火柴拼正方形\_LeetCode\_473\_MatchsticksToSquare

# 火柴拼正方形\_LeetCode\_473\_MatchsticksToSquare

## 题目介绍

\* 难度：Medium

\* 题目：LeetCode\_473\_MatchsticksToSquare\_火柴拼正方形

\* DateTime：2018-10-06 16:09 京东集团总部

\* https://leetcode.com/problems/matchsticks-to-square/description/

\* 题目介绍：

\* Remember the story of Little Match Girl? By now, you know exactly

\* what matchsticks the little match girl has, please find out

\* a way you can make one square by using up all those matchsticks.

\* You should not break any stick, but you can link them up,

\* and each matchstick must be used exactly one time.

\* <p>

\* Your input will be several matchsticks the girl has,

\* represented with their stick length. Your output will either be true or false,

\* to represent whether you could make one square using all the matchsticks

\* the little match girl has.

\* <p>

\* Example 1:

\* Input: [1,1,2,2,2]

\* Output: true

\* <p>

\* Explanation: You can form a square with length 2,

\* one side of the square came two sticks with length 1.

\* Example 2:

\* Input: [3,3,3,3,4]

\* Output: false

\* Explanation: You cannot find a way to form a square with all the matchsticks.

\*

\* Note:

\* The length sum of the given matchsticks is in the range of 0 to 10^9.

\* The length of the given matchstick array will not exceed 15.

\* 中文：领扣上的翻译：

\* 还记得童话《卖火柴的小女孩》吗？现在，你知道小女孩有多少根火柴，

\* 请找出一种能使用所有火柴拼成一个正方形的方法。不能折断火柴，可以把火柴连接起来，并且每根火柴都要用到。

\* 输入为小女孩拥有火柴的数目，每根火柴用其长度表示。输出即为是否能用所有的火柴拼成正方形。

\*

\* 示例 1:

\* 输入: [1,1,2,2,2]

\* 输出: true

\* 解释: 能拼成一个边长为2的正方形，每边两根火柴。

\* 示例 2:

\*

\* 输入: [3,3,3,3,4]

\* 输出: false

\*

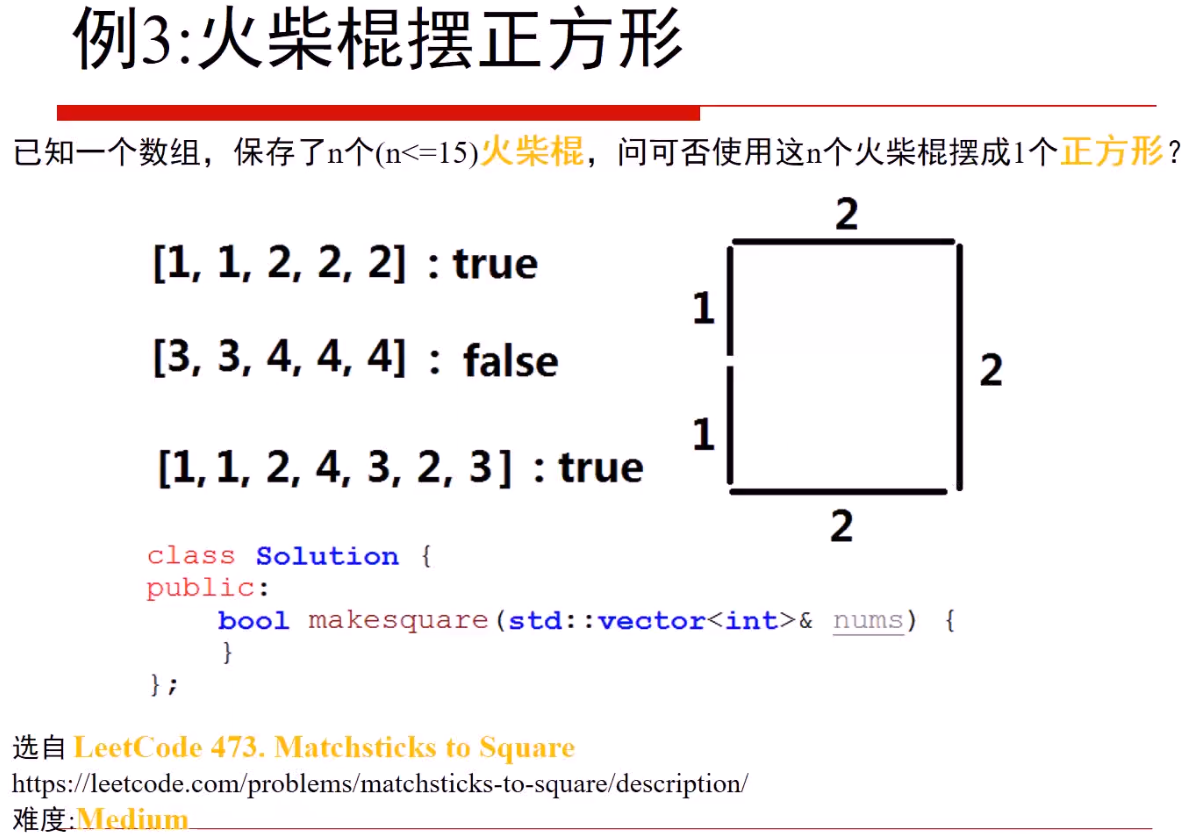
\* 解释: 不能用所有火柴拼成一个正方形。

\* 注意:

\*

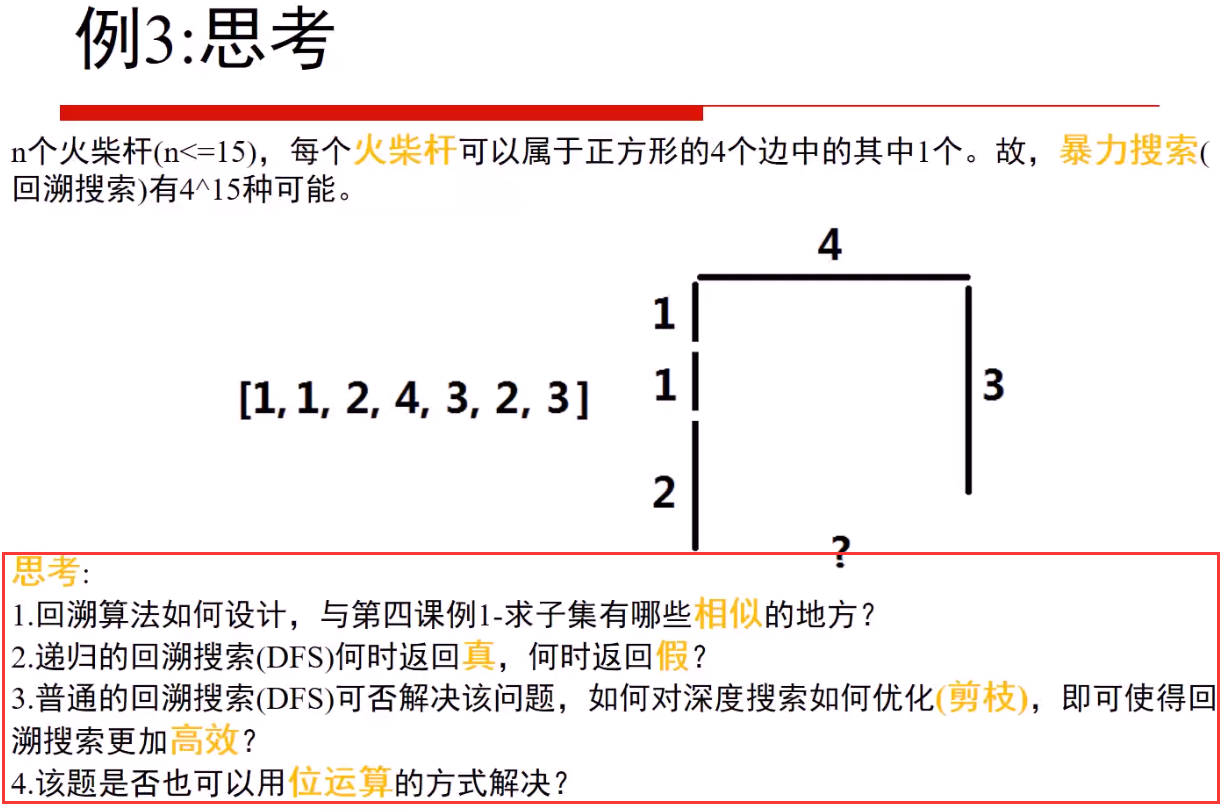
\* 给定的火柴长度和在 0 到 10^9之间。

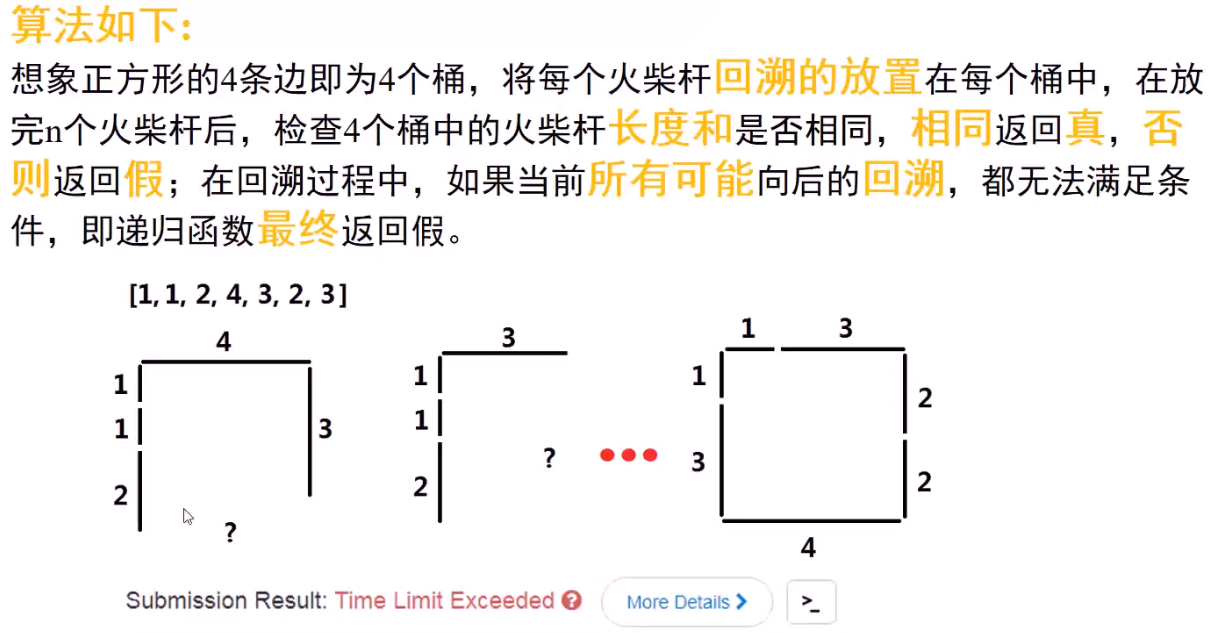
\* 火柴数组的长度不超过15。

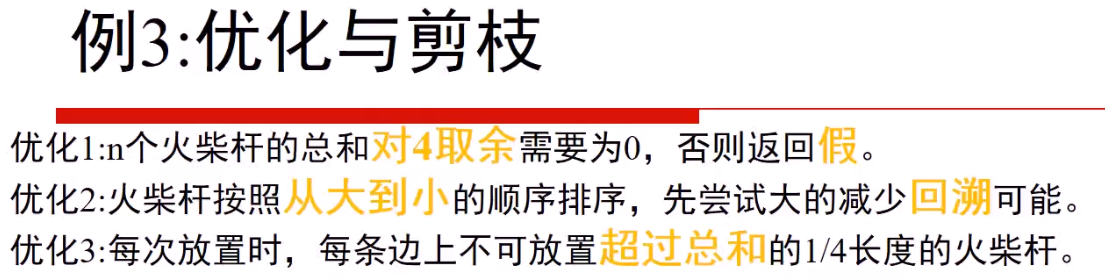


## 思路分析

### 方法1：深度优先搜索。DFS







\* 思路分析：采用深度优先搜索DFS。

\* 但是，直接深度搜索复杂度太高，15个火柴，则复杂度为4^15，因此必须进行优化搜索。

\* 深度优先搜索：每次捞起一根火柴，都有四种选择，然后各自的下一个也是四个选择，15根火柴，则4^15种情况。

\* 优化搜索：

\* 优化1：火柴杆的个数必须大于等于4个，且所有火柴杆的长度总和必须大于4且是4的整数倍，否则直接返回false；

\* 优化2：对火柴杆按照长度从大到小排序,每次都是先安排长的火柴杆；

\* 优化3: 每一边放置火柴杆时，该边的火柴总长度不能大于火柴杆总长度的1/4。

\* 注意：上述优化是针对深度搜索的优化，如果直接使用优化1/2/3的原则，不再进行深度遍历，是不行的。

\* 如[5,5,5,5,4,4,4,4,3,3,3,3]。过程中会出现10,10,4,4；此时10和10将无法再排火柴了，说明此路是走不通的；

\* 正确结果为：5+4+3；5+4+3；5+4+3；5+4+3；

### 方法2：位运算法

\* 方法2：位运算法。

\* 分成三步走：

\* 第一步:找出所有和等于一边值average的索引位为1对应的值，放到集合subOne中，

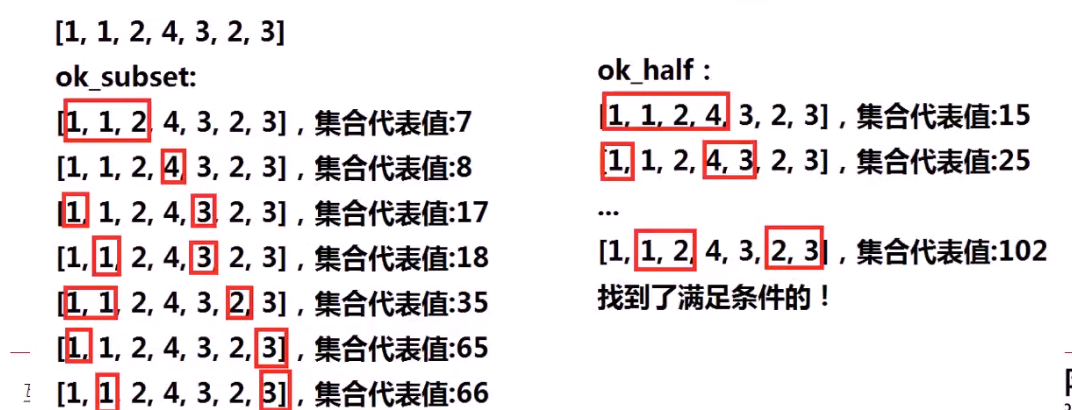
\* 如[1,1,2,2,2]；和等于2的有[00011,00100,01000,10000]=[3,4,8,16]；即为集合subOne；(数组下标j对应：1<<j)

\* 第二步：将subOne集合中的元素两两相与操作，若结果等于0，说明没有使用共同的火柴，将两者或的结果存到集合subTwo中，

\* 表示满足两边的情况；

\* 第三步：将subTwo集合中的元素两两相与操作，若结果等于0，说明存在四种不重叠使用火柴的情况，即返回true。





## Java代码

### 方法1：深度优先搜索。DFS

public boolean makesquare(int[] nums) {

if (nums.length < 4) return false;

int sum = 0;

for (int num : nums) sum += num;

if (sum % 4 != 0) return false;//优化1

int average = sum / 4;

Arrays.sort(nums);//排序<默认是升序>//优化2

int[] bucket = new int[4];

return DFS(nums, nums.length-1, average, bucket);

}

/\*\*

\* DFS：深度优先搜索方法

\* @param index : 需要安排火柴的下标

\* @param average ：每一边的火柴总长度

\*/

private boolean DFS(int[] nums, int index, int average, int[] bucket) {

if (index < 0) {//所有火柴都已经使用

return bucket[0] == average && bucket[1] == average

&& bucket[2] == average && bucket[3] == average;

}

for (int j = 0; j < 4; j++) {

**if (bucket[j] + nums[index] > average) continue;// 排除这些；优化3**

bucket[j] += nums[index];

if(DFS(nums, index - 1, average, bucket))//深度搜索

return true;

bucket[j] -= nums[index];//还原

}

return false;

}



### 方法2：位运算法。

/\*\*

\* 方法2:位运算方法

\*/

public boolean makesquare\_2(int[] nums) {

if (nums.length < 4) return false;

int sum = 0;

for (int num : nums) sum += num;

if (sum % 4 != 0) return false;

int average = sum / 4;

List<Integer> subOne = new ArrayList<Integer>();

List<Integer> subTwo = new ArrayList<Integer>();

//第一步：将所有满足拼接一边的位对应值存到subOne中

int len = nums.length;

int count = 1 << len;//如果是4个火柴，一共是2^4=16种情况

for (int i = 0; i < count; i++) {

int tempSum = 0;

boolean flag = true;//表示对i的所有元素是否加完

for (int j = 0; j < len; j++) {

if ((i & (1 << j)) != 0) {//表示第j个元素对应i是选中了

tempSum += nums[j];

}

//优化下，避免了1111111，其实没有必要全都加一遍，加到第三个，就已经超过average了

if (tempSum > average) {

flag = false;

break;

}

}

if (flag && tempSum == average) {

subOne.add(i);

}

}

//第二步：将subOne中元素两两相与，若为0，表示使用的火柴不冲突

for (int i = 0; i < subOne.size(); i++) {

for (int j = i + 1; j < subOne.size(); j++) {

if ((subOne.get(i) & subOne.get(j)) == 0) {

subTwo.add(subOne.get(i) | subOne.get(j));

}

}

}

//第三步：将subTwo中元素两两相与，若为0，表示拼成正方形使用的火柴都不冲突

for (int i = 0; i < subTwo.size(); i++) {

for (int j = i + 1; j < subTwo.size(); j++) {

if ((subTwo.get(i) & subTwo.get(j)) == 0) {

return true;

}

}

}

return false;

}

