

题目描述

给定一个数组nums，将元素分为若干个组，使得每组和相等，求出满足条件的所有分组中，组内元素和的最小值。

输入描述

第一行输入 m
接着输入m个数，表示此数组nums
数据范围：1<=m<=50, 1<=nums[i]<=50

输出描述

最小拆分数组和

用例

输入	7 4 3 2 3 5 2 1
输出	5
说明	可以等分的情况有： 4 个子集 (5) , (1,4) , (2,3) , (2,3) 2 个子集 (5, 1, 4) , (2,3, 2,3) 但最小的为5。

题目解析

本题就是[华为机试 - 叠积木_伏城之外的博客-CSDN博客_叠积木 算法](#)
的变种题，解法一致。

JavaScript算法源码

```
1  /* JavaScript Node ACM模式 控制台输入获取 */
2  const readline = require("readline");
3
4  const rl = readline.createInterface({
5    input: process.stdin,
6    output: process.stdout,
7  });
8
9  const lines = [];
10 rl.on("line", (line) => {
11   lines.push(line);
12
13   if (lines.length === 2) {
14     const m = parseInt(lines[0]);
15     const arr = lines[1].split(" ").map(Number);
16
17     console.log(getResult(arr, m));
18
19     lines.length = 0;
20   }
21 });
22
23 function getResult(arr, m) {
24   const sum = arr.sort((a, b) => b - a).reduce((p, c) => p + c);
25
26   while (m) {
27     if (canPartitionMSubsets([...arr], sum, m)) return sum / m;
28     m--;
29   }
30
31   return sum;
32 }
33
34 function canPartitionMSubsets(arr, sum, m) {
35   if (sum % m !== 0) return false;
36
37   const subSum = sum / m;
38 }
```

JavaScript算法源码

```
1  /* JavaScript Node ACM模式 控制台输入获取 */
2  const readline = require("readline");
3
4  const rl = readline.createInterface({
5    input: process.stdin,
6    output: process.stdout,
7  });
8
9  const lines = [];
10 rl.on("line", (line) => {
11   lines.push(line);
12
13   if (lines.length === 2) {
14     const m = parseInt(lines[0]);
15     const arr = lines[1].split(" ").map(Number);
16
17     console.log(getResult(arr, m));
18
19     lines.length = 0;
20   }
21 });
22
23 function getResult(arr, m) {
24   const sum = arr.sort((a, b) => b - a).reduce((p, c) => p + c);
25
26   while (m) {
27     if (canPartitionMSubsets([...arr], sum, m)) return sum / m;
28     m--;
29   }
30
31   return sum;
32 }
33
34 function canPartitionMSubsets(arr, sum, m) {
35   if (sum % m !== 0) return false;
36
37   const subSum = sum / m;
38
39   if (subSum < arr[0]) return false;
40
41   while (arr[0] === subSum) {
42     arr.shift();
43     m--;
44   }
45
46   const buckets = new Array(m).fill(0);
47
48   return partition(arr, 0, buckets, subSum);
49 }
50
51 function partition(arr, index, buckets, subSum) {
52   if (index === arr.length) return true;
53
54   const select = arr[index];
55
56   for (let i = 0; i < buckets.length; i++) {
57     if (i > 0 && buckets[i] === buckets[i - 1]) continue;
58     if (select + buckets[i] <= subSum) {
59       buckets[i] += select;
60       if (partition(arr, index + 1, buckets, subSum)) return true;
61       buckets[i] -= select;
62     }
63   }
64
65   return false;
66 }
```


Java算法源码

感谢m0_71826536指正41行错误，41行在用例

```
5
5 5 5 5 5
```

时会出现越界异常

```
1  import java.util.LinkedList;
2  import java.util.Scanner;
3
4  public class Main {
5    public static void main(String[] args) {
6      Scanner sc = new Scanner(System.in);
7
8      int m = sc.nextInt();
9
10     LinkedList<Integer> link = new LinkedList<>();
11     for (int i = 0; i < m; i++) {
12       link.add(sc.nextInt());
13     }
14 }
```

 伏城之外 已关注

👍 0 🗨 6 📄 0 📁 专栏目录 已订阅

Java算法源码

感谢m0_71826536指正41行错误，41行在用例

```
5
5 5 5 5 5
```

时会出现越界异常

```
1 import java.util.LinkedList;
2 import java.util.Scanner;
3
4 public class Main {
5     public static void main(String[] args) {
6         Scanner sc = new Scanner(System.in);
7
8         int m = sc.nextInt();
9
10        LinkedList<Integer> link = new LinkedList<>();
11        for (int i = 0; i < m; i++) {
12            link.add(sc.nextInt());
13        }
14
15        System.out.println(getResult(link, m));
16    }
17
18    public static int getResult(LinkedList<Integer> link, int m) {
19        link.sort((a, b) -> b - a);
20
21        int sum = 0;
22        for (Integer ele : link) {
23            sum += ele;
24        }
25
26        while (m > 0) {
27            LinkedList<Integer> link_cp = new LinkedList<>(link);
28            if (canPartitionMSubsets(link_cp, sum, m)) return sum / m;
29            m--;
30        }
31
32        return sum;
33    }
34
35    public static boolean canPartitionMSubsets(LinkedList<Integer> link, int sum, int m)
36    {
37        if (sum % m != 0) return false;
38        int subSum = sum / m;
39
40        if (subSum < link.get(0)) return false;
41
42        // while (link.get(0) == subSum) { // 此段代码可能越界
43        while (link.size() > 0 && link.get(0) == subSum) {
44            link.removeFirst();
45            m--;
46        }
47
48        int[] buckets = new int[m];
49        return partition(link, 0, buckets, subSum);
50    }
51
52    public static boolean partition(LinkedList<Integer> link, int index, int[] buckets, int
53    {
54        if (index == link.size()) return true;
55        int select = link.get(index);
56
57        for (int i = 0; i < buckets.length; i++) {
58            if (i > 0 && buckets[i] == buckets[i - 1]) continue;
59            if (select + buckets[i] <= subSum) {
60                buckets[i] += select;
61                if (partition(link, index + 1, buckets, subSum)) return true;
62                buckets[i] -= select;
63            }
64        }
65
66        return false;
67    }
68 }
```

Python算法源码

```
1 # 输入获取
2 m = int(input())
3 link = list(map(int, input().split()))
4
```

伏城之外 已关注

👍 0 🗨 0 ⭐ 6 📁 0 📖 0 专栏目录 已订阅

Python算法源码

```
1 # 输入获取
2 m = int(input())
3 link = list(map(int, input().split()))
4
5
6 # 算法入口
7 def getResult(link, m):
8     link.sort(reverse=True)
9
10    sumV = 0
11    for ele in link:
12        sumV += ele
13
14    while m > 0:
15        if canPartitionMSubsets(link[:], sumV, m):
16            return int(sumV / m)
17        m -= 1
18
19    return sumV
20
21
22 def canPartitionMSubsets(link, sumV, m):
23     if sumV % m != 0:
24         return False
25
26     subSum = sumV / m
27
28     if subSum < link[0]:
29         return False
30
31     while len(link) > 0 and link[0] == subSum:
32         link.pop(0)
33         m -= 1
34
35     buckets = [0] * m
36
37     return partition(link, 0, buckets, subSum)
38
39
40 def partition(link, index, buckets, subSum):
41     if index == len(link):
42         return True
43
44     select = link[index]
45
46     for i in range(len(buckets)):
47         if i > 0 and buckets[i] == buckets[i - 1]:
48             continue
49
50         if select + buckets[i] <= subSum:
51             buckets[i] += select
52             if partition(link, index + 1, buckets, subSum):
53                 return True
54             buckets[i] -= select
55
56     return False
57
58
59 # 算法调用
60 print(getResult(link, m))
```