# 문제. 익스트림 베스킨라빈스

당신은 잘 알려진 술게임 '베스킨라빈스 게임'을 더욱 어렵게 만들어 '익스트림 베스킨라빈스 게임'을 만들고자 한다.

두 명이서 하는 베스킨라빈스 게임은 아래와 같이 진행된다.

- 첫 시작 시 게임 수는 `1`부터 시작한다.
- 자신의 차례에 `1`, `2`, `3` 중 하나를 골라 이 수를 게임 수에 더한다.
- 차례는 번갈아 가며 진행하며, 자기 차례에 게임 수가 `31` 이상의 수가 되면 패배한다.

당신은 기존의 게임에 다음과 같이 룰을 변경하였다.

- 게임은 두 명이서 진행한다.
- 첫 시작 시 게임 수는 `0`부터 시작한다.
- 도달하면 게임을 종료하는 수 `31` 대신 자연수 `target`으로 한다.
- `target`값 이상이 되면 게임에서 패배하는 대신, 게임에서 승리한다.
- `1` 이상 `3` 이하의 수 대신, `1` 이상 `r` 이하의 수를 선택할 수 있다.
- 수를 선택할 때에는 한 번 사용한 수는 재사용할 수 없다.

예를 들면, `target = 10`, `r = 4`인 경우 아래와 같은 게임 양상이 있을 수 있다.

- 첫번째 사람은 `1`~`4` 중 아무거나 선택할 수 있다. `4`를 선택하면 현재 수는 `4`가 된다.
- 두번째 사람은 `1`, `2`, `3` 중 선택할 수 있다. `2`를 선택하면 현재 수는 `6`이 된다.
- 첫번째 사람은 `1`, `3` 중 선택할 수 있다. `1`을 선택하면 현재 수는 `7`이 된다.
- 두번째 사람은 `3`을 선택할 수 있다. `3`을 선택하면 현재 수는 `10`이 된다.
- 두번째 사람이 `target` 값에 도달하였으므로 두 번째 사람이 게임에서 승리한다.

위 경우에는 어떠한 방식으로 플레이하더라도 결국 두 번째 사람이 게임에서 승리하게 된다.

당신은 주어진 `target`과 `r`에 대해서, 첫번째 사람이 항상 승리할 수 있는지 여부를 논리값으로 반환하시오.

단, 두 사람은 모두 항상 자신이 이길 수 있는 최선의 수를 선택한다. 또한 사용할 수 있는 모든 수를 더해도 `target`에 도달할 수 없으면 승리할 수 없는 것으로 본다.

### 입력 설명

- `0 < target <= 300`
- `0 < r <= 20`

## 예시 입출력 예시

### #1

- 입력
  - `target = 10`
  - `r = 3`

- 반환값
  - `false`

- 설명

  본문에 제시된 예시로, 반드시 두번째 사람이 승리한다. 즉, 첫번째 사람은 반드시 패배한다.

### #2

- 입력
  - `target = 5`
  - `r = 3`

- 반환값
  - `true`

- 설명

  아래와 같이 첫번째 사람이 무조건 승리할 수 있다.

  - 첫번째 사람은 `1`를 선택할 수 있다. 상대가 `5`에 도달하지 못하게 하기 위해 `1`을 택하는 것이 최선이다. 현재 게임 수는 `1`이 된다.
  - 두번째 사람은 `2`, `3` 중 하나를 택할 수 있다. `2`를 택하면 현재 게임 수는 `3`이 되고, `3`을 택하면 현재 게임 수는 `4`가 된다.
  - 첫번째 사람은 두번째 사람의 선택과 무관하게 나머지 하나를 택하게 된다. 그러면 현재

게임 수는 `6`이 되어 게임에서 무조건 승리한다.

## 해설. 익스트림 베스킨라빈스

- **Python3 정답 코드 예시**

```
from functools import cache



def solution(target, r):
    if sum([i + 1 for i in range(r)]) < target:
        return False

    @cache
    def dfs(total, mask):
        answer = False
        for i in range(r):
            if mask >> i & 1 == 0:
                if total + i + 1 >= target:
                    answer = True
                    break
                new_mask = mask | 1 << i
                if not dfs(total + i + 1, new_mask):
                    answer = True
                    break
        return answer

    return dfs(0, 0)
```

- **Java 정답 코드 예시**

```
import java.util.HashMap;
import java.util.List;
import java.util.Map;
```

```
import java.util.stream.IntStream;

class Solution {
    int target;
    int r;
    Map<List<Integer>, Boolean> dp;

    public boolean solution(int target, int r) {
        this.target = target;
        this.r = r;
        this.dp = new HashMap<>();

        if (IntStream.range(0, r+1).sum() < target) {
            return false;
        }

        return dfs(0, 0);
    }

    boolean dfs(int total, int mask) {
        boolean answer = false;

        if (dp.containsKey(List.of(total, mask))) {
            return dp.get(List.of(total, mask));
        }

        for (int i = 0; i < r; i++) {
            if ((mask >> i & 1) == 0) {
                if (total + i + 1 >= target) {
                    answer = true;
                    break;
                }
```

```
                int newMask = mask | 1 << i;
                if (!dfs(total + i + 1, newMask)) {
                    answer = true;
                    break;
                }
            }
        }

        dp.put(List.of(total, mask), answer);
        return answer;
    }
}
```