문제. 협동력 점수

N명의 학생을 K개의 그룹으로 나누고자 한다. 각 학생은 협동력 점수를 가지고 있으며, 단 하나의 그룹에만 속할 수 있다. 모든 학생은 반드시 각자 특정한 하나의 그룹에 배치된다.

모든 학생에 대한 그룹 배치가 끝나고 나면, 각 그룹의 전투력을 계산할 수 있다. 이때 특정한 그룹의 전투력은 그룹에 포함된 각 학생에 대하여 (해당 그룹에 속한 구성원 수 X 해당 학생의 협동력 점수)을 계산하여 모두 더한 값이다. 어떤 학생의 협동력 점수는 음수(-)일 수 있으며, 결과적으로 특정한 그룹의 전투력은 음수가 될 수도 있다.

이때, N명의 학생으로 K개의 그룹을 구성하는 모든 경우를 고려했을 때, **가장 전투력이 높은 그룹과 전투력이 낮은 그룹의 전투력 차이의 최솟값**을 구하여라. 단, 모든 K개의 그룹에 최소한 1명의 학생은 배치되어야 한다.

예를 들어 $N = 4$명의 학생들의 협동력 점수가 다음의 표와 같으며, 그룹의 수 $K = 2$라고 하자.

학생 번호	**1**	**2**	**3**	**4**
협동력 점수	-1	0	3	-2

가장 먼저, 다음과 같이 두 그룹을 결성했다고 가정하자.

그룹 번호	**1**	**2**
학생 번호 구성	[1, 2]	[3, 4]

이때 전투력은 다음과 같이 계산되므로, 두 그룹의 전투력 차이는 4이다.

(1) 1번 그룹의 전투력: -2 = 2 * (-1) + 2 * 0
(2) 2번 그룹의 전투력: 2 = 2 * 3 + 2 * (-2)

혹은 다음과 같이 두 그룹을 결성했다고 가정하자.

그룹 번호	1	2
학생 번호 구성	[2]	[1, 3, 4]

이때 전투력은 다음과 같이 계산되므로, 두 그룹의 전투력 차이는 0이다.

(1) 1번 그룹의 전투력: 0 = 1 * 0
(2) 2번 그룹의 전투력: 0 = 3 * (-1) + 3 * 3 + 3 * (-2)

따라서, 현재 입력 예시에서의 최적의 해는 0이다.

입력 조건

가장 먼저 학생의 수 N이 자연수로 주어진다. N은 2 이상 7 이하의 자연수다.
이어서 그룹의 수 K가 자연수로 주어진다. K는 2 이상 N 이하의 자연수다.
이후에 각 학생의 협동력 점수 정보가 담긴 배열 arr가 주어진다. 각 학생들의 협동력 점수는 -10 이상 10 이하의 정수이다.

출력 조건

N명의 학생으로 K개의 그룹을 구성하는 모든 경우를 고려했을 때, 가장 전투력이 높은 그룹과 전투력이 낮은 그룹의 전투력 차이의 최솟값을 반환한다.

입출력 예시

N	K	arr	정답
4	2	[-1, 0, 3, -2]	0
5	4	[2, 1, 1, 2, 3]	2

해설 3. 협동력 점수

각 학생은 자신이 배치될 그룹의 번호로 K개의 수 중에서 1개를 골라야 한다. 이때, 각 그룹의 번호는 중복하여 선택될 수 있기 때문에, 본 문제는 **중복 순열**을 계산하는 문제와 유사하다. 중복 순열 문제에서 백트래킹을 활용하여 모든 경우의 수를 계산하기 위해 필요한 최악의 경우 시간 복잡도는 $O(K^N)$이다.

이때, K개의 그룹에는 최소한 1명의 학생은 배치되어야 한다. 따라서 각 그룹에 대하여 최소 1명의 학생이 배치된 경우만을 모두 고려하여, 해당 경우에서 각 그룹의 전투력을 계산하는 방식으로 문제를 해결할 수 있다.

- **Python3 정답 코드 예시**

```
def dfs(N, K, arr, depth):
    # 모든 중복 순열을 확인하는 부분
    if depth == N:  # K개의 선택지(그룹)에서 N명이 각자 하나씩 선택한 상황
        groups = [[] for _ in range(K)]  # 각 그룹에 속한 학생 목록
        # 각 학생을 확인하며 각 그룹에 배정
        for i in range(N):
            index = selected[i]
            groups[index].append(arr[i])
        # 1명도 배치되지 않은 그룹이 있는 경우 무시
        for i in range(K):
            if len(groups[i]) == 0: return
        min_score = int(1e9)  # 가장 전투력이 낮은 그룹
        max_score = -int(1e9)  # 가장 전투력이 높은 그룹
        # 각 그룹을 하나씩 확인하며
        for group in groups:
            # 해당 그룹의 전투력 계산
            score = 0
            for power in group:
                score += len(group) * power
            min_score = min(min_score, score)
            max_score = max(max_score, score)
        # 가장 전투력이 높은 그룹과 전투력이 낮은 그룹의 전투력 차이의 최솟값
        global answer
        answer = min(answer, abs(min_score - max_score))
        return
    # 0번부터 K-1번까지 하나씩 그룹의 인덱스(index)를 확인하며
    for i in range(K):
```

```
        selected.append(i)  # 현재 원소 선택
        dfs(N, K, arr, depth + 1)  # 재귀 함수 호출
        selected.pop()  # 현재 원소 선택 취소

selected = []  # 현재 중복 순열에 포함된 원소의 인덱스(index)

# 학생의 수(N), 그룹의 수(K), 협동력 점수 배열(arr)
def solution(N, K, arr):
    # 가장 전투력이 높은 그룹과 전투력이 낮은 그룹의 전투력 차이의 최솟값
    global answer
    answer = int(1e9)
    dfs(N, K, arr, 0)
    return answer
```

- **Java 정답 코드 예시**

```
import java.util.*;

class Solution {
    // 가장 전투력이 높은 그룹과 전투력이 낮은 그룹의 전투력 차이의 최솟값
    public static int answer = (int) 1e9;
    // 현재 중복 순열에 포함된 원소의 인덱스(index)
    public static int[] selected = new int[7];

    public static void dfs(int N, int K, int[] arr, int depth) {
        // 모든 중복 순열을 확인하는 부분
        if (depth == N) { // K개의 선택지(그룹)에서 N명이 각자 하나씩 선택한 상황
            // 각 그룹에 속한 학생 목록
            ArrayList<Integer>[] groups = new ArrayList[K];
            for (int i = 0; i < K; i++) {
                groups[i] = new ArrayList<Integer>();
            }
            // 각 학생을 확인하며 각 그룹에 배정
            for (int i = 0; i < N; i++) {
                int index = selected[i];
                groups[index].add(arr[i]);
            }
            // 1명도 배치되지 않은 그룹이 있는 경우 무시
            for (int i = 0; i < K; i++) {
                if (groups[i].size() == 0) return;
```

```
            }
            int minScore = (int) 1e9; // 가장 전투력이 낮은 그룹
            int maxScore = (int) -1e9; // 가장 전투력이 높은 그룹
            // 각 그룹을 하나씩 확인하며
            for (int i = 0; i < K; i++) {
                // 해당 그룹의 전투력 계산
                int score = 0;
                for (int j = 0; j < groups[i].size(); j++) {
                    int power = groups[i].get(j);
                    score += groups[i].size() * power;
                }
                minScore = Math.min(minScore, score);
                maxScore = Math.max(maxScore, score);
            }
            // 가장 전투력이 높은 그룹과 전투력이 낮은 그룹의 전투력 차이의 최솟값
            answer = Math.min(answer, Math.abs(minScore - maxScore));
            return;
        }
        // 0번부터 K-1번까지 하나씩 그룹의 인덱스(index)를 확인하며
        for (int i = 0; i < K; i++) {
            selected[depth] = i; // 현재 원소 선택
            dfs(N, K, arr, depth + 1); // 재귀 함수 호출
        }
    }

    // 학생의 수(N), 그룹의 수(K), 협동력 점수 배열(arr)
    public static int solution(int N, int K, int[] arr) {
        dfs(N, K, arr, 0);
        return answer;
    }
}
```