题目描述：

为了充分发挥 GPU 算力，需要尽可能多的将任务交给 GPU 执行，现在有一个任务数组，数组元素表示在这 1 秒内新增的任务个数且每秒都有新增任务，假设 GPU 最多一次执行 n 个任务，一次执行耗时 1 秒，在保证 GPU 不空闲情况下，最少需要多长时间执行完成

输入描述：

第一个参数为 GPU 一次最多执行的任务个数，取值范围[1, 10000]

第二个参数为任务数组长度，取值范围[1, 10000]

第三个参数为任务数组，数字范围[1, 10000]

输出描述：

执行完所有任务最少需要多少秒

补充说明：

示例 1

输入：

```
3

5

1 2 3 4 5
```

输出：

```
6
```

说明：

一次最多执行 3 个任务，最少耗时 6s

示例 2

输入：

```
4
5
5 4 1 1 1
```

输出：

```
5
```

说明：

一次最多执行 4 个任务，最少耗时 5s

```
import java.util.*;

// 注意类名必须为 Main, 不要有任何 package xxx 信息
public class Main {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        // 注意 hasNext 和 hasNextLine 的区别
        while (in.hasNextLine()) { // 注意 while 处理多个 case
            int gpu = Integer.parseInt(in.nextLine());
            int count = Integer.parseInt(in.nextLine());
            String[] numString = in.nextLine().split(" ");
            int[] nums = new int[count];
            for (int i = 0; i < count; i ++) {
                nums[i] = Integer.parseInt(numString[i]);
            }
            int[] gpus = new int[gpu];
            int min;
            Queue<Integer> taskQueue = new LinkedList<>();
            for (int i = 0; i < nums[0]; i ++) {
                taskQueue.add(1);
            }
            int time = 0;
            while (!taskQueue.isEmpty()) {
                for (int j = 0; j < gpu && !taskQueue.isEmpty(); j ++) {
                    taskQueue.poll();
                }
                time ++;
                for (int i = 0; time < count && i < nums[time] ; i ++) {
                    taskQueue.add(1);
                }
            }

            System.out.println(time);
        }
    }
```

}