

알튜비튜

동적 계획법

오늘은 '다이나믹 프로그래밍'이라고도 불리는 동적 계획법 알고리즘에 대해 배웁니다.
과거에 구한 해를 현재 해를 구할 때 활용하는 알고리즘이죠. 문제에 많이 나오는 굉장히 중요한 알고리즘 중 하나예요.

동적 계획법

- 특정 범위까지의 값을 구하기 위해 이전 범위의 값을 활용하여 효율적으로 값을 얻는 기법
- 이전 범위의 값을 저장(Memoization)함으로써 시간적, 공간적 효율 얻음

프로그래머스 : 가장 큰 정사각형 찾기 Lv.2

문제

- 0과 1로 채워진 표에서,
1로 이루어진 가장 큰 정사각형의 넓이를 구하는 문제

제한 사항

- 표는 2차원 배열
- 행과 열의 크기: 1,000 이하의 자연수

가장 큰 정사각형 찾기



0	1	1	1
1	1	1	1
1	1	1	1
0	0	0	1

가장 큰 정사각형 찾기

0	1	1	1
1	1	1	1
1	1	1	1
0	0	0	1

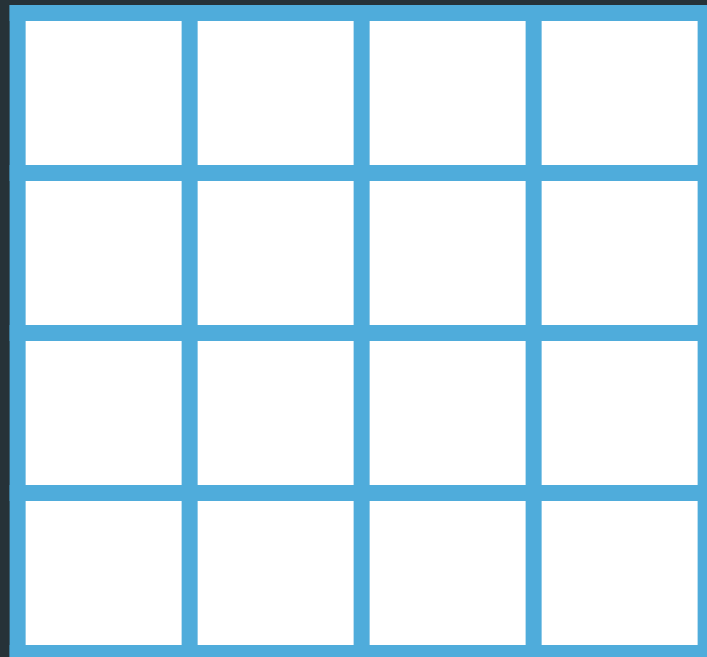
→ 9

점화식

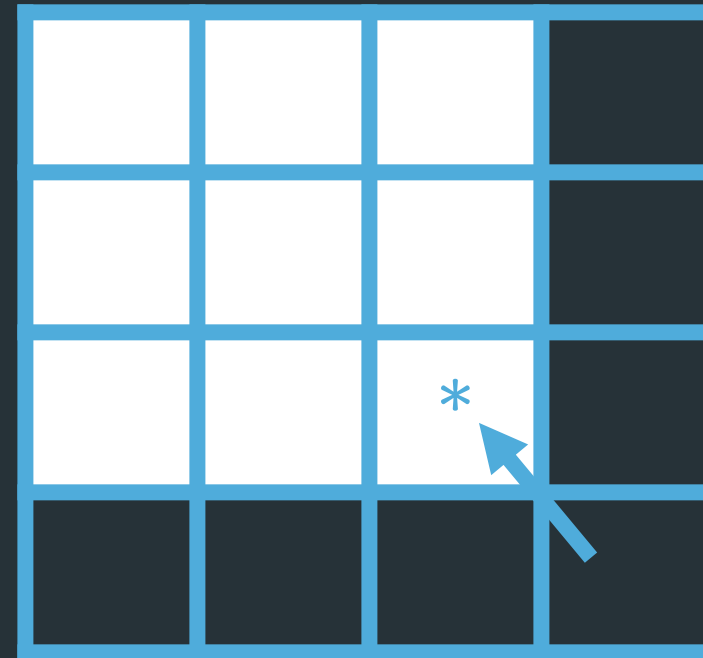
- $dp[i][j] = (i, j)$ 를 우측 하단 꼭짓점으로 하는 가장 큰 정사각형의 한 변의 길이

점화식을 세워봅시다

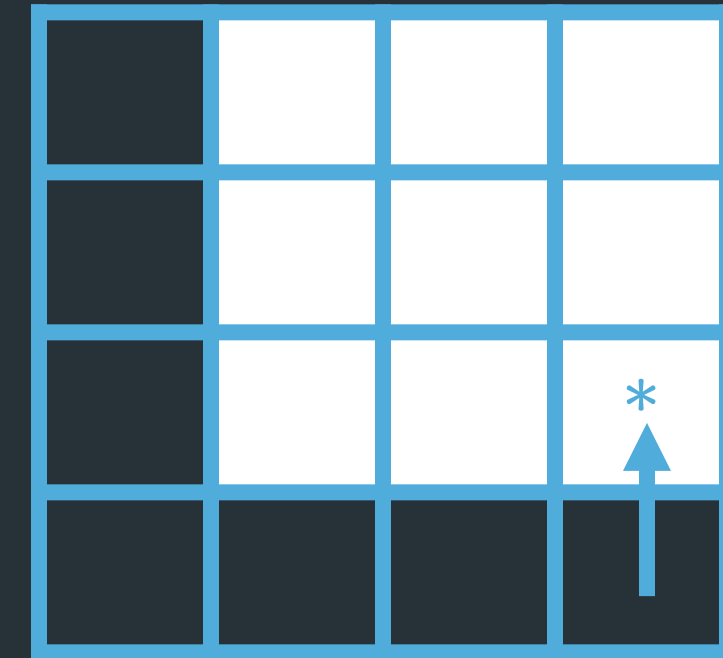
접근



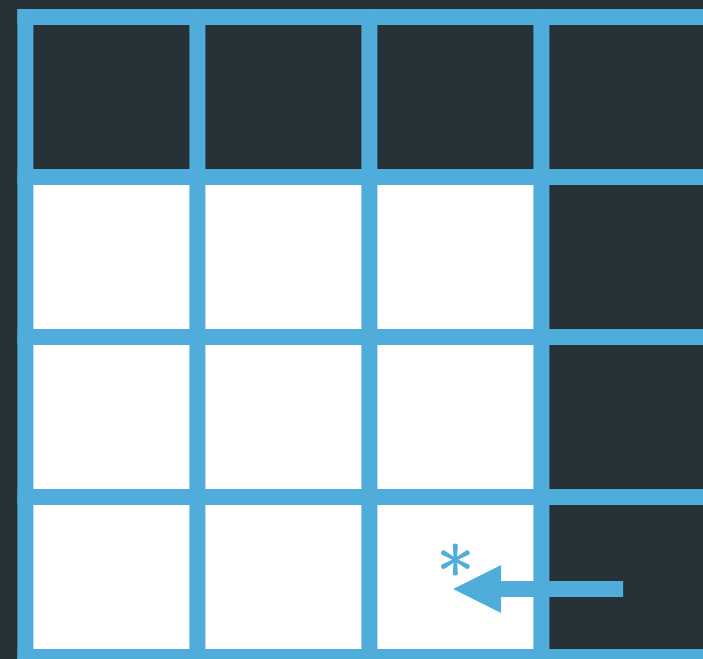
● 4 X 4 정사각형



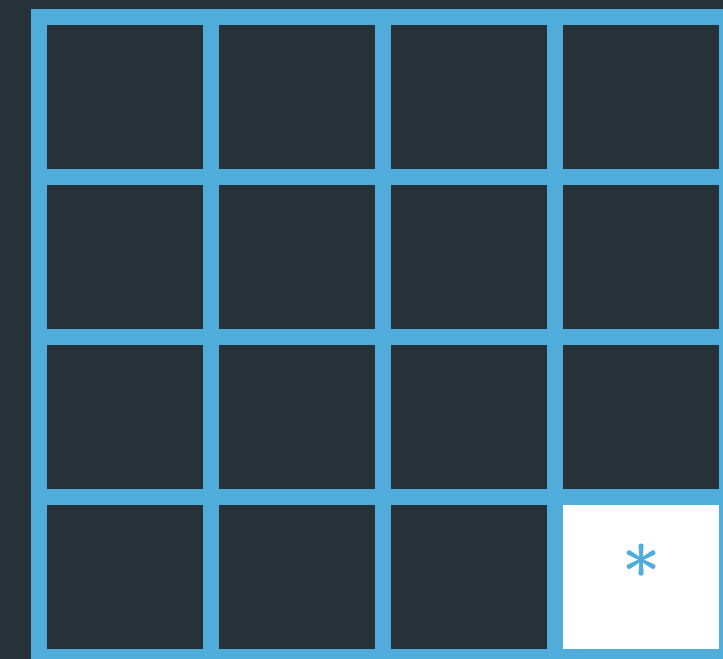
● ↖ 3 X 3 정사각형



● ↑ 3 X 3 정사각형



● ← 3 X 3 정사각형



● 꼭짓점 1 X 1 정사각형

점화식을 세워봅시다

1	1	1	1	0
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
0	1	0	1	0

점화식을 세워봅시다

1	1	1	1	0
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
0	1	0	1	0

점화식을 세워봅시다

1	1	1	1	0
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
0	1	0	1	0

점화식을 세워봅시다

1	1	1	1	0
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
0	1	0	1	0

점화식을 세워봅시다

1	1	1	1	0
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
0	1	0	1	0

점화식을 세워봅시다

접근

- $dp[i-1][j] = 2$
→ $dp[i][j]$ 는 3보다 커질 수 없음

1	1	1	1	0
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
0	1	0	1	0

점화식을 세워봅시다

접근

- $dp[i-1][j-1] = 3$
→ $dp[i][j]$ 는 4보다 커질 수 없음

1	1	1	1	0
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
0	1	0	1	0

점화식을 세워봅시다

접근

- $dp[i][j-1] = 4$
→ $dp[i][j]$ 는 5보다 커질 수 없음

1	1	1	1	0
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
0	1	0	1	0

점화식을 세워봅시다

1	1	1	1	0
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
0	1	0	1	0

- $dp[i-1][j] = 2$
→ $dp[i][j]$ 는 3보다 커질 수 없음

1	1	1	1	0
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
0	1	0	1	0

- $dp[i-1][j-1] = 3$
→ $dp[i][j]$ 는 4보다 커질 수 없음

1	1	1	1	0
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
0	1	0	1	0

- $dp[i][j-1] = 4$
→ $dp[i][j]$ 는 5보다 커질 수 없음

점화식을 세워봅시다

1	1	1	1	0
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
0	1	0	1	0

- $dp[i-1][j] = 2$
→ $dp[i][j]$ 는 3보다 커질 수 없음

1	1	1	1	0
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
0	1	0	1	0

- $dp[i-1][j-1] = 3$
→ $dp[i][j]$ 는 4보다 커질 수 없음

1	1	1	1	0
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
0	1	0	1	0

- $dp[i][j-1] = 4$
→ $dp[i][j]$ 는 5보다 커질 수 없음

점화식을 세워봅시다

1	1	1	1	0
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
0	1	0	1	0

1	1	1	1	0
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
0	1	0	1	0

1	1	1	1	0
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
0	1	0	1	0

→ 가장 작은 값이 $dp[i][j]$ 를 결정한다!

점화식

- $dp[i][j]$ = (i, j) 를 우측 하단 꼭짓점으로 하는 가장 큰 정사각형의 한 변의 길이
- ∴ board[i][j] = 1인 경우:
 $dp[i][j] = \min(dp[i-1][j-1], dp[i-1][j], dp[i][j-1]) + 1$
board[i][j] = 0인 경우:
 $dp[i][j] = 0$

/<> 9084번 : 동전 - Gold 5

문제

- 동전의 종류가 주어질 때,
주어진 금액을 만드는 모든 방법을 세는 문제

제한 사항

- 테스트케이스 개수 T : $1 \leq T \leq 10$
- 동전의 가지 수 N : $1 \leq N \leq 20$
- 각 동전의 금액: 1원 ~ 10,000원
- 만들어야 할 금액 M : $1 \leq M \leq 10,000$

예제 입력

```
3
2
1 2
1000
3
1 5 10
100
2
5 7
22
```

예제 출력

```
501
121
1
```

문제

- 동전의 종류가 주어질 때,
주어진 금액을 만드는 모든 방법을 세는 문제

접근

- i 원을 만들려면, $(i - \text{coin})$ 원을 만든 다음 coin원짜리 동전을 추가하면 됨
- 가능한 모든 coin을 고려해주어야 함

- 냅색 문제와 비슷하게 2차원 DP로도, 1차원 DP로도 풀 수 있어요!

왜 2차원 DP?

- 1차원 DP로 하면 안 되는 이유는?
 - 그 전 물품까지의 정보만 사용해야 하기 때문
- 지금처럼 증가하면서 검사할 때 1차원 DP를 사용하게 되면?
 - 해당 물품을 또 사용하는 경우 생길 수 있음!
 - 따라서 2차원을 사용하며 각 물품을 행으로 구분해서 중복 방지

냅색 1차원 DP 풀이

기존 증가 방식

dp



감소 방식

dp



뒤에서부터 채워지므로 중복 사용 방지

왜 2차원 DP?

- 1차원 DP로 하면 안 되는 이유는?
→ 그 전 물품까지의 정보만 사용해야 하기 때문
- 지금처럼 증가하면서 검사할 때 1차원 DP를 사용하게 되면?
→ 해당 물품을 또 사용하는 경우 생길 수 있음!
→ 따라서 2차원을 사용하며 각 물품을 행으로 구분해서 중복 방지
- 각 동전의 개수는 무한
→ 중복 사용 가능 + 모든 방법의 수를 세려면 오히려 중복을 고려해야 함!

- 목표 금액: i 원
- 현재 고려하고 있는 동전의 금액: coin 원
- 두 가지 경우로 나눌 수 있음
- 1. 현재 동전을 사용하지 않는 경우 $\rightarrow i$ 원을 만들어야 함
- 2. 현재 동전을 사용하는 경우 $\rightarrow (i - \text{coin})$ 원을 만들어야 함

$\rightarrow DP[i] = DP[i] + DP[i - \text{coin}]$

/<> 20923번 : 숫자 할리갈리 게임 - Silver 1

문제

1. 게임 시작 시 도도와 수연이는 N장의 카드로 구성된 덱을 받음 (그라운드는 비어 있음)
2. 도도와 수연이는 차례대로 덱의 가장 위에 있는 카드를 그라운드에 내려놓음
3. 종을 치는 사람이 그라운드의 카드 더미를 모두 가져감
 - a. 수연: 그라운드 위의 카드의 숫자 합이 5가 될 때 종을 침
 - b. 도도: 카드의 숫자가 5가 나올 때 종을 침
4. 카드 더미를 가져갈 때는 상대방의 그라운드, 자신의 그라운드 순으로 가져와 자신의 덱 아래에 합침
5. M번 게임을 진행한 후 더 많은 카드를 가진 사람이 승리, 카드의 수가 같은 경우 비김
 - a. 게임 도중 덱에 있는 카드가 다 떨어지면 상대가 승리
6. 2~4 과정을 진행하는 것을 한 번 진행한 것으로 볼 때 M번 진행 후 승리한 사람은?

/<> 20923번 : 숫자 할리갈리 게임 - Silver 1

제한 사항

- 도도와 수연이가 가지는 카드의 개수의 범위는 $1 \leq N \leq 30,000$
- 게임 진행 횟수의 범위는 $1 \leq M \leq 2,500,000$
- 각각의 카드는 1이상 5이하의 자연수가 적혀 있음

예제 입력1

```
10 12
1 2
2 2
1 2
2 3
3 1
2 2
2 5
2 1
5 1
2 3
```

예제 출력1

```
do
```

예제 입력2

```
1 1
5 2
```

예제 출력2

```
su
```

예제 입력3

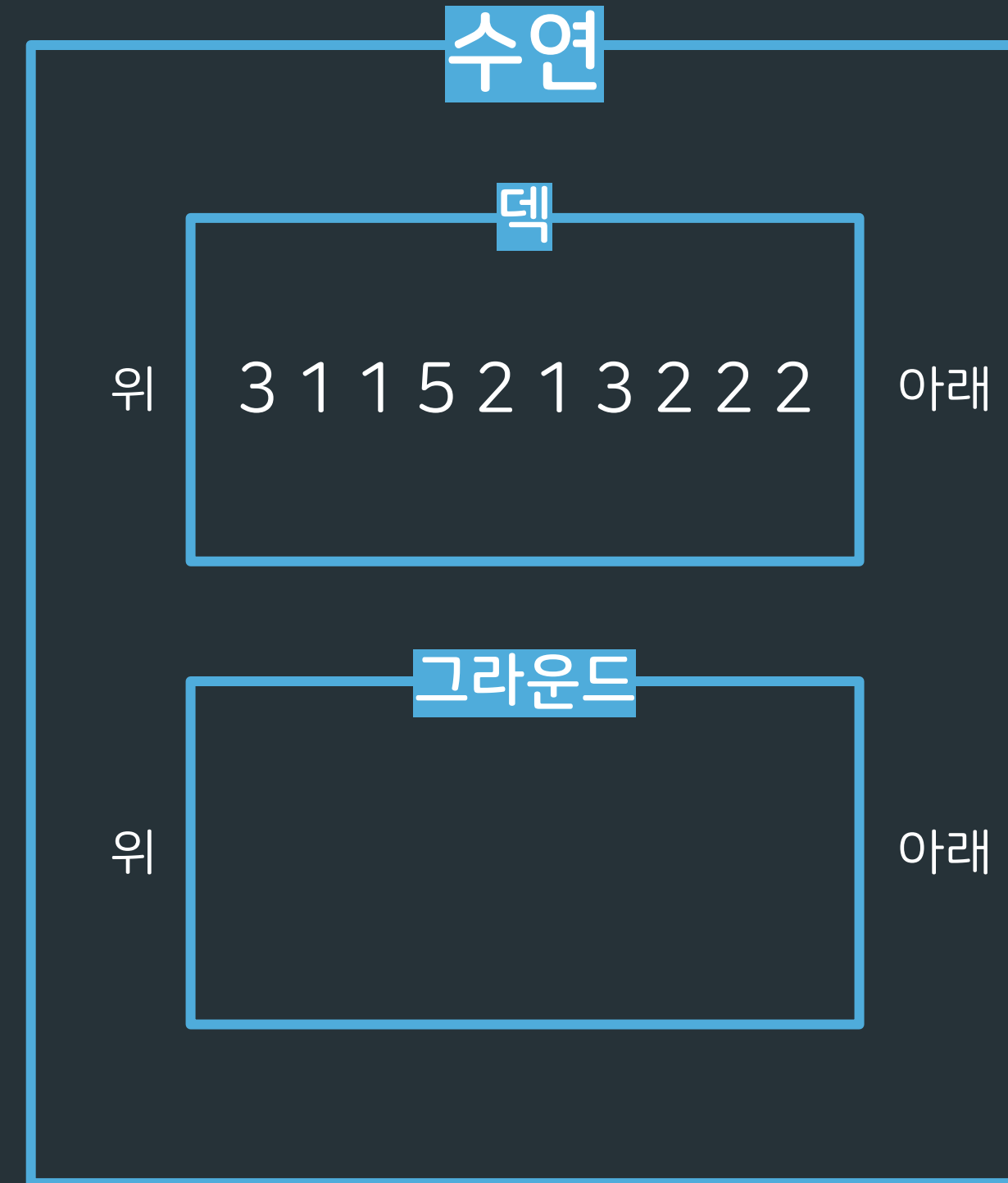
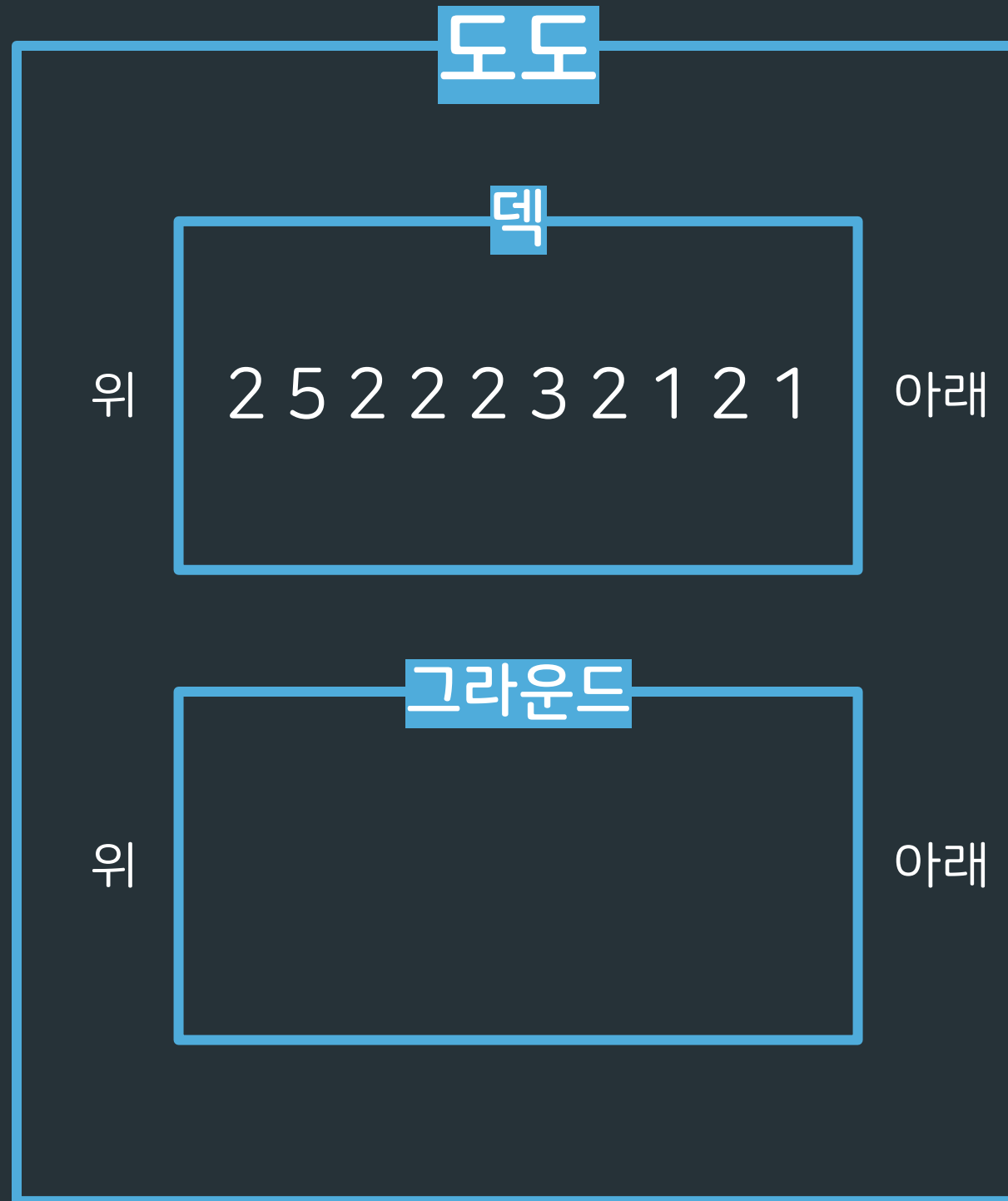
```
3 4
1 2
2 2
1 1
```

예제 출력3

```
dosu
```

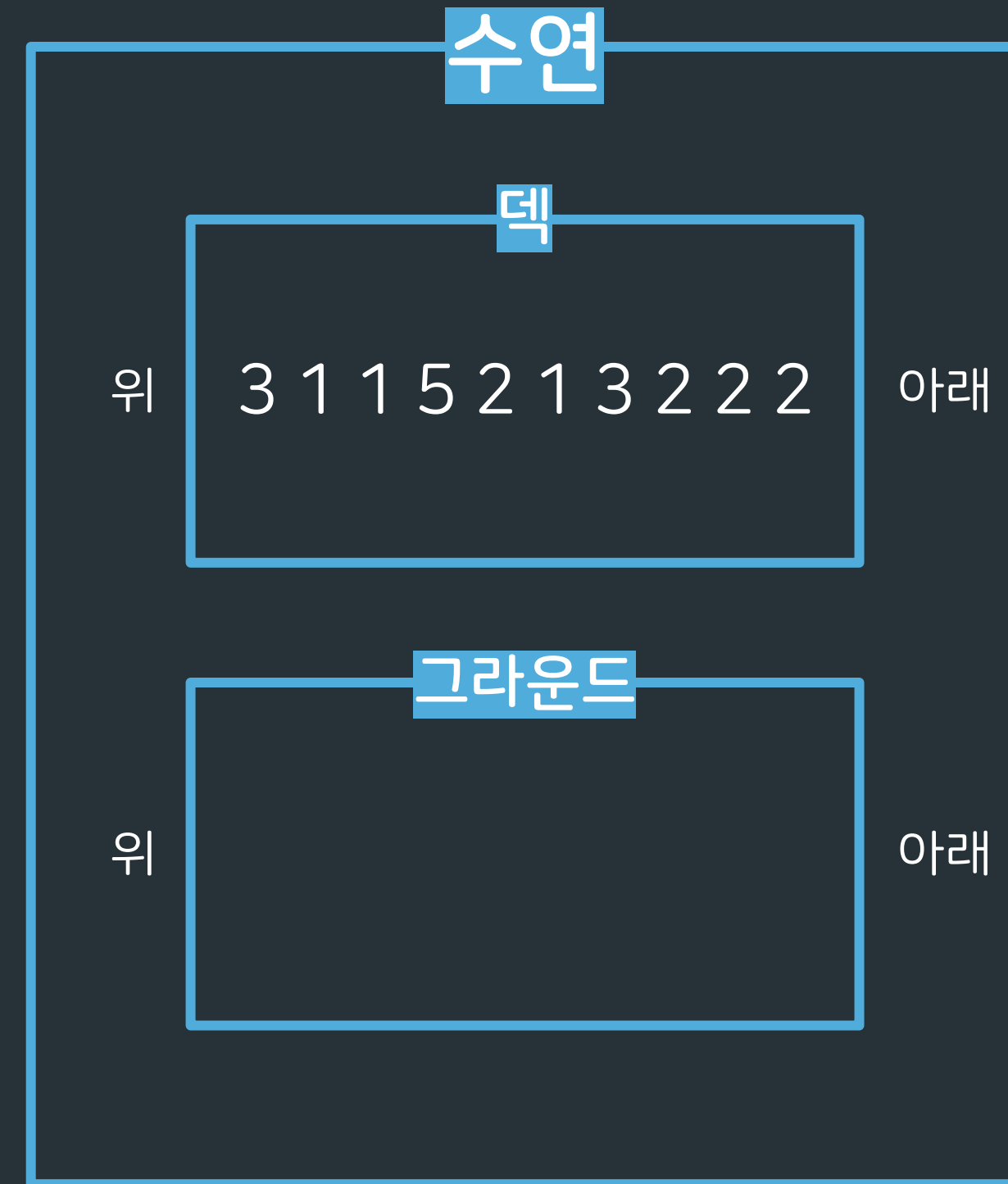
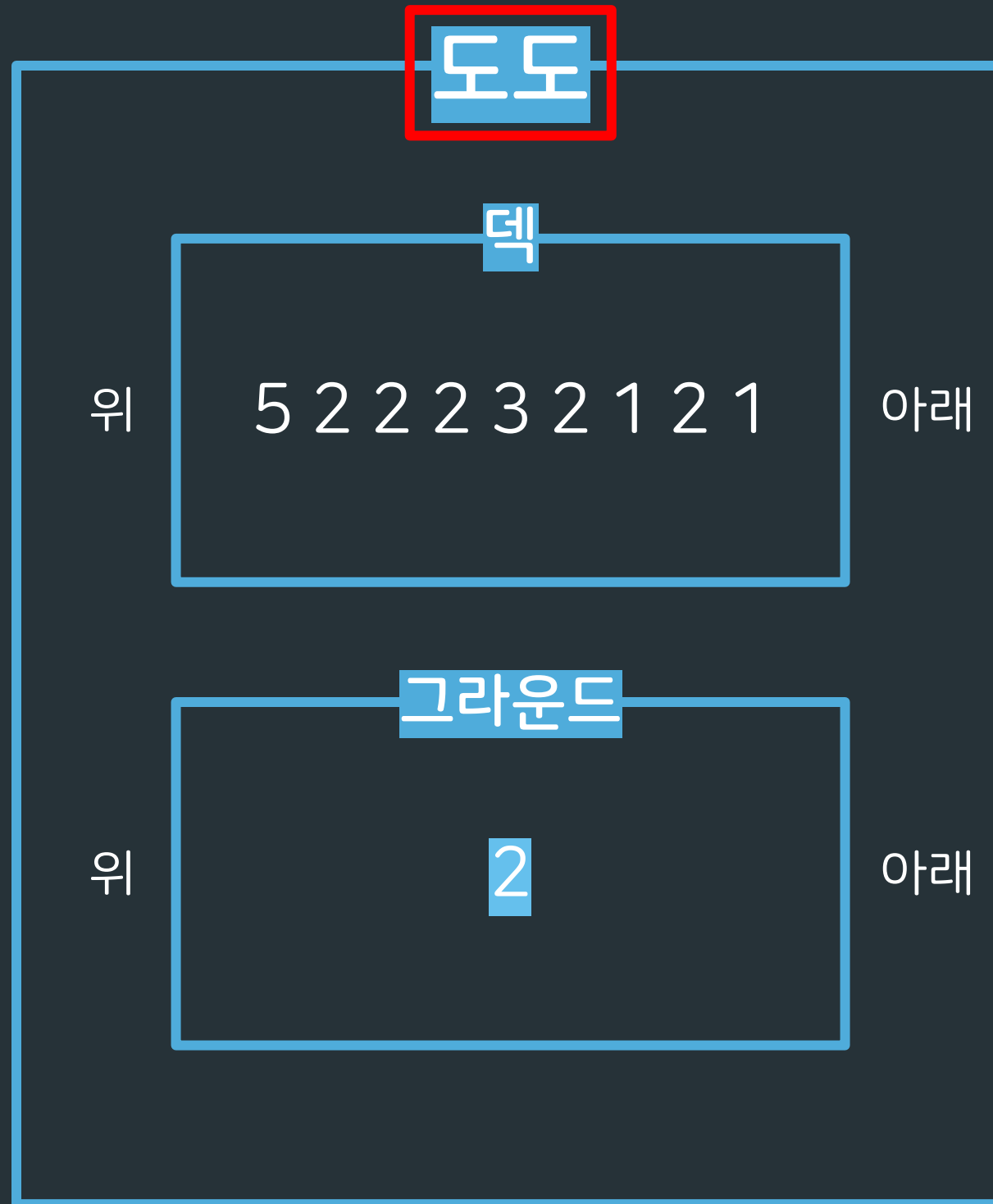
순서대로 게임을 진행해보자

M = 0



순서대로 게임을 진행해보자

M = 1



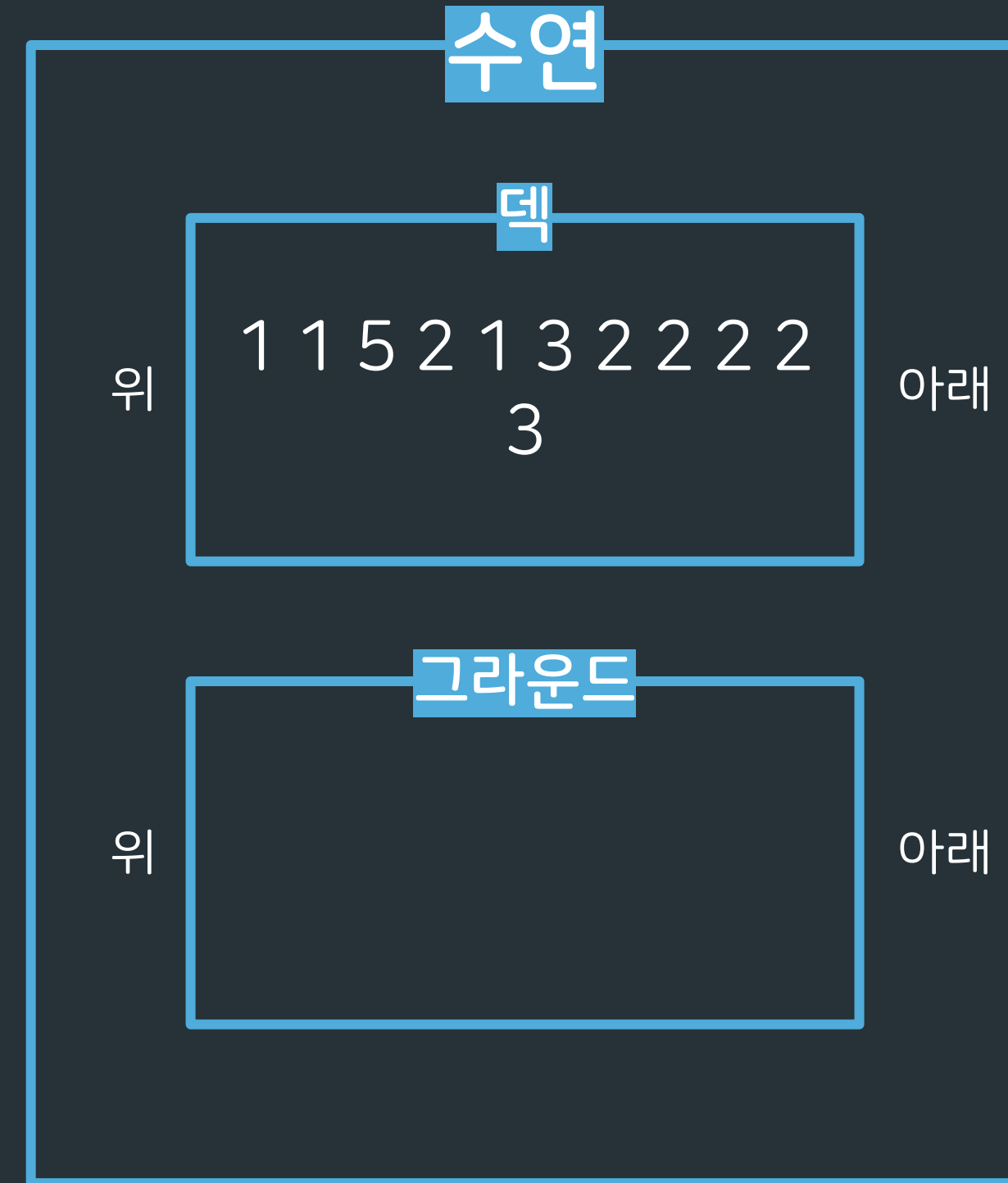
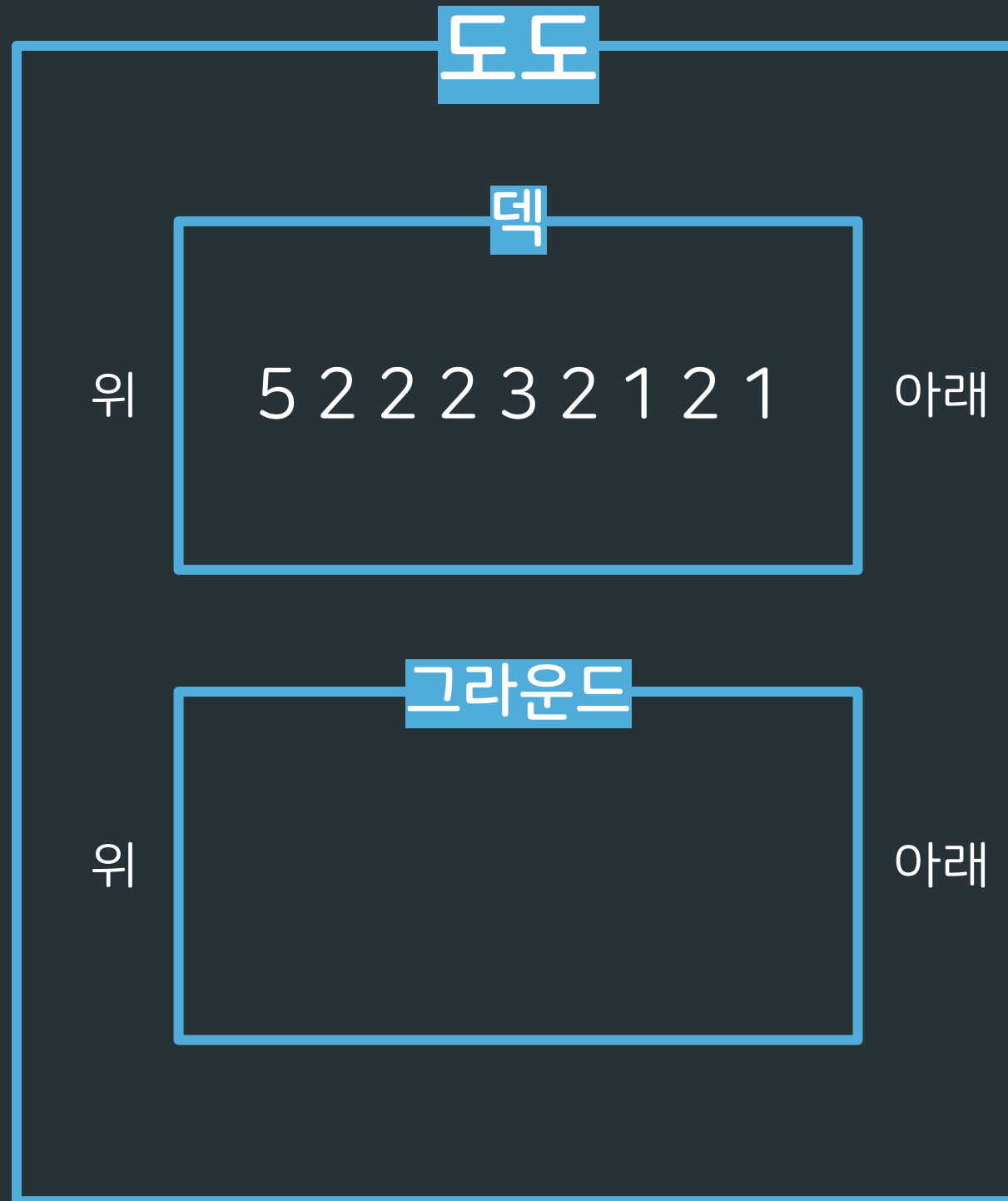
순서대로 게임을 진행해보자

M = 2



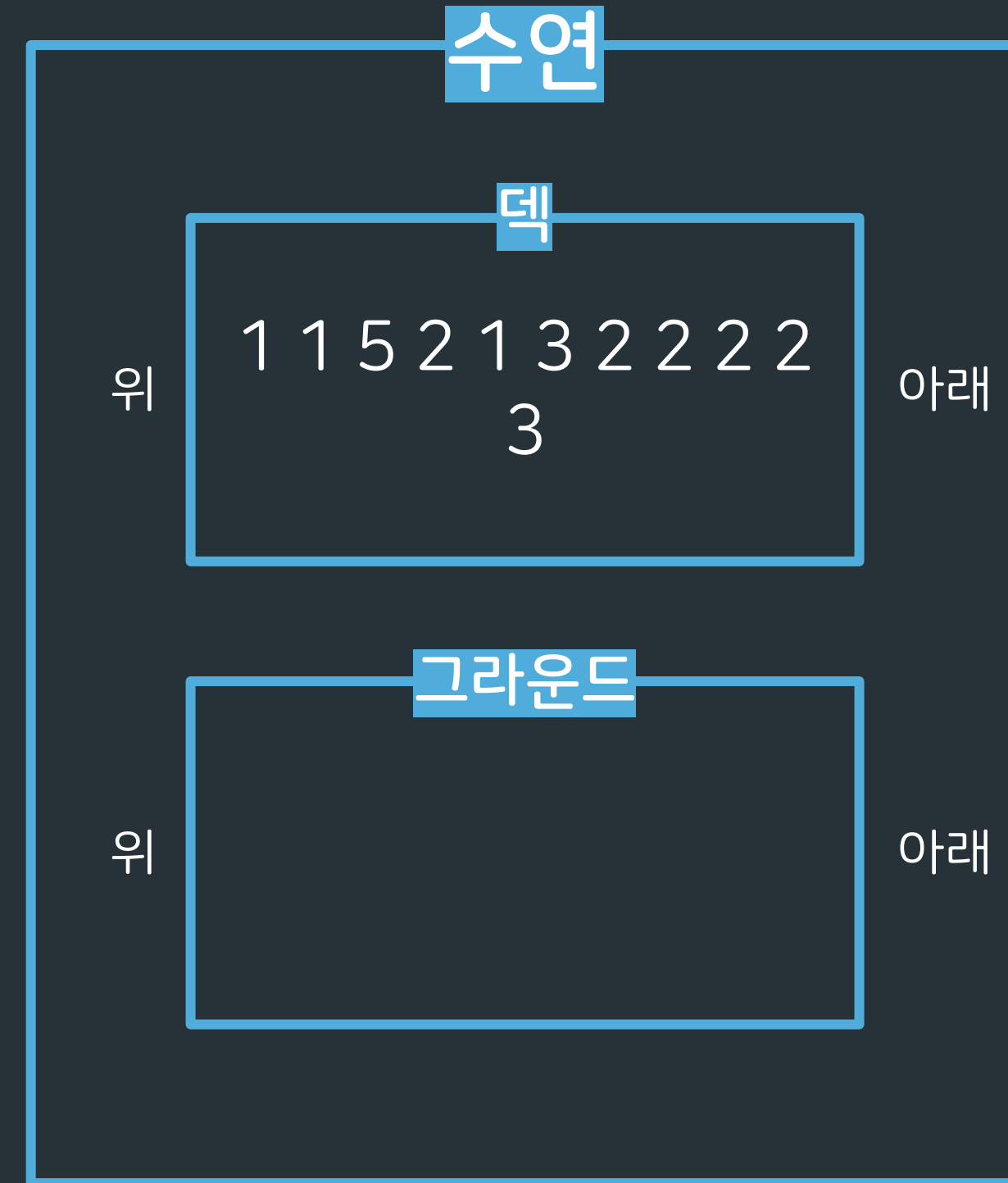
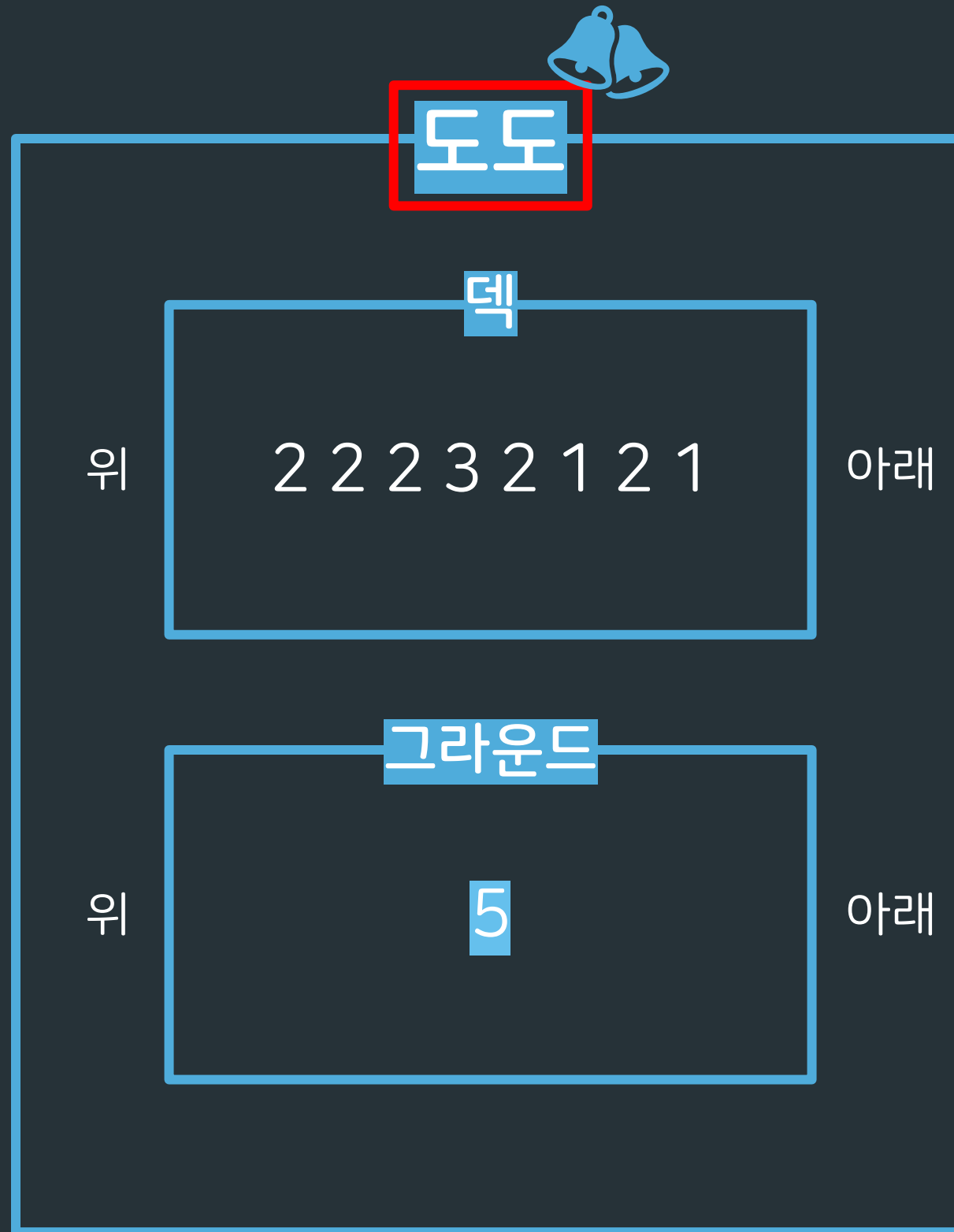
순서대로 게임을 진행해보자

M = 2



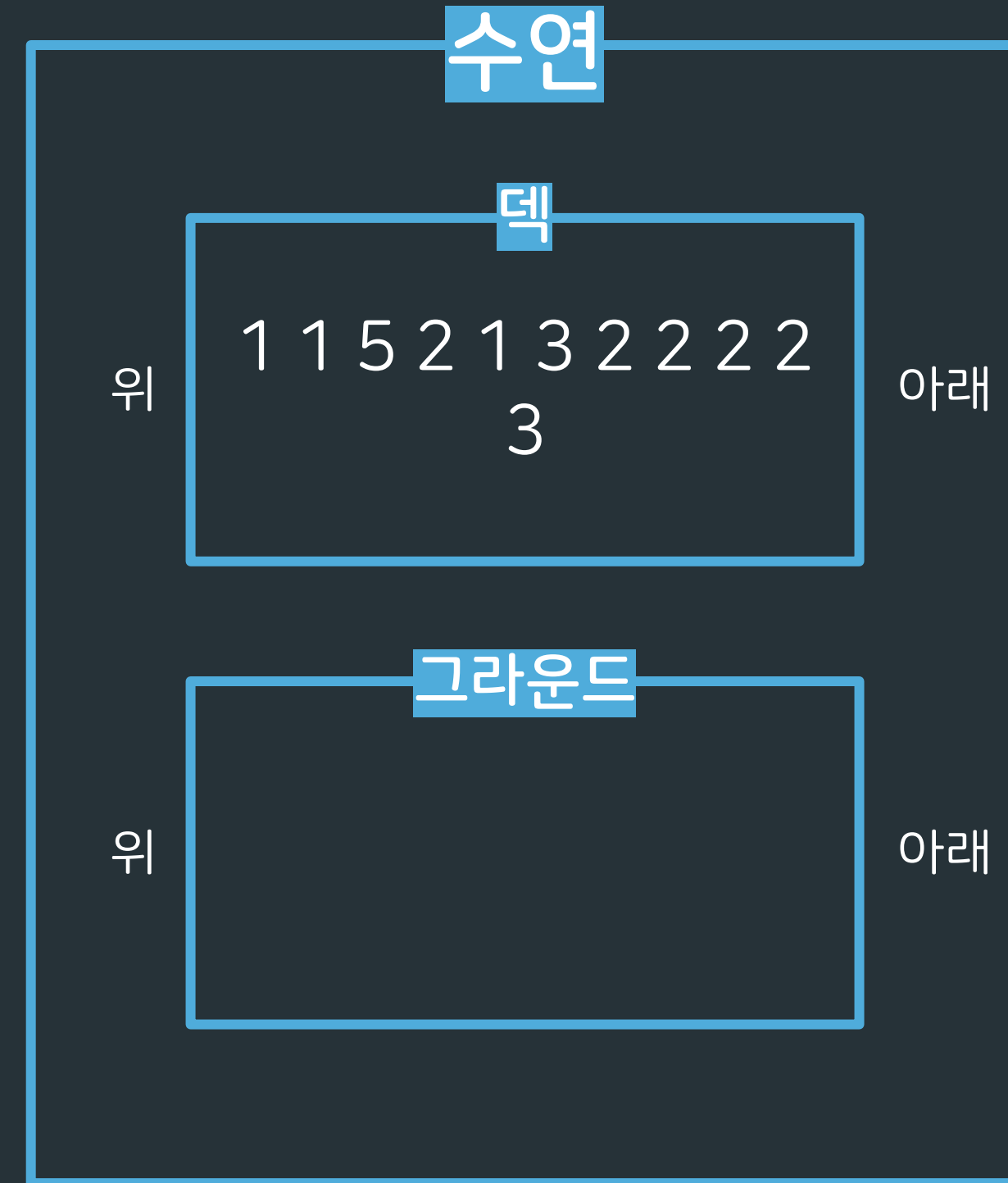
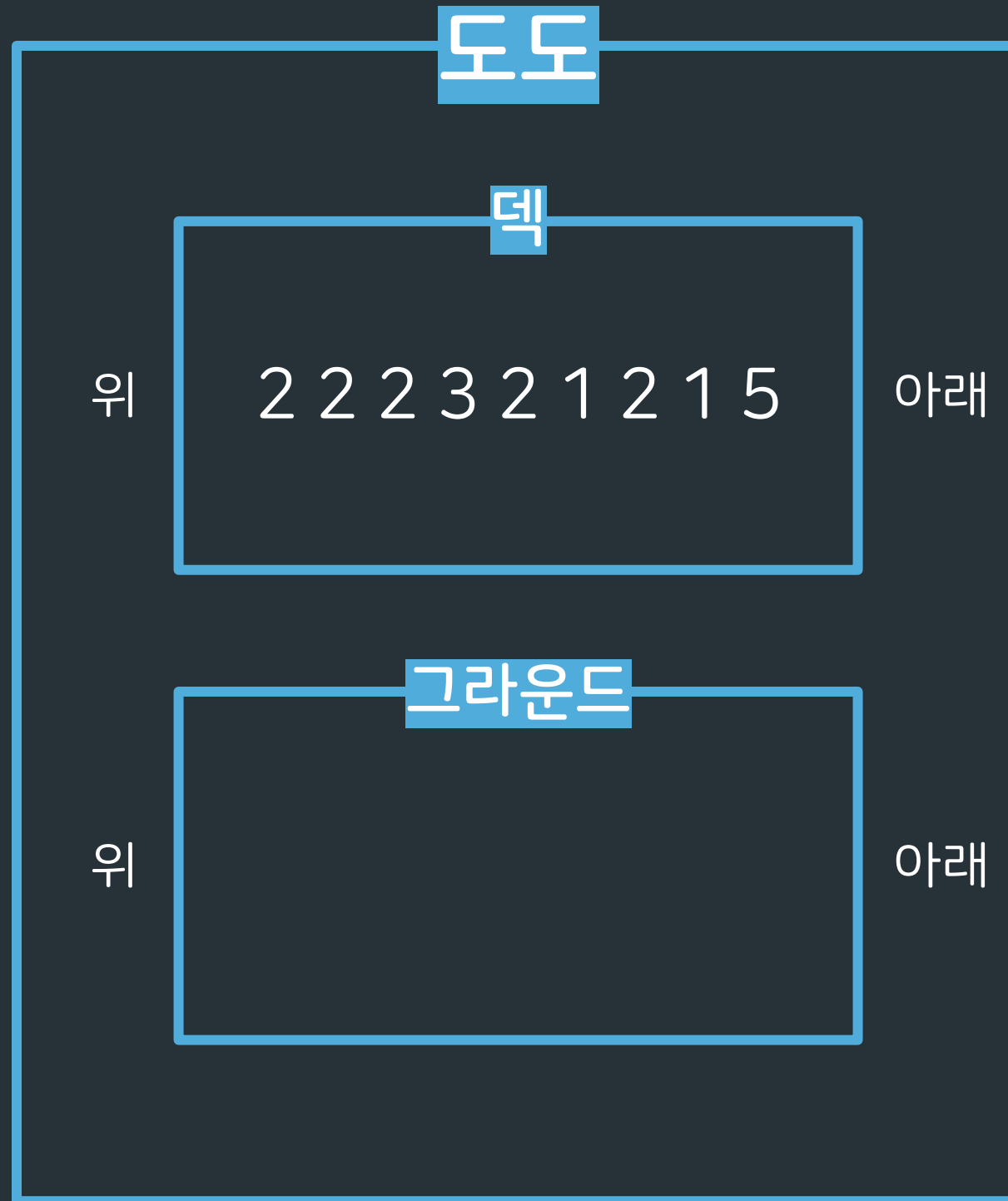
순서대로 게임을 진행해보자

M = 3



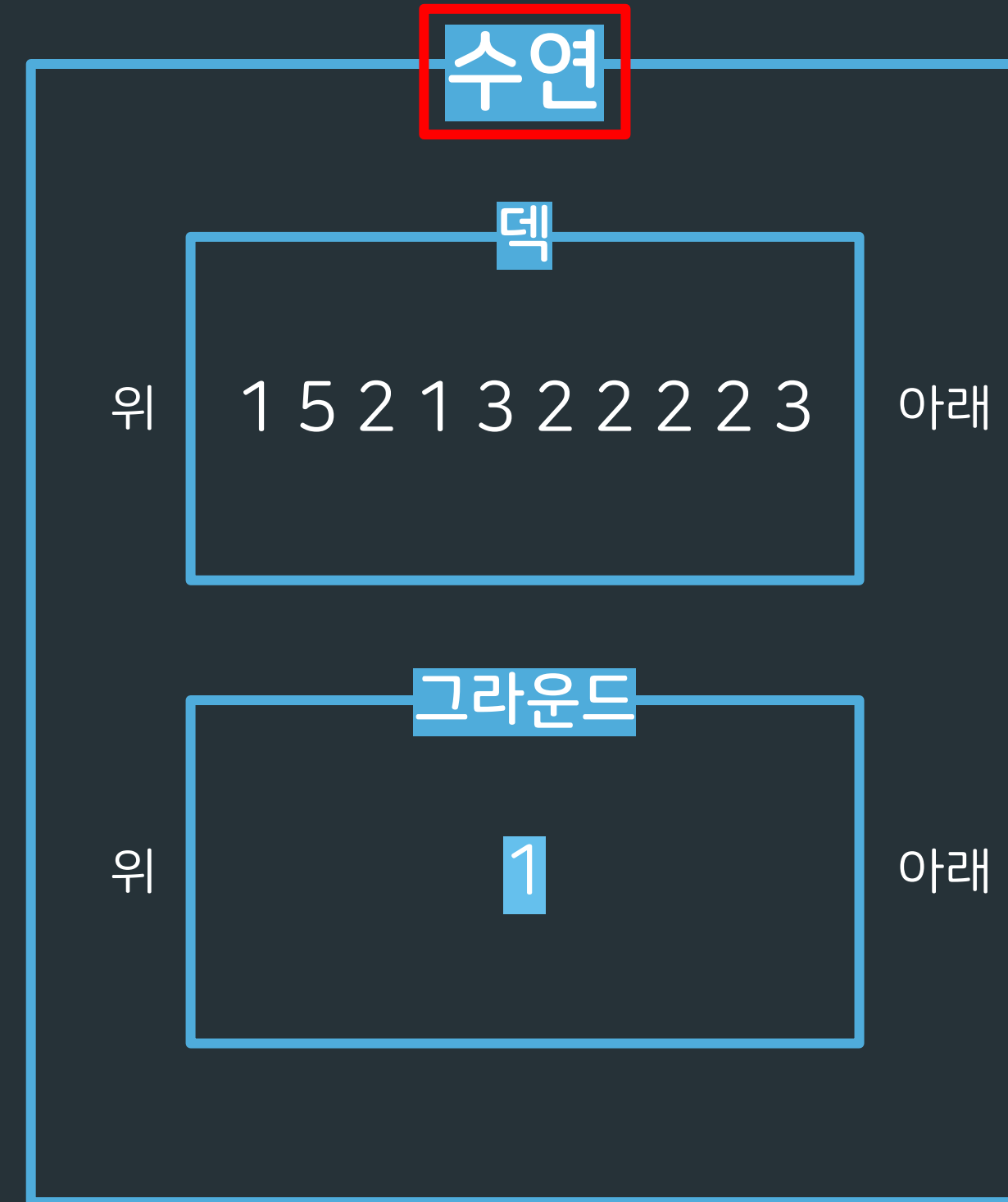
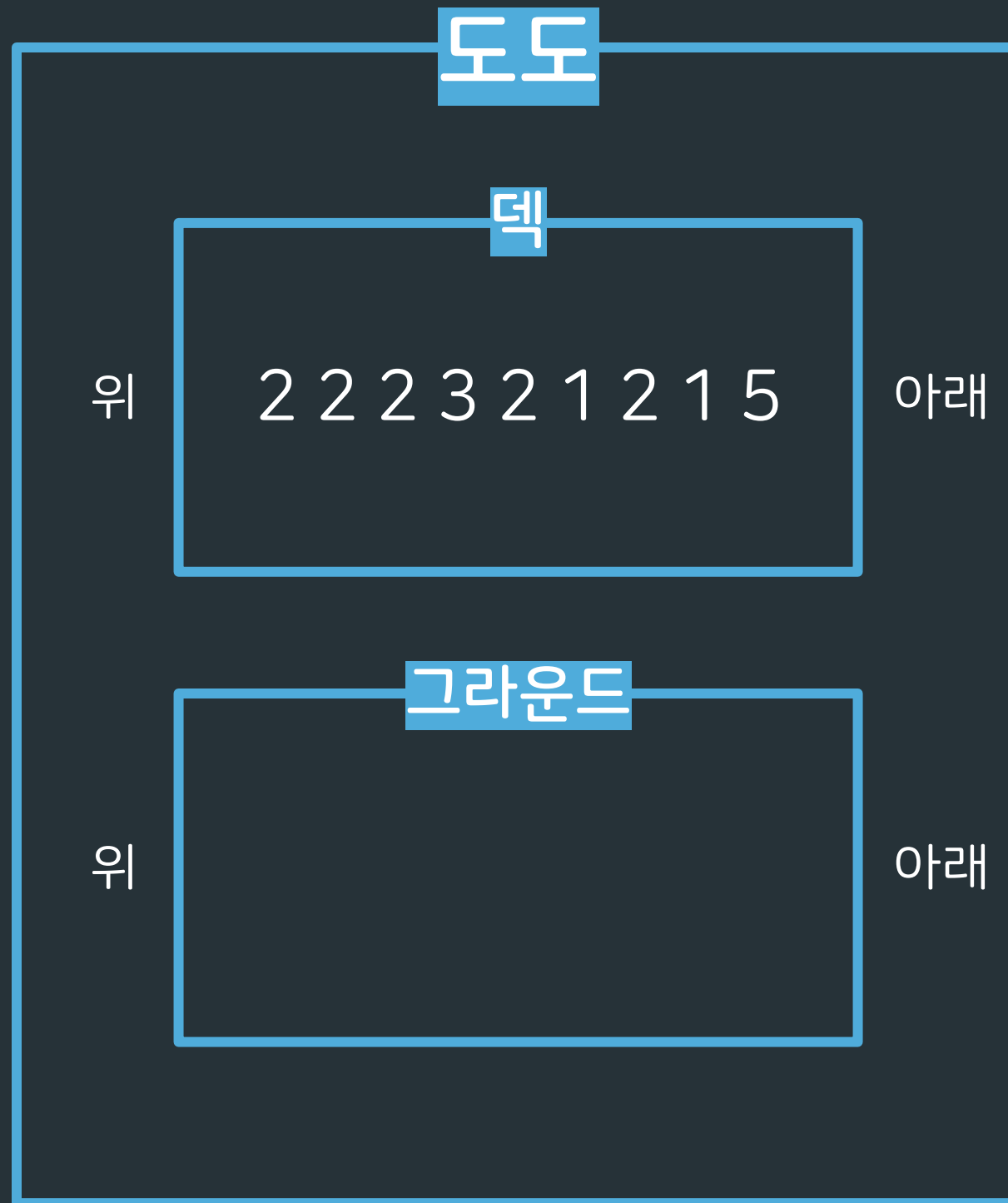
순서대로 게임을 진행해보자

M = 3



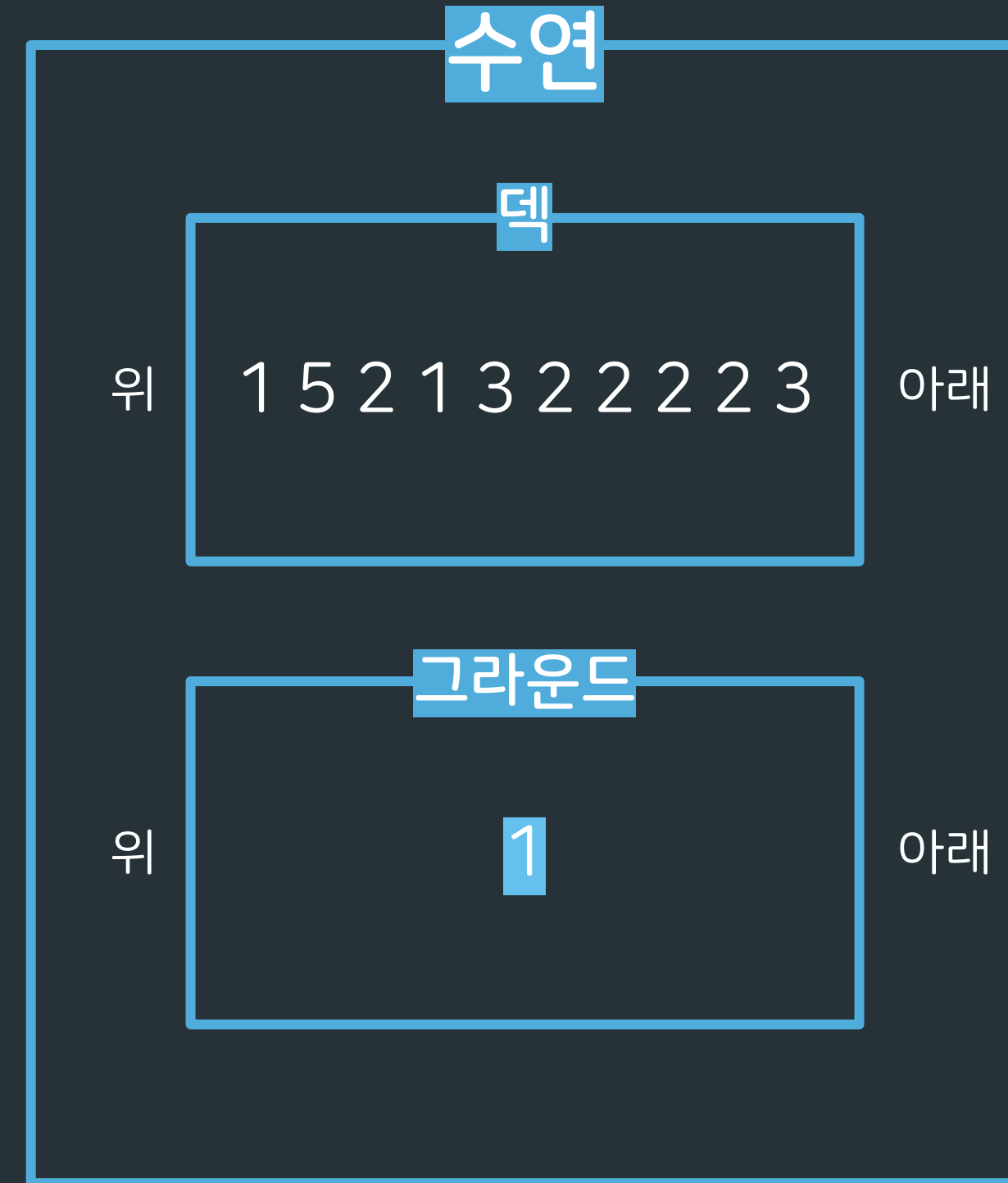
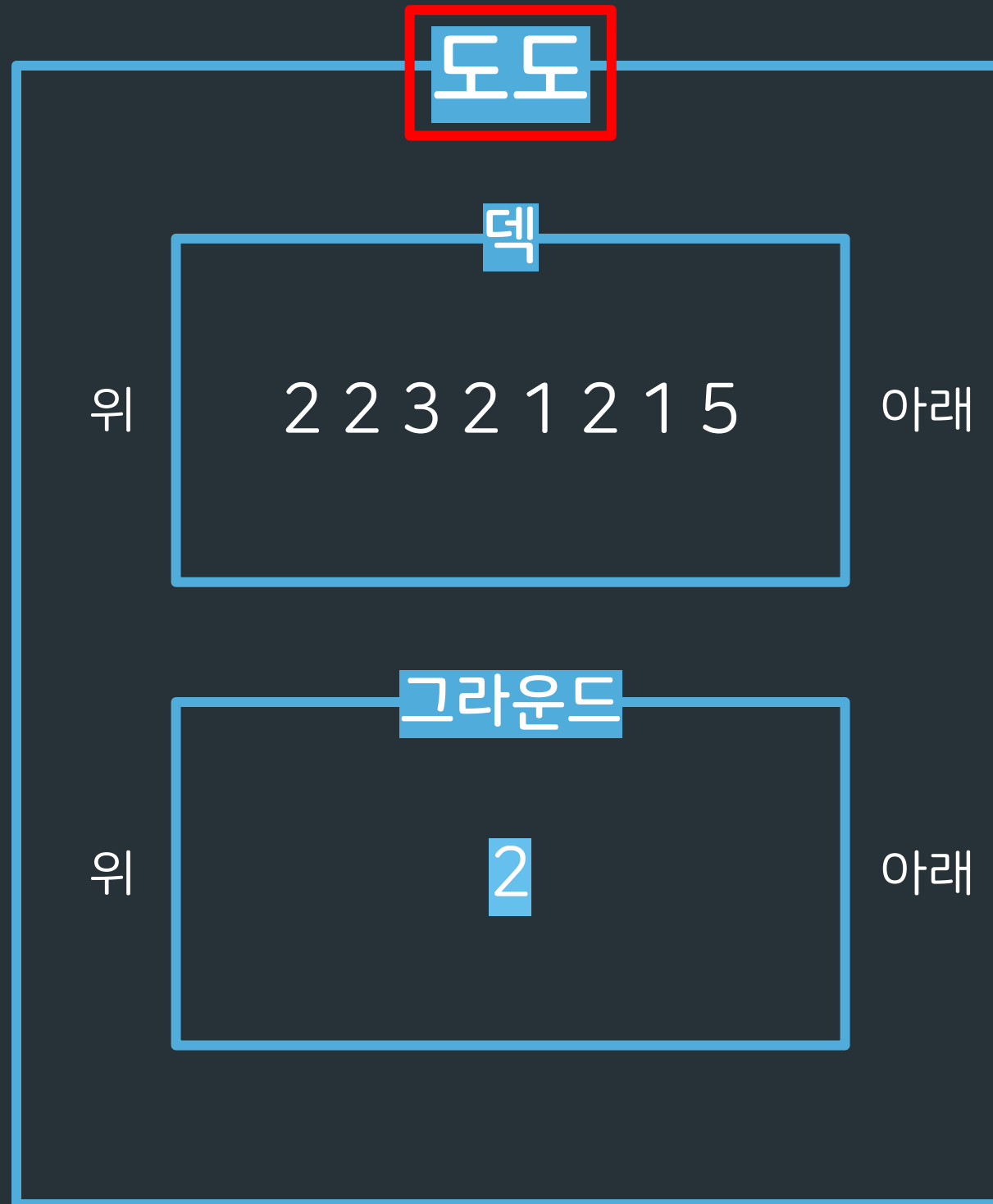
순서대로 게임을 진행해보자

M = 4



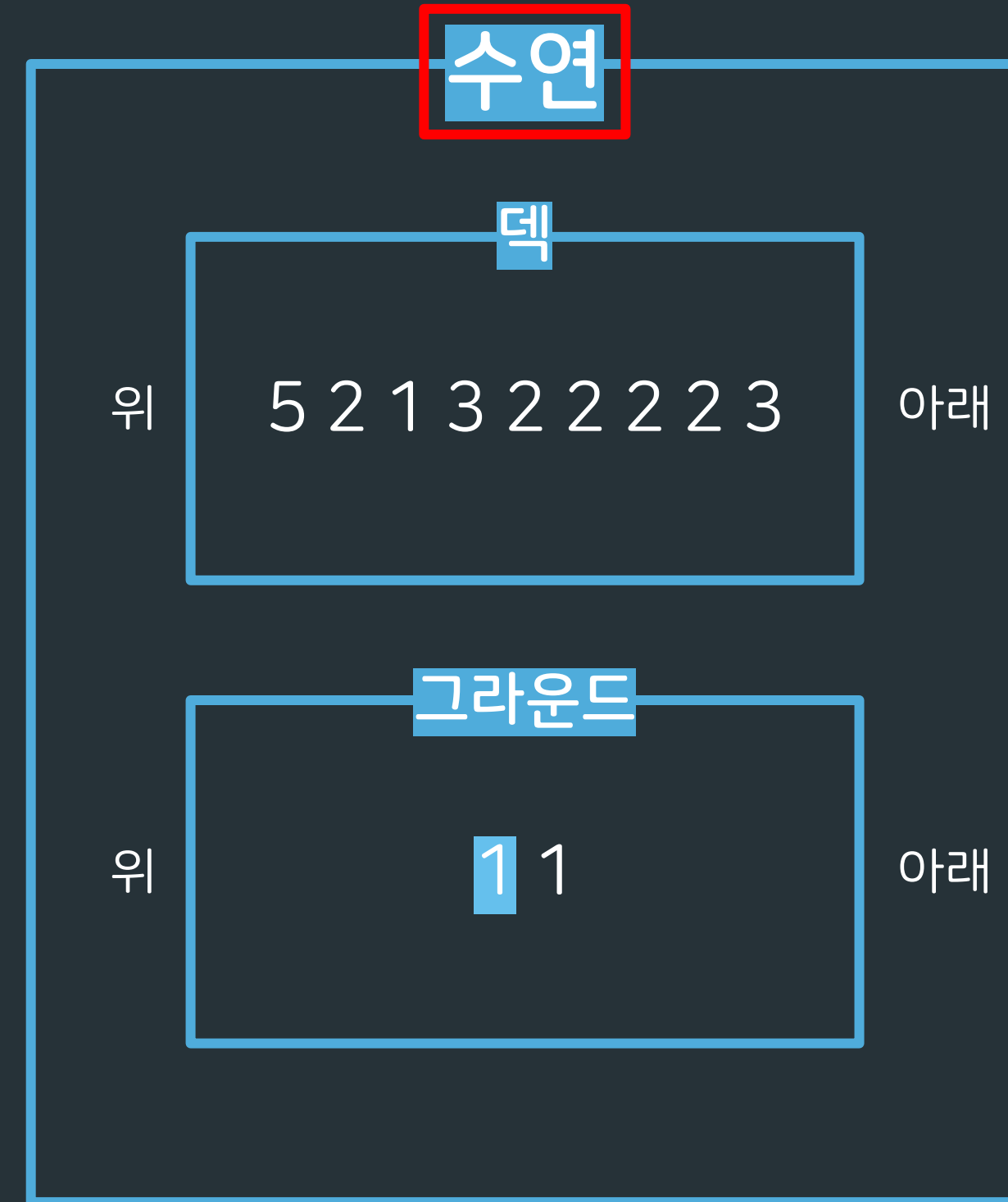
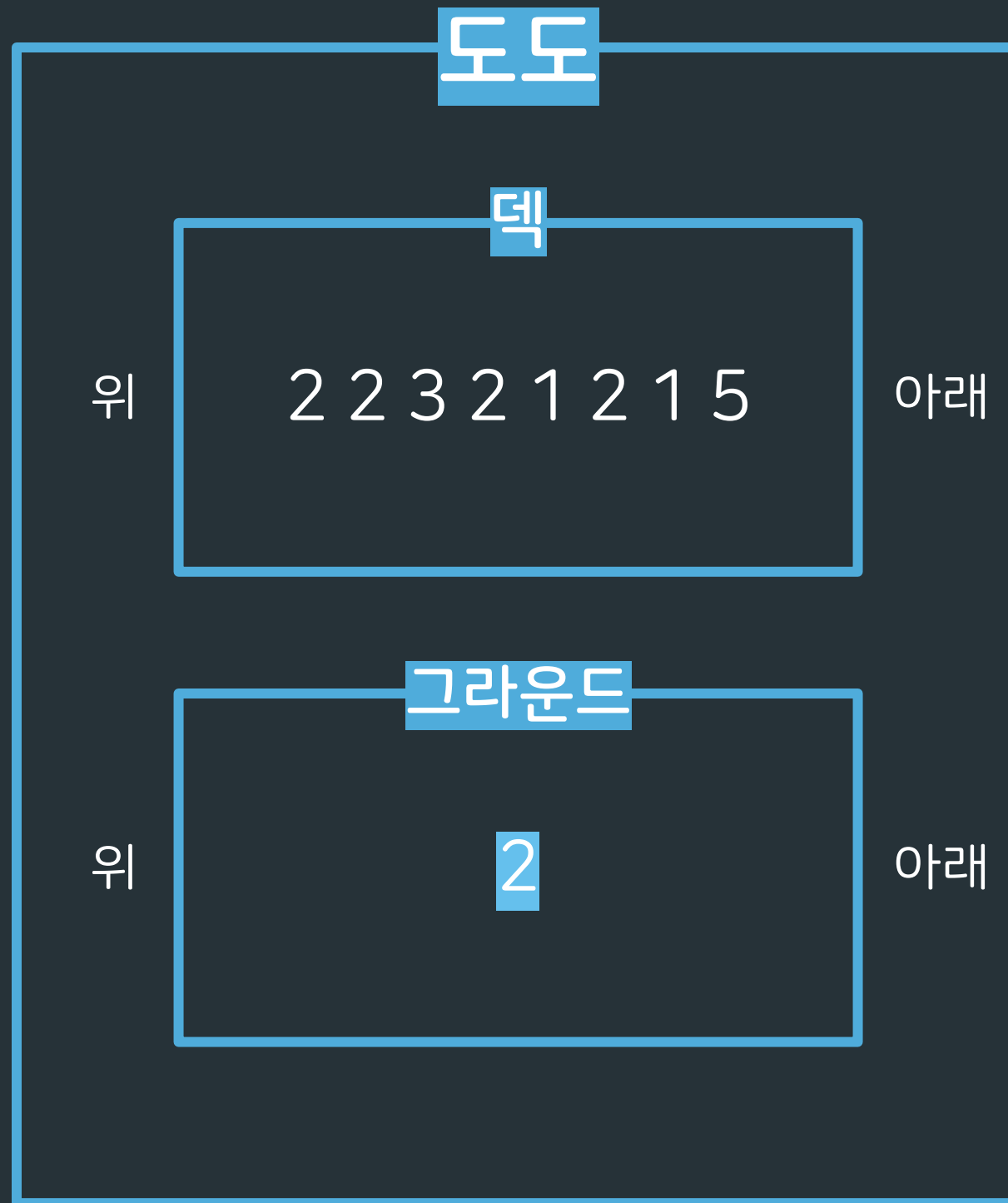
순서대로 게임을 진행해보자

M = 5



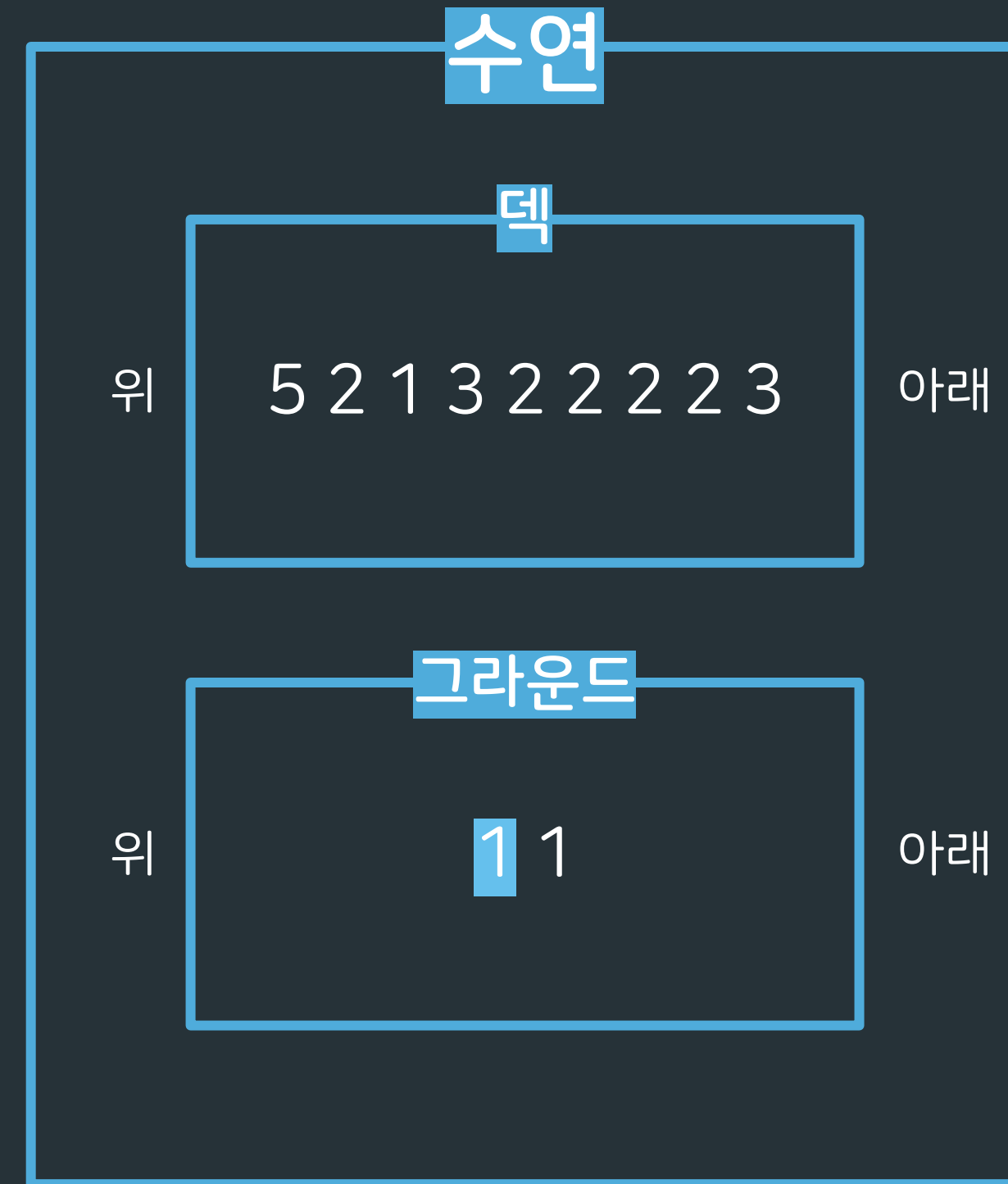
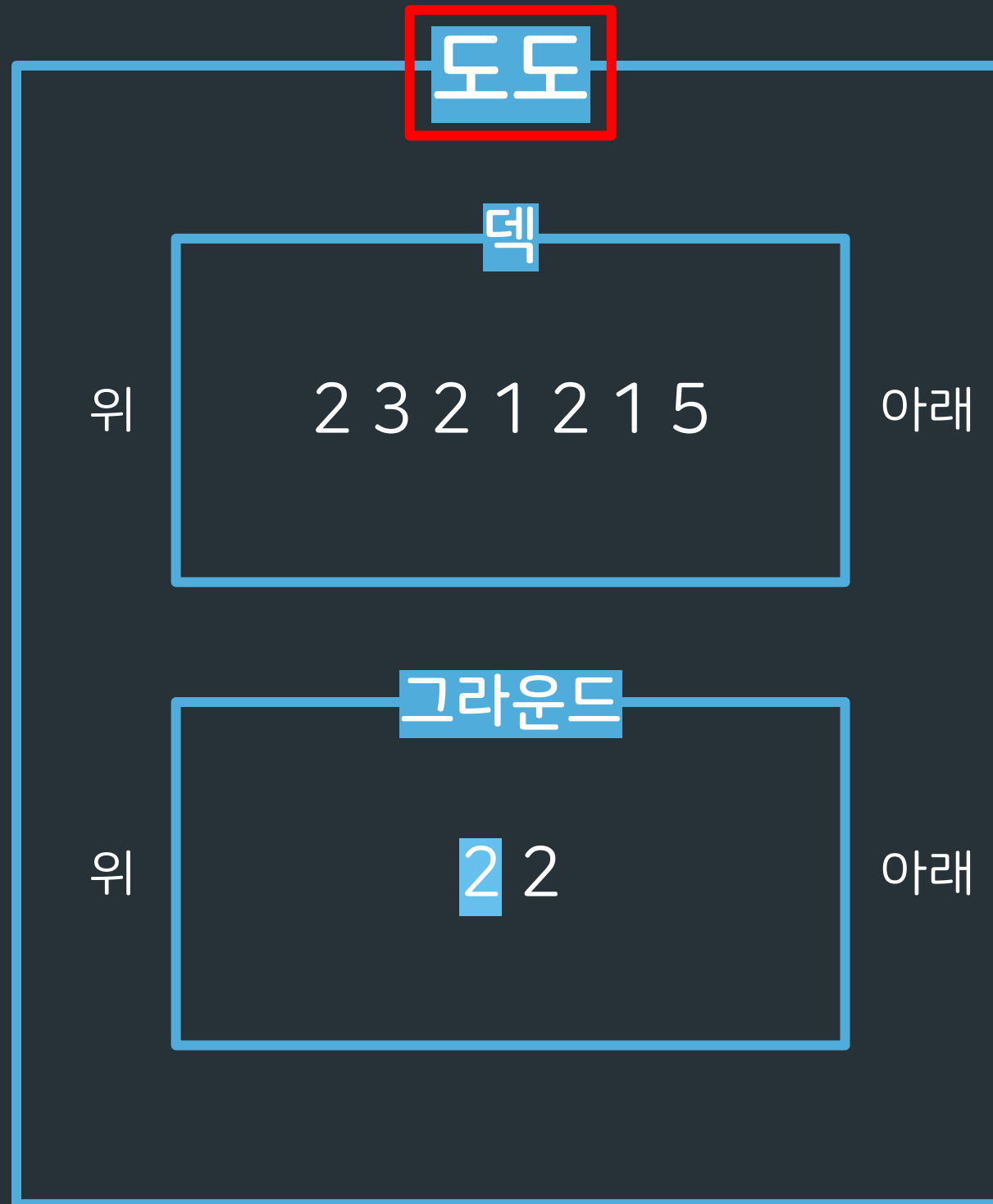
순서대로 게임을 진행해보자

M = 6



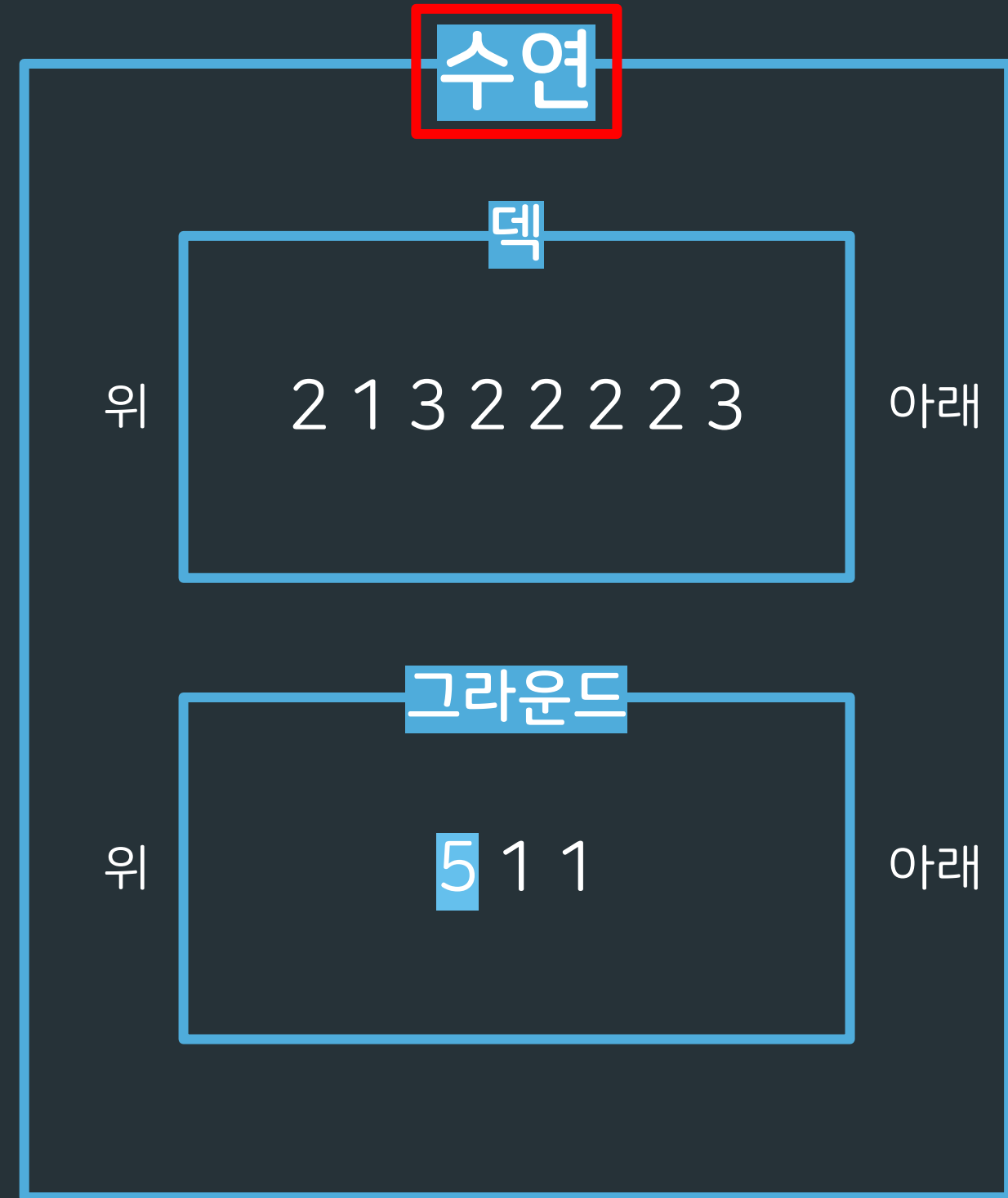
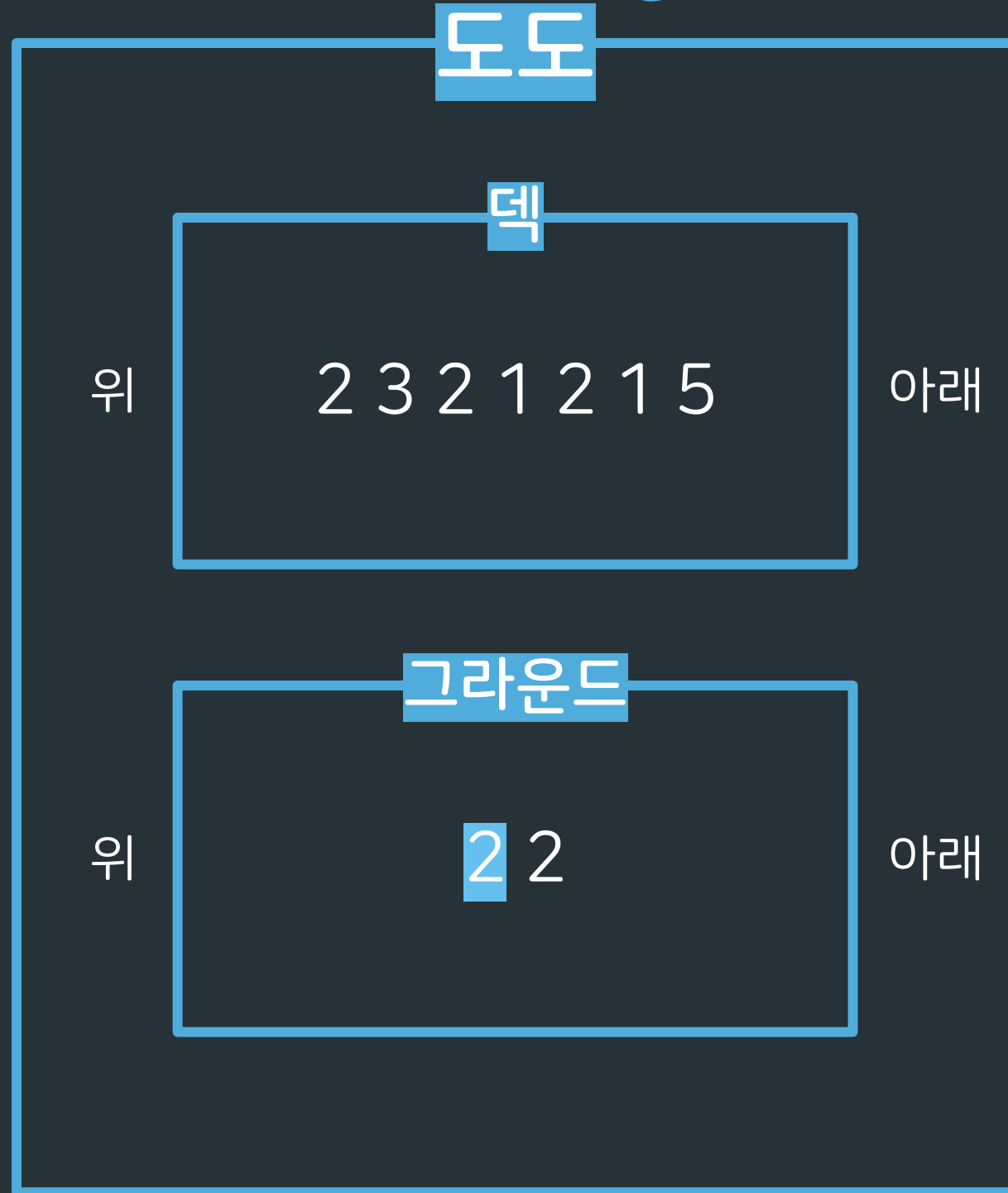
순서대로 게임을 진행해보자

M = 7



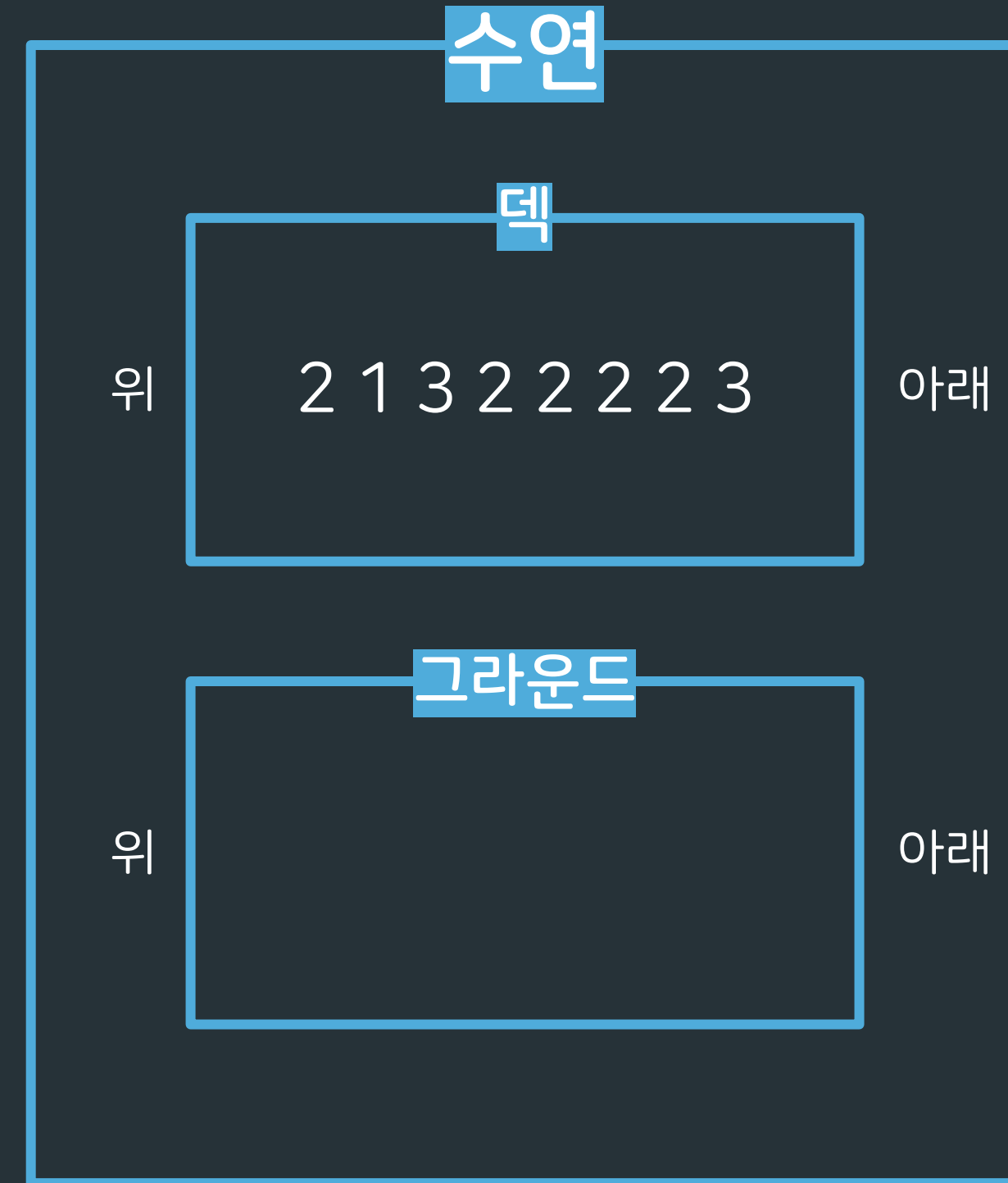
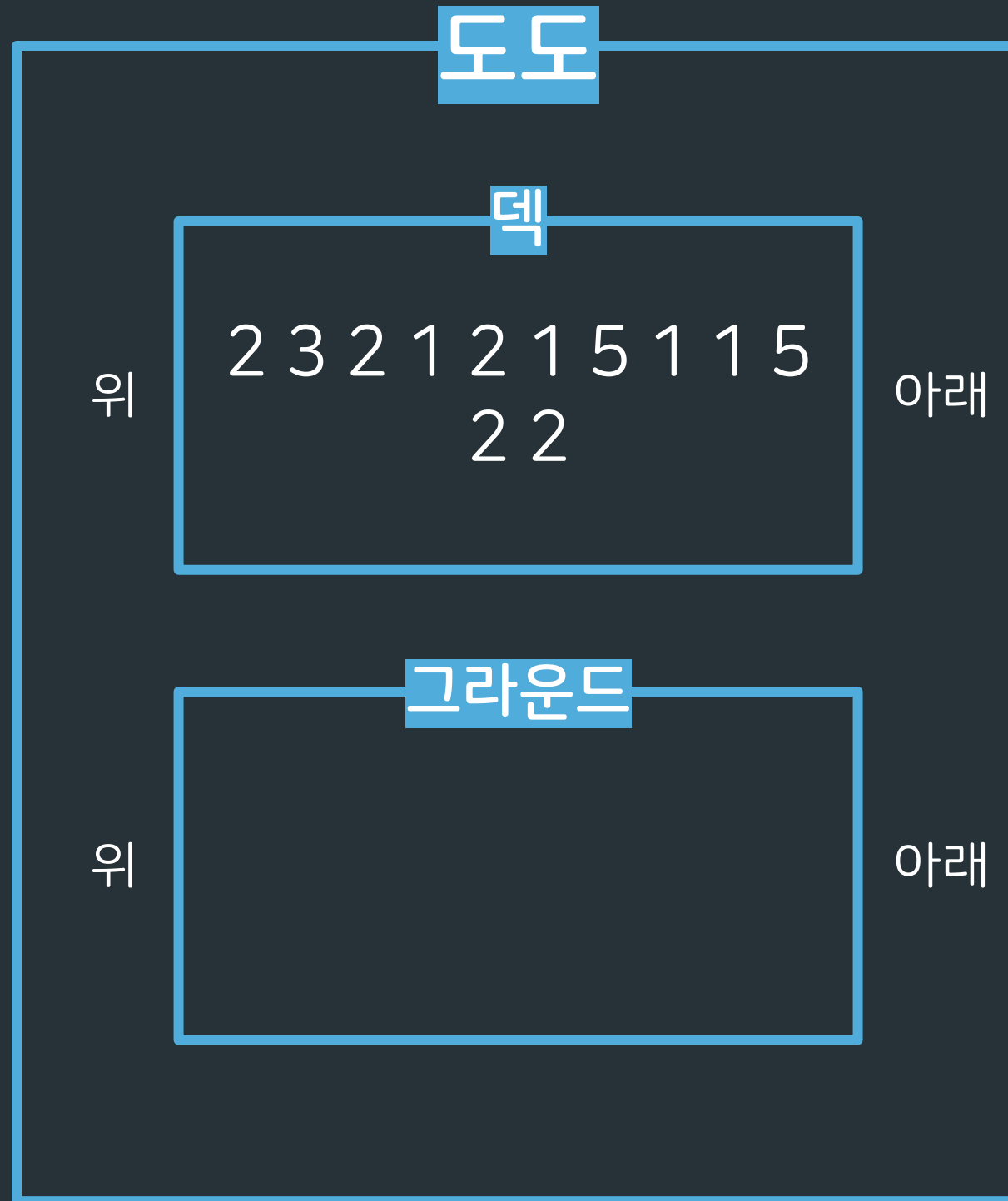
순서대로 게임을 진행해보자

M = 8



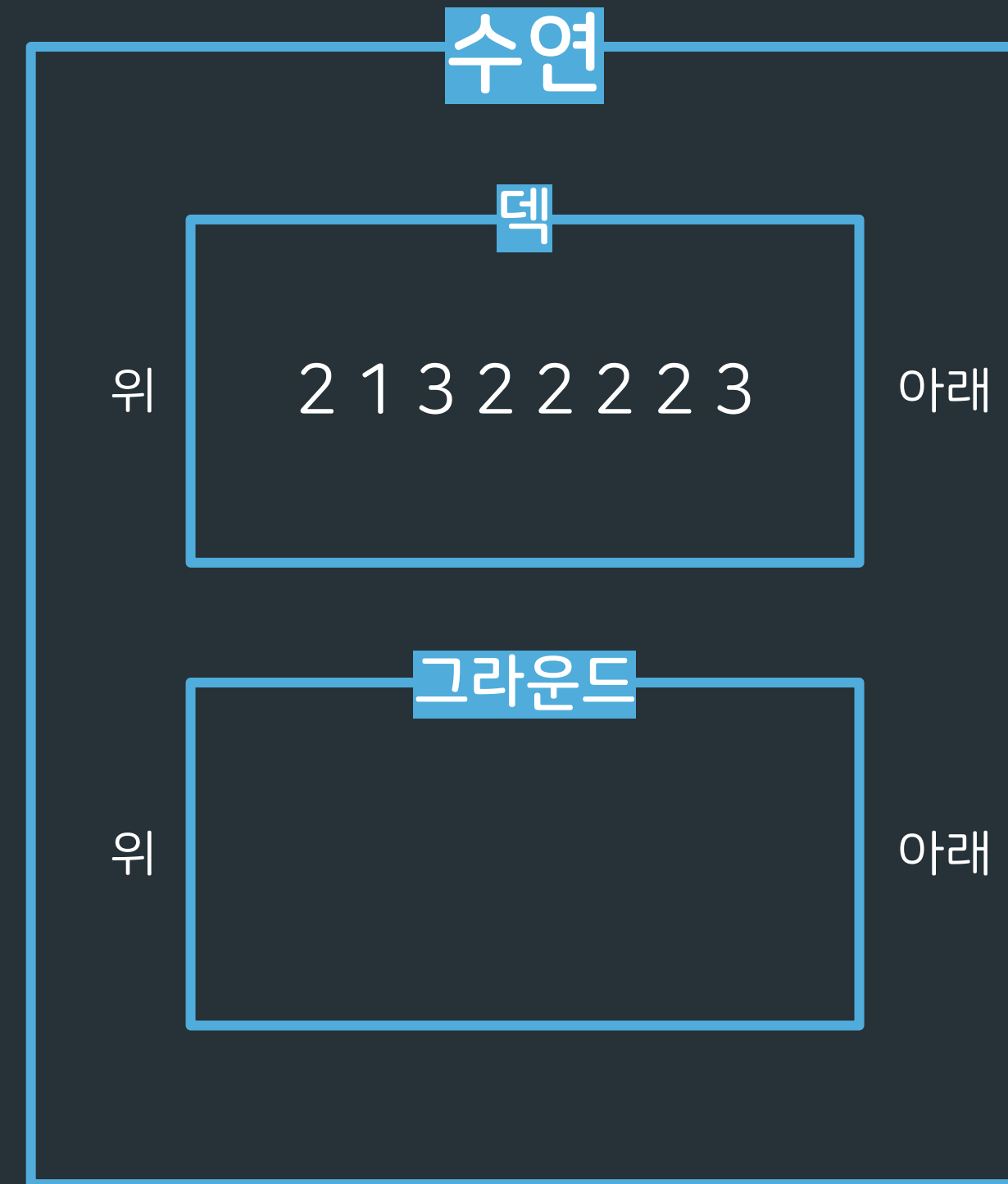
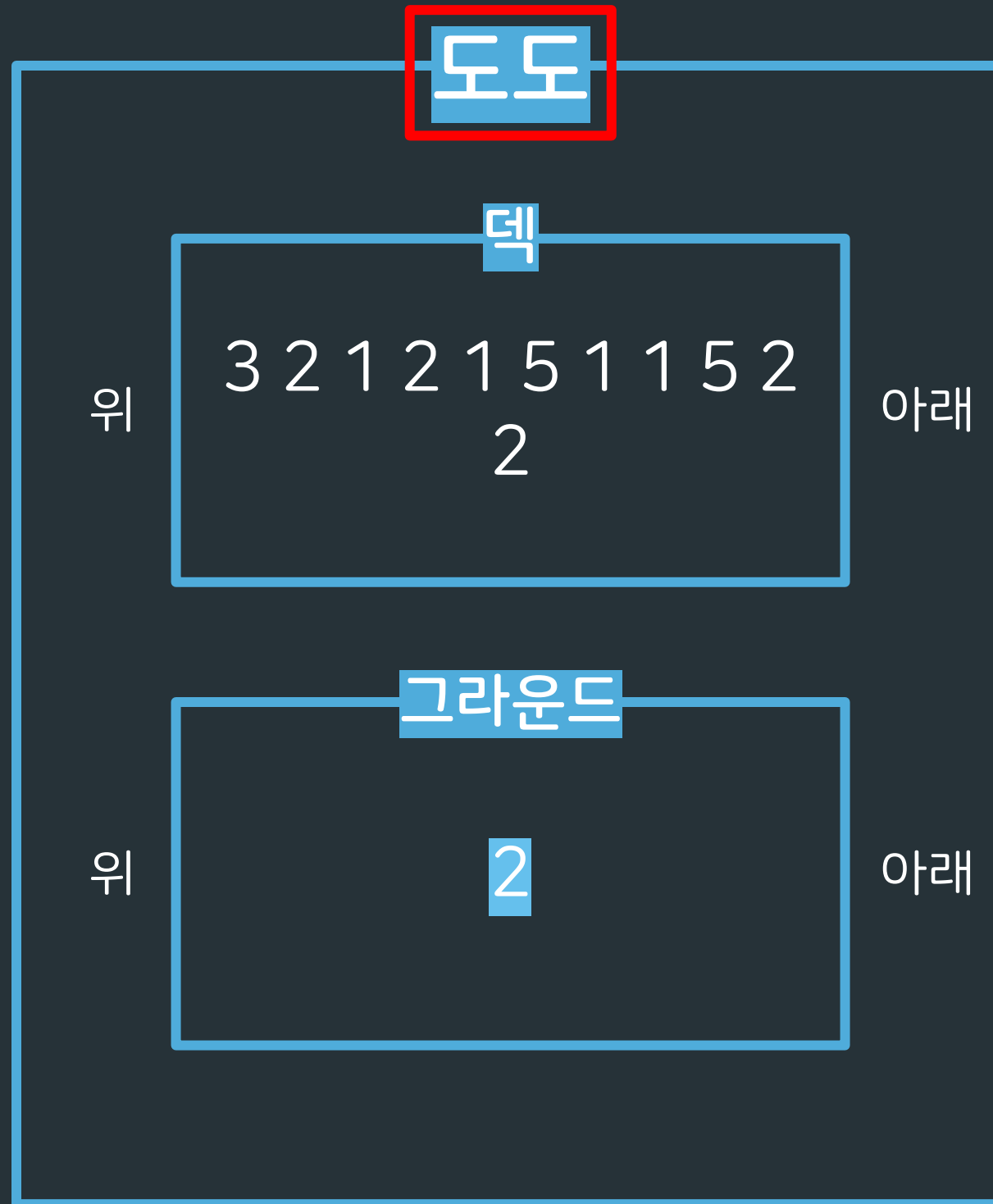
순서대로 게임을 진행해보자

M = 8



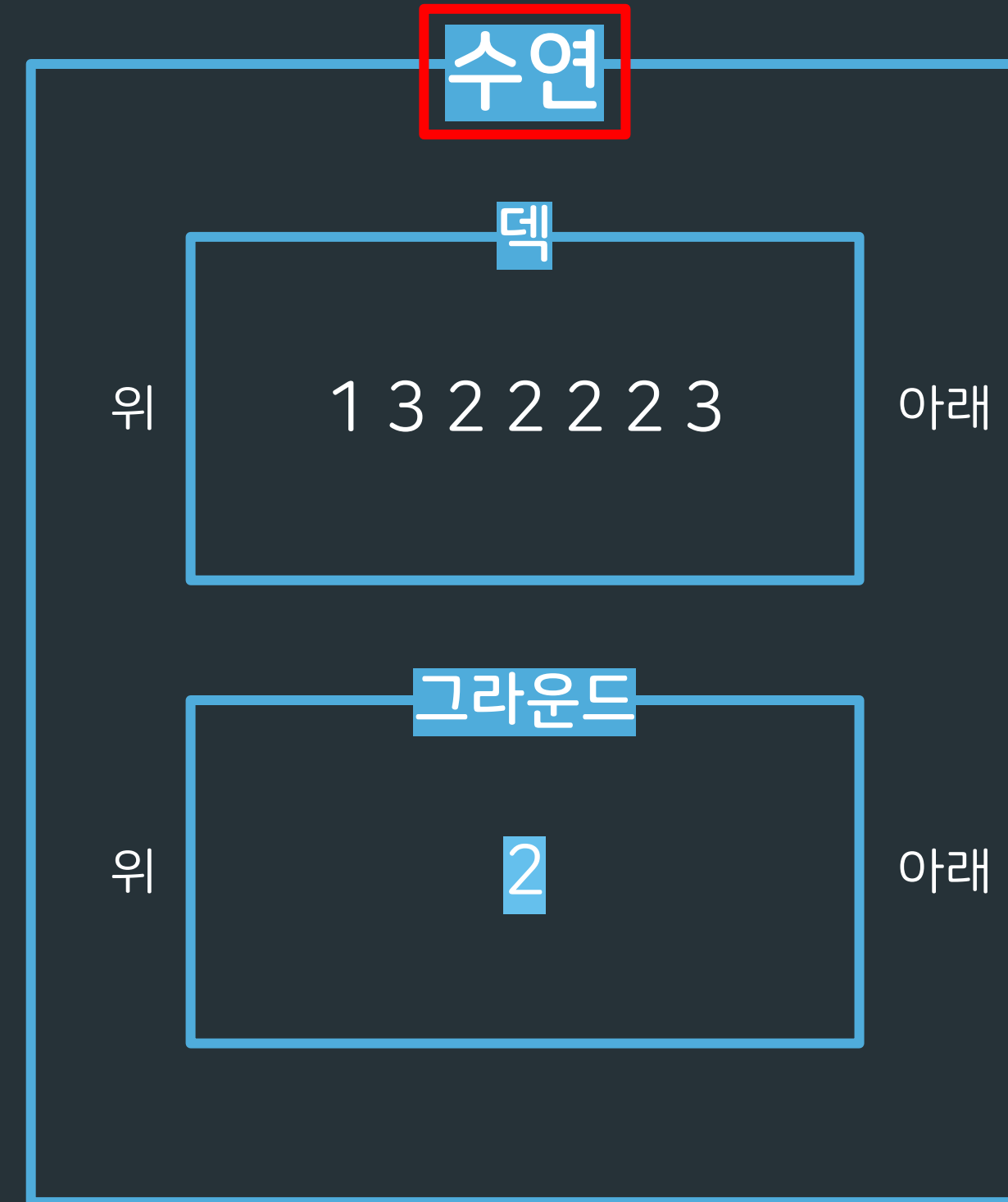
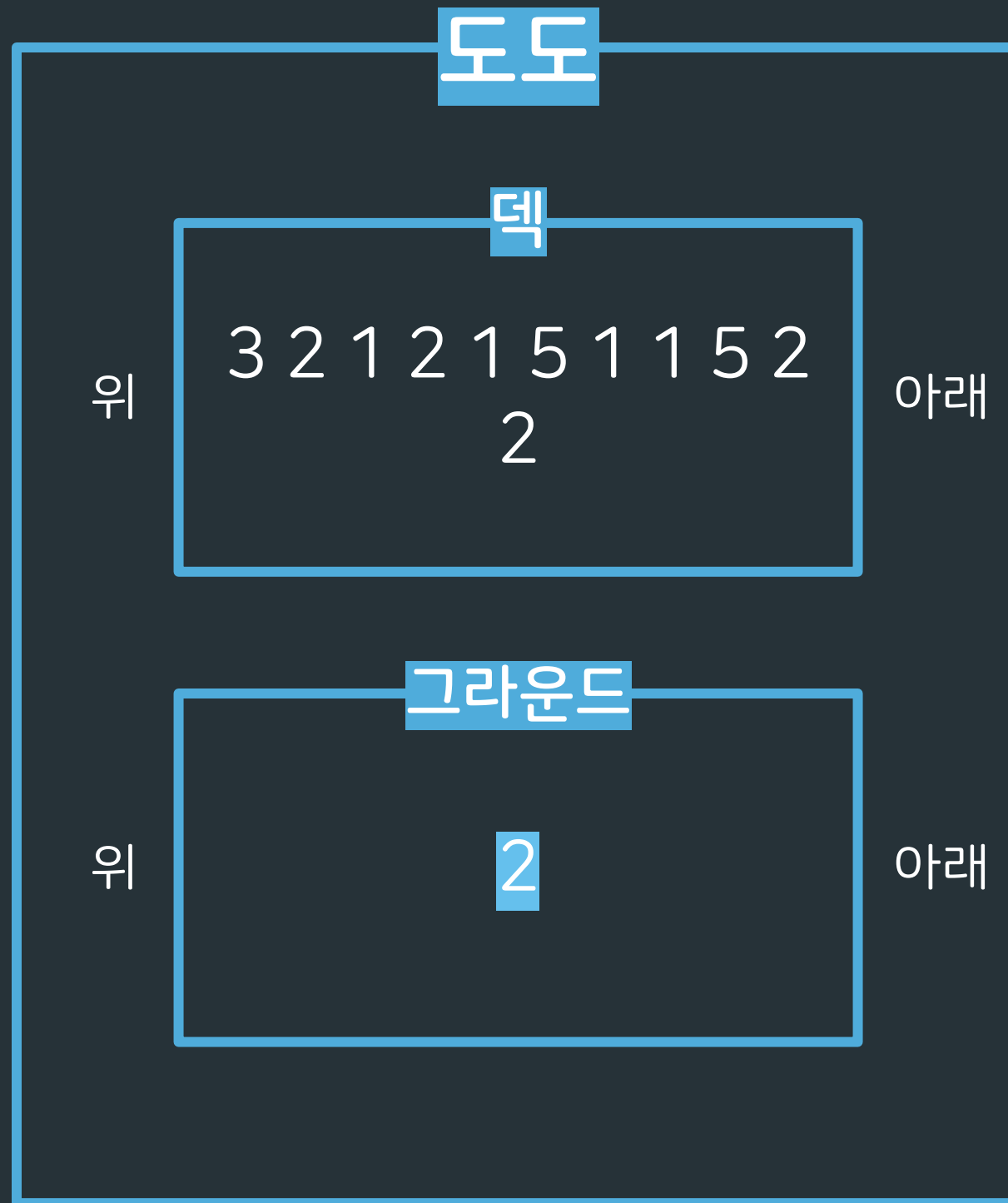
순서대로 게임을 진행해보자

M = 9



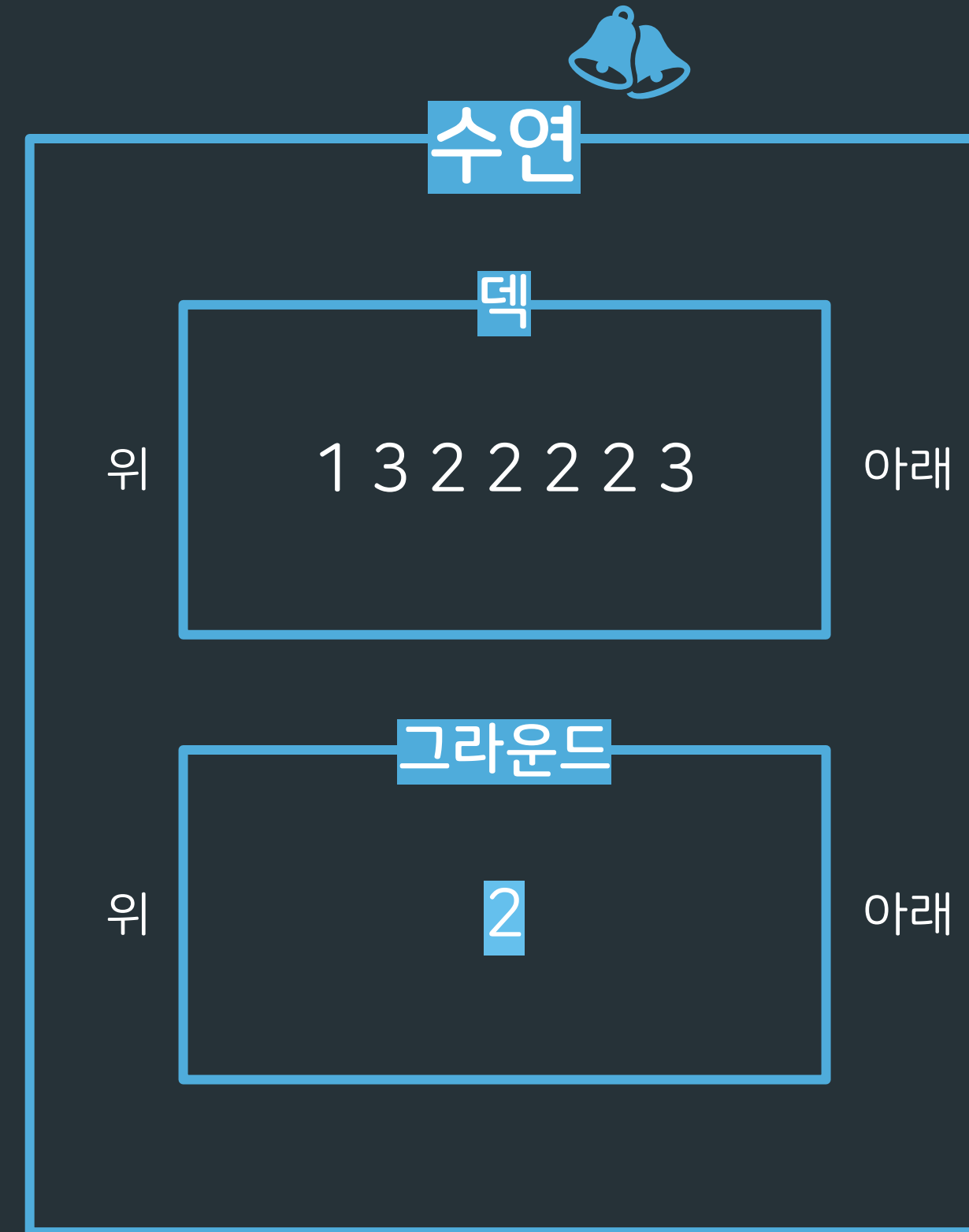
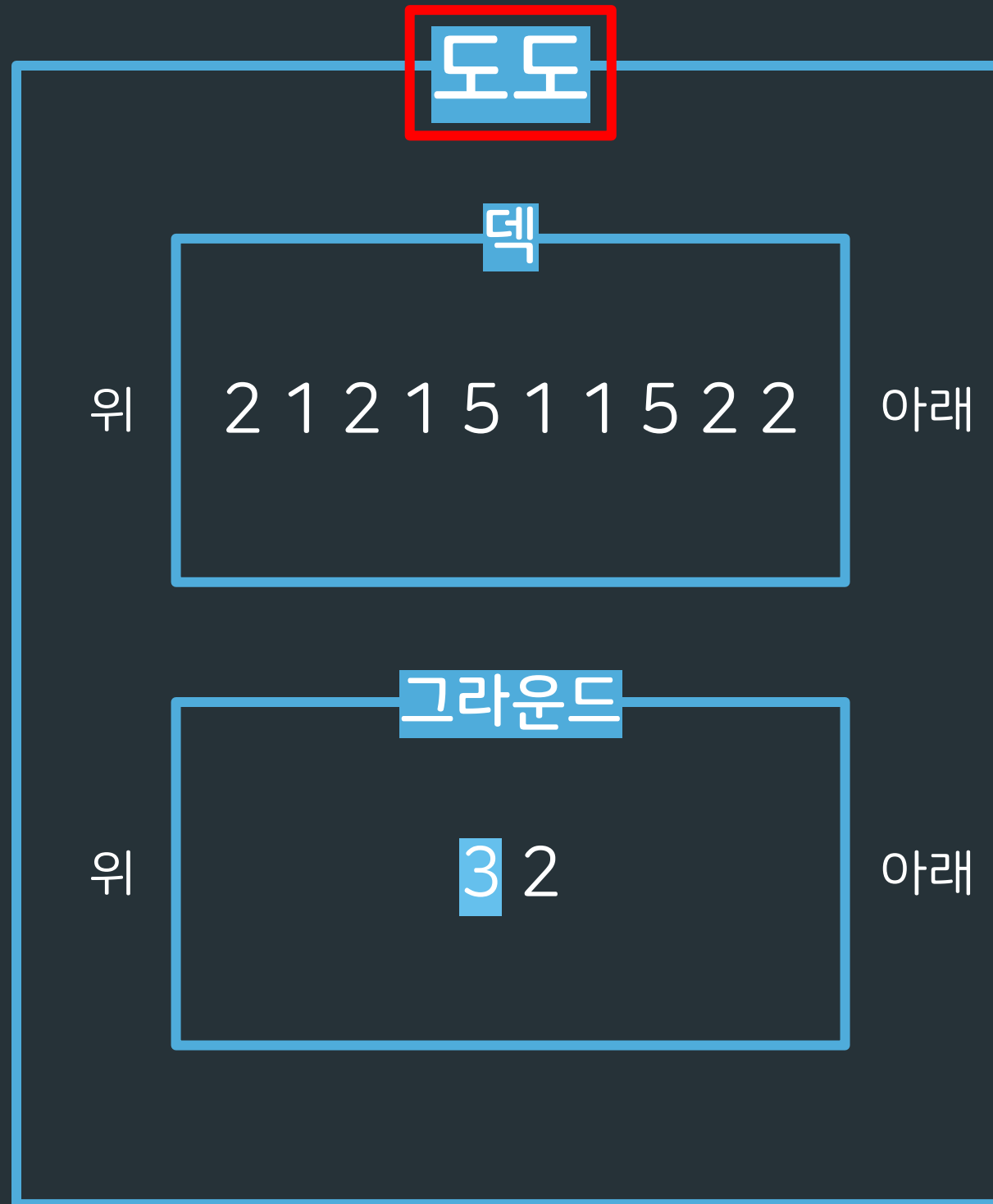
순서대로 게임을 진행해보자

M = 10



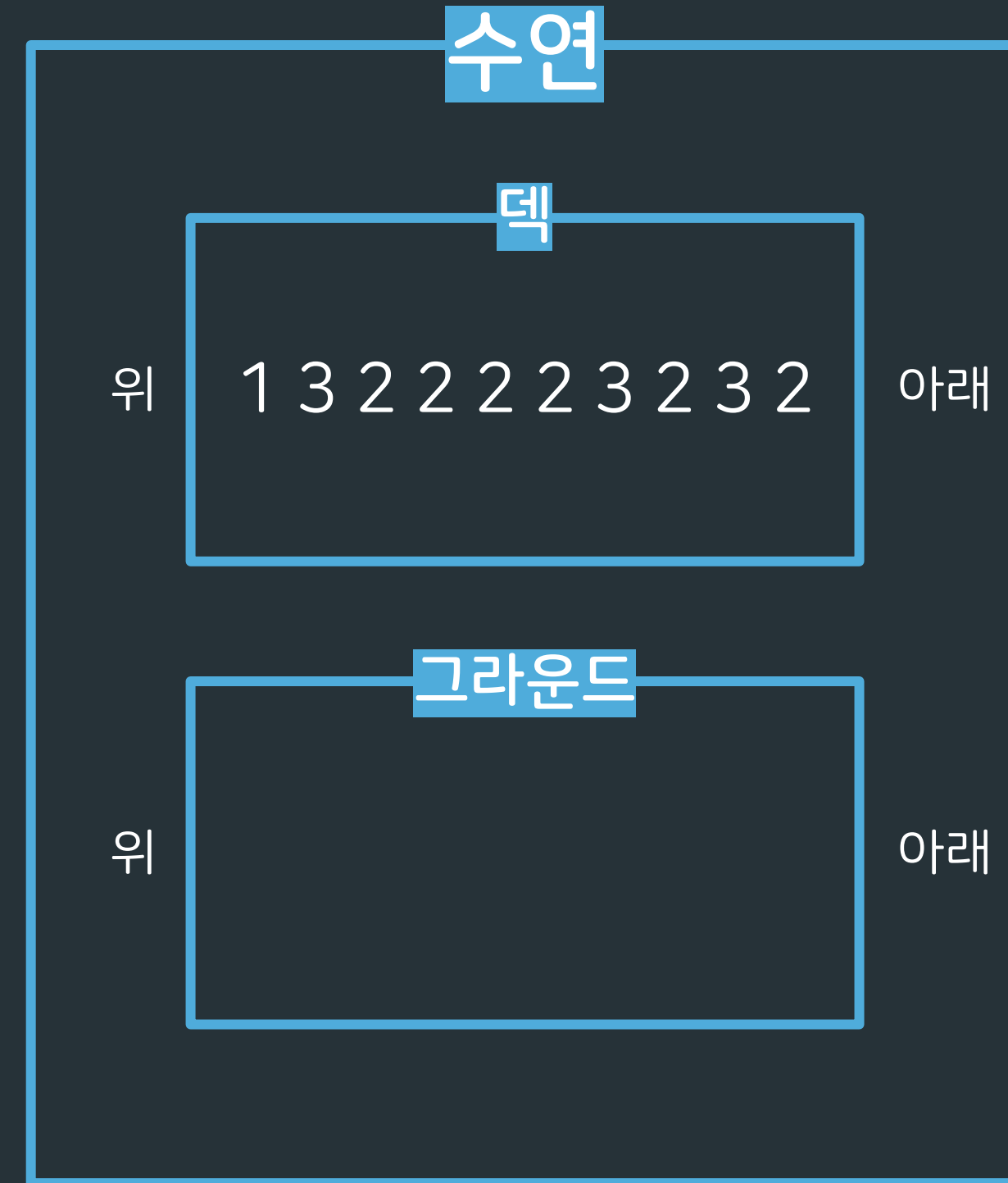
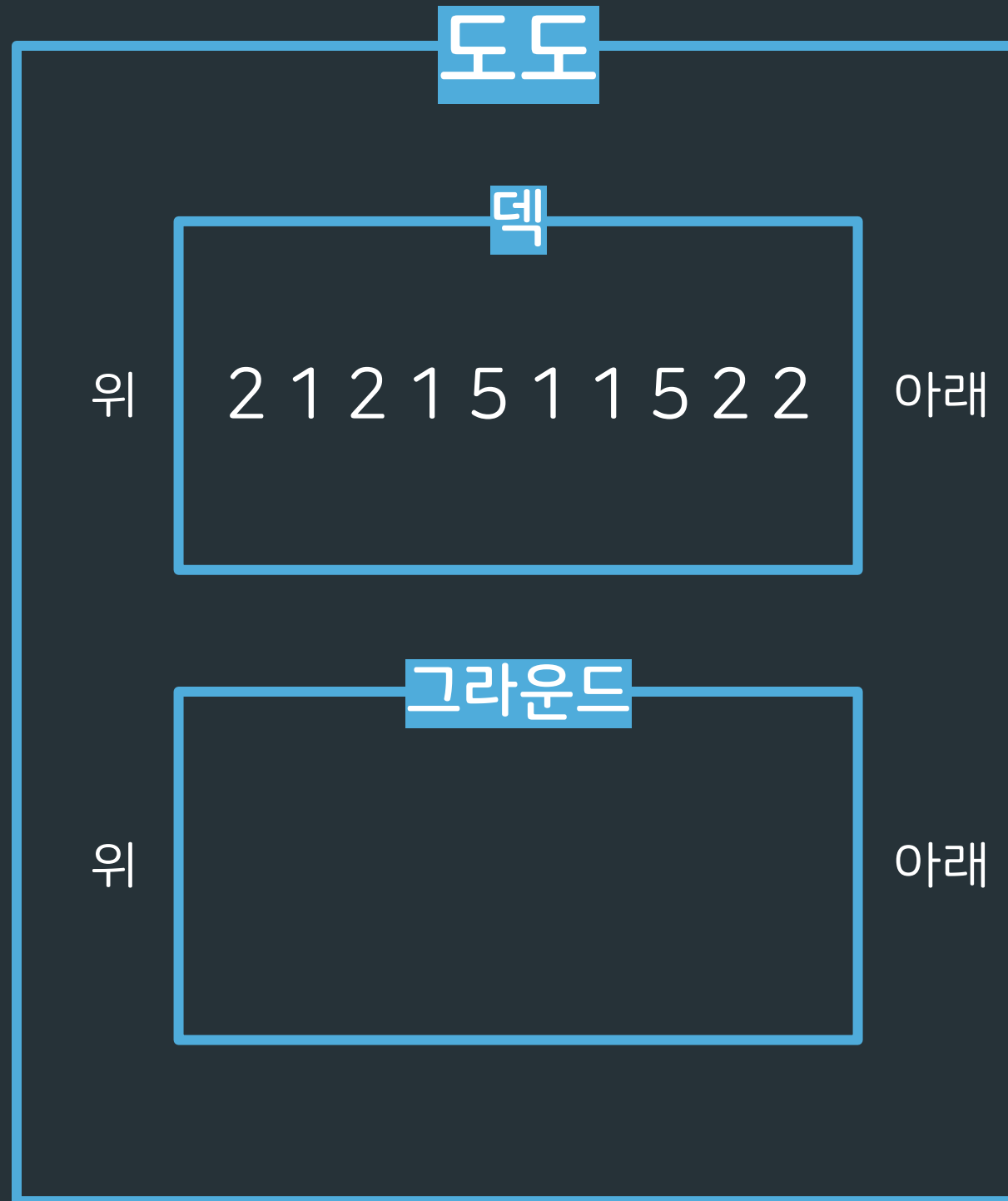
순서대로 게임을 진행해보자

M = 11



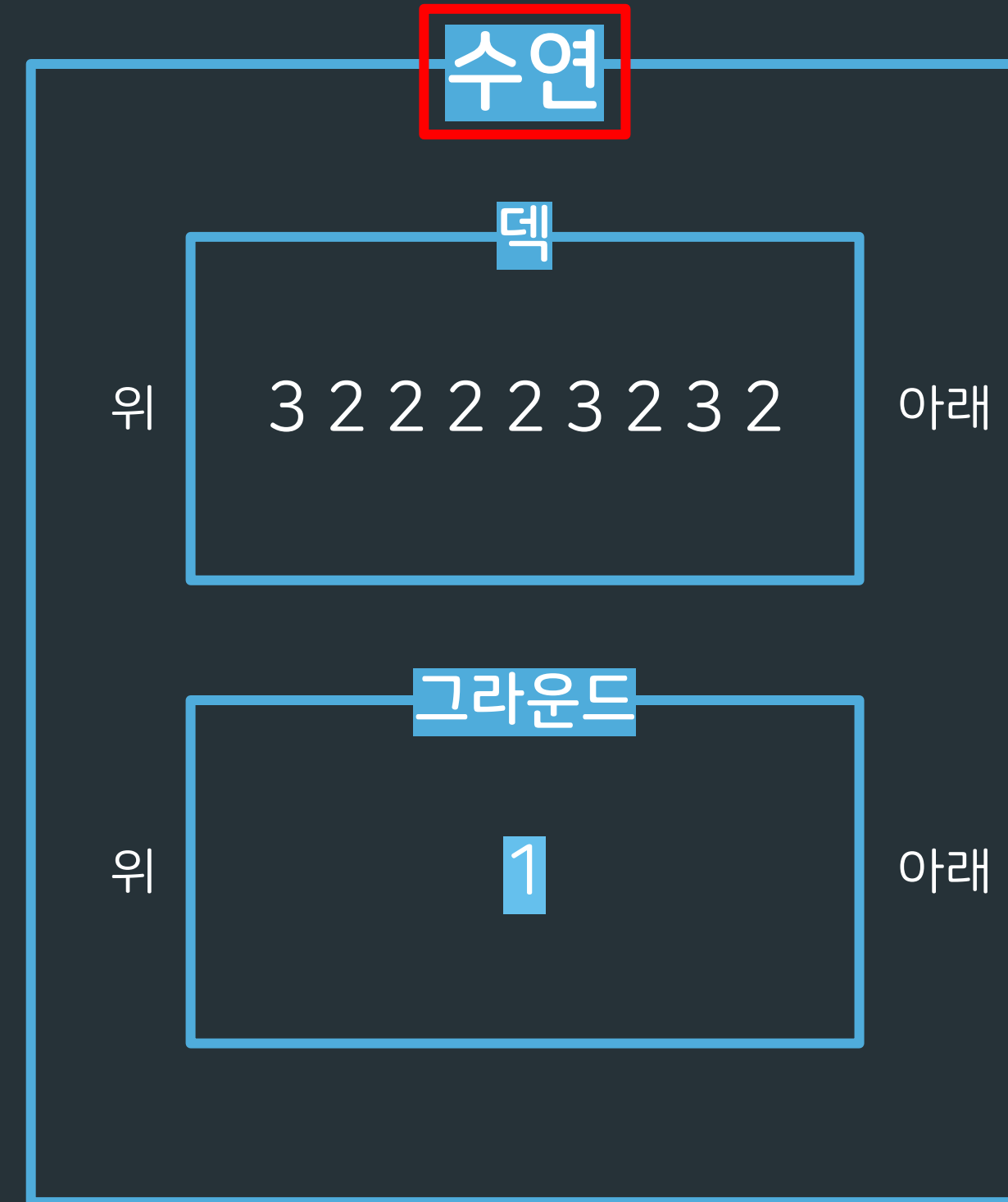
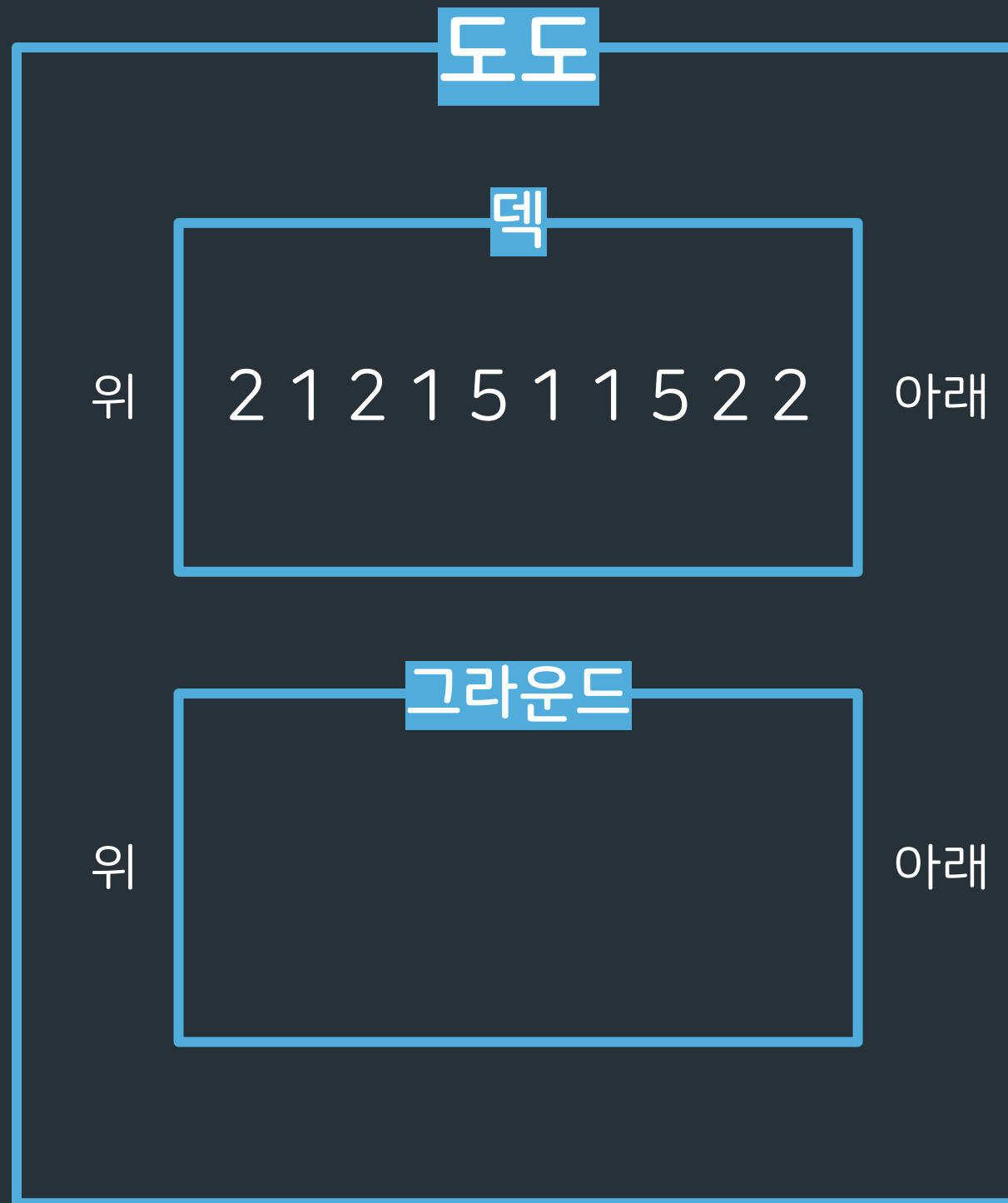
순서대로 게임을 진행해보자

M = 11



순서대로 게임을 진행해보자

M = 12



순서대로 게임을 진행해보자

도도

덱

2 1 2 1 5 1 1 5 2 2

10장

수연

덱

3 2 2 2 2 3 2 3 2

9장

>

카드를 더 많이 가지고 있는 도도의 승리!

도도, 수연은 각각 덱과 그라운드를 가짐(deque)

1. 카드를 덱에서 한 장씩 내려 놓음 (도도 먼저)
2. 어떤 플레이어가 종을 칠 수 있는지 판단 (도도/수연/없음)
 - a. 종을 친 경우 그라운드의 카드를 덱으로 이동
3. 종료 조건 만족 시 승리한 사람 판단

추가로 풀어보면 좋은 문제!

/<> 9655번 : 돌 게임 - Silver 5

/<> 9095번 : 1, 2, 3 더하기 - Silver 3

/<> 2156번 : 포도주 시식 - Silver 1

/<> 9251번 : LCS - Gold 5