## 取出尽量少的球

时间限制：1s 空间限制：32MB 限定语言：不限

### 题目描述：

某部门开展Family Day开放日活动，其中有个从桶里取球的游戏，游戏规则如下：有N个容量一样的小桶等距排开，且每个小桶都默认装了数量不等的小球，每个小桶所装的小球数量记录在数组bucketBallNums中，游戏开始时，要求所有桶的小球总数不能超过SUM，如果小球总数超过SUM，则需对所有的小桶统一设置一个容量最大值maxCapacity，并需将超过容量最大值的小球拿出来，直至小桶里的小球数量小于maxCapacity；请您根据输入的数据，计算从每个小桶里拿出的小球数量?

### 限制规则一：

如果所有小桶的小球总和小于SUM，则无需设置容量值，并且无需从小桶中拿球出来，返回结果[]；

### 限制规则二：

如果所有小桶的小球总和大于SUM，则需设置容量最大值maxCapacity，并且需从小桶中拿球出来，返回从每个小桶拿出的小球数量组成的数组；

### 输入描述：

第一行输入2个正整数，数字之间使用空格隔开，其中第一个数字表示SUM；第二个数字表示bucketBallNums数组长度；

第二行输入N个正整数，数字之间使用空格隔开，表示bucketBallNums的每一项；

### 输出描述：

从每个小桶里拿出的小球数量，并使用一维数组表示

### 补充说明：

1 <= bucketBallNums[i] <= 10^9
1 <= bucketBallNums.length = N <= 10^5
1 <= maxCapacity <= 10^9
1 <= SUM <= 10^9

## 示例1

**输入：**

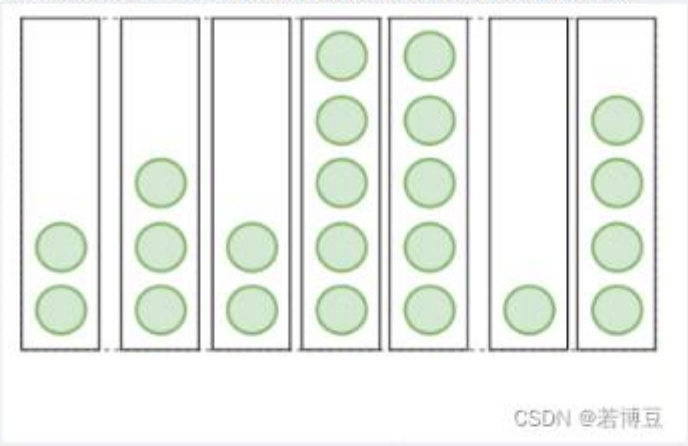
14 7
2 3 2 5 5 1 4

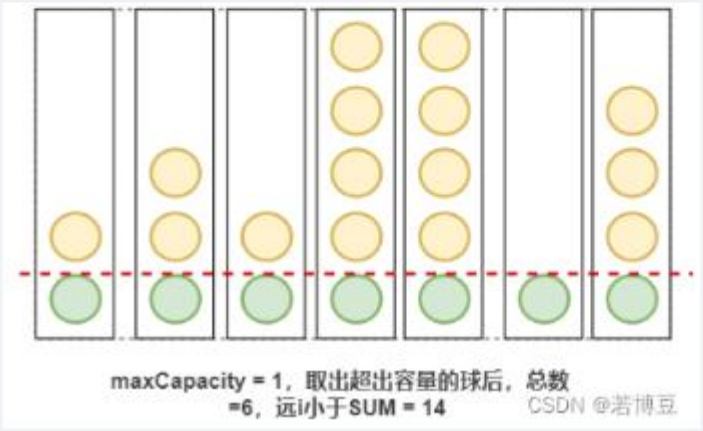
**输出：**

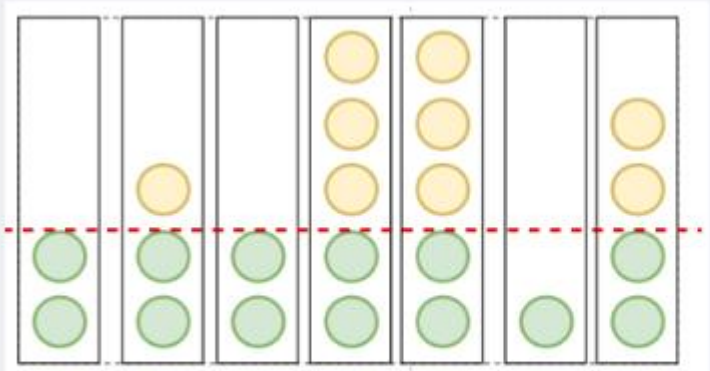
[0,1,0,3,3,0,2]

**说明：**

小球总数为22，SUM=14，超出范围了，需从小桶取球，maxCapacity=1，取出球后，桶里剩余小球总和为7，远小于14；maxCapacity=2，取出小球后，桶里剩余小球总和为13，小于最大值；maxCapacity=3，取出小球后，桶里剩余小球总和为16，大于14；因此maxCapacity=2，每个小桶小球数量大于2的都需要拿出来；

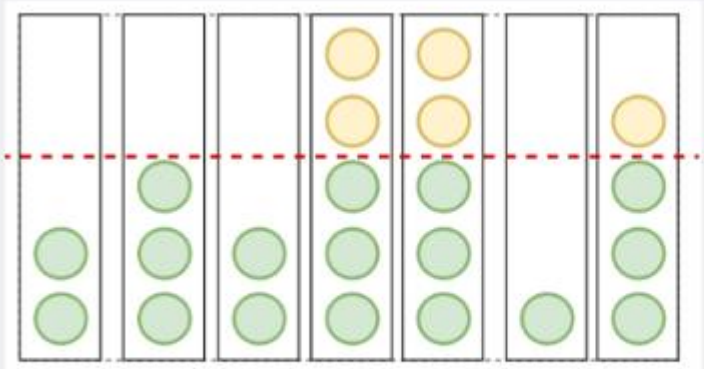




maxCapacity = 2，取出超出容量的球后，总数=13，小于SUM = 14





maxCapacity = 3，取出超出容量的球后，总数=17，大于SUM = 14

## 示例2

**输入：**

3 3
1 2 3

**输出：**

[0,1,2]

**说明：**

小球总数为6，SUM=3，超出范围了，需从小桶取球，maxCapacity=1，则小球总数为3，从1号桶取0个球，1号桶取1个球，2号桶取2个球；

## 示例3

**输入：**

6 2
3 2

**输出：**

[]

**说明：**

小球总数为5，SUM=6，在范围内，无需从小桶取球；

## 解题思路：

**先求出maxCapacity的最小值和最大值。**

最小值：SUM/bucketBallNums，比如SUM=15，bucketBallNums=7；
则maxCapacity至少需要2层，因为2层只有14个球，而三层至多可能有21个球。
最大值：所有小桶里面球数最多的那个

## 如例1所示：

对maxCapacity从14/7=2遍历到5；
当maxCapacity=2时，取出多余的球后，剩余球为13，小于14
当maxCapacity=3是，取出多余的球后，剩余球为17，大于14
此时确定maxCapacity=2，且取出的球的数组为[0,1,0,3,3,0,2]

```
public class Main{

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        int SUM = sc.nextInt();
        int ballNumsLen = sc.nextInt();

        int[] ballNums = new int[ballNumsLen];
        int ballCount = 0;    //球的总数
        for( int i=0; i<ballNumsLen; i++){
            ballNums[i] = sc.nextInt();         //各个管子中的球的个数
            ballCount += ballNums[i];
        }



        if(SUM < ballCount){       //球的总数大于 SUM 时需要处理
            int min = SUM/ballNumsLen;           //maxCapacity 的最小值
            int max = Arrays.stream(ballNums).max().getAsInt(); //maxCapacity 的最大值

            int[] tempOut = new int[ballNumsLen];     //各个管子移除的球的个数数组（暂时
存放）
            int[] ballOut = {};     //各个管子移除的球的个数数组
            for(int i=min; i<=max; i++){

                for(int j=0; j<ballNumsLen; j++){     //每个管子都需要移除 i 个球
                    tempOut[j] = ballNums[j] - i > 0 ? ballNums[j] - i : 0;
                }

                int outCount = Arrays.stream(tempOut).sum();       //取出的球的总数
                int remainCount = ballCount - outCount; //剩余的球的总数

                if(remainCount > SUM){   //剩下的球大于 SUM，则输出
                    break;
                }else {
                    ballOut = Arrays.copyOf( tempOut, tempOut.length);
                }
            }

            System.out.println(ballOut.length     ==     0     ?     Arrays.toString(tempOut)     :
Arrays.toString(ballOut));
```

```
            }else {
                System.out.println("[]");
            }

        }
    }
```