

문제

어떤 수 X 가 1보다 큰 제곱수로 나누어 떨어지지 않을 때, 제곱 L.L.수라고 한다. 제곱수는 정수의 제곱이다. min과 max가 주어지면, min과 max를 포함한 사이에 제곱 L.L.수가 몇 개 있는지 출력한다.

입력

첫째 줄에 min과 max가 주어진다. min은 1보다 크거나 같고, 1,000,000,000,000보다 작거나 같은 자연수이고, max는 min보다 크거나 같고, min+1,000,000보다 작거나 같은 자연수이다.

출력

첫째 줄에 [min,max]구간에 제곱 L.L.수가 몇 개인지 출력한다.

예제 입력 1 복사

1 10

예제 출력 1 복사

7

중요!

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓

★ 1보다 큰 제곱수 부터 → 최소 2^2 이상.

2^2 2^3 3^2 2^4 5^2 3^3 2^5 7^2

→ ① 소수의 제곱들...

4 8 9 16 25 27 32 49

(제곱 한 수의 모습 포함)

생각나는 풀이: 소수들을 제곱해 나가면서 min/max 구간 안에 있는 수라면 answer에 +1 한다. x^2 이 max를 넘으면 break 한다.

재귀?

시간 초과로 본 풀.

1016번 문제는 제곱 \ll 수라는 이름으로 소개되어 있습니다. 최소와 최댓값을 입력받았을 때, 최솟값과 최댓값을 포함한 사이의 값들 중 제곱 \ll 수의 개수를 찾는 문제입니다. 문제에 의하면 제곱 \ll 수는 "어떤 수 X 가 1보다 큰 제곱수로 나누어 떨어지지 않을 때의 X "를 의미합니다. 일반적으로, 제곱수는 자연수의 제곱을 의미합니다.

문제만 봤을 때, 구현 자체는 생각보다 간단할 수 있습니다. min부터 max까지 숫자를 순서대로 나열한 list에서 max보다 작거나 같은 제곱 수의 배수들을 모두 제외하면 된다고 생각할 수 있지요. 하지만, min의 범위가 1,000,000,000,000 까지 허용되기 때문에 이 방법은 다시 생각해 봐야합니다. 반면, max는 min보다 크고 min + 1,000,000 보다는 작거나 같은 수이므로 입력받을 수 있는 수의 크기에 비해 다루야 하는 데이터의 수는 상대적으로 작은 것을 알 수 있습니다.

이러한 조건에서, 만약 앞에서 말한 것처럼 max보다 작은 제곱 수들을 모두 구하고, 또 구해진 각각의 제곱수에 대해 min부터 max까지의 수들이 나누어 떨어지는지 일일이 확인하여 문제를 풀이한다면, min이 1,000,000,000,000과 같은 극단적인 수로 주어질 때 시간 초과를 충분히 예상할 수 있는 상황입니다. 실제로 이렇게 해서 시간 초과를 이미 경험한 상태이기도 했구요.

Hint!

시간초과 문제를 해결하기 위해 다음 방법을 생각해 풀어했습니다:

1. 먼저, min과 max까지의 수의 개수를 길이로 갖고 모든 요소가 1인 list를 만듭니다. 예를 들면, min이 20, max가 50일 경우 20~50까지의 숫자는 총 31개이므로 길이가 31이고 모든 요소의 값이 1인 list, [1, 1, 1, ..., 1] 을 생성합니다. 이 list를 validation list 라고 부르겠습니다.
2. max보다 작은 모든 제곱 수를 구해 list를 만들고 이 list를 가지고 for문을 돌립니다.
 1. 이 때, 먼저 min보다 큰 최초의 제곱수의 배수를 구합니다. min이 20, max가 50이고 현재 for문에 들어온 제곱수가 16이라면 32가 그 값이 됩니다.
 2. 이전 단계에서 계산한 32 에 min 값을 빼면 12이고 이 값은 validation list의 의미상 숫자 32의 index와 같습니다. min부터 max까지의 숫자 중 13번째 (index가 12이면 list의 13번째 숫자이죠?) 숫자라는 뜻입니다. 그리고 validation list에서 이 index의 값을 0으로 대체 합니다. 제곱수의 배수이기 때문에 제곱 \ll 수가 아니기 때문입니다.
3. validation list에서 방금 0으로 대체 시킨 index에서 현재 제곱수 만큼 떨어진 요소의 값을 반복하여 0으로 대체합니다. 이전 단계에서 현재 제곱수가 16이고 12의 index 값(32)이 0으로 대체 되었으므로, $12 + 16 = 28$ 의 index 값도 0으로 대체 됩니다. 이 index에 해당하는 값(48)도 제곱수(16)의 배수이기 때문이죠. 단, 이 작업은 index가 validation list의 길이를 초과할 경우 중단됩니다.

3. validation list의 총합이 제곱 \ll 수의 개수가 됩니다.

이렇게 풀이하여 시간 초과를 해결할 수 있었습니다. 나누어 떨어지는지 검토할 필요가 없는 숫자들까지 모두 순회하여 검토하는 것보다는 배수의 특징을 이용하여 우리가 목표로 하는 최대 1,000,000개의 숫자들에 대해서만 제곱수의 배수를 구하는 것이 효과적일 것이라는 생각으로 출발하게 되었네요.

min : 20 / max : 50

validation: [20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34 ... 50]

square : 16 → 32 / cur_idx = 32 - 20 = 12

validation[cur_idx] = 0 (cur_idx <= max - min)

cur_idx += square