深度优先搜索算法

171491106 安南 硬件一班

1. 实验题目

深度优先搜索算法（DFS）

1. 算法简介
2. 深度优先搜索属于图算法的一种，英文缩写为DFS（Depth First Search.）其过程简要来说是对每一个可能的分支路径深入到不能再深入为止，而且每个[节点](https://baike.baidu.com/item/%E8%8A%82%E7%82%B9)只能访问一次。
3. 实验目的
4. 成功编写一个深度优先算法，将1-4，几个数字进行全排列，列出所有的可能和个数。
5. 1-9任意一个数字实现 ABC + DEF = GHI; 列出实现的式子和个数
6. 实验代码

代码一：

**package** 深度优先;

**public** **class** 算法 {

**static** **int**[] *arr* = **new** **int**[5];//存储目标数组

**static** **int**[] *result* = **new** **int**[5];

**static** **int** *step\_total* = 5;

**static** **int** *count* = 0;

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

*full*(1);

System.***out***.println("total count:" + *count*);;

}

**public** **static** **void** full(**int** step) {

// **TODO** Auto-generated method stub

**if**(step >= *step\_total*) {

System.***out***.println(*arr*[1] + "" + *arr*[2]+ "" + *arr*[3] + "" + *arr*[4]);

*count*++;

**return**;

}

//重复内容

**for**(**int** i = 1;i < *step\_total*;i++) {

**if**(*result*[i] == 0) {//如果当前数没有被排列

*result*[i] = 1;//标记当前数已经被排列

*arr*[step] = i;

*full*(step + 1);

*result*[i] = 0;

}

}

}

}

代码二：

**package** 深度优先;

**public** **class** 算法\_1 {

**static** **int** [] *arr* = **new** **int**[10];

**static** **int** [] *result* = **new** **int**[10];

**static** **int** *step\_total* = 10;

**static** **int** *count* = 0;

**public** **static** **void** dfs1 (**int** step) {

**if** (step >= *step\_total*) {

**int** ABC = Integer.*parseInt*("" + *arr*[1] + *arr*[2] + *arr*[3]);

**int** DEF = Integer.*parseInt*("" + *arr*[4] + *arr*[5] + *arr*[6]);

**int** GHI = Integer.*parseInt*("" + *arr*[7] + *arr*[8] + *arr*[9]);

**if** (ABC + DEF == GHI) {

System.***out***.println(*arr*[1] + "" + *arr*[2] + "" + *arr*[3] + "+"

+ *arr*[4] + "" + *arr*[5] + "" + *arr*[6] + "=" + *arr*[7] + "" + *arr*[8] + "" +*arr*[9]);

*count* ++;

**return**;

}

}

//重复且不相等的生成arr[]

**for** (**int** i = 1; i < *step\_total*; i++) {

**if** (*result*[i] == 0) {

*result*[i] = 1;

*arr*[step] = i;

*dfs1*(step + 1);

*result*[i] = 0;

}

}

}

**public** **static** **void** main(String[] args) {

*dfs1*(1);

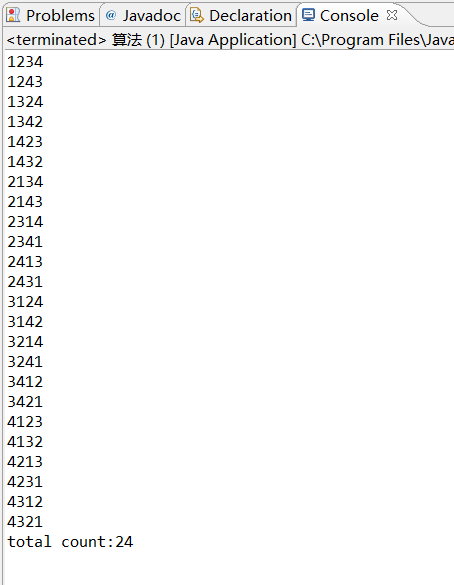
System.***out***.println("Total count : " + *count*);

}

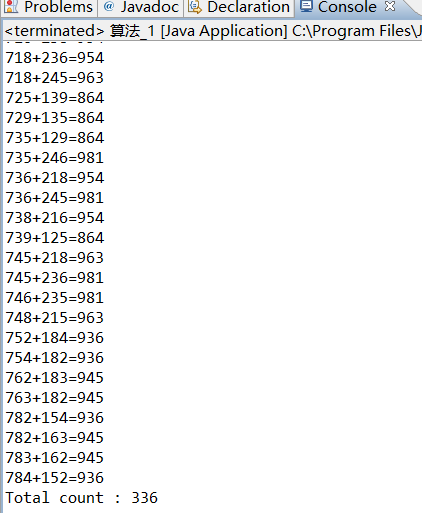
}

1. 实验结果

代码一：



代码二：



1. 实验总结

DFS适合给定初始状态跟目标状态，要求判断从初始状态到目标状态是否有解的问题。大致思路是从第一层进入后依次进入后面的几层，而不重复进入第一层，直到把一条路径上的走到尽头，然后回到上一个节点，走另外一条路到尽头，知道遍历完所有的节点。