1. 입국심사

문제 설명

n 명이 입국심사를 위해 줄을 서서 기다리고 있습니다. 각 입국심사대에 있는 심사관마다 심사하는데 걸리는 시간은 다릅니다.

처음에 모든 심사대는 비어있습니다. 한 심사대에서는 동시에 한 명만 심사를 할 수 있습니다. 가장 앞에 서 있는 사람은 비어 있는 심사대로 가서 심사를 받을 수 있습니다. 하지만 더 빨리 끝나는 심사대가 있으면 기다렸다가 그곳으로 가서 심사를 받을 수도 있습니다.

모든 사람이 심사를 받는데 걸리는 시간을 최소로 하고 싶습니다.

입국심사를 기다리는 사람 수 n, 각 심사관이 한 명을 심사하는데 걸리는 시간이 담긴 배열 times 가 매개변수로 주어질 때, 모든 사람이 심사를 받는데 걸리는 시간의 최솟값을 return 하도록 solution 함수를 작성해주세요.

제한사항

- 입국심사를 기다리는 사람은 1 명 이상 1,000,000,000 명 이하입니다.
- 각 심사관이 한 명을 심사하는데 걸리는 시간은 1 분 이상 1,000,000,000 분 이하입니다.
- 심사관은 1 명 이상 100,000 명 이하입니다.

입출력 예

n	times	return
6	[7, 10]	28

입출력 예 설명

가장 첫 두 사람은 바로 심사를 받으러 갑니다.

7 분이 되었을 때, 첫 번째 심사대가 비고 3 번째 사람이 심사를 받습니다.

10 분이 되었을 때, 두 번째 심사대가 비고 4 번째 사람이 심사를 받습니다.

14 분이 되었을 때, 첫 번째 심사대가 비고 5 번째 사람이 심사를 받습니다.

20 분이 되었을 때, 두 번째 심사대가 비지만 6 번째 사람이 그곳에서 심사를 받지 않고 1 분을 더 기다린 후에 첫 번째 심사대에서 심사를 받으면 28 분에 모든 사람의 심사가 끝납니다.

출처

※ 공지 - 2019 년 9 월 4 일 문제에 새로운 테스트 케이스를 추가하였습니다.
도움을 주신 weaver9651 님께 감사드립니다.

```
import java.util.*;

class Solution {
   public long solution(int n, int[] times) {
      Arrays.sort(times);
      return binerySearch(times, n, times[times.length - 1]);
   }

   long binerySearch(int[] times, int n, long max){
      long left = 1, right = max * n;
      long mid = 0;
      long ans = Long.MAX_VALUE;

      while(left <= right){
         mid = (left + right) / 2;

         if(isPassed(times, n, mid)){
            ans = ans > mid ? mid : ans;
            right = mid - 1;
         } else {
            left = mid + 1;
         }
      }
      return ans;
   }

   boolean isPassed(int[] times, int n, long mid){
      long amount = 0;

      for(int i = 0 ; i < times.length ; ++i){
         amount += mid / times[i];
      }

      if(amount >= n) return true;
      else return false;
   }
}
```

https://velog.io/@hyeon930/%ED%94%84%EB%A1%9C%EA%B7%B8%EB%9E%98%EB%A8%B8%EC%8A%A4-%EC%9E%85%EA%B5%AD%EC%8B%AC%EC%82%AC-Java

2. 징검다리

문제 설명

출발지점부터 distance 만큼 떨어진 곳에 도착지점이 있습니다. 그리고 그사이에는 바위들이 놓여있습니다. 바위 중 몇 개를 제거하려고 합니다.
예를 들어, 도착지점이 25 만큼 떨어져 있고, 바위가 [2, 14, 11, 21, 17] 지점에 놓여있을 때 바위 2 개를 제거하면 출발지점, 도착지점, 바위 간의 거리가 아래와 같습니다.

제거한 바위의 위치	각 바위 사이의 거리	거리의 최솟값
[21, 17]	[2, 9, 3, 11]	2
[2, 21]	[11, 3, 3, 8]	3
[2, 11]	[14, 3, 4, 4]	3
[11, 21]	[2, 12, 3, 8]	2
[2, 14]	[11, 6, 4, 4]	4

위에서 구한 거리의 최솟값 중에 가장 큰 값은 4 입니다.

출발지점부터 도착지점까지의 거리 distance, 바위들이 있는 위치를 담은 배열 rocks, 제거할 바위의 수 n 이 매개변수로 주어질 때, 바위를 n 개 제거한 뒤 각 지점 사이의 거리의 최솟값 중에 가장 큰 값을 return 하도록 solution 함수를 작성해주세요.

제한사항

- 도착지점까지의 거리 distance 는 1 이상 1,000,000,000 이하입니다.
- 바위는 1 개 이상 50,000 개 이하가 있습니다.
- n 은 1 이상 바위의 개수 이하입니다.

입출력 예

distance	rocks	n	return
25	[2, 14, 11, 21, 17]	2	4

입출력 예 설명

문제에 나온 예와 같습니다.

출처

※ 공지 - 2020 년 2 월 17 일 테스트케이스가 추가되었습니다.

```java
import java.util.*;

class Solution {
    public int solution(int distance, int[] rocks, int n) {
        // 이분탐색은 오름차순으로 정렬되어있는 경우를 전제로한다.
        Arrays.sort(rocks);
        return binerySearch(distance, rocks, n);
    }

    int binerySearch(int distance, int[] rocks, int n){
        long ans = 0;
        long left = 1, right = distance, mid = 0;

        while(left <= right){
            int cnt = 0;
            int prev = 0;
            mid = (left + right) / 2;

            for(int i = 0 ; i < rocks.length ; ++i){
                if(rocks[i] - prev < mid){
                // mid보다 작은 값이 존재한다는 뜻으로
                    // 해당 돌을 제거한다.
                    cnt++;
                } else {
                                // mid보다 크거나 같은 값이 존재하므로
                    // prev를 현재 돌로 초기화한다.
                    prev = rocks[i];
                }
            }

            // 마지막 돌과 도착점 사이의 거리도 확인한다.
            if(distance - prev < mid) cnt++;

            if(cnt <= n){
                // 주어진 n 보다 작거나 같은 만큼 돌을 없애서
                // 최솟값 x를 만들 수 있다.
                ans = mid > ans ? mid : ans;
                left = mid + 1;
            } else {
                right = mid - 1;
            }
        }
        return (int) ans;
    }
}
```

https://velog.io/@hyeon930/%ED%94%84%EB%A1%9C%EA%B7%B8%EB%9E%98%EB%A8%B8%EC%8A%A4-%EC%A7%95%EA%B2%80%EB%8B%A4%EB%A6%AC-Java