

Chapter. 02

알고리즘

동적 프로그래밍 Dynamic Programming

FAST CAMPUS
ONLINE

알고리즘 공채 대비반 I

강사. 류호석

Chapter. 02

알고리즘



동적 프로그래밍(Dynamic Programming)

I 동적 프로그래밍(Dynamic Programming)이란?

Dynamic := 동적인, 변화하는

Programming := 문제를 해결하는

문제의 크기를 변화하면서 정답을 계산하는데,

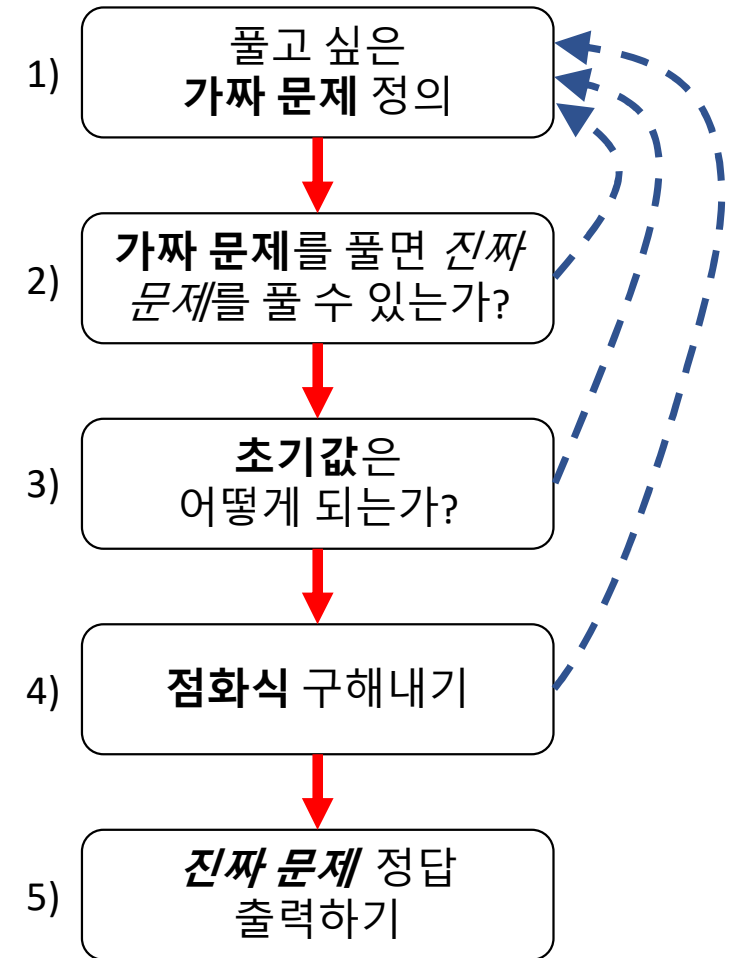
작은 문제의 결과를 *이/용*해서 큰 문제의 정답을 빠르게 계산하는 알고리즘

I 동적 프로그래밍(Dynamic Programming)이란?

1. 문제가 원하는 정답을 찾기 위해 가장 먼저 완전 탐색 (Brute-Force Search) 접근을 시도해본다.
2. 근데, 완전 탐색 과정에서 탐색하게 되는 경우가 **지나치게** 많아서 도저히 안 될 것 같다.
3. 이럴 때, 모든 경우를 **빠르게** 탐색하는 방법으로 *Dynamic Programming* 접근을 시도해볼 수 있다.

➔ **규격화된 문제 풀이 순서를 외워서 훈련해야 합니다.**

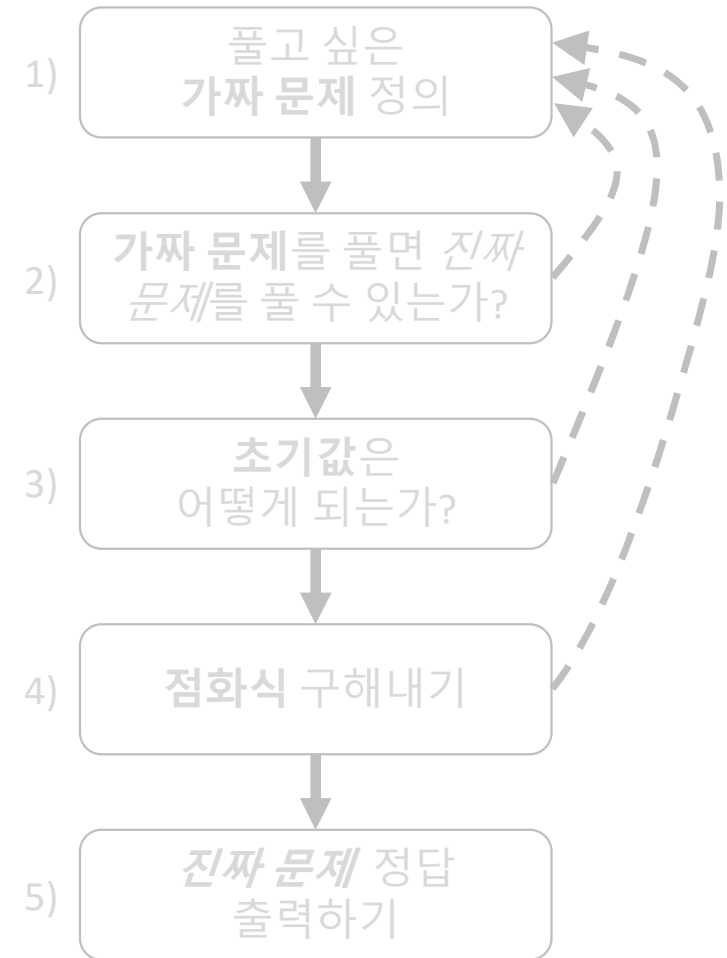
I 동적 프로그래밍(Dynamic Programming)이란?



I 동적 프로그래밍(Dynamic Programming)이란?

1) 풀고 싶은 가짜 문제 정의

예시)

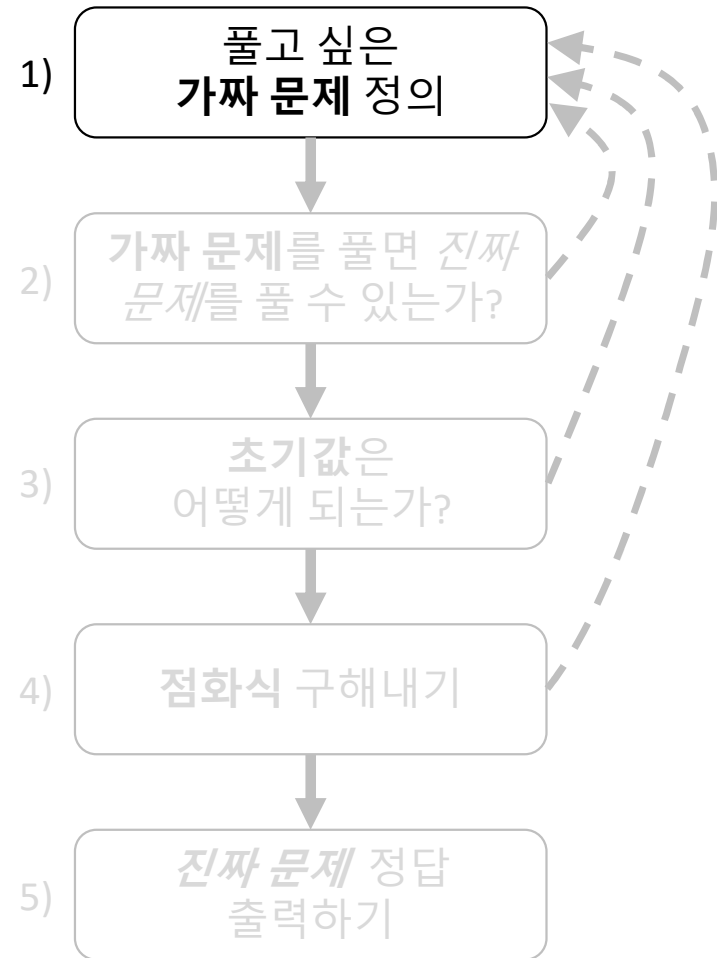


I 동적 프로그래밍(Dynamic Programming)이란?

1) 풀고 싶은 가짜 문제 정의

예시)

- $Dy[i] := 1 \sim i$ 번 원소에 대해서 조건을 만족하는 경우의 수
- $Dy[i][j] := i \sim j$ 번 원소에 대해서 조건을 만족하는 최댓값
- $Dy[i][j] :=$ 수열 $A[1...i]$ 와 수열 $B[1...j]$ 에 대해서 무언가를 계산한 값



I 동적 프로그래밍(Dynamic Programming)이란?

2) 가짜 문제를 풀면 진짜 문제를 풀 수 있는가?

예시)

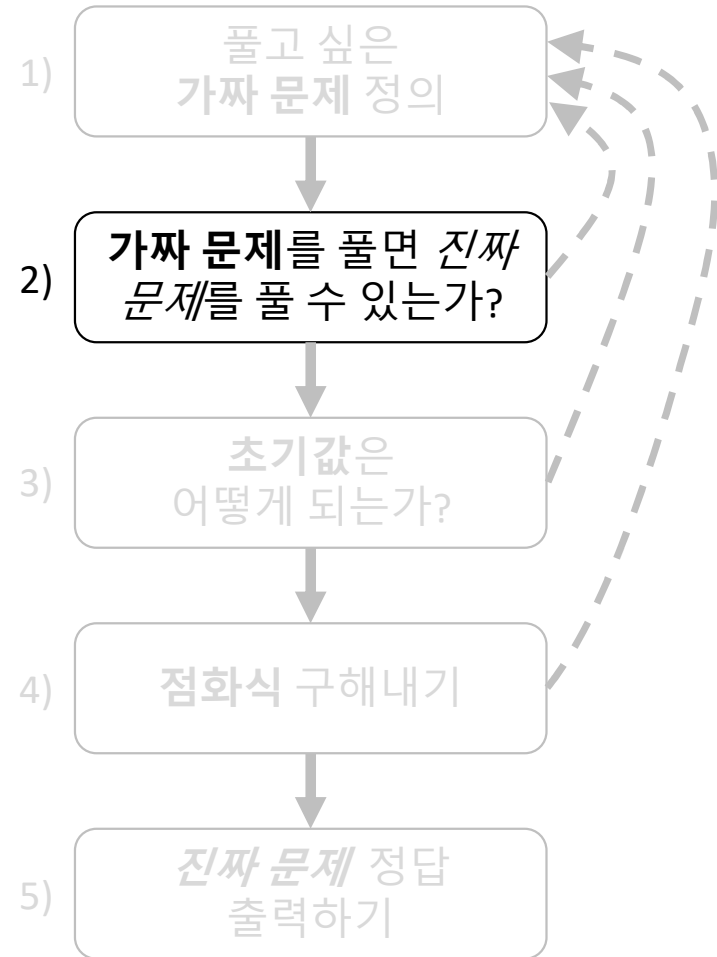
진짜 문제

→ 수열 $A[1...N]$ 에서 조건을 만족하는 부분 수열의 개수

가짜 문제

→ $Dy[i] :=$ 수열 $A[1...i]$ 에서 조건을 만족하는 부분 수열의 개수

가짜 문제를 푼다면 $Dy[1], Dy[2], \dots, Dy[N]$ 을 모두 계산했을 것이니까, $Dy[N]$ 에 적혀있는 값이 곧 진짜 문제가 원하는 값이다.



I 동적 프로그래밍(Dynamic Programming)이란?

3) step

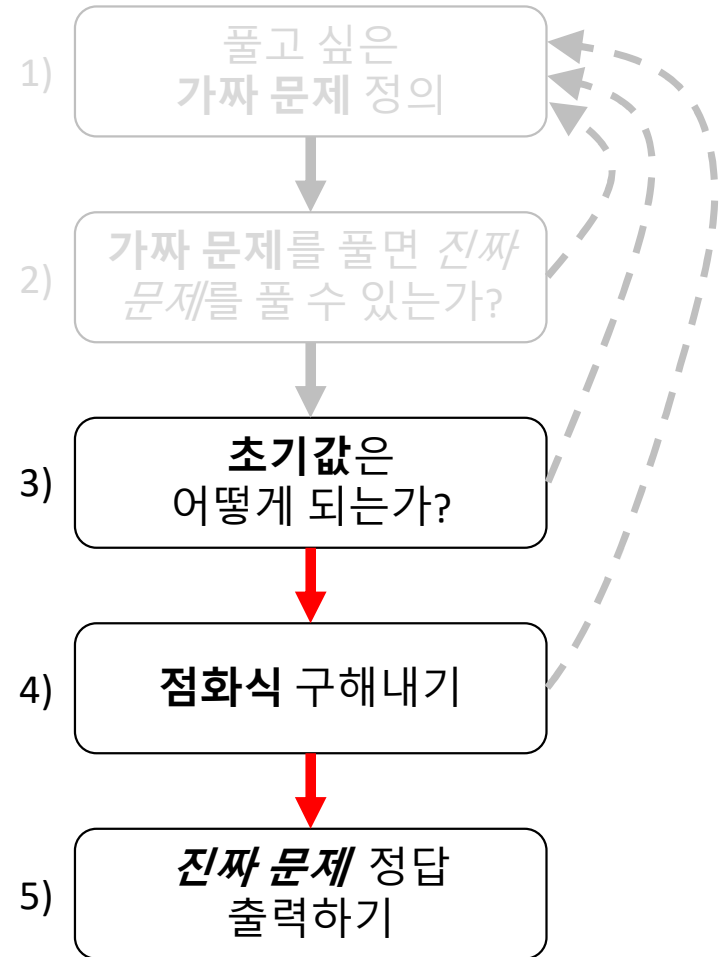
➔ 가장 작은 문제 해결하기

4) step

➔ 3)에서 계산한 것을 기반으로, 점점 더 큰 문제들을 해결하면서 Dy 배열을 가득 계산하는 과정

5) step

➔ 1) ~ 4)가 성공적으로 끝난다면 Dy 배열을 이용하여 진짜 문제 해결하기



Chapter. 02 알고리즘

| [BOJ 9095 – 1, 2, 3 더하기](#)

난이도: 2

$1 \leq N \leq 11$

정수 n 이 주어졌을 때, n 을 1, 2, 3의 합으로 나타내는 방법의 수를 구하는 프로그램을 작성하시오.

$$4 = 1+1+1+1$$

$$4 = 1+1+2$$

$$4 = 1+2+1$$

$$4 = 2+1+1$$

$$4 = 2+2$$

$$4 = 1+3$$

$$4 = 3+1 \quad \text{답: 7}$$

I 완전 탐색 접근

완전 탐색 접근을 통해서 모든 경우를 직접 하나하나 찾아내 보자.

$$10 = 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1$$

$$10 = 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 2$$

...

$$10 = 3 + 3 + 3 + 1$$

➔ N이 커질수록 탐색해야 하는 경우가 엄청 많아진다.

I Dynamic Programming 접근

1) 풀고 싶은 가짜 문제 정의

Hint) 진짜 문제 먼저 써보기

진짜 문제 := 주어진 N 에 대해서 N 을 1, 2, 3의 합으로 표현하는 경우의 수

가짜 문제 := $Dy[i] = i$ 를 1, 2, 3의 합으로 표현하는 경우의 수

i	1	2	3	4	5	6	7
Dy[i]							

(레벨 1: 진짜 문제랑 똑같은 가짜 문제인 경우)

I Dynamic Programming 접근

2) **가짜 문제를 풀면 진짜 문제를 풀 수 있는가?**

Dy 배열을 가득 채울 수만 있다면? **진짜 문제**에 대한 답은 $Dy[N]$ 이다.

i	1	2	3	4	5	6	7
Dy[i]							

I Dynamic Programming 접근

3) 초기값은 어떻게 되는가?

초기값: 쪼개지 않아도 풀 수 있는 “작은 문제”들에 대한 정답

i	1	2	3	4	5	6	7
Dy[i]	1	2	4				

I Dynamic Programming 접근

4) 점화식 구해내기

1. $Dy[i]$ 계산에 필요한 탐색 경우를 공통점끼리 묶어 내기 (Partitioning)
2. 묶어낸 부분의 정답을 Dy 배열을 이용해서 빠르게 계산해보기

I Dynamic Programming 접근

4-1) $Dy[i]$ 계산에 필요한 탐색 경우를 공통점끼리 묶어 내기 (Partitioning) $Dy[5]$ 계산에 필요한 탐색 경우들

$$\begin{array}{ccccccc}
 1 + 1 + 1 + 1 + 1 & & & & 1 + 1 + 2 + 1 & 1 + 1 + 3 & \\
 & & & & & & \\
 1 + 2 + 1 + 1 & 1 + 3 + 1 + 1 & & & 3 + 2 & & \\
 1 + 1 + 1 + 2 & 3 + 1 + 1 & & & 2 + 2 + 1 & & \\
 & & & & & & 2 + 3 \\
 1 + 2 + 2 & 2 + 1 + 1 + 1 & 2 + 1 + 2 & & & &
 \end{array}$$

I Dynamic Programming 접근

4-1) $Dy[i]$ 계산에 필요한 탐색 경우를 공통점끼리 묶어 내기 (Partitioning)

Dy[5] 계산에 필요한 탐색 경우들

$1 + 3 + 1 + 1$		
$1 + 1 + 1 + 1 + 1$	$3 + 2$	
$1 + 2 + 1 + 1$	$2 + 1 + 2$	$1 + 1 + 3$
$3 + 1 + 1$	$1 + 2 + 2$	$2 + 3$
$1 + 1 + 2 + 1$	$1 + 1 + 1 + 2$	
$2 + 2 + 1$		
$2 + 1 + 1 + 1$		

I Dynamic Programming 접근

4-2) 묶어낸 부분의 정답을 Dy 배열을 이용해서 빠르게 계산해보기

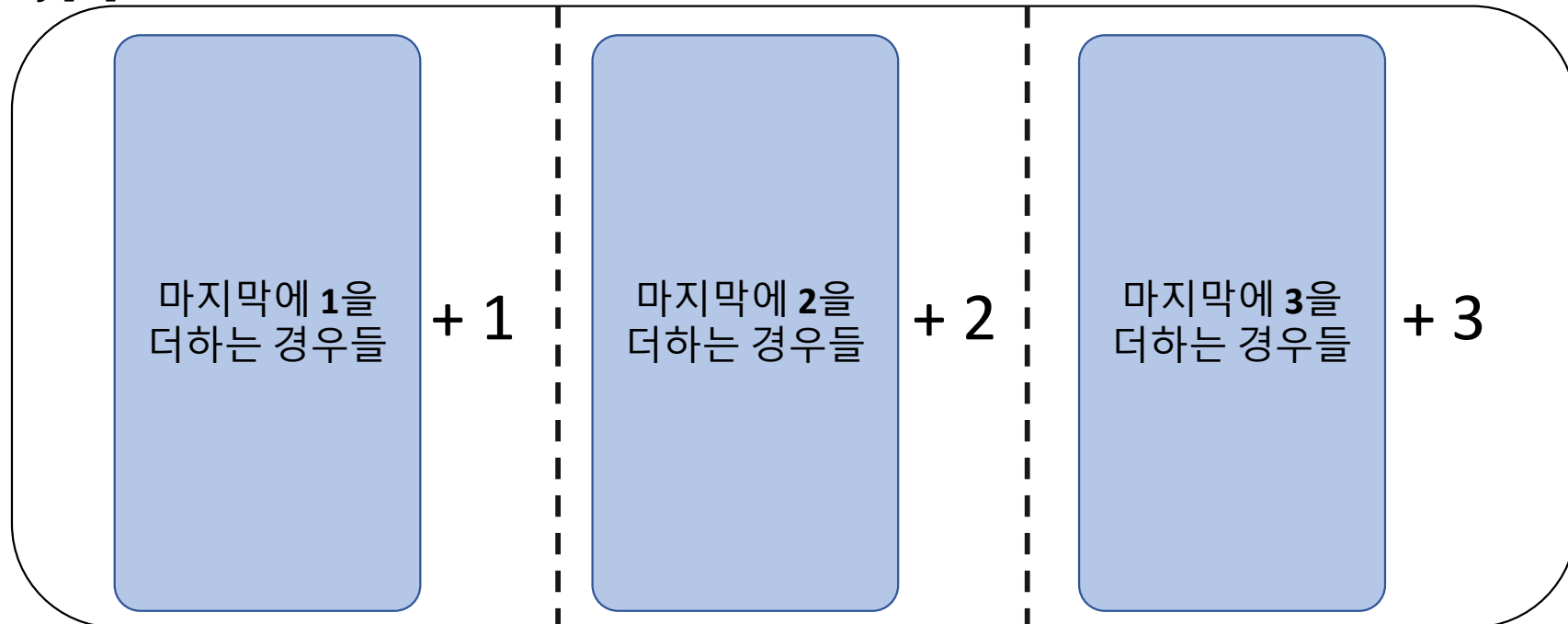
★ 전체 경우의 수 = (각 Partition의 경우의 수) 들의 합

1 + 3 + 1 + 1		
1 + 1 + 1 + 1 + 1	3 + 2	
1 + 2 + 1 + 1	2 + 1 + 2	1 + 1 + 3
3 + 1 + 1	1 + 2 + 2	2 + 3
1 + 1 + 2 + 1	1 + 1 + 1 + 2	
2 + 2 + 1		
2 + 1 + 1 + 1		

I Dynamic Programming 접근

4-2) 묶어낸 부분의 정답을 Dy 배열을 이용해서 빠르게 계산해보기

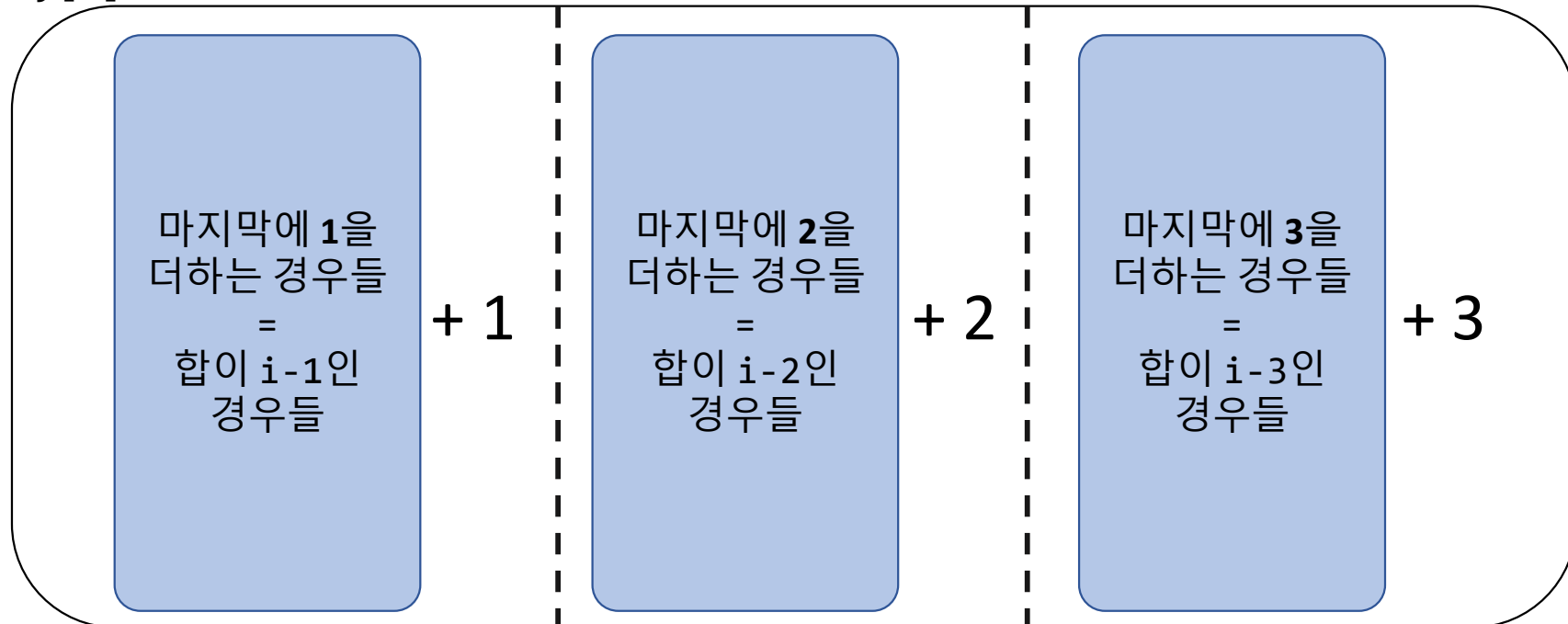
$Dy[i]$ 계산에 필요한 탐색 경우들



I Dynamic Programming 접근

4-2) 묶어낸 부분의 정답을 Dy 배열을 이용해서 빠르게 계산해보기

$Dy[i]$ 계산에 필요한 탐색 경우들



I Dynamic Programming 접근

4) 점화식 구해내기

$Dy[i]$

= (($i-1$)을 만들고 1 더하는 경우의 수) + (($i-2$)을 만들고 2 더하는 경우의 수) + (($i-3$)을 만들고 3 더하는 경우의 수)

= $Dy[i-1] + Dy[i-2] + Dy[i-3]$

I Dynamic Programming 접근

4) 점화식 구해내기

$$Dy[i] = Dy[i-1] + Dy[i-2] + Dy[i-3]$$

i	1	2	3	4	5	6	7
Dy[i]	1	2	4				

I 시간, 공간 복잡도 계산하기

완전 탐색을 통해 모든 경우를 세면 정답의 개수만큼의 시간이 걸리지만, Dy 배열을 1번지부터 N번지 까지 채우는 것은 $O(N)$ 이라는 시간 복잡도면 충분하다!

다수의 테스트 케이스를 처리하기 전에 모든 N에 대해 정답을 구해놓자.

구현

```
static int[] Dy;

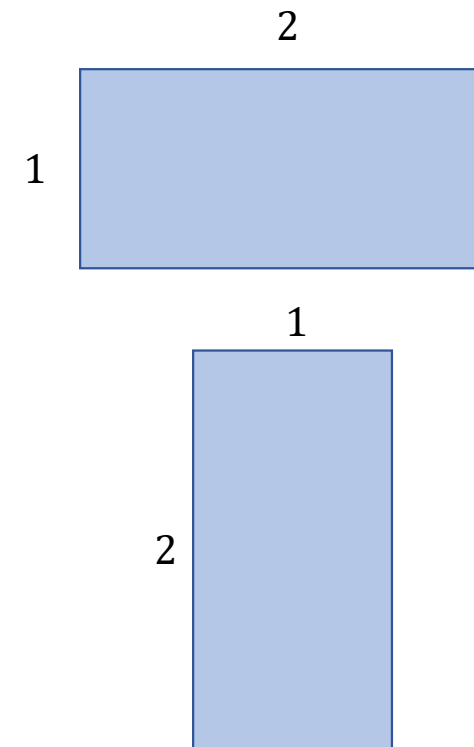
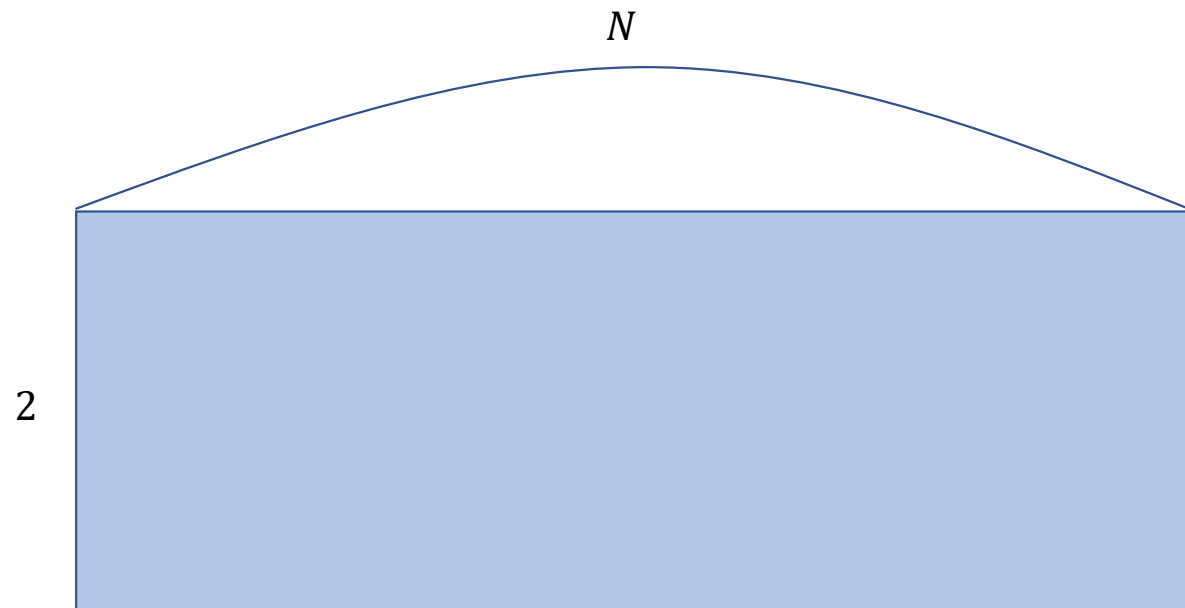
static void preprocess() {
    Dy = new int[15];
    // 초기값 구하기
    /* TODO */

    // 점화식을 토대로 Dy 배열 채우기
    /* TODO */
}
```


I BOJ 11726 – 2xN 타일링

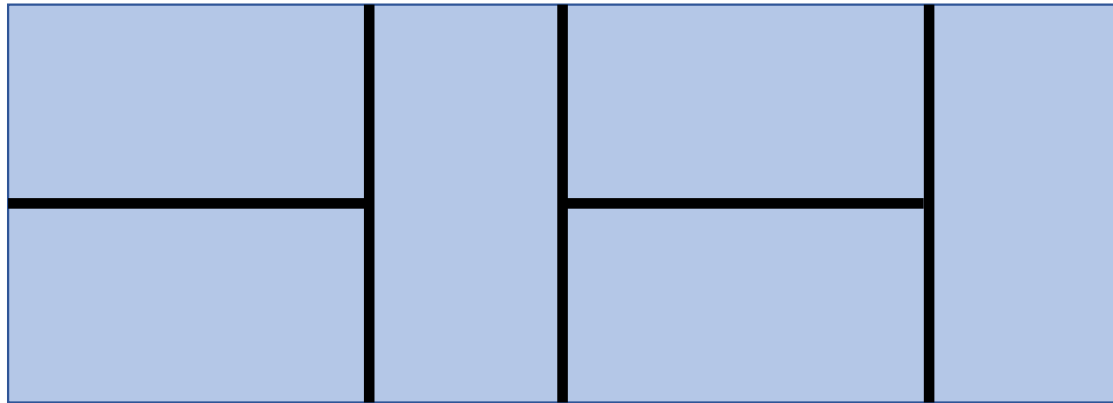
난이도: 2

$$1 \leq N \leq 1,000$$



I 완전 탐색 접근

완전 탐색 접근을 통해서 모든 경우를 직접 하나하나 찾아내 보자.



➔ N 이 커질수록 탐색해야 하는 경우가 엄청 많아진다.

I Dynamic Programming 접근

1) 풀고 싶은 가짜 문제 정의

Hint) 진짜 문제 먼저 써보기

진짜 문제 := 주어진 N 에 대해서 $2 \times N$ 타일링 경우의 수

가짜 문제 := $Dy[i] = 2 \times i$ 타일링 경우의 수

i	1	2	3	4	5	6	7
Dy[i]							

(레벨 1: 진짜 문제랑 똑같은 가짜 문제인 경우)

I Dynamic Programming 접근

2) **가짜 문제**를 풀면 **진짜 문제**를 풀 수 있는가?

Dy 배열을 가득 채울 수만 있다면? **진짜 문제**에 대한 대답은 $Dy[N]$ 이다.

i	1	2	3	4	5	6	7
Dy[i]							

I Dynamic Programming 접근

3) 초기값은 어떻게 되는가?

초기값: 쪼개지 않아도 풀 수 있는 “작은 문제”들에 대한 정답

i	1	2	3	4	5	6	7
Dy[i]	1	2					

I Dynamic Programming 접근

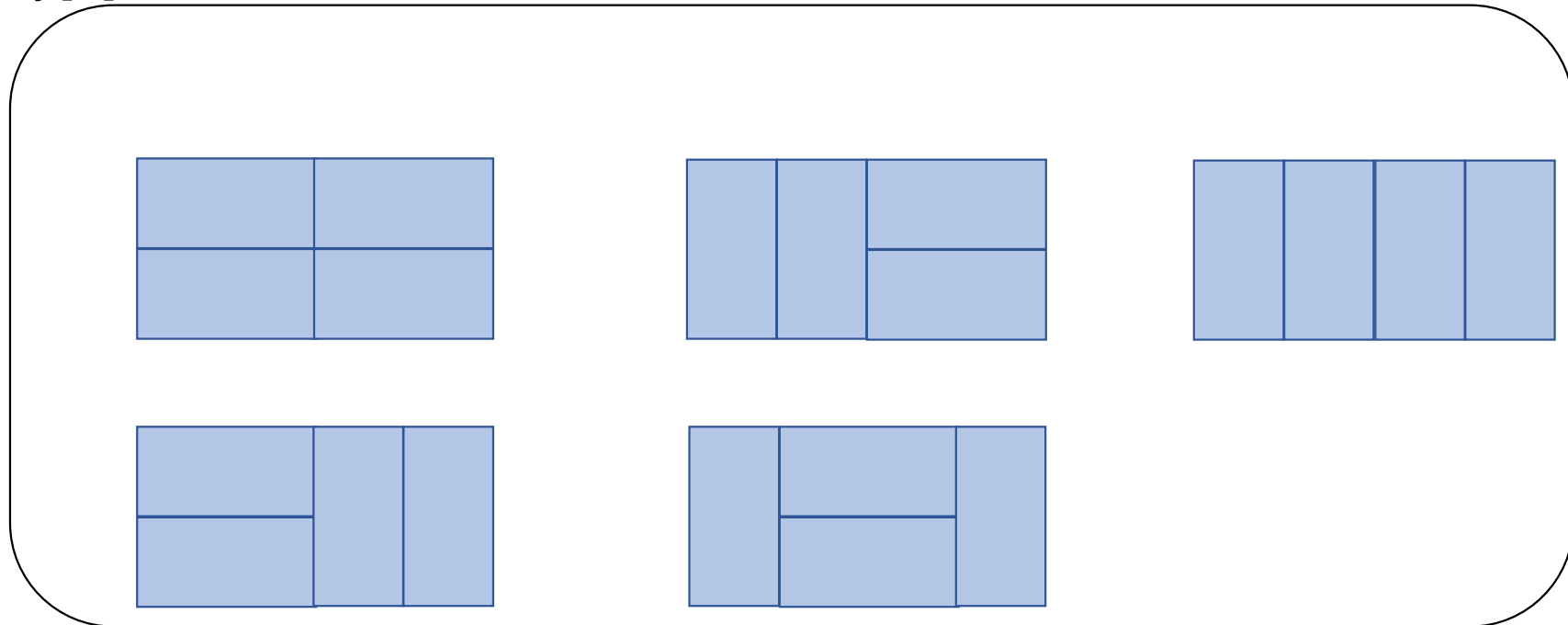
4) 점화식 구해내기

1. $Dy[i]$ 계산에 필요한 탐색 경우를 공통점끼리 묶어 내기 (Partitioning)
2. 묶어낸 부분의 정답을 Dy 배열을 이용해서 빠르게 계산해보기

I Dynamic Programming 접근

4-1) $Dy[i]$ 계산에 필요한 탐색 경우를 공통점끼리 묶어 내기 (Partitioning)

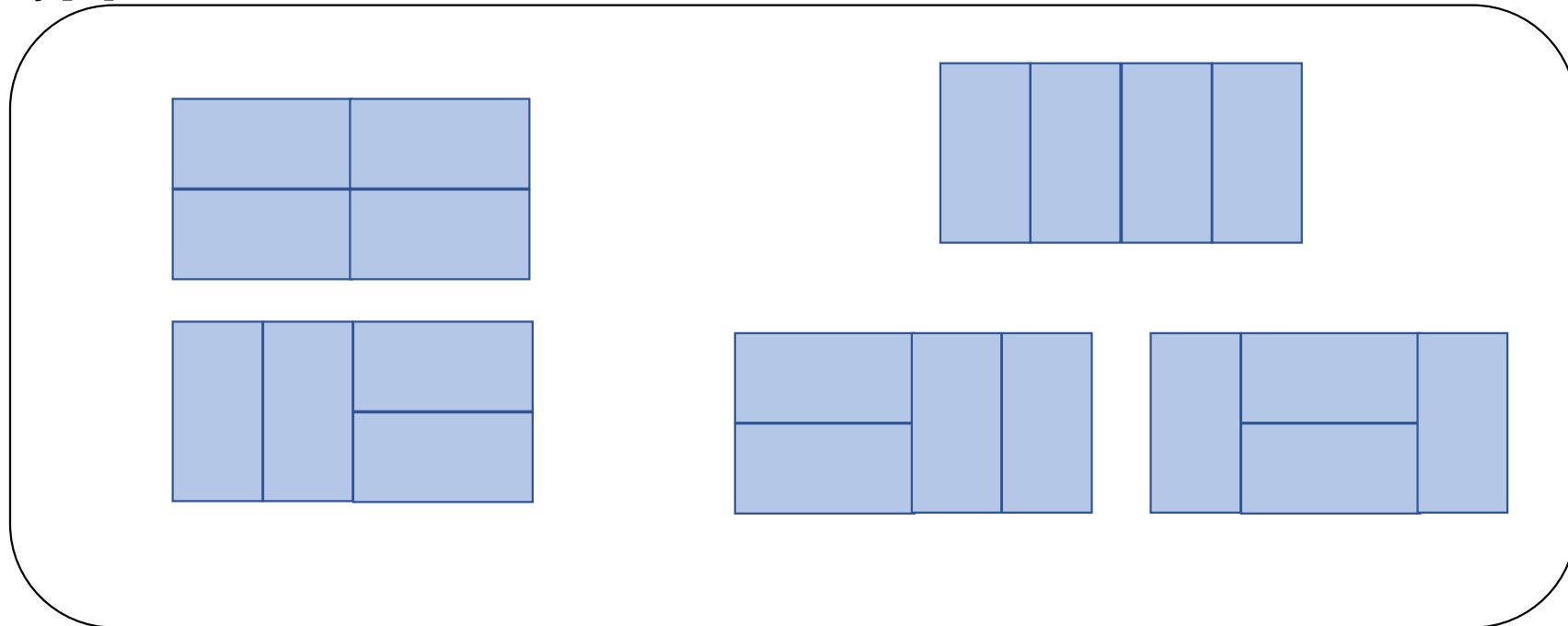
$Dy[4]$ 계산에 필요한 탐색 경우들



I Dynamic Programming 접근

4-1) $Dy[i]$ 계산에 필요한 탐색 경우를 공통점끼리 묶어 내기 (Partitioning)

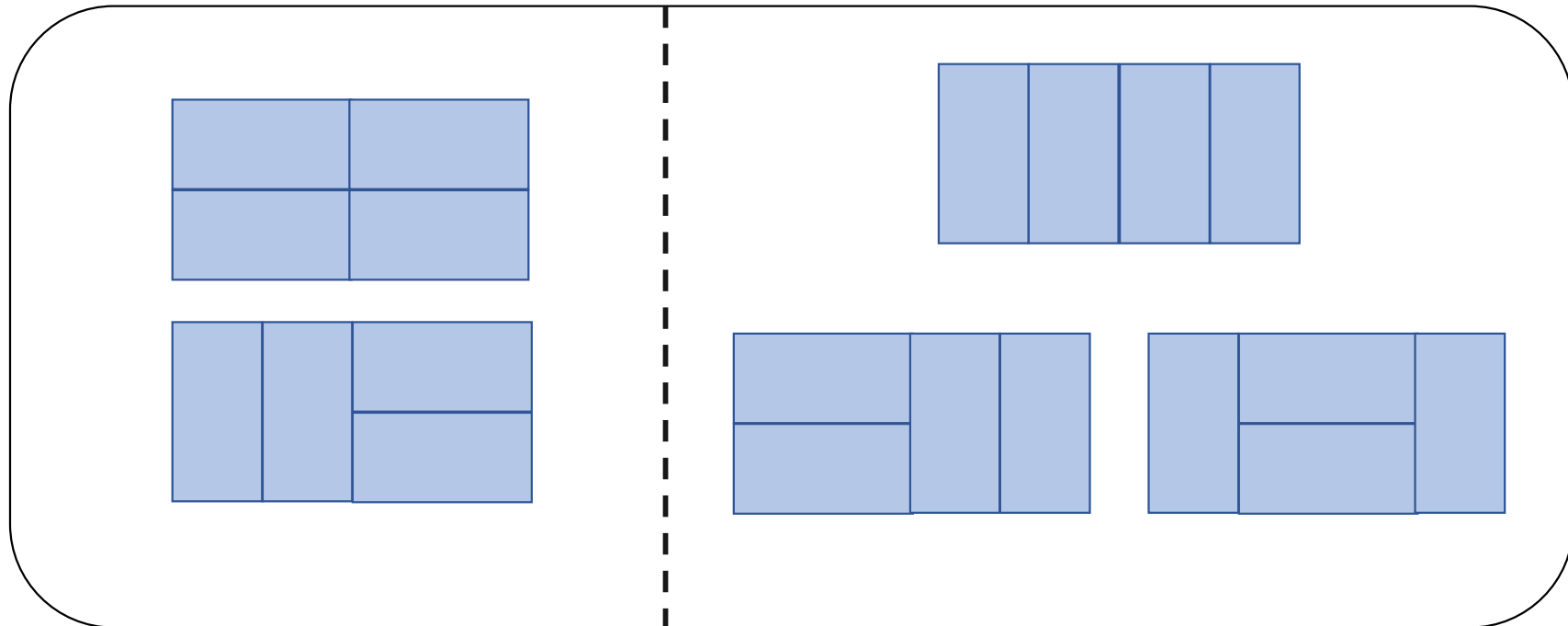
$Dy[4]$ 계산에 필요한 탐색 경우들



I Dynamic Programming 접근

4-2) 묶어낸 부분의 정답을 Dy 배열을 이용해서 빠르게 계산해보기

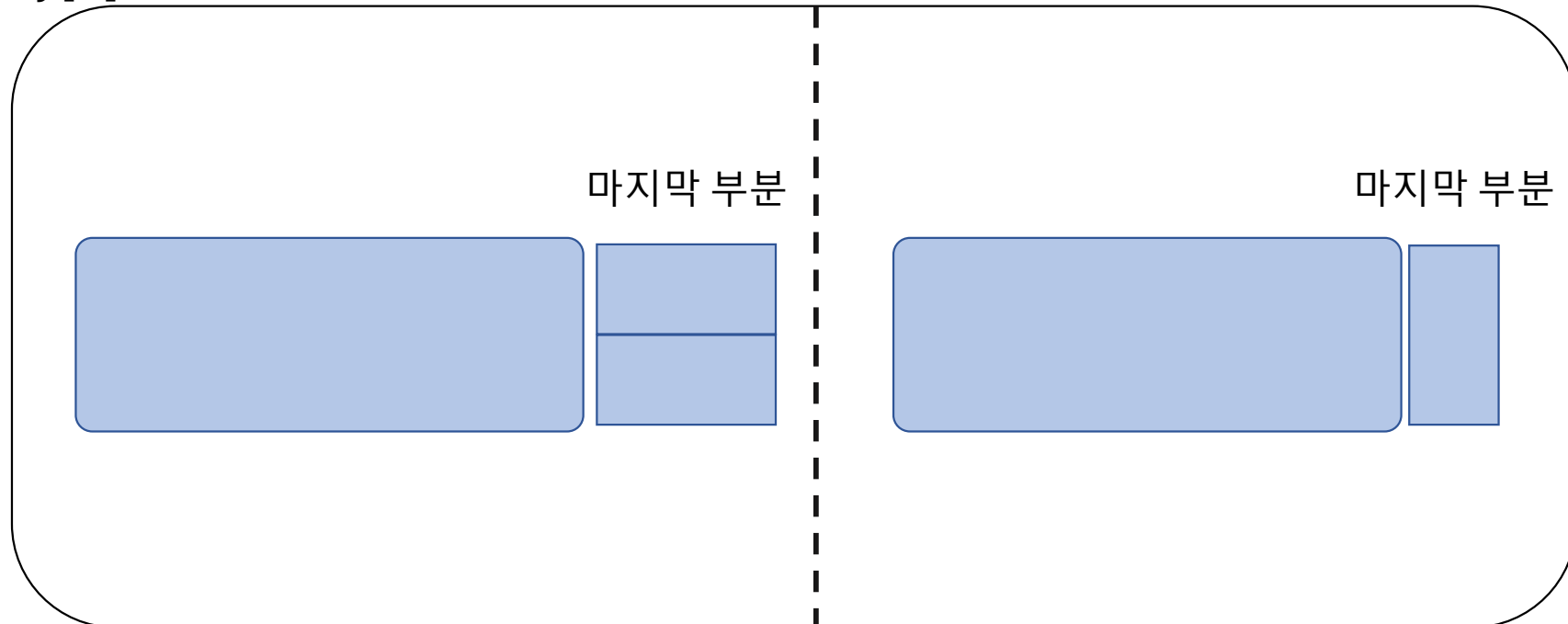
★ 전체 경우의 수 = (각 Partition의 경우의 수) 들의 합



I Dynamic Programming 접근

4-2) 묶어낸 부분의 정답을 Dy 배열을 이용해서 빠르게 계산해보기

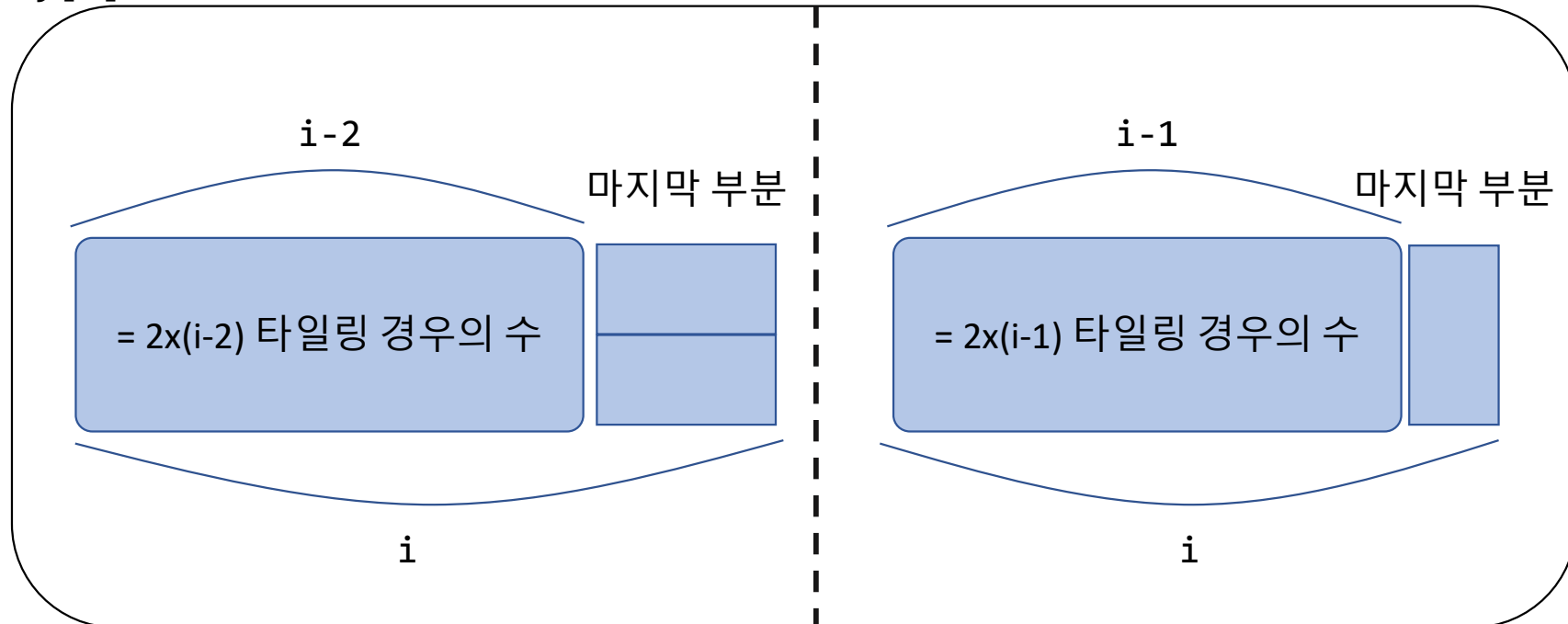
$Dy[i]$ 계산에 필요한 탐색 경우들



I Dynamic Programming 접근

4-2) 묶어낸 부분의 정답을 Dy 배열을 이용해서 빠르게 계산해보기



$Dy[i]$ 계산에 필요한 탐색 경우들



I Dynamic Programming 접근

4) 점화식 구해내기

 $Dy[i]$

= ($2x(i-2)$ 을 만들고  붙이는 경우의 수)
 + ($2x(i-1)$ 을 만들고  붙이는 경우의 수)

 $= Dy[i-1] + Dy[i-2]$

I Dynamic Programming 접근

4) 점화식 구해내기

$$Dy[i] = (Dy[i-1] + Dy[i-2]) \% 10,007$$

i	1	2	3	4	5	6	7
Dy[i]	1	2					

I 시간, 공간 복잡도 계산하기

완전 탐색을 통해 모든 경우를 세면 정답의 개수만큼
의 시간이 걸리지만, Dy 배열을 1번지부터 N번지 까지
채우는 것은 $O(N)$ 이라는 시간 복잡도면 충분하다!

구현

```
static void pro() {  
    Dy = new int[N + 1];  
    // 초기값 구하기  
    /* TODO */  
  
    // 점화식을 토대로 Dy 배열 채우기  
    /* TODO */  
  
    System.out.println(Dy[N]);  
}
```

I 연습 문제

- BOJ 1003 – 피보나치 함수
- BOJ 10870 – 피보나치 수 5
- BOJ 15988 – 1, 2, 3 더하기 3
- BOJ 15991 – 1, 2, 3 더하기 6
- BOJ 11052 – 카드 구매하기
- BOJ 2011 – 암호 코드

이외의 추천 문제가 추가되면 Github 자료에 코드 업로드