**回溯法求全排列问题程序报告**

姓名：吴静 学号：2113285 班级：信息安全二班

**一、源码部分**

#include<iostream>

using namespace std;

int n;

int num = 0;

int arr[40] = { 0 };

void backtrace(int depth) {

if (depth == n) {

num++;

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

cout << arr[i];

}

cout << endl;

return;

}

for (int i = depth; i < n; i++) {//总共有n个数字，需要进行n次循环（有n个数字可以在第一位）

swap(arr[depth], arr[i]);

backtrace(depth + 1);

swap(arr[depth], arr[i]);

}

}

int main() {

cout << "请输入n的值：" << endl;

cin >> n;

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

arr[i] = i + 1;

}

cout << "用回溯法得出1~"<<n<<"的全排列有：" << endl;

backtrace(0);

cout << "总共个数为：" << endl;

cout << num;

}

**二、说明部分**

本题主要考察对回溯法和全排列问题的理解运用。

对于回溯法，基本框架如下：

void BackTrace(int t){//第t层

if(t>n)//n为树的高度，此时t为叶子结点

Output(t);//输出结果

else{

for(int i=f(n,t);i<=g(n,t);i++){//f(n,t)为开始结点,g(n,t)为结束结点，遍历

x[i]=h(i);//记录可选值

if(Constraint(t)&&Bound(t))//限界函数

BackTrace(t+1)//满足限界函数，往下一层移动

}

}

}

而对于n个元素的全排列，这是一个n层的解空间树，这个题需要做的就是确定每一层取哪个值。

递归：对于第depth层，它可以确定depth层后面的所有层，即在确定该层元素后，要调用一次backtrace(depth+1)确定后面的元素

输出：当查询的层数depth与我们输入的元素数目n相等时，该次查询结束，输出结果：顺序输出arr中的元素。

回溯：在寻找到一个完整的解空间后，需要向上层回溯，继续寻找下一个可能的值（这里可能的值是还没有在前面的层出现过的），这里运用一个思路，就是对于该层元素的选择，可以看成未排序数组部分第一个元素的确定，即第一个元素和哪一个元素进行交换，即为若是将第j个元素作为该层元素，即将第j个元素和arr[depth]进行交换，可以使用函数swap。

例如：对于第depth层（i==depth）

a[i]取a[i]---------------------🡪swap(a[i],a[i])

a[i]取a[i+1]-------------------🡪swap(a[i],a[i+1])

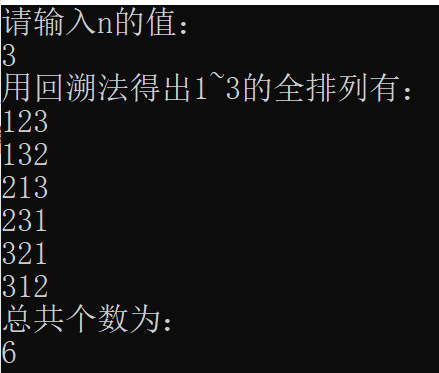
……………

a[i]取a[n-1]-------------------🡪swap(a[i],a[n-1])

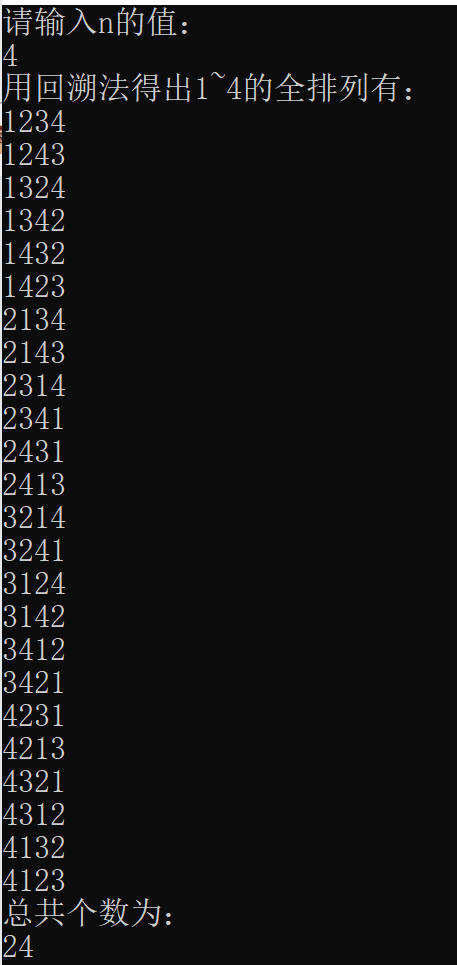
由上可知，此时可以用一个for循环，令i从depth开始，每次加一，直至i==n时结束。

而在递归调用确定之后的函数之后，要把此时交换后的两个数换回来，方便之后的回溯调用，这里对应“撤销选择，回到回溯算法之前的状态”一步。

**三、运行示例**



以n==3为例，此时得到全排列为123，132，213，231，321，312总共6个，结果正确。



以n==4为例，得到全排列为1234，1243等24个，结果正确。