youngyangyang04 Merge pull request #302 from z80160280/master ... History

5 contributors

196 lines (159 sloc) 5.37 KB

> 该用set的时候，还是得用set

第202题. 快乐数

https://leetcode-cn.com/problems/happy-number/

编写一个算法来判断一个数 n 是不是快乐数。

「快乐数」定义为：对于一个正整数，每一次将该数替换为它每个位置上的数字的平方和，然后重复这个过程直到这个数变为 1，也可能是 无限循环 但始终变不到 1。如果 可以变为 1，那么这个数就是快乐数。

如果 n 是快乐数就返回 True ；不是，则返回 False 。

示例：

输入：19
输出：true
解释：
1^2 + 9^2 = 82
8^2 + 2^2 = 68
6^2 + 8^2 = 100
1^2 + 0^2 + 0^2 = 1

思路

这道题目看上去貌似一道数学问题，其实并不是！

题目中说了会 **无限循环**，那么也就是说**求和的过程中，sum会重复出现，这对解题很重要！**

正如：关于哈希表，你该了解这些！中所说，**当我们遇到了要快速判断一个元素是否出现集合里的时候，就要考虑哈希法了。**

所以这道题目使用哈希法，来判断这个sum是否重复出现，如果重复了就是return false， 否则一直找到sum为1为止。

判断sum是否重复出现就可以使用unordered_set。

还有一个难点就是求和的过程，如果对取数值各个位上的单数操作不熟悉的话，做这道题也会比较艰难。

C++代码如下：

```
class Solution {
public:
    // 取数值各个位上的单数之和
    int getSum(int n) {
        int sum = 0;
        while (n) {
            sum += (n % 10) * (n % 10);
            n /= 10;
        }
        return sum;
    }
    bool isHappy(int n) {
        unordered_set<int> set;
        while(1) {
            int sum = getSum(n);
            if (sum == 1) {
                return true;
            }
            // 如果这个sum曾经出现过，说明已经陷入了无限循环了，立刻return false
            if (set.find(sum) != set.end()) {
                return false;
            } else {
                set.insert(sum);
            }
            n = sum;
        }
    }
};
```

其他语言版本

Java：

```
class Solution {
    public boolean isHappy(int n) {
```

```
        Set<Integer> record = new HashSet<>();
        while (n != 1 && !record.contains(n)) {
            record.add(n);
            n = getNextNumber(n);
        }
        return n == 1;
    }

    private int getNextNumber(int n) {
        int res = 0;
        while (n > 0) {
            int temp = n % 10;
            res += temp * temp;
            n = n / 10;
        }
        return res;
    }
}
```

Python：

```
class Solution:
    def isHappy(self, n: int) -> bool:
        set_ = set()
        while 1:
            sum_ = self.getSum(n)
            if sum_ == 1:
                return True
            #如果这个sum曾经出现过，说明已经陷入了无限循环了，立刻return false
            if sum_ in set_:
                return False
            else:
                set_.add(sum_)
            n = sum_

    #取数值各个位上的单数之和
    def getSum(self, n):
        sum_ = 0
        while n > 0:
            sum_ += (n%10) * (n%10)
            n //= 10
        return sum_
```

Go：

```
func isHappy(n int) bool {
    m := make(map[int]bool)
    for n != 1 && !m[n] {
        n, m[n] = getSum(n), true
    }
    return n == 1
}

func getSum(n int) int {
    sum := 0
```

```
    for n > 0 {
        sum += (n % 10) * (n % 10)
        n = n / 10
    }
    return sum
}
```

javaScript:

```
function getN(n) {
    if (n == 1 || n == 0) return n;
    let res = 0;
    while (n) {
        res += (n % 10) * (n % 10);
        n = parseInt(n / 10);
    }
    return res;
}

var isHappy = function(n) {
    const sumSet = new Set();
    while (n != 1 && !sumSet.has(n)) {
        sumSet.add(n);
        n = getN(n);
    }
    return n == 1;
};

// 使用环形链表的思想 说明出现闭环 退出循环
var isHappy = function(n) {
    if (getN(n) == 1) return true;
    let a = getN(n), b = getN(getN(n));
    // 如果 a === b
    while (b !== 1 && getN(b) !== 1 && a !== b) {
        a = getN(a);
        b = getN(getN(b));
    }
    return b === 1 || getN(b) === 1 ;
};
```

- 作者微信：程序员Carl
- B站视频：代码随想录
- 知识星球：代码随想录