난수 생성 함수 구현

1. 숫자 `1` 혹은 `0`을 임의로 return 하는 `get_1_or_0` 함수를 구현하세요.

```
import random

def get_1_or_0() -> int:
    return random.randint(0, 1)
```

- Python 표준 라이브러리를 사용해 `0` 또는 `1`을 반환하도록 작성했습니다.

2. 1.에서 작성한 `get_1_or_0`을 이용하여 숫자 `n`을 인자로 받아 `0~n` 사이의 임의의 정수를 반환하는 `get_random` 함수를 구현하세요. (단, 절대!! 난수 생성 함수를 직접 사용해서는 안됩니다.)

```
def get_random(n: int) -> int:
    if n < 0:
        raise ValueError("n은 0 이상이어야 한다.")

    bits = n.bit_length()

    while True:
        value = 0
        for _ in range(bits):
            value = (value << 1) | get_1_or_0()

        if value <= n:
            return value
```

- 앞선 함수에서 `0`과 `1`만을 반환하기 때문에 2진수를 사용해 함수를 구현했습니다.
- 입력받은 10진수를 2진수로 변환했을 때의 길이만큼을 반복하며 `0`과 `1`을 사용해 2진수의 비트를 구성해 임의의 정수를 반환하도록 했습니다.
- 예를 들어 `5(10진수)`의 경우 `101(2진수)`로 변환할 수 있으며, 길이는 3이 됩니다.
- 이때 `110(2진수)`, `111(2진수)`의 경우 `5`의 값을 초과하므로 이때는 while문이 종료되지 않고 새로운 정수를 반환하도록 합니다.
- 각 비트는 동전 던지기와 동일하게 0과 1이 동일한 확률로 생성되므로, 모든 이진수 조합은 균등한 확률을 가지며 결과적으로 0부터 n까지의 값이 특정 값에 치우치지 않고 생성됩니다.

3. 2.에서 작성한 `get_random` 함수에 대한 테스트 함수를 최대한 넓은 범위를 cover할 수 있도록 작성하세요.

```
def test_edge_cases():
    assert get_random(0) == 0
    assert get_random(1) in (0, 1)
    print("✅ 경계값(0, 1) 테스트 통과")

def test_range(n: int, trials: int = 10000):
    for _ in range(trials):
        v = get_random(n)
        assert 0 <= v <= n
    print(f"✅ 범위(0 ~ {n}) 테스트 통과")
```

- 시드를 주는 `get_1_or_0` 에서는 `0`과 `1`만 반환할 수 있으므로 `0`과 `1`에 대해 경계값을 테스트했습니다.
- 2번에서 작성한 함수에서 랜덤으로 입력받을 `n`에 대해 테스트를 진행해 범위를 벗어나는 경우가 존재하는지 테스트 했습니다.

```
from collections import defaultdict

def test_distribution(n: int, trials: int = 100000):
    freq = defaultdict(int)

    for _ in range(trials):
        freq[get_random(n)] += 1

    print(f"\n📊 Distribution test (n={n})")
    for i in range(n + 1):
        print(f"{i}: {freq[i] / trials:.4f}")
```

- 비트는 0과 1로만 이루어지므로 균등한 비율로 각 숫자들이 생성됩니다.
- 그래서 입력받는 n에 대해 임의의 정수를 어떠한 비율로 생성하는지 확인하고자 했습니다.

```
if __name__ == "__main__":
    random.seed()

    test_edge_cases() # 0과 1에 대해 테스트
    test_range(10)
    test_range(100)
    test_range(1000)
    test_distribution(5) # 0~5 사이의 값이 어떠한 비율로 생성되는지 테스트
```

<결과>

```
✅ 경계값(0, 1) 테스트 통과
✅ 범위(0 ~ 10) 테스트 통과
✅ 범위(0 ~ 100) 테스트 통과
✅ 범위(0 ~ 1000) 테스트 통과

📊 Distribution test (n=5)
0: 0.1666
1: 0.1666
2: 0.1671
3: 0.1658
4: 0.1683
5: 0.1656
```