习题课（枚举）

TEACHING COURSEWARE POWPOINT

授课时间：2025.07.26

目录

01

题目描述

TEACHING
COURSEWARE

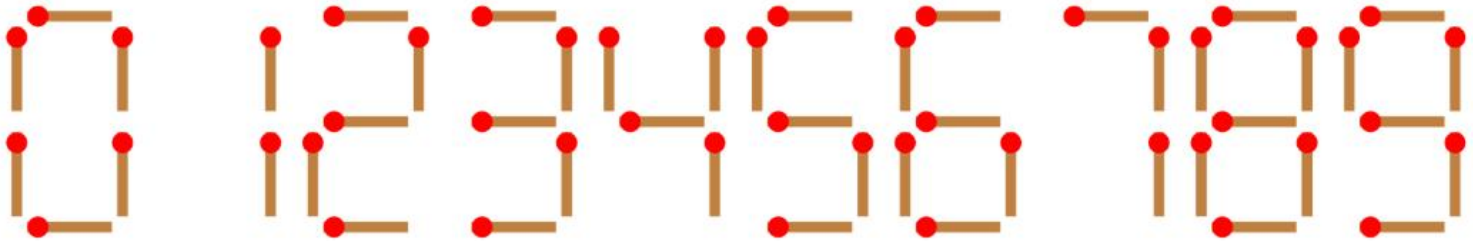
TEACH

题目描述

[NOIP 2008 提高组] 火柴棒等式

题目描述

给你 n 根火柴棍，你可以拼出多少个形如 $A+B=C$ 的等式？等式中的 A、B、C 是用火柴棍拼出的整数（若该数非零，则最高位不能是 0）。用火柴棍拼数字 $0\sim9$ 的拼法如图所示：



注意：

1. 加号与等号各自需要两根火柴棍；
2. 如果 $A\neq B$，则 $A+B=C$ 与 $B+A=C$ 视为不同的等式（$A,B,C\geq0$）；
3. n 根火柴棍必须全部用上。

输入格式

一个整数 $n(1\leq n\leq 24)$。

输出格式

一个整数，能拼成的不同等式的数目。

题目描述

[NOIP 2008 提高组] 火柴棒等式

样例1：
输入：
14
输出：
2

样例2：
输入：
18
输出：
9

说明/提示

【输入输出样例 1 解释】

2 个等式为 $0+1=1$ 和 $1+0=1$。

【输入输出样例 2 解释】

9 个等式为

$0+4=4$、$0+11=11$、$1+10=11$、$2+2=4$、$2+7=9$、$4+0=4$、$7+2=9$、$10+1=11$、$11+0=11$。

02

知识回归

TEACHING
COURSEWARE

TEACH

知识回归

“在计算机科学中，枚举算法（也叫‘穷举法’）是指：明确问题的所有可能解的范围（即‘解空间’），逐一列举所有候选解，对每个候选解验证是否满足问题的条件，最终得到所有有效解的算法。

简单说，就是让计算机‘按顺序把所有可能的情况过一遍，符合要求的就记下来。”

核心思想：

“枚举算法的核心可以浓缩成两个动作：

第一个是‘遍历’：系统地、不重不漏地列出解空间里的所有候选解；

第二个是‘验证’：对每个候选解，用问题的条件检查它是否有效。

03

题目解答

TEACHING
COURSEWARE

TEACH

题目解答

先将拼出数字零到九需要用到的火柴根数存进s 数组里。

再写一个函数，返回值是所用火柴根数的和。

假设进入函数的这个数小于十，那么直接返回，否则将这个的每一位都拆开，判断每一位要用的根数，最后相加。

04

标准程序

TEACHING
COURSEWARE

TEACH

标准程序


```cpp
#include<bits/stdc++.h>
using namespace std;

// s[i]表示数字i需要的火柴棍数量（0~9）
// 对应题目中给出的数字拼法：0需6根，1需2根，2需5根，以此类推
int s[10] = {6, 2, 5, 5, 4, 5, 6, 3, 7, 6};
int n;          // 输入的火柴棍总数
int ans = 0;    // 符合条件的等式数量
```

标准程序


```
// 计算数字x需要的火柴棍总数
int fun(int x){
    int sum = 0;
    if(x < 10){  // 若x是个位数，直接取对应的值
        sum = s[x];
    } else {     // 若x是多位数，逐位计算并累加
        int t = x;  // 临时变量存储x，用于拆解每一位
        while(t != 0){
            sum += s[t % 10];  // 取最后一位数字，累加其火柴棍数
            t /= 10;           // 移除最后一位
        }
    }
    return sum;
}
```

标准程序

```
int main(){
    cin >> n;  // 输入火柴棍总数

    // 枚举所有可能的A（i）和B（j）
    // 范围设为0~1000：因为n最大24，去掉加号和等号的4根，剩余20根
    // A、B、C的火柴棍总和不超过20，因此数字不会太大，1000足够覆盖所有可能
    for(int i = 0; i <= 1000; i++){
        for(int j = 0; j <= 1000; j++){
            int c = i + j;  // 计算C = A + B
            // 计算A、B、C的火柴棍总数，需等于n-4（因为加号和等号共需4根）
            if(fun(i) + fun(j) + fun(c) == n - 4){
                ans++;  // 符合条件，计数加1
            }
        }
    }

    cout << ans << "\n";  // 输出符合条件的等式数量
    return 0;
}
```