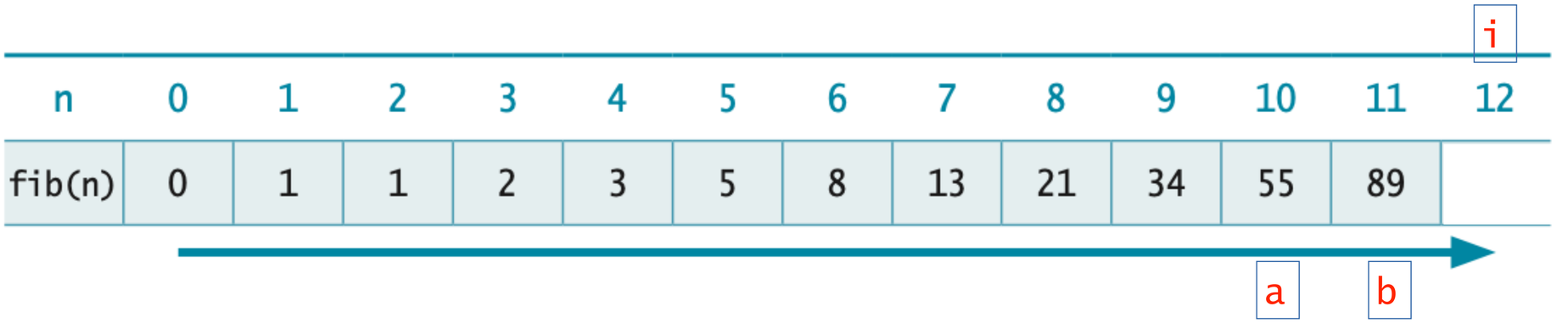
code : 7-3.py

```
1  def fib(n):
2      if n > 1:
3          i = 2
4          a, b = 0, 1
5          while i <= n:
6              a, b = b, a + b
7              i += 1
8          return b
9      else:
10         return n
```



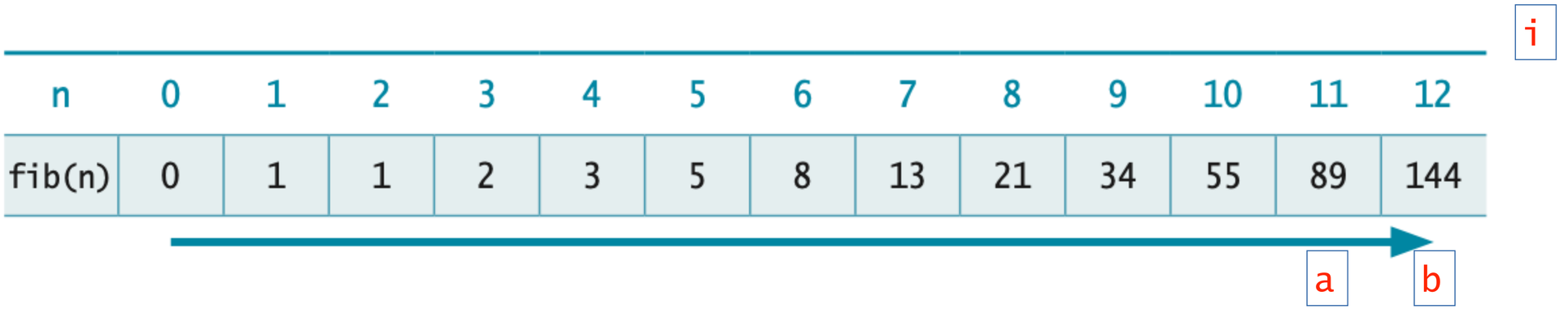
code : 7-3.py

```
1  def fib(n):
2      if n > 1:
3          i = 2
4          a, b = 0, 1
5          while i <= n:
6              a, b = b, a + b
7              i += 1
8          return b
9      else:
10         return n
```



code : 7-3.py

```
1  def fib(n):
2      if n > 1:
3          i = 2
4          a, b = 0, 1
5          while i <= n:
6              a, b = b, a + b
7              i += 1
8          return b
9      else:
10         return n
```



표채워풀기 알고리즘의 계산 복잡도

```
>>> run_fib(60)
fib(60) => 1548008755920
1e-05 seconds
>>> run_fib(120)
fib(120) => 5358359254990966640871840
2e-05 seconds
>>> run_fib(240)
fib(240) => 642020148637230941269017774288731118023075486236 80
3e-05 seconds
>>> run_fib(480)
fib(480) => 92168457176568747129804505627262024155673605659807947771
11390
85033164481367485698164696022619228736 0
7e-05 seconds
```

code : 7-3.py

```
1  def fib(n):
2      if n > 1:
3          i = 2
4          a, b = 0, 1
5          while i <= n:
6              a, b = b, a + b
7              i += 1
8          return b
9      else:
10         return n
```

while
루프

code : 7-4.py

```
1  def fib(n):
2      if n > 1:
3          a, b = 0, 1
4          for _ in range(2, n+1):
5              a, b = b, a + b
6          return b
7      else:
8          return n
```

for
루프

code : 7-3.py

```
def fib(n):
    if n > 1:
        i = 2
        a, b = 0, 1
        while i <= n:
            a, b = b, a
            i += 1
        return b
    else:
        return n
```

code : 7-5.py

```
def fibseq(n):
    fibs = [0, 1]
    for i in range(2, n+1):
        fibs.append(fibs[i-1] + fibs[i-2])
    return fibs
```

```
>>> fibseq(12)
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144]
```

code : 7-6.py

```
def fib(n):
    return fibseq(n)[-1]
```

CHAPTER 7

표채워풀기

조합

Combination

서로 다른 n개의 원소에서
순서에 상관없이
r개를 뽑는 가지수

$${}_nC_r \quad \binom{n}{r}$$

조합

Combination

서로 다른 6개의 원소에서
순서에 상관없이
2개를 뽑는 가지수

$${}_6C_2$$

a b c d e f

a b
a c b c
a d b d c d
a e b e c e d e
a f b f c f d f e f

조합

Combination

서로 다른 6개의 원소에서
순서에 상관없이
2개를 뽑는 가지수

$${}_6C_2$$

a b c d e f

a b
a c b c
a d b d c d
a e b e c e d e
a f b f c f c f e f

서로 다른 6개의 원소에서
순서에 상관없이
4개를 뽑는 가지수

$${}_6C_4$$

a b c d e f

c d e f
b d e f a d e f
b c e f a c e f a b e f
b c d f a c d f a b d f a b c f
b c d e a c d e a b d e a b d e a b c d

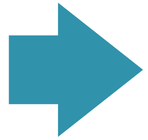
조합

Combination

$$ {}_nC_r = \begin{cases} {}_{n-1}C_{r-1} + {}_{n-1}C_r & \text{if } r \neq 0 \text{ and } r \neq n \\ 1 & \text{if } r = 0 \\ 1 & \text{if } r = n \end{cases} $$

code : 7-7.py

```
1  def comb(n,r):
2      if r != 0 and r != n:
3          return comb(n-1,r-1) + comb(n-1,r)
4      else:
5          return 1
```



>>>>>>>>>>> 제어 구조의 설계 원리를 중심으로 배우는 >>>>>>>>>>>

프로그래밍의 정석

파이썬

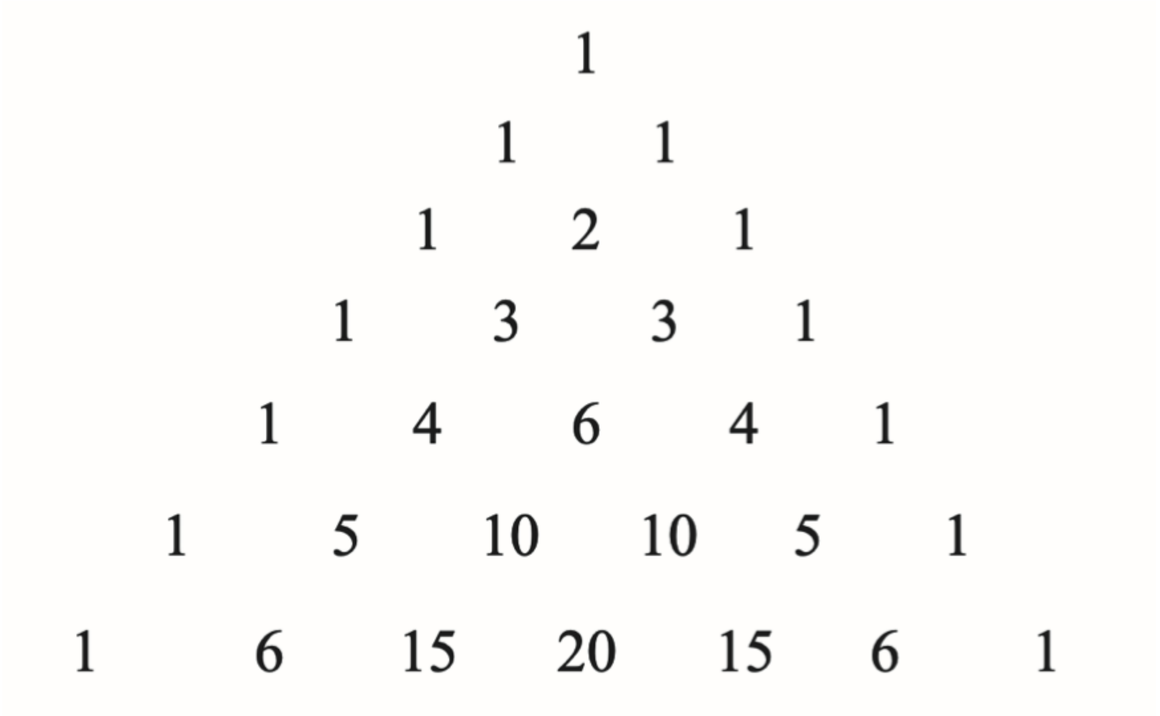
도경구 지음

실습 7.1 comb 함수의 실행 시간 측정 함수

comb 함수 호출의 실행 시간을 측정하는 함수 run_comb를 7.1절의 run_fib 함수와 같은 요령으로 작성하여, 위에서 제시한 호출 사례의 실제 실행 시간을 측정해 보자.

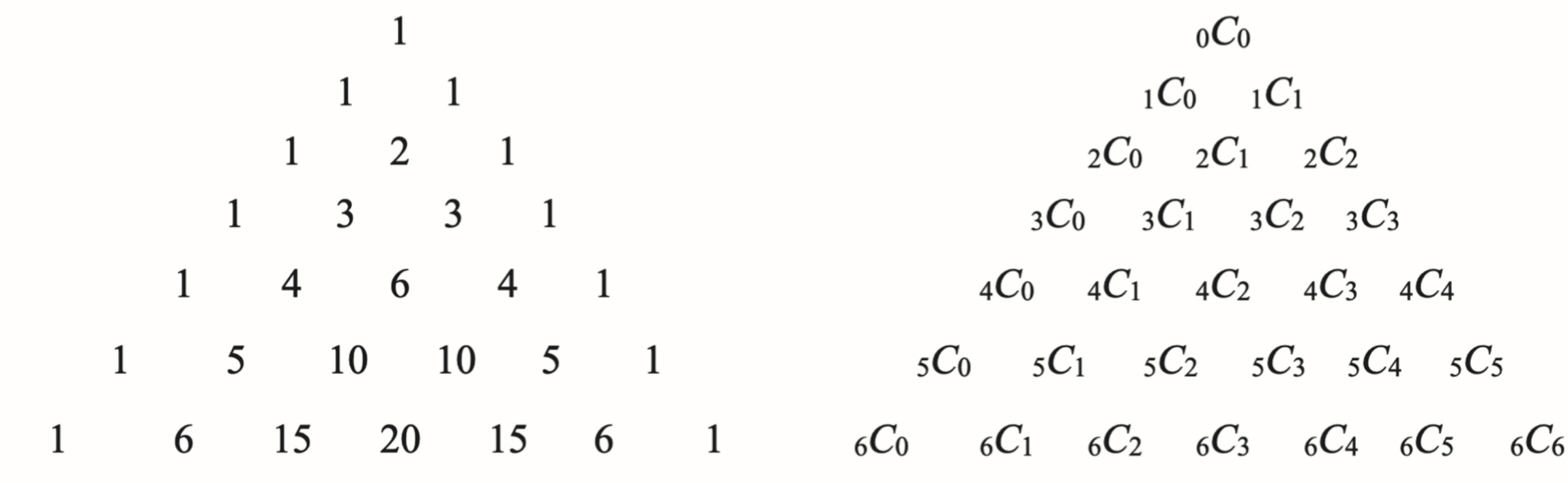
파스칼의 삼각형

블레즈 파스칼(Blaise Pascal, 1623~1662년)



파스칼의 삼각형

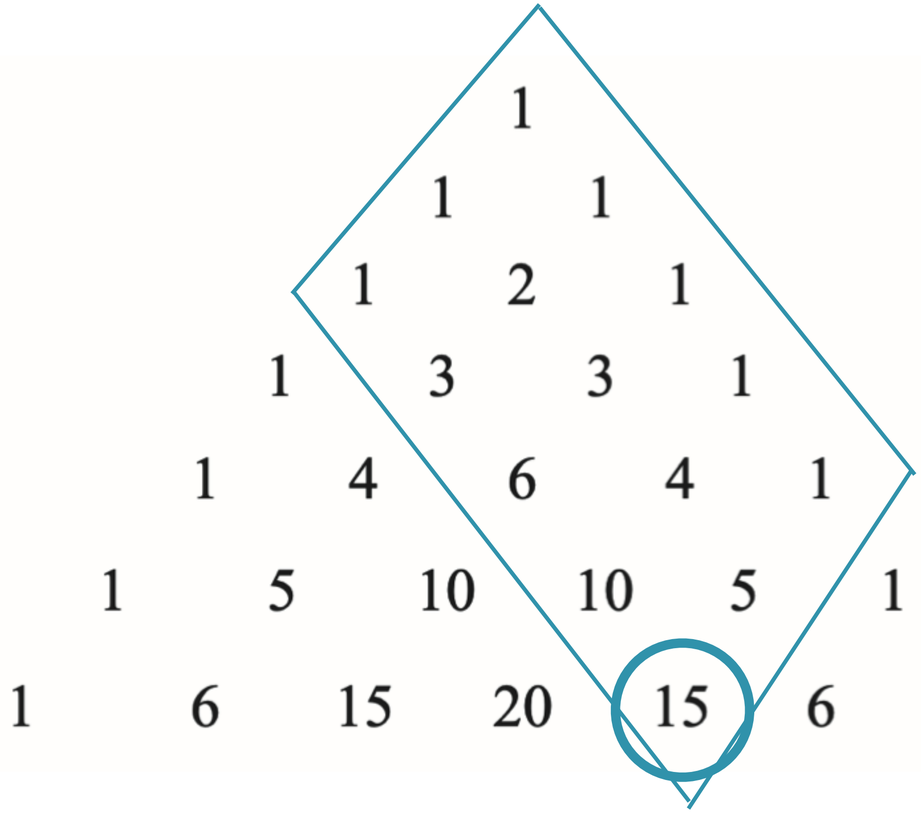
블레즈 파스칼(Blaise Pascal, 1623~1662년)

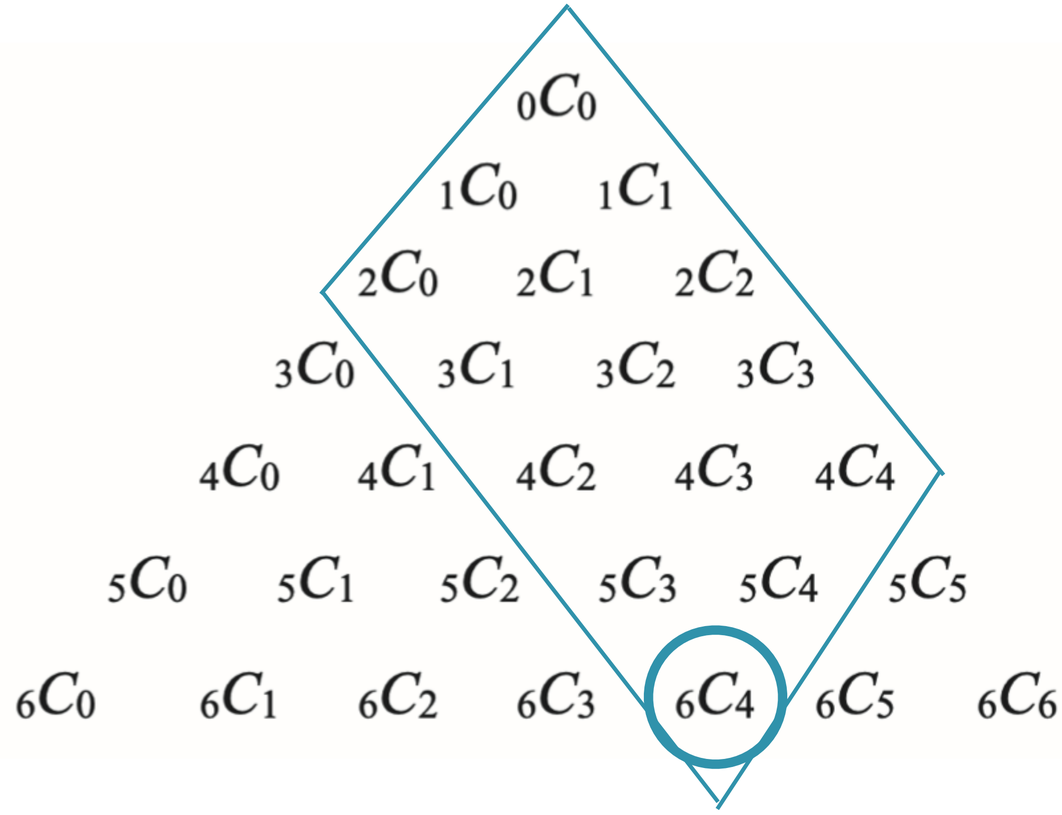


파스칼의 삼각형

블레즈 파스칼(Blaise Pascal, 1623~1662년)

표채워풀기 알고리즘

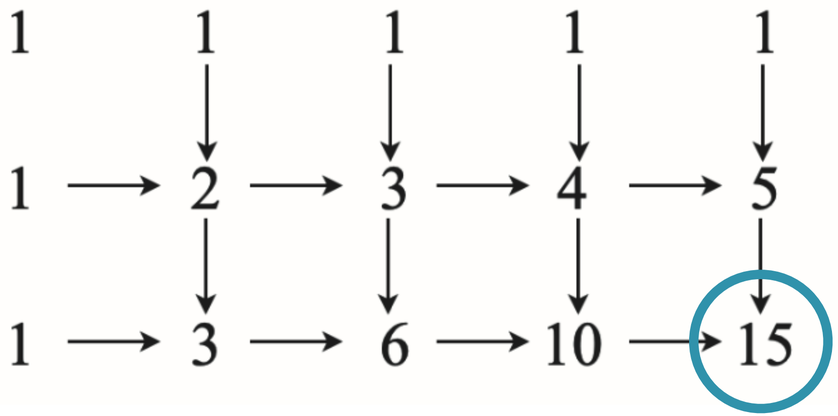


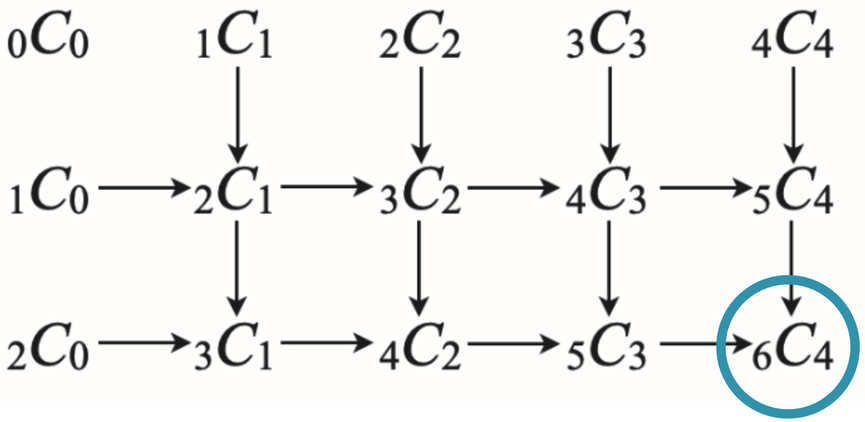


파스칼의 삼각형

블레즈 파스칼(Blaise Pascal, 1623~1662년)

표채워풀기 알고리즘

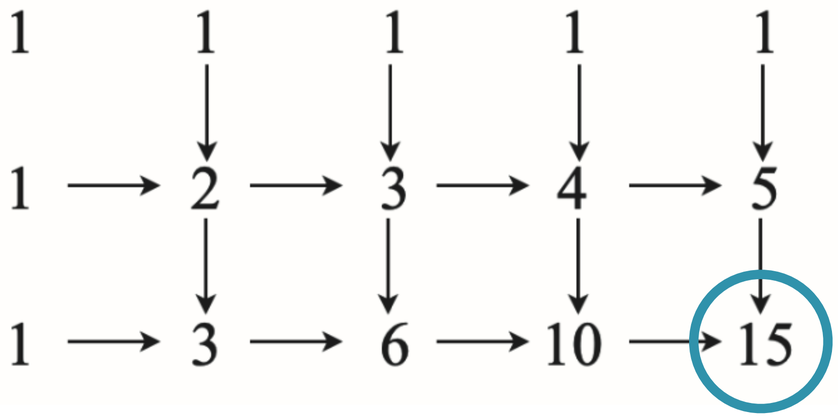


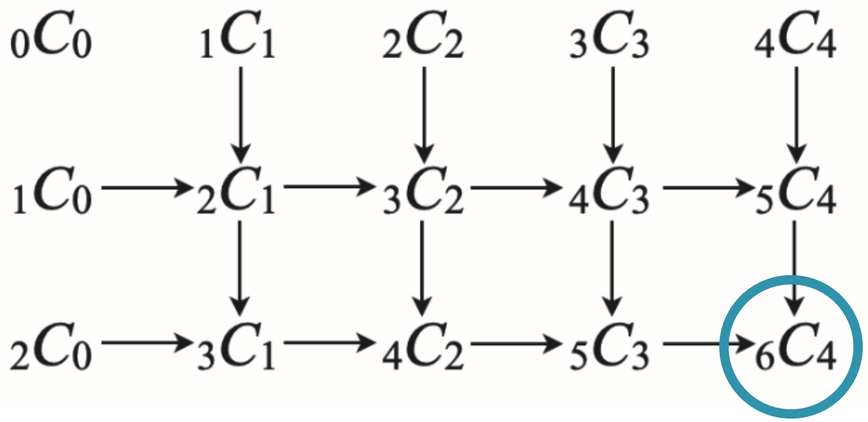


파스칼의 삼각형

블레즈 파스칼(Blaise Pascal, 1623~1662년)

표채워풀기 알고리즘





행렬
Matrix

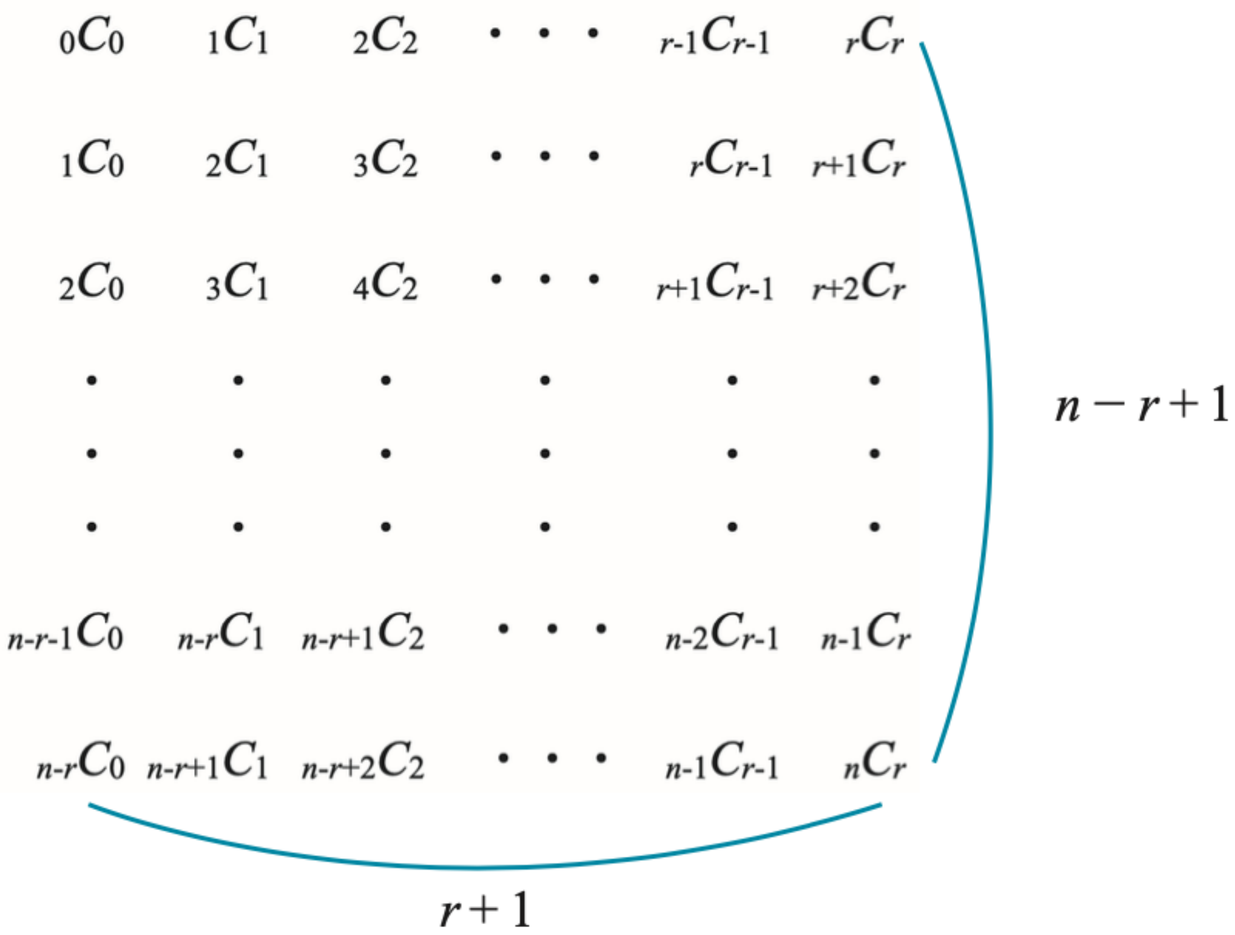
행 = 가로줄 = row
렬 = 세로줄 = column

row x column
3 x 5

파스칼의 삼각형

블레즈 파스칼(Blaise Pascal, 1623~1662년)

표채워풀기 알고리즘



파스칼의 삼각형

표채워풀기 구현

$_6C_4$

1	1	1	1	1
1	2	3	4	5
1	3	6	10	15

파스칼의 삼각형

표채워풀기 구현

$_6C_4$

```
1  1  1  1  1
1  2  3  4  5
1  3  6  10 15
```

```
[[1, 1, 1, 1, 1],
 [1, 2, 3, 4, 5],
 [1, 3, 6, 10, 15]]
```

중첩 리스트
Nested List

파스칼의 삼각형

표채워풀기 구현

$_6C_4$

```
1  1  1  1  1
1  2  3  4  5
1  3  6  10 15
```

```
[[1, 1, 1, 1, 1],
 [1, 2, 3, 4, 5],
 [1, 3, 6, 10, 15]]
```

중첩 리스트
Nested List

```
[[1, 1, 1, 1, 1], [1, 2, 3, 4, 5], [1, 3, 6, 10, 15]]
```

파스칼의 삼각형

표채워풀기 구현

$_6C_4$

1 1 1 1 1
1 2 3 4 5
1 3 6 10 15

```
[[1, 1, 1, 1, 1],
 [1, 2, 3, 4, 5],
 [1, 3, 6, 10, 15]]
```

중첩 리스트
Nested List

```
[[1, 1, 1, 1, 1], [1, 2, 3, 4, 5], [1, 3, 6, 10, 15]]
```

```
   [[1, 1, 1, 1, 1], [1], [1]]
=>
```

파스칼의 삼각형

표채워풀기 구현

$_6C_4$

```
1  1  1  1  1
1  2  3  4  5
1  3  6  10 15
```

```
[[1, 1, 1, 1, 1],
 [1, 2, 3, 4, 5],
 [1, 3, 6, 10, 15]]
```

중첩 리스트
Nested List

```
[[1, 1, 1, 1, 1], [1, 2, 3, 4, 5], [1, 3, 6, 10, 15]]
```

```
   [[1, 1, 1, 1, 1], [1], [1]]
=> [[1, 1, 1, 1, 1], [1, 2], [1]]
=>
```

파스칼의 삼각형

표채워풀기 구현

$_6C_4$

```
1  1  1  1  1
1  2  3  4  5
1  3  6  10 15
```

```
[[1, 1, 1, 1, 1],
 [1, 2, 3, 4, 5],
 [1, 3, 6, 10, 15]]
```

중첩 리스트
Nested List

```
[[1, 1, 1, 1, 1], [1, 2, 3, 4, 5], [1, 3, 6, 10, 15]]
```

```
   [[1, 1, 1, 1, 1], [1], [1]]
=> [[1, 1, 1, 1, 1], [1, 2], [1]]
=>
```

파스칼의 삼각형

표채워풀기 구현

$_6C_4$

```
1  1  1  1  1
1  2  3  4  5
1  3  6  10 15
```

```
[[1, 1, 1, 1, 1],
 [1, 2, 3, 4, 5],
 [1, 3, 6, 10, 15]]
```

중첩 리스트
Nested List

```
[[1, 1, 1, 1, 1], [1, 2, 3, 4, 5], [1, 3, 6, 10, 15]]
```

```
   [[1, 1, 1, 1, 1], [1], [1]]
=> [[1, 1, 1, 1, 1], [1, 2], [1]]
=> [[1, 1, 1, 1, 1], [1, 2, 3], [1]]
=>
```

파스칼의 삼각형

표채워풀기 구현

$_6C_4$

```
1  1  1  1  1
1  2  3  4  5
1  3  6  10 15
```

```
[[1, 1, 1, 1, 1],
 [1, 2, 3, 4, 5],
 [1, 3, 6, 10, 15]]
```

중첩 리스트
Nested List

```
[[1, 1, 1, 1, 1], [1, 2, 3, 4, 5], [1, 3, 6, 10, 15]]
```

```
   [[1, 1, 1, 1, 1], [1], [1]]
=> [[1, 1, 1, 1, 1], [1, 2], [1]]
=> [[1, 1, 1, 1, 1], [1, 2, 3], [1]]
=>
```

파스칼의 삼각형

표채워풀기 구현

$_6C_4$

```
1  1  1  1  1
1  2  3  4  5
1  3  6  10 15
```

```
[[1, 1, 1, 1, 1],
 [1, 2, 3, 4, 5],
 [1, 3, 6, 10, 15]]
```

중첩 리스트
Nested List

```
[[1, 1, 1, 1, 1], [1, 2, 3, 4, 5], [1, 3, 6, 10, 15]]
```

```
   [[1, 1, 1, 1, 1], [1], [1]]
=> [[1, 1, 1, 1, 1], [1, 2], [1]]
=> [[1, 1, 1, 1, 1], [1, 2, 3], [1]]
=> [[1, 1, 1, 1, 1], [1, 2, 3, 4], [1]]
=>
```

파스칼의 삼각형

표채워풀기 구현

$_6C_4$

```
1  1  1  1  1
1  2  3  4  5
1  3  6  10 15
```

```
[[1, 1, 1, 1, 1],
 [1, 2, 3, 4, 5],
 [1, 3, 6, 10, 15]]
```

중첩 리스트
Nested List

```
[[1, 1, 1, 1, 1], [1, 2, 3, 4, 5], [1, 3, 6, 10, 15]]
```

```
   [[1, 1, 1, 1, 1], [1], [1]]
=> [[1, 1, 1, 1, 1], [1, 2], [1]]
=> [[1, 1, 1, 1, 1], [1, 2, 3], [1]]
=> [[1, 1, 1, 1, 1], [1, 2, 3, 4], [1]]
=>
```

파스칼의 삼각형

표채워풀기 구현

$_6C_4$

```
1  1  1  1  1
1  2  3  4  5
1  3  6  10 15
```

```
[[1, 1, 1, 1, 1],
 [1, 2, 3, 4, 5],
 [1, 3, 6, 10, 15]]
```

중첩 리스트
Nested List

```
[[1, 1, 1, 1, 1], [1, 2, 3, 4, 5], [1, 3, 6, 10, 15]]
```

```
   [[1, 1, 1, 1, 1], [1], [1]]
=> [[1, 1, 1, 1, 1], [1, 2], [1]]
=> [[1, 1, 1, 1, 1], [1, 2, 3], [1]]
=> [[1, 1, 1, 1, 1], [1, 2, 3, 4], [1]]
=> [[1, 1, 1, 1, 1], [1, 2, 3, 4, 5], [1]]
=>
```

파스칼의 삼각형

표채워풀기 구현

$_6C_4$

```
1  1  1  1  1
1  2  3  4  5
1  3  6  10 15
```

```
[[1, 1, 1, 1, 1],
 [1, 2, 3, 4, 5],
 [1, 3, 6, 10, 15]]
```

중첩 리스트
Nested List

```
[[1, 1, 1, 1, 1], [1, 2, 3, 4, 5], [1, 3, 6, 10, 15]]
```

```
   [[1, 1, 1, 1, 1], [1], [1]]
=> [[1, 1, 1, 1, 1], [1, 2], [1]]
=> [[1, 1, 1, 1, 1], [1, 2, 3], [1]]
=> [[1, 1, 1, 1, 1], [1, 2, 3, 4], [1]]
=> [[1, 1, 1, 1, 1], [1, 2, 3, 4, 5], [1]]
=>
```

파스칼의 삼각형

표채워풀기 구현

$_6C_4$

```
1  1  1  1  1
1  2  3  4  5
1  3  6  10 15
```

```
[[1, 1, 1, 1, 1],
 [1, 2, 3, 4, 5],
 [1, 3, 6, 10, 15]]
```

중첩 리스트
Nested List

```
[[1, 1, 1, 1, 1], [1, 2, 3, 4, 5], [1, 3, 6, 10, 15]]
```

```
   [[1, 1, 1, 1, 1], [1], [1]]
=> [[1, 1, 1, 1, 1], [1, 2], [1]]
=> [[1, 1, 1, 1, 1], [1, 2, 3], [1]]
=> [[1, 1, 1, 1, 1], [1, 2, 3, 4], [1]]
=> [[1, 1, 1, 1, 1], [1, 2, 3, 4, 5], [1]]
=> [[1, 1, 1, 1, 1], [1, 2, 3, 4, 5], [1, 3]]
=>
```

파스칼의 삼각형

표채워풀기 구현

$_6C_4$

```
1  1  1  1  1
1  2  3  4  5
1  3  6  10 15
```

```
[[1, 1, 1, 1, 1],
 [1, 2, 3, 4, 5],
 [1, 3, 6, 10, 15]]
```

중첩 리스트
Nested List

```
[[1, 1, 1, 1, 1], [1, 2, 3, 4, 5], [1, 3, 6, 10, 15]]
```

```
   [[1, 1, 1, 1, 1], [1], [1]]
=> [[1, 1, 1, 1, 1], [1, 2], [1]]
=> [[1, 1, 1, 1, 1], [1, 2, 3], [1]]
=> [[1, 1, 1, 1, 1], [1, 2, 3, 4], [1]]
=> [[1, 1, 1, 1, 1], [1, 2, 3, 4, 5], [1]]
=> [[1, 1, 1, 1, 1], [1, 2, 3, 4, 5], [1, 3]]
=>
```

파스칼의 삼각형

표채워풀기 구현

$_6C_4$

```
1  1  1  1  1
1  2  3  4  5
1  3  6  10 15
```

```
[[1, 1, 1, 1, 1],
 [1, 2, 3, 4, 5],
 [1, 3, 6, 10, 15]]
```

중첩 리스트
Nested List

```
[[1, 1, 1, 1, 1], [1, 2, 3, 4, 5], [1, 3, 6, 10, 15]]
```

```
   [[1, 1, 1, 1, 1], [1], [1]]
=> [[1, 1, 1, 1, 1], [1, 2], [1]]
=> [[1, 1, 1, 1, 1], [1, 2, 3], [1]]
=> [[1, 1, 1, 1, 1], [1, 2, 3, 4], [1]]
=> [[1, 1, 1, 1, 1], [1, 2, 3, 4, 5], [1]]
=> [[1, 1, 1, 1, 1], [1, 2, 3, 4, 5], [1, 3]]
=> [[1, 1, 1, 1, 1], [1, 2, 3, 4, 5], [1, 3, 6]]
=>
```

파스칼의 삼각형

표채워풀기 구현

$_6C_4$

```
1  1  1  1  1
1  2  3  4  5
1  3  6  10 15
```

```
[[1, 1, 1, 1, 1],
 [1, 2, 3, 4, 5],
 [1, 3, 6, 10, 15]]
```

중첩 리스트
Nested List

```
[[1, 1, 1, 1, 1], [1, 2, 3, 4, 5], [1, 3, 6, 10, 15]]
```

```
   [[1, 1, 1, 1, 1], [1], [1]]
=> [[1, 1, 1, 1, 1], [1, 2], [1]]
=> [[1, 1, 1, 1, 1], [1, 2, 3], [1]]
=> [[1, 1, 1, 1, 1], [1, 2, 3, 4], [1]]
=> [[1, 1, 1, 1, 1], [1, 2, 3, 4, 5], [1]]
=> [[1, 1, 1, 1, 1], [1, 2, 3, 4, 5], [1, 3]]
=> [[1, 1, 1, 1, 1], [1, 2, 3, 4, 5], [1, 3, 6]]
=>
```

파스칼의 삼각형

표채워풀기 구현

$_6C_4$

1 1 1 1 1
1 2 3 4 5
1 3 6 10 15

```
[[1, 1, 1, 1, 1],
 [1, 2, 3, 4, 5],
 [1, 3, 6, 10, 15]]
```

중첩 리스트
Nested List

```
[[1, 1, 1, 1, 1], [1, 2, 3, 4, 5], [1, 3, 6, 10, 15]]
```

```
   [[1, 1, 1, 1, 1], [1], [1]]
=> [[1, 1, 1, 1, 1], [1, 2], [1]]
=> [[1, 1, 1, 1, 1], [1, 2, 3], [1]]
=> [[1, 1, 1, 1, 1], [1, 2, 3, 4], [1]]
=> [[1, 1, 1, 1, 1], [1, 2, 3, 4, 5], [1]]
=> [[1, 1, 1, 1, 1], [1, 2, 3, 4, 5], [1, 3]]
=> [[1, 1, 1, 1, 1], [1, 2, 3, 4, 5], [1, 3, 6]]
=> [[1, 1, 1, 1, 1], [1, 2, 3, 4, 5], [1, 3, 6, 10]]
=>
```

파스칼의 삼각형

표채워풀기 구현

$_6C_4$

```
1  1  1  1  1
1  2  3  4  5
1  3  6  10 15
```

```
[[1, 1, 1, 1, 1],
 [1, 2, 3, 4, 5],
 [1, 3, 6, 10, 15]]
```

중첩 리스트
Nested List

```
[[1, 1, 1, 1, 1], [1, 2, 3, 4, 5], [1, 3, 6, 10, 15]]
```

```
   [[1, 1, 1, 1, 1], [1], [1]]
=> [[1, 1, 1, 1, 1], [1, 2], [1]]
=> [[1, 1, 1, 1, 1], [1, 2, 3], [1]]
=> [[1, 1, 1, 1, 1], [1, 2, 3, 4], [1]]
=> [[1, 1, 1, 1, 1], [1, 2, 3, 4, 5], [1]]
=> [[1, 1, 1, 1, 1], [1, 2, 3, 4, 5], [1, 3]]
=> [[1, 1, 1, 1, 1], [1, 2, 3, 4, 5], [1, 3, 6]]
=> [[1, 1, 1, 1, 1], [1, 2, 3, 4, 5], [1, 3, 6, 10]]
=>
```

파스칼의 삼각형

표채워풀기 구현

$_nC_r$

```
1  1  1  1  1
1  2  3  4  5
1  3  6  10 15
```

```
[[1, 1, 1, 1, 1],
 [1, 2, 3, 4, 5],
 [1, 3, 6, 10, 15]]
```

중첩 리스트
Nested List

```
[[1, 1, 1, 1, 1], [1, 2, 3, 4, 5], [1, 3, 6, 10, 15]]
```

```
   [[1, 1, 1, 1, 1], [1], [1]]
=> [[1, 1, 1, 1, 1], [1, 2], [1]]
=> [[1, 1, 1, 1, 1], [1, 2, 3], [1]]
=> [[1, 1, 1, 1, 1], [1, 2, 3, 4], [1]]
=> [[1, 1, 1, 1, 1], [1, 2, 3, 4, 5], [1]]
=> [[1, 1, 1, 1, 1], [1, 2, 3, 4, 5], [1, 3]]
=> [[1, 1, 1, 1, 1], [1, 2, 3, 4, 5], [1, 3, 6]]
=> [[1, 1, 1, 1, 1], [1, 2, 3, 4, 5], [1, 3, 6, 10]]
=> [[1, 1, 1, 1, 1], [1, 2, 3, 4, 5], [1, 3, 6, 10, 15]]
```

파스칼의 삼각형

표채워풀기 구현

$_nC_r$

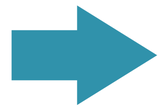
r+1

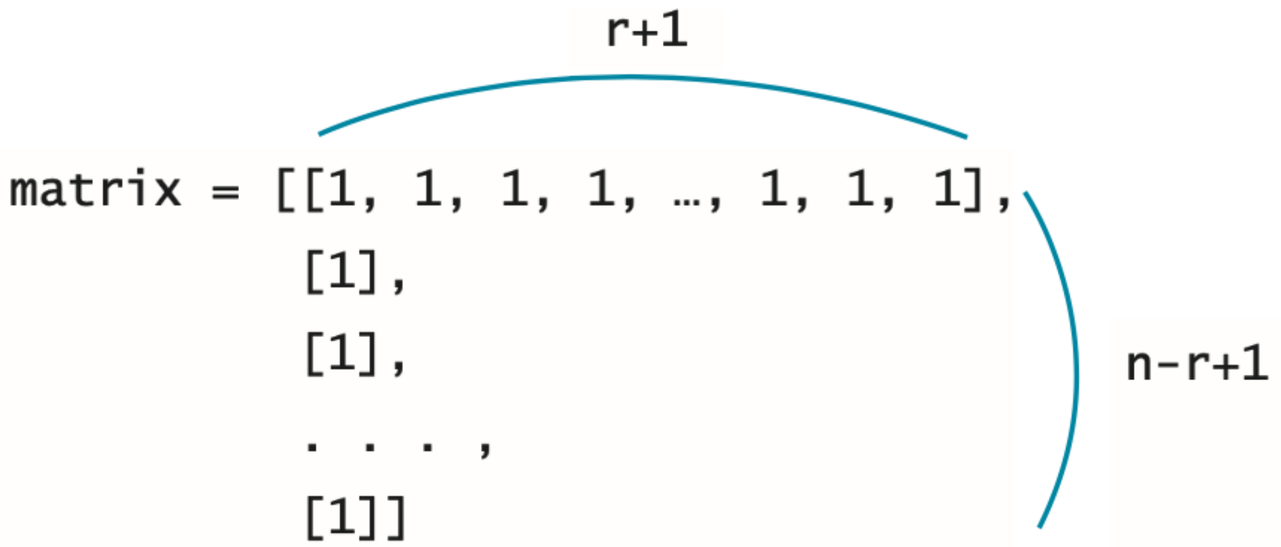
```
matrix = [[1, 1, 1, 1, …, 1, 1, 1],
          [1],
          [1],
          . . . ,
          [1]]
```

n-r+1

리스트 축약

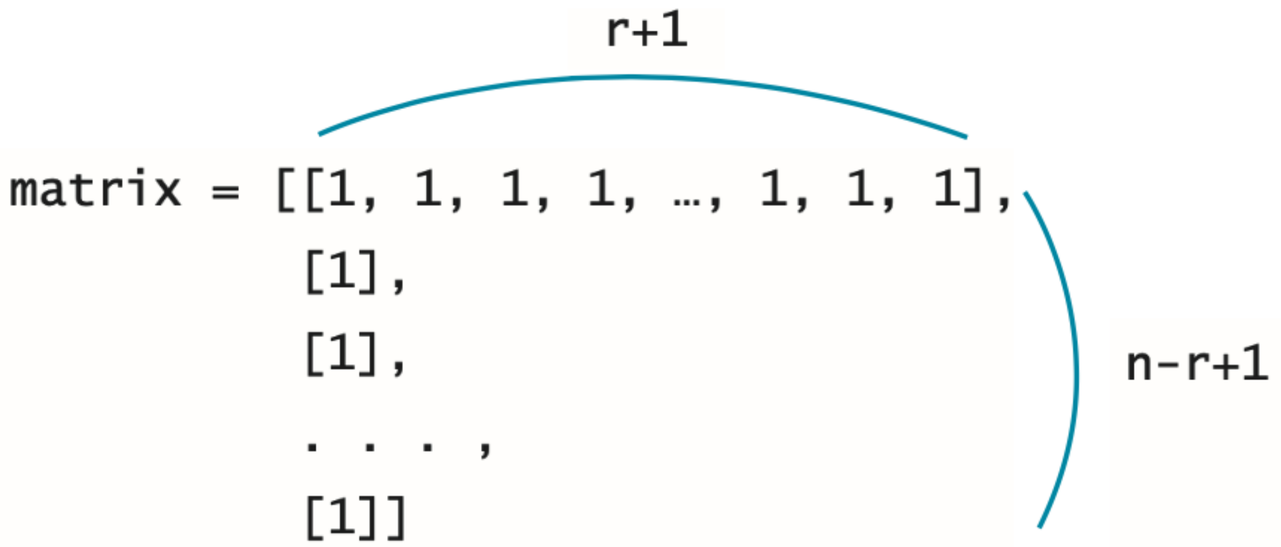
List Comprehension





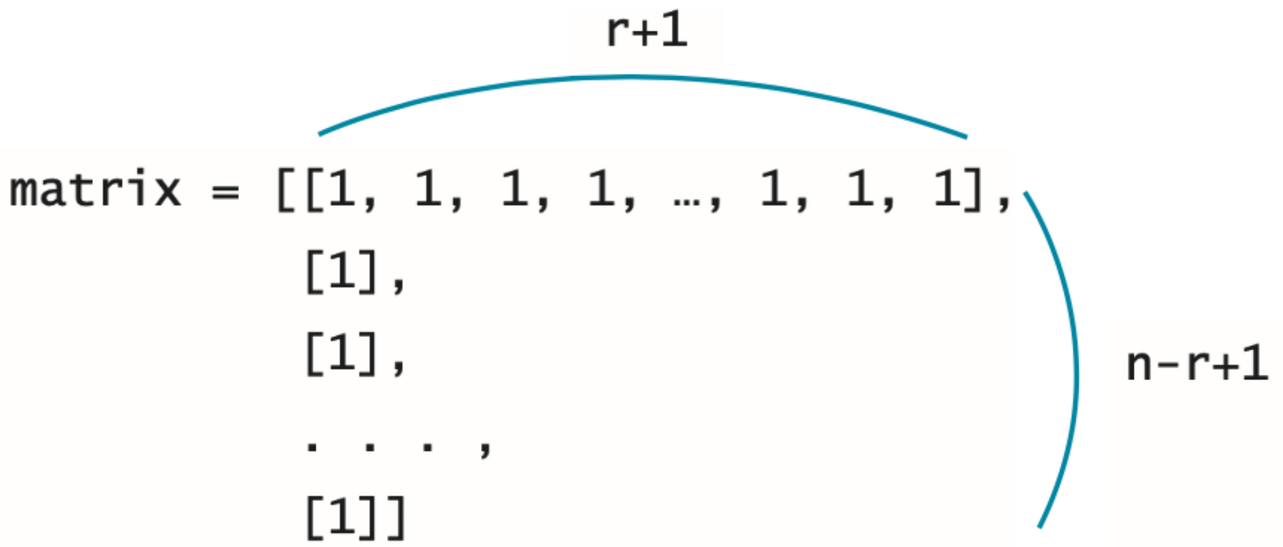
code : 7-8.py

```
1  def comb_pascal(n,r):
2      row0 = [1 for _ in range(r+1)]
3
4
5
6
7
8
```



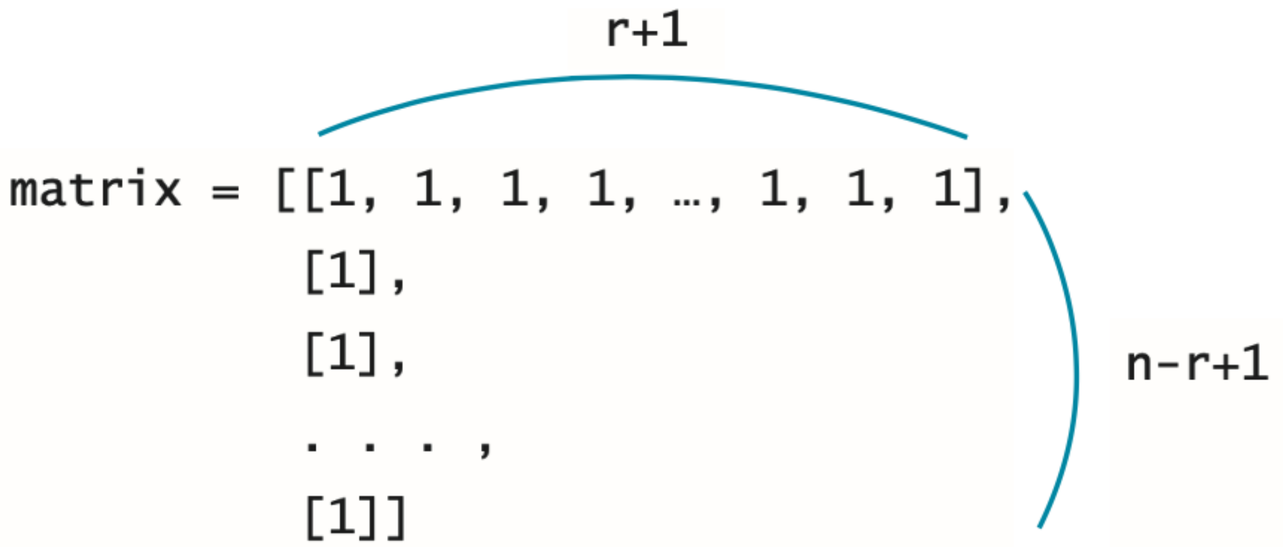
code : 7-8.py

```
1  def comb_pascal(n,r):
2      row0 = [1 for _ in range(r+1)]
3      matrix = [row0] + [[1] for _ in range(n-r)]
4
5
6
7
8
```



code : 7-8.py

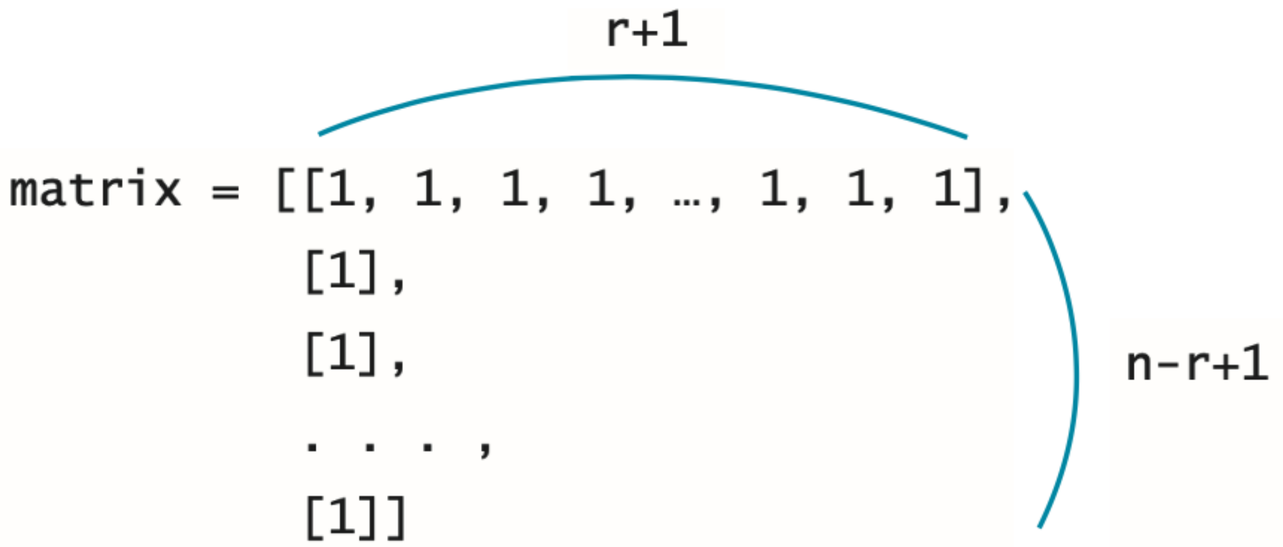
```
1  def comb_pascal(n,r):
2      row0 = [1 for _ in range(r+1)]
3      matrix = [row0] + [[1] for _ in range(n-r)]
4      for i in range(1, n-r+1):
5          for j in range(1, r+1):
6
7
8
```



code : 7-8.py

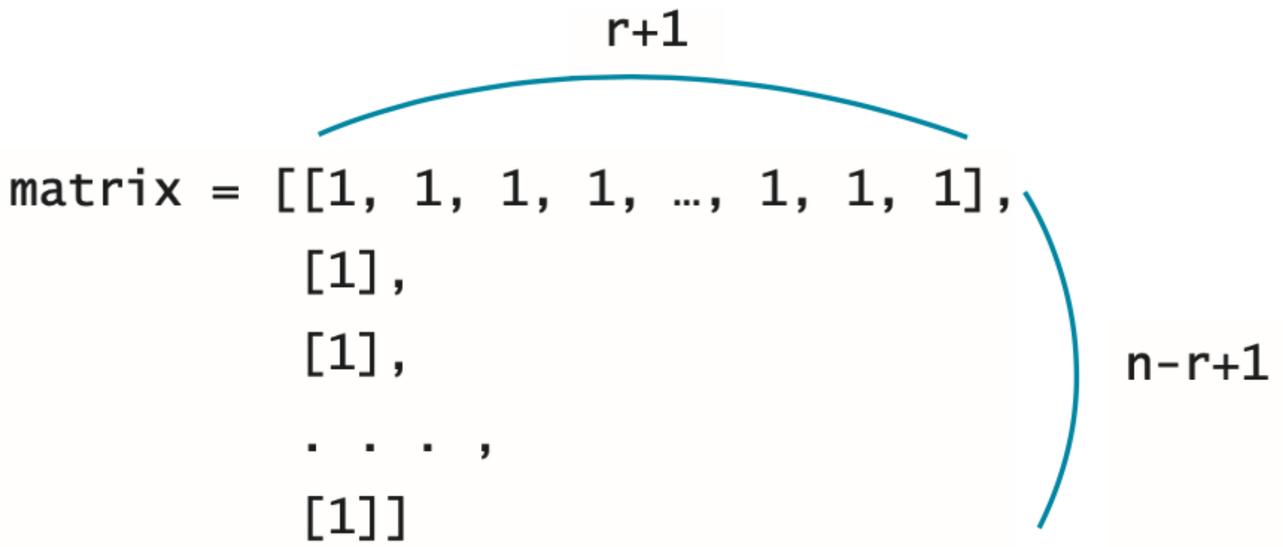
```
def comb_pascal(n,r):
    row0 = [1 for _ in range(r+1)]
    matrix = [row0] + [[1] for _ in range(n-r)]
    for i in range(1, n-r+1):
        for j in range(1, r+1):
            newvalue = matrix[i][j-1] + matrix[i-1][j]


```



code : 7-8.py

```
1  def comb_pascal(n,r):
2      row0 = [1 for _ in range(r+1)]
3      matrix = [row0] + [[1] for _ in range(n-r)]
4      for i in range(1, n-r+1):
5          for j in range(1, r+1):
6              newvalue = matrix[i][j-1] + matrix[i-1][j]
7              matrix[i].append(newvalue)
8
```



code : 7-8.py

```
1  def comb_pascal(n,r):
2      row0 = [1 for _ in range(r+1)]
3      matrix = [row0] + [[1] for _ in range(n-r)]
4      for i in range(1, n-r+1):
5          for j in range(1, r+1):
6              newvalue = matrix[i][j-1] + matrix[i-1][j]
7              matrix[i].append(newvalue)
8      return matrix[n-r][r]
```

>>>>>>>>>>> 제어 구조의 설계 원리를 중심으로 배우는 >>>>>>>>>>>

프로그래밍의 정석

파이썬

도경구 지음

생능출판

p.349

실습 7.2 comb_pascal 함수의 실행 시간 측정

〈실습 7.1〉에서 작성한 함수 run_comb를 사용하여 comb_pascal 함수의 실행 시간을 측정하여 comb 함수에 비해서 얼마나 개선되었는지 비교하자.

code : 7-9.py

```python
def comb_pascal(n,r):
    vector = [1 for _ in range(r+1)]
    for _ in range(1, n-r+1):
        for j in range(1, r+1):
            vector[j] = vector[j-1] + vector[j]
    return vector[r]
```

```
comb_pascal(6,4)

vector = [1, 1, 1, 1, 1]
      =>
```

code : 7-9.py

```
1  def comb_pascal(n,r):
2      vector = [1 for _ in range(r+1)]
3      for _ in range(1, n-r+1):
4          for j in range(1, r+1):
5              vector[j] = vector[j-1] + vector[j]
6      return vector[r]
```

```
comb_pascal(6,4)

vector = [1, 1, 1, 1, 1]
       => [1, 2, 1, 1, 1]
       =>
```

code : 7-9.py

```python
def comb_pascal(n,r):
    vector = [1 for _ in range(r+1)]
    for _ in range(1, n-r+1):
        for j in range(1, r+1):
            vector[j] = vector[j-1] + vector[j]
    return vector[r]
```

```
comb_pascal(6,4)

vector = [1, 1, 1, 1, 1]
      => [1, 2, 1, 1, 1]
      => [1, 2, 3, 1, 1]
      =>
```

code : 7-9.py

```python
def comb_pascal(n,r):
    vector = [1 for _ in range(r+1)]
    for _ in range(1, n-r+1):
        for j in range(1, r+1):
            vector[j] = vector[j-1] + vector[j]
    return vector[r]
```

```
comb_pascal(6,4)

vector = [1, 1, 1, 1, 1]
      => [1, 2, 1, 1, 1]
      => [1, 2, 3, 1, 1]
      => [1, 2, 3, 4, 1]
      =>
```

code : 7-9.py

```python
def comb_pascal(n,r):
    vector = [1 for _ in range(r+1)]
    for _ in range(1, n-r+1):
        for j in range(1, r+1):
            vector[j] = vector[j-1] + vector[j]
    return vector[r]
```

```
comb_pascal(6,4)

vector = [1, 1, 1, 1, 1]
      => [1, 2, 1, 1, 1]
      => [1, 2, 3, 1, 1]
      => [1, 2, 3, 4, 1]
      => [1, 2, 3, 4, 5]
      =>
```

code : 7-9.py

```python
def comb_pascal(n,r):
    vector = [1 for _ in range(r+1)]
    for _ in range(1, n-r+1):
        for j in range(1, r+1):
            vector[j] = vector[j-1] + vector[j]
    return vector[r]
```

```
comb_pascal(6,4)

vector = [1, 1, 1, 1, 1]
      => [1, 2, 1, 1, 1]
      => [1, 2, 3, 1, 1]
      => [1, 2, 3, 4, 1]
      => [1, 2, 3, 4, 5]
      => [1, 3, 3, 4, 5]
      =>
```

code : 7-9.py

```python
def comb_pascal(n,r):
    vector = [1 for _ in range(r+1)]
    for _ in range(1, n-r+1):
        for j in range(1, r+1):
            vector[j] = vector[j-1] + vector[j]
    return vector[r]
```

```
comb_pascal(6,4)

vector = [1, 1, 1, 1, 1]
      => [1, 2, 1, 1, 1]
      => [1, 2, 3, 1, 1]
      => [1, 2, 3, 4, 1]
      => [1, 2, 3, 4, 5]
      => [1, 3, 3, 4, 5]
      => [1, 3, 6, 4, 5]
      =>
```

code : 7-9.py

```python
def comb_pascal(n,r):
    vector = [1 for _ in range(r+1)]
    for _ in range(1, n-r+1):
        for j in range(1, r+1):
            vector[j] = vector[j-1] + vector[j]
    return vector[r]
```

```
comb_pascal(6,4)

vector = [1, 1, 1, 1, 1]
      => [1, 2, 1, 1, 1]
      => [1, 2, 3, 1, 1]
      => [1, 2, 3, 4, 1]
      => [1, 2, 3, 4, 5]
      => [1, 3, 3, 4, 5]
      => [1, 3, 6, 4, 5]
      => [1, 3, 6, 10, 5]
      =>
```

code : 7-9.py

```python
def comb_pascal(n,r):
    vector = [1 for _ in range(r+1)]
    for _ in range(1, n-r+1):
        for j in range(1, r+1):
            vector[j] = vector[j-1] + vector[j]
    return vector[r]
```

```
comb_pascal(6,4)

vector = [1, 1, 1, 1, 1]
      => [1, 2, 1, 1, 1]
      => [1, 2, 3, 1, 1]
      => [1, 2, 3, 4, 1]
      => [1, 2, 3, 4, 5]
      => [1, 3, 3, 4, 5]
      => [1, 3, 6, 4, 5]
      => [1, 3, 6, 10, 5]
      => [1, 3, 6, 10, 15]
```

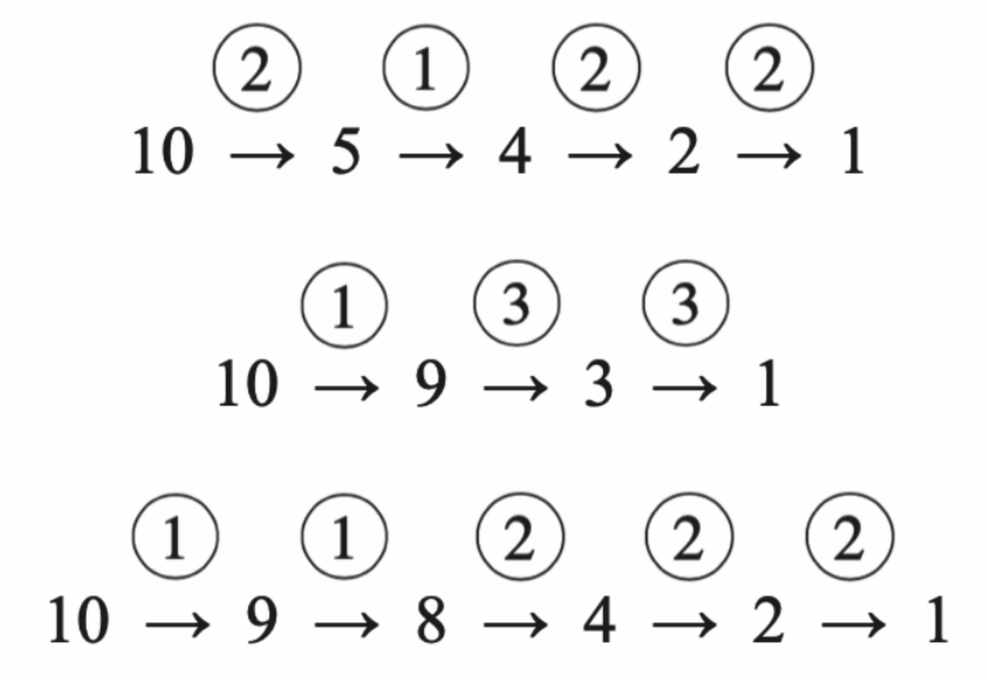
CHAPTER 7

표채워풀기

1까지 줄이는 최소 스텝

다음의 3가지 스텝 중 하나를 선택하여 주어진 양수 n을 줄일 수 있다고 하자.

- 스텝 ① : 1을 뺀다.
- 스텝 ② : 2로 나누어지면, 2로 나눈다.
- 스텝 ③ : 3으로 나누어지면, 3으로 나눈다.



1까지 줄이는 최소 스텝

임의의 양수 n을 1까지 줄여나가는 가장 짧은 경로의 길이를 구하는 1까지 줄이는 최소 스텝minimum steps to one 문제를 풀어보자. 이 문제를 푸는 식은 다음과 같이 재귀로 표현할 수 있다.

$$\text{minsteps}(n) = \begin{cases} 1 + \mathit{minimum}\{\text{minsteps}(n-1), \text{minsteps}(n/2), \text{minsteps}(n/3)\} & \text{if } n > 1 \\ 0 & \text{if } n = 1 \end{cases}$$

$$10 \xrightarrow{2} 5 \xrightarrow{1} 4 \xrightarrow{2} 2 \xrightarrow{2} 1$$

$$10 \xrightarrow{1} 9 \xrightarrow{3} 3 \xrightarrow{3} 1$$

$$10 \xrightarrow{1} 9 \xrightarrow{1} 8 \xrightarrow{2} 4 \xrightarrow{2} 2 \xrightarrow{2} 1$$

재귀 알고리즘

code : 7-10.py

```
1  def minsteps(n):
2      if n > 1:
3          steps = minsteps(n-1)
4          if n % 2 == 0:
5              steps = min(steps, minsteps(n//2))
6          if n % 3 == 0:
7              steps = min(steps, minsteps(n//3))
8          return 1 + steps
9      else:
10         return 0
```

code : 7-10.py

```python
def minsteps(n):
    if n > 1:
        steps = minsteps(n-1)
        if n % 2 == 0:
            steps = min(steps, minsteps(n//2))
        if n % 3 == 0:
            steps = min(steps, minsteps(n//3))
        return 1 + steps
    else:
        return 0
```

code : 7-11.py

```python
def run_minsteps(n):
    from time import perf_counter
    start = perf_counter()
    answer = minsteps(n)
    finish = perf_counter()
    print("minsteps(", n, ") => ", answer, sep="")
    print(round(finish-start), "seconds")
```

```
>>> run_minsteps(23)
minsteps(23) => 6
0 seconds
>>> run_minsteps(237)
minsteps(237) => 8
1 seconds
>>> run_minsteps(317)
minsteps(317) => 10
4 seconds
>>> run_minsteps(514)
minsteps(514) => 8
59 seconds
>>> run_minsteps(711)
minsteps(711) => 9
431 seconds
>>> run_minsteps(908)
minsteps(908) => 11
2106 seconds
```

탐욕 알고리즘

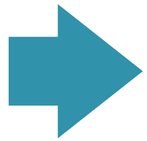
함수호출	답	줄이기 절차
minsteps(3)	1	3 → 1 탐욕, 정답
minsteps(4)	2	4 → 2 → 1 탐욕, 정답
minsteps(7)	3	7 → 6 → 2 → 1 탐욕, 정답
minsteps(10)	3	10 → 5 → 4 → 2 → 1 탐욕 10 → 9 → 3 → 1 정답
minsteps(23)	6	23 → 22 → 11 → 10 → 5 → 4 → 2 → 1 탐욕 23 → 22 → 21 → 7 → 6 → 2 → 1 정답
minsteps(237)	8	237 → 79 → 78 → 26 → 13 → 12 → 4 → 2 → 1 탐욕, 정답
minsteps(317)	10	317 → 316 → 158 → 79 → 78 → 26 → 13 → 12 → 4 → 2 → 1 탐욕, 정답
minsteps(514)	8	514 → 257 → 256 → 128 → 64 → 32 → 16 → 8 → 4 → 2 → 1 탐욕 514 → 513 → 171 → 57 → 19 → 18 → 6 → 2 → 1 정답

- 스텝 ① : 1을 뺀다.
- 스텝 ② : 2로 나누어지면, 2로 나눈다.
- 스텝 ③ : 3으로 나누어지면, 3으로 나눈다.

메모해두기 알고리즘

code : 7-12.py

```
1  def minsteps(n):
2      memo = [0 for _ in range(n+1)]
3      def loop(n):
4          if n > 1:
5              if memo[n] == 0:
6                  steps = loop(n - 1)
7                  if n % 2 == 0:
8                      steps = min(steps, loop(n // 2))
9                  if n % 3 == 0:
10                     steps = min(steps, loop(n // 3))
11                 memo[n] = steps + 1
12             return memo[n]
13         else:
14             return 0
15     return loop(n)
```



표채워풀기 알고리즘

code : 7-13.py

```
1  def minsteps(n):
2      memo = [0 for _ in range(n+1)]
3      pass # Write your for-loop here.
4
5
6
7
8
9
10     return memo[n]
```

실습 7.3 표채워풀기 알고리즘 구현

상향식으로 memo[2]부터 시작하여 memo[n]까지 memo 리스트를 차례로 모두 채우는 표채워풀기 방식으로 for 문을 사용하여 아래 뼈대코드에 맞추어 함수를 다시 짜보자. for 문을 사용하므로 인수가 아무리 커도 잘 작동할 것이다.

code : 7-13.py

```
1  def minsteps(n):
2      memo = [0 for _ in range(n+1)]
3      pass # Write your for-loop here.
4
5
6
7
8
9
10     return memo[n]
```

run_minsteps로 테스트하여 작동 여부 및 실행 시간을 확인해 보자.

CHAPTER 7

표채워풀기

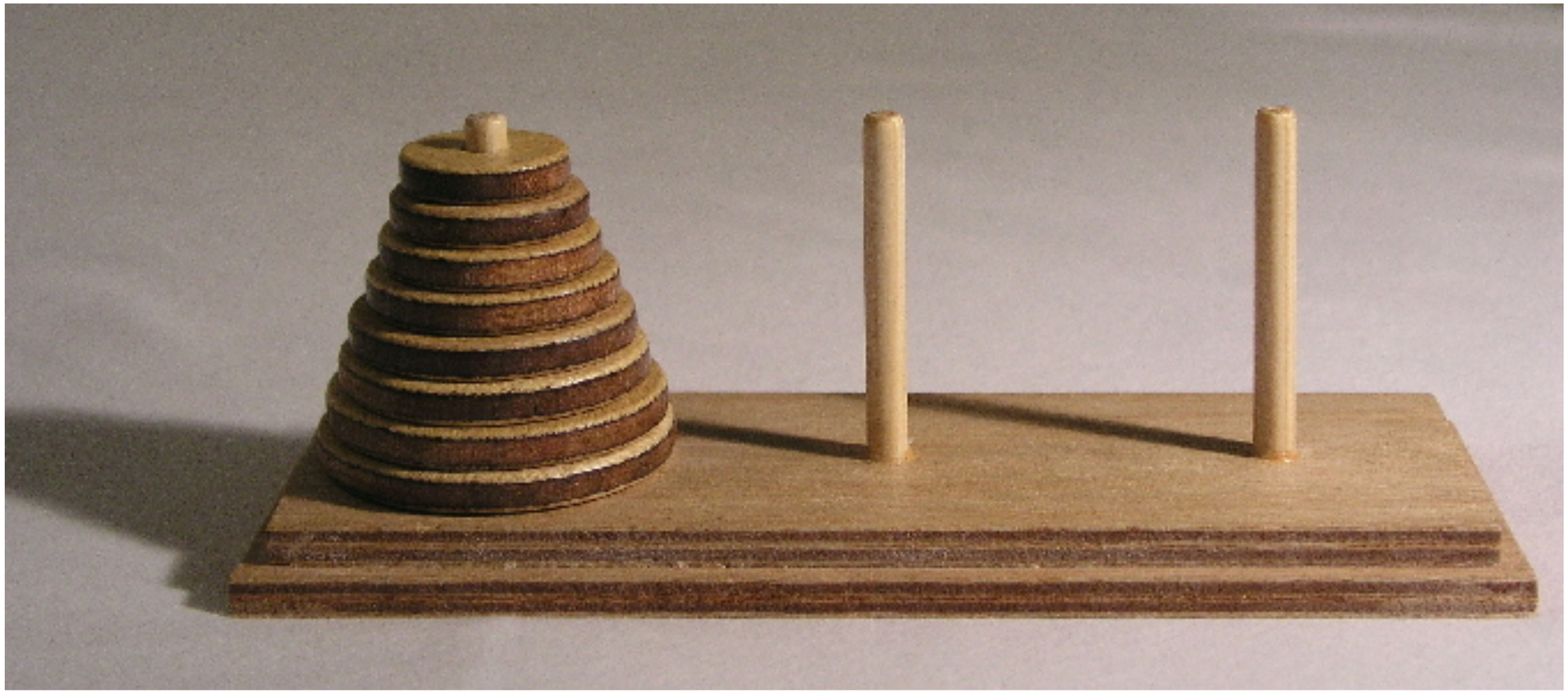
7.1 피보나치 수열

7.2 조합

7.3 1까지 줄이는 최소 스텝

✔ 7.4 하노이의 탑

하노이의 탑



(출처 : https://en.wikipedia.org/wiki/Tower_of_Hanoi)

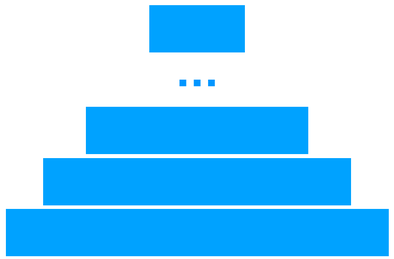
하노이의 탑

(출처 : https://en.wikipedia.org/wiki/Tower_of_Hanoi)

하노이의 탑

n개 원반을 출발말뚝에서 도착말뚝으로 옮기는 방법

1. (재귀) 출발말뚝에 있는 $n-1$개의 원반을 임시말뚝으로 옮긴다.
2. 출발말뚝의 맨 아래에 남아 있는 가장 큰 원반을 도착말뚝으로 옮긴다.
3. (재귀) 임시말뚝에 있는 $n-1$개의 원반을 도착말뚝으로 옮긴다.

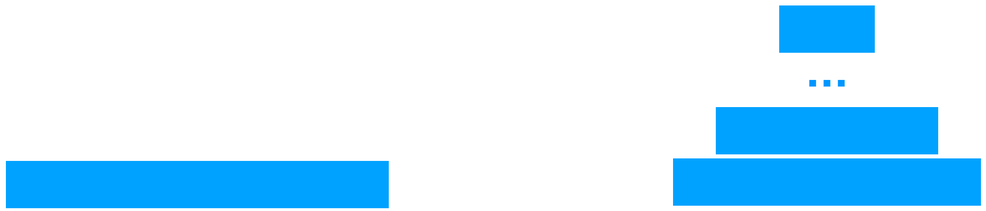


출발말뚝 임시말뚝 도착말뚝

하노이의 탑

n개 원반을 출발말뚝에서 도착말뚝으로 옮기는 방법

1. (재귀) 출발말뚝에 있는 $n-1$개의 원반을 임시말뚝으로 옮긴다.
2. 출발말뚝의 맨 아래에 남아 있는 가장 큰 원반을 도착말뚝으로 옮긴다.
3. (재귀) 임시말뚝에 있는 $n-1$개의 원반을 도착말뚝으로 옮긴다.

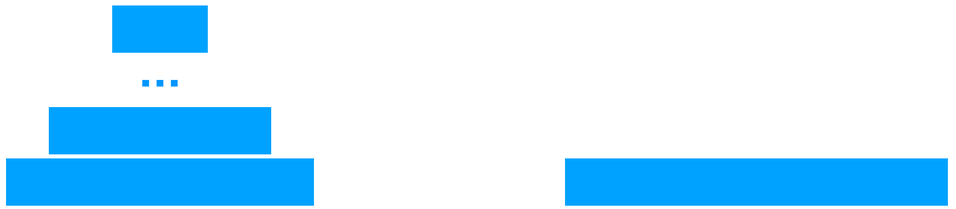


출발말뚝 임시말뚝 도착말뚝

하노이의 탑

n개 원반을 출발말뚝에서 도착말뚝으로 옮기는 방법

1. (재귀) 출발말뚝에 있는 $n-1$개의 원반을 임시말뚝으로 옮긴다.
2. 출발말뚝의 맨 아래에 남아 있는 가장 큰 원반을 도착말뚝으로 옮긴다.
3. (재귀) 임시말뚝에 있는 $n-1$개의 원반을 도착말뚝으로 옮긴다.

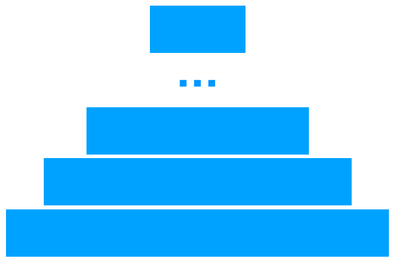


출발말뚝 임시말뚝 도착말뚝

하노이의 탑

n개 원반을 출발말뚝에서 도착말뚝으로 옮기는 방법

1. (재귀) 출발말뚝에 있는 $n-1$개의 원반을 임시말뚝으로 옮긴다.
2. 출발말뚝의 맨 아래에 남아 있는 가장 큰 원반을 도착말뚝으로 옮긴다.
3. (재귀) 임시말뚝에 있는 $n-1$개의 원반을 도착말뚝으로 옮긴다.



출발말뚝 임시말뚝 도착말뚝

하노이의 탑

n개 원반을 출발말뚝에서 도착말뚝으로 옮기는 방법

1. (재귀) 출발말뚝에 있는 $n-1$개의 원반을 임시말뚝으로 옮긴다.
2. 출발말뚝의 맨 아래에 남아 있는 가장 큰 원반을 도착말뚝으로 옮긴다.
3. (재귀) 임시말뚝에 있는 $n-1$개의 원반을 도착말뚝으로 옮긴다.

code : 7-14.py

```
1 def tower_of_hanoi(n, source, destin, temp):
2     if n > 1:
3         tower_of_hanoi(n-1, source, temp, destin)
4         print("Move a disk from", source, "to", destin)
5         tower_of_hanoi(n-1, temp, destin, source)
6     else:
7         print("Move a disk from", source, "to", destin)
```

하노이의 탑

실행시간 측정

code : 7-15.py

```
1  def tower_of_hanoi(n, source, destin, temp):
2      global count
3      if n > 1:
4          tower_of_hanoi(n-1, source, temp, destin)
5          count += 1
6          tower_of_hanoi(n-1, temp, destin, source)
7      else:
8          count += 1
9
10 for n in [4,6,8,16,24,25,26,27,28]:
11     count = 0
12     from time import perf_counter
13     start = perf_counter()
14     tower_of_hanoi(n, "A", "C", "B")
15     finish = perf_counter()
16     cpu_time = round(finish-start,1)
17     print(n, "disks:", count, "moves in", cpu_time, "seconds")
```

```
4 disks: 15 moves in 0.0 seconds
6 disks: 63 moves in 0.0 seconds
8 disks: 255 moves in 0.0 seconds
16 disks: 65535 moves in 0.0 seconds
22 disks: 4194303 moves in 0.9 seconds
23 disks: 8388607 moves in 1.8 seconds
24 disks: 16777215 moves in 3.6 seconds
25 disks: 33554431 moves in 7.3 seconds
26 disks: 67108863 moves in 14.8 seconds
27 disks: 134217727 moves in 29.4 seconds
28 disks: 268435455 moves in 59.4 seconds
```

p.362

실습 7.4 tower_of_hanoi 함수의 실행 시간 예측

위의 테스트 결과 원반이 28개일 때 실행 시간이 약 1분 걸렸다. 그러면 실행 시간이 하루(24시간)가 넘게 되는 원반의 최소 개수는 몇 개인지 손으로 계산하여 예측해보자.

>>>>>>>>>>> 제어 구조의 설계 원리를 중심으로 배우는 >>>>>>>>>>>

프로그래밍의 정석

파이썬

도경구 지음

생능출판

CHAPTER 7

표채워풀기