최고의 집합

문제 설명

자연수 n 개로 이루어진 중복 집합(multi set, 편의상 이후에는 "집합"으로 통칭) 중에 다음 두 조건을 만족하는 집합을 최고의 집합이라고 합니다.

1. 각 원소의 합이 S가 되는 수의 집합
2. 위 조건을 만족하면서 각 원소의 곱 이 최대가 되는 집합

예를 들어서 자연수 2개로 이루어진 집합 중 합이 9가 되는 집합은 다음과 같이 4개가 있습니다.
{ 1, 8 }, { 2, 7 }, { 3, 6 }, { 4, 5 }
그중 각 원소의 곱이 최대인 { 4, 5 }가 최고의 집합입니다.

집합의 원소의 개수 n과 모든 원소들의 합 s가 매개변수로 주어질 때, 최고의 집합을 return 하는 solution 함수를 완성해주세요.

제한사항

- 최고의 집합은 **오름차순으로 정렬된 1차원 배열(list, vector)** 로 return 해주세요.
- 만약 최고의 집합이 존재하지 않는 경우에 **크기가 1인 1차원 배열(list, vector)** 에 `-1` 을 채워서 return 해주세요.
- 자연수의 개수 n은 1 이상 10,000 이하의 자연수입니다.
- 모든 원소들의 합 s는 1 이상, 100,000,000 이하의 자연수입니다.

입출력 예

n	s	result
2	9	[4, 5]
2	1	[-1]
2	8	[4, 4]

초기 접근

```
1. 재귀를 통해 N개를 뽑고, N개가 뽑혔을 때 최대값과 비교하여 갱신한다.

=> N이 최대 1만.. => 재귀를 1만번? ㄷㄷ..

연산도 많고, 비효율적이다.
```

N개의 곱을 가장 크게 하려면 어떻게 하면 될까?

★ Point (문제에서는 중복집합 (같은 값이 포함되는) 으로 구성되었다.

만약 목표(S)가 15

{ 1, 14 } == 14

{ 2, 13 } == 26

{ 3, 12 } == 36

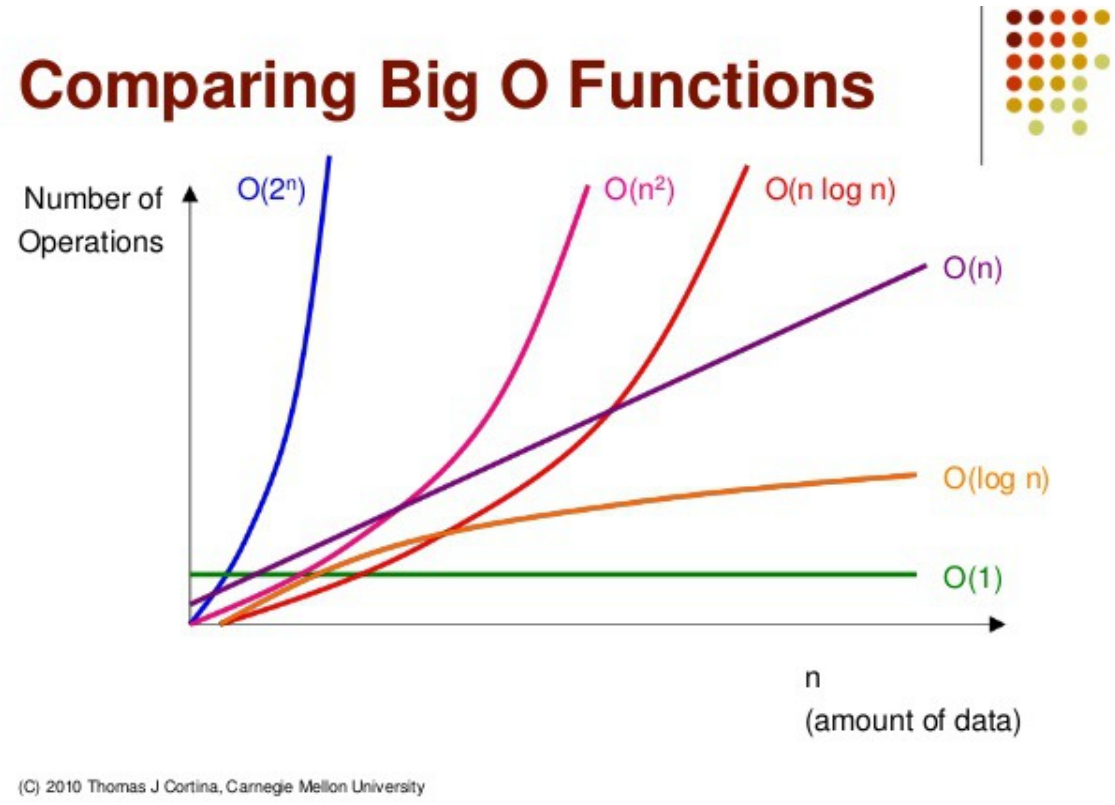
{ 4, 11 } == 44

{ 5, 10 } == 50

{ 6, 9 } == 54

{ 7, 8 } == 56

곱해져야 하는 값의 차이가 비슷할 수록 지수 승 형태로 커지게 됨 (O(n^숫자갯수)



faster $O(1) < O(\log n) < O(n) < O(n \log n) < O(n^2) < O(2^n)$ slower

(상수함수 < 로그함수 < 선형함수 < 다항함수 < 지수함수)

풀이

1. 전체 값을 갯수로 나눈 값을 균등하게 갖는다.

2. 그 나머지 값을 각 항에 0이 될때 까지 1씩 더해줘 완성된 값의 곱이 최대 값이 된다.

코드

```
package Algo_Study_Programmers;

import java.io.BufferedReader;

public class Solution_최고의집합 {

    public static int[] solution(int n, int s) {

      if( n > s ) {
        return new int[] { -1 };
      }

      int[] answer = new int[n];

      for(int i = 0; i < n; i++) {
        answer[i] = s / n;
      }

      for(int i = 0; i < s % n; i++) {
        answer[i]++;
      }
        return answer;
    }
  public static void main(String[] args) {
    int n = 2;
    int s = 9;

    solution(n, s);
```

```
    }
}
```