

## 算法设计与分析实验报告

实验名称：子集和数问题(回溯算法)

### 一、问题陈述，相关背景、应用及研究现状的综述分析

#### 1.问题陈述：

给定  $N$  个数，和一个整数  $M$ ，判定是否可以从  $N$  个数中取出若干个数，使它们的和等于  $M$ 。输出：YES 或者 NO。把  $N$  个数看成一个集合，问题就是从这个集合中选出一个子集，使这个子集满足和是  $M$ ；

## 二、模型拟制、算法设计和正确性证明

子集和数问题和 0-1 背包问题有些类似，解空间树也是一个子集树，将所有数字按升序排列，则从某一节点向左子树搜索表示选择对应的数字，向右子树搜索表示不选择对应数字，从最小的数字开始，向叶节点进行搜索，每有一个满足条件的解就输出出来。

在选择某一个数字是否加入子集时，可以根据数字的正数和负数来进行分类讨论限界函数。利用一个类的静态成员来存储数据，如 `current_sum`（现在的数字之和），`left_positive_sum`（未选择的正数之和），`left_negative_sum`（未选择的负数之和），这样做可以避免重复递归申请新的变量空间。

### 三、时间和空间复杂性分析

#### 子集和数问题也是 NP 完全问题

时间复杂度:  $O(2^n)$

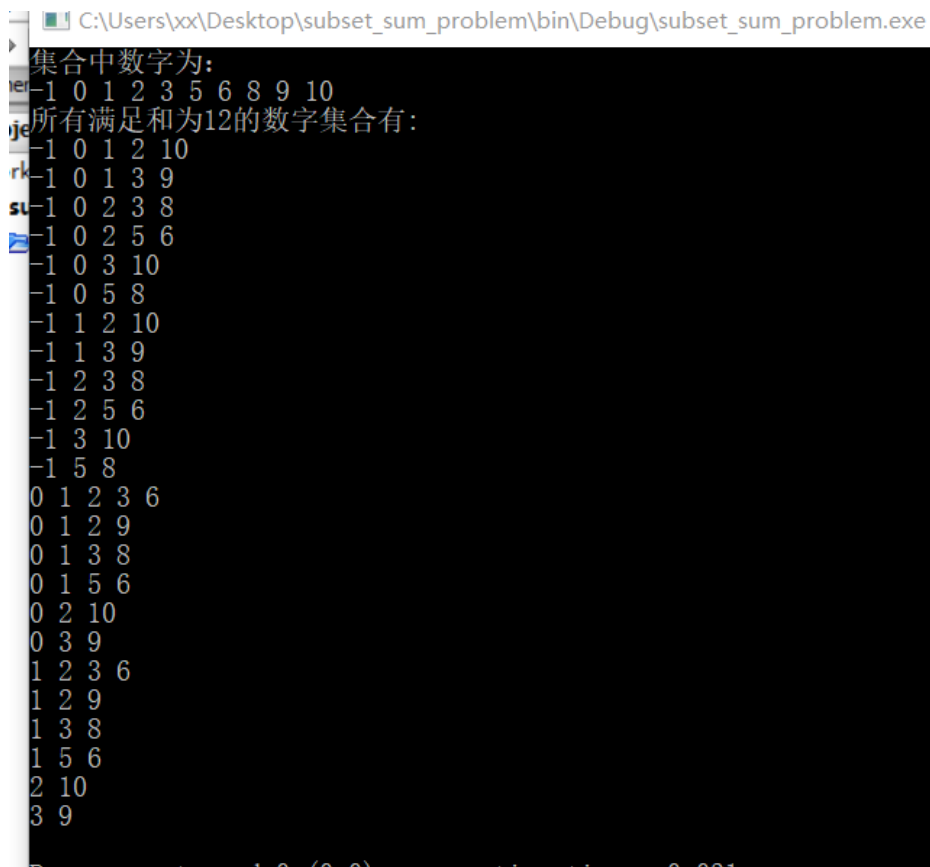
空间复杂度:

需要几个变量来存储中间数据, 一个 `bool` 数组来构造解, 空间复杂度为  $O(n)$  ;

## 四、程序实现和实验测试过程

源程序见 Subset\_sum\_problem.cpp

测试过程：



```
C:\Users\xx\Desktop\subset_sum_problem\bin\Debug\subset_sum_problem.exe
集合中数字为:
-1 0 1 2 3 5 6 8 9 10
所有满足和为12的数字集合有:
-1 0 1 2 10
-1 0 1 3 9
-1 0 2 3 8
-1 0 2 5 6
-1 0 3 10
-1 0 5 8
-1 1 2 10
-1 1 3 9
-1 2 3 8
-1 2 5 6
-1 3 10
-1 5 8
0 1 2 3 6
0 1 2 9
0 1 3 8
0 1 5 6
0 2 10
0 3 9
1 2 3 6
1 2 9
1 3 8
1 5 6
2 10
3 9
```

## 五、总结

子集和数问题一般情况下是 NP 难的，不论是动态规划还是回溯法，都较难找到多项式级别时间复杂度的算法。

利用类的静态变量可以节省一定的递归申请的变量空间，但是要小心注意递归回退时要恢复静态变量，以便再次递归。