题目描述

某软件系统会在运行过程中持续产生日志，系统每天运行N单位时间，运行期间每单位时间产生的日志条数保行在数组records中。records[i]表示第i单位时间内产生日志条数。
由于系统磁盘空间限制，每天可记录保存的日志总数上限为total条。
如果一天产生的日志总条数大于total，则需要对当天内每单位时间产生的日志条数进行限流后保存，请计算每单位时间最大可保存日志条数limit，以确保当天保存的总日志条数不超过total。
对于单位时间内产生日志条数不超过limit的日志全部记录保存;
对于单位时间内产生日志条数超过limit的日志，则只记录保存limit条日志；
如果一天产生的日志条数总和小于等于total，则不需要启动限流机制．result为-1。
请返回result的最大值或者-1。

输入描述

第一行为系统某一天运行的单位时间数N,1<=N<=10^5
第二行为表示这一天每单位时间产生的日志数量的数组records[]，0 <= records[i] <= 10^5
第三行为系统一天可以保存的总日志条数total。1 <= total <= 10^9

输出描述

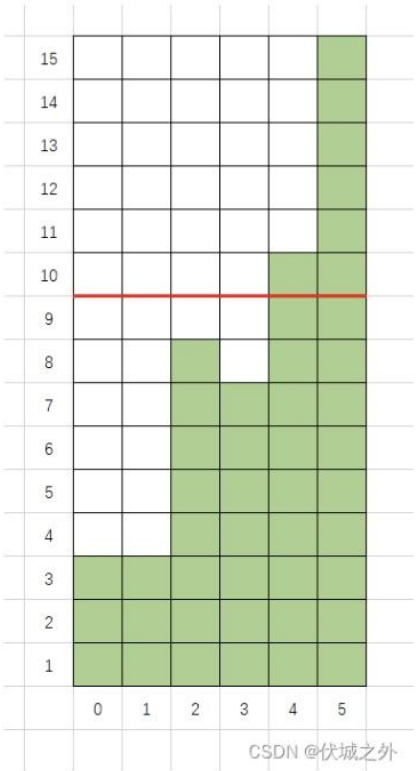
每单位时间内最大可保存的日志条数limit，如果不需要启动限流机制，返回-1。

用例

输入	6 3 3 8 7 10 15 40
输出	9
说明	无

题目解析

用例图示如下

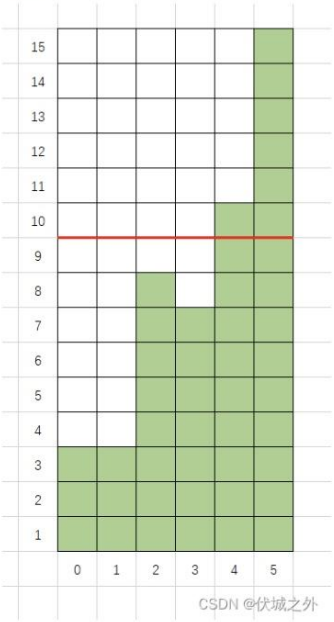


本题其实和华为OD机试 - 开放日活动_伏城之外的博客-CSDN博客

几乎一模一样，本题将采用二分查找策略解题。

题目解析

用例图示如下



本题其实和华为OD机试 - 开放日活动_伏城之外的博客-CSDN博客

几乎一模一样，本题将采用 二分查找 策略解题。

首先，根据华为OD机试 - 开放日活动_伏城之外的博客-CSDN博客

我们可以知道，本题有一个理想的limit，值为total / n，取这个limit，得到限流后总日志数只可能小于等于total，原因很简单，大家可以自己想想。

因此，我们可以将 total / n 当成最小limit取值，而最大limit取值其实就是max(records)，即初始时：

- max_limit = max(records)
- min_limit = total / n

我们每次都取min_limit和max_limit的二分值作为测试limit，测试逻辑如下：

- 遍历records每一个record，如果record数量小于等于limit，则不做限流，如果record大于limit，则限流为limit条，然后求限流后日志总条数tmp

得到日志总条数后，如果

- tmp > total，则说明limit取小了，则应该提高limit值，即让min_limit = limit
- tmp < total，则说明limit取大了，则应该降低limit值，即让max_limit = limit
- tmp == total，则说明limit取得刚刚好，此时的limit就是最佳limit

当然我们可能无法遇到tmp == total的情况，此时我们应该定义一个ans变量，保存tmp < total时的limit值，如果最终没有tmp == total的情况，则最后应该返回ans作为题解。

另外，在这些逻辑之前，我们可以先对records求和sum，如果sum <= total，则不需要做上面逻辑。

上面算法的时间复杂度为 O(log(total) * N)

- 1<=N<=10^5
- 1 <= total <= 10^9

因此性能还算比较优异。

JavaScript算法源码

```javascript
/* JavaScript Node ACM模式 控制台输入获取 */
const readline = require("readline");

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout,
});

const lines = [];
rl.on("line", (line) => {
  lines.push(line);

  if (lines.length === 3) {
    const n = lines[0] - 0;
```

JavaScript算法源码

```javascript
/* JavaScript Node ACM模式 控制台输入获取 */
const readline = require("readline");

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout,
});

const lines = [];
rl.on("line", (line) => {
  lines.push(line);

  if (lines.length === 3) {
    const n = lines[0] - 0;
    const records = lines[1].split(" ").map(Number);
    const total = lines[2] - 0;

    console.log(getResult(n, records, total));
    lines.length = 0;
  }
});

/**
 *
 * @param {*} n 系统某一天运行的单位时间数N
 * @param {*} records 这一天每单位时间产生的日志数量的数组
 * @param {*} total 系统一天可以保存的总日志条数total
 * @return {*} 每单位时间内最大可保存的日志条数limit，如果不需要启动限流机制，返回-1
 */
function getResult(n, records, total) {
  const sum = records.reduce((p, c) => p + c);

  // 如果一天产生的日志条数总和小于等于total，则不需要启动限流机制，result为-1
  if (sum <= total) return -1;

  // records.sort((a, b) => a - b);

  // let max_limit = records.at(-1);
  let max_limit = Math.max(...records);
  let min_limit = Math.floor(total / n);

  let ans = min_limit;

  while (max_limit - min_limit > 1) {
    let limit = Math.floor((max_limit + min_limit) / 2);

    let tmp = 0;
    records.forEach((record) => {
      tmp += Math.min(record, limit);
    });

    if (tmp > total) {
      max_limit = limit;
    } else if (tmp < total) {
      min_limit = limit;
      ans = limit;
    } else {
      return limit;
    }
  }

  return ans;
}
```

Java算法源码

```java
import java.util.Arrays;
import java.util.Scanner;

public class Main {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    int n = sc.nextInt();

    int[] records = new int[n];
    for (int i = 0; i < n; i++) {
      records[i] = sc.nextInt();
    }

    int total = sc.nextInt();

    System.out.println(getResult(n, records, total));
  }

  /**
   * @param n 系统某一天运行的单位时间数N
   * @param records 这一天每单位时间产生的日志数量的数组
   * @param total 系统一天可以保存的总日志条数total
   * @return 每单位时间内最大可保存的日志条数limit，如果不需要启动限流机制，返回-1
```

Java算法源码

```java
import java.util.Arrays;
import java.util.Scanner;

public class Main {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    int n = sc.nextInt();

    int[] records = new int[n];
    for (int i = 0; i < n; i++) {
      records[i] = sc.nextInt();
    }

    int total = sc.nextInt();

    System.out.println(getResult(n, records, total));
  }

  /**
   * @param n 系统某一天运行的单位时间数N
   * @param records 这一天每单位时间产生的日志数量的数组
   * @param total 系统一天可以保存的总日志条数total
   * @return 每单位时间内最大可保存的日志条数limit，如果不需要启动限流机制，返回-1
   */
  public static int getResult(int n, int[] records, int total) {
    int sum = Arrays.stream(records).reduce(Integer::sum).getAsInt();

    // 如果一天产生的日志条数总和小于等于total，则不需要启动限流机制，result为-1
    if (sum <= total) return -1;

    //    Arrays.sort(records);

    //    int max_limit = records[records.length - 1];
    int max_limit = Arrays.stream(records).max().getAsInt();
    int min_limit = total / n;

    int ans = min_limit;
    while (max_limit - min_limit > 1) {
      int limit = (max_limit + min_limit) / 2;

      int tmp = 0;
      for (int record : records) {
        tmp += Math.min(record, limit);
      }

      if (tmp > total) {
        max_limit = limit;
      } else if (tmp < total) {
        min_limit = limit;
        ans = limit;
      } else {
        return limit;
      }
    }

    return ans;
  }
}
```

Python算法源码

```python
# 输入获取
n = int(input())
records = list(map(int, input().split()))
total = int(input())


# 算法入口
def getResult(n, records, total):
    """
    :param n: 系统某一天运行的单位时间数N
    :param records: 这一天每单位时间产生的日志数量的数组
    :param total: 系统一天可以保存的总日志条数total
    :return: 每单位时间内最大可保存的日志条数limit，如果不需要启动限流机制，返回-1
    """
    sumV = sum(records)

    # 如果一天产生的日志条数总和小于等于total，则不需要启动限流机制，result为-1
    if sumV <= total:
        return -1

    # records.sort()

    # max_limit = records[-1]
    max_limit = max(records)
    min_limit = int(total / n)

    ans = min_limit
    while max_limit - min_limit > 1:
```

Python算法源码

复制

```python
# 输入获取
n = int(input())
records = list(map(int, input().split()))
total = int(input())


# 算法入口
def getResult(n, records, total):
    """
    :param n: 系统某一天运行的单位时间数N
    :param records: 这一天每单位时间产生的日志数量的数组
    :param total: 系统一天可以保存的总日志条数total
    :return: 每单位时间内最大可保存的日志条数limit，如果不需要启动限流机制，返回-1
    """
    sumV = sum(records)

    # 如果一天产生的日志条数总和小于等于total，则不需要启动限流机制，result为-1
    if sumV <= total:
        return -1

    # records.sort()

    # max_limit = records[-1]
    max_limit = max(records)
    min_limit = int(total / n)

    ans = min_limit
    while max_limit - min_limit > 1:
        limit = int((max_limit + min_limit) / 2)

        tmp = 0
        for record in records:
            tmp += min(record, limit)

        if tmp > total:
            max_limit = limit
        elif tmp < total:
            min_limit = limit
            ans = limit
        else:
            return limit

    return ans


# 调用算法
print(getResult(n, records, total))
```