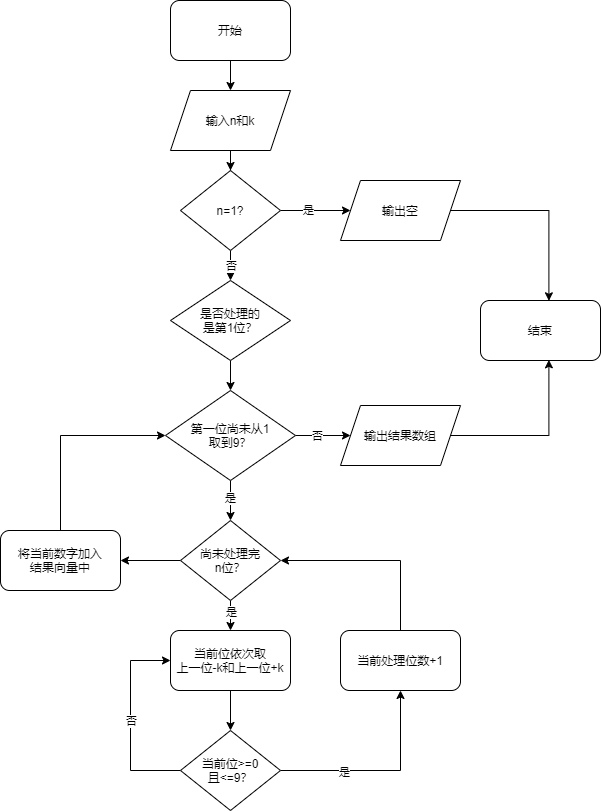
作业1-编程

# 问题重述

输入正整数n和非负数k，本问题需要我们寻找到所有的n位数，其中每相邻的两位数之绝对差为k，并将结果按递增顺序打印出。

# 算法设计

为了解决本问题，采用的算法主要是深度优先搜索（Deep-First Search，即DFS）。其流程图如下所示：



# 算法代码

函数getDifferenceVector()有两个参数n和k，在其内部调用递归函数getDifferenceDfs()来完成计算满足问题的全部数字。

代码如下所示：

**vector**<int> **getDifferenceVector**(int n, int k) {

**vector**<int> res;

**getDifferenceDfs**(res, 0, 0, n, k);

    return res;

}

而函数getDifferenceDfs()是递归函数，其result参数用于存储满足问题的数字，index用于记忆当前递归层（当index等于n，表示当前处理数字位数已满足题目所需），number表示当前正在处理的数字（该数字每进入一个递归层会乘以10并加上新的一位数），n和k则是题目所给输入。

需要注意到：为了使得结果按照有序输出，在每一个递归层中都是先考虑上一位减去k再考虑上一位加上k的情况，这样即可使在第n层递归时加入的数字有一个先后次序。

其代码如下所示：

void **getDifferenceDfs**(**vector**<int>& result,

    int index, int number, int n, int k) {

    if (n == 1) {

        return;

    }

    if (index == n) {

        result.**push\_back**(number);

        return;

    }

    if (index == 0) {

        for (int i = 1; i <= 9; ++i) {

**getDifferenceDfs**(result, 1, i, n, k);

        }

    }

    else {

        int lastNumber = number % 10;

        if (lastNumber - k >= 0) {

**getDifferenceDfs**(result, index + 1, lastNumber - k + number \* 10, n, k);

        }

        if (k != 0 && lastNumber + k <= 9) {

**getDifferenceDfs**(result, index + 1, lastNumber + k + number \* 10, n, k);

        }

    }

}

# 算法分析

由于本算法中使用了深度优先搜索的思路，因此在每一个递归层都有两种可能的选择（即当前位等于上一位加k或者等于上一位减k）。

数字一共n位，每一位可供选择数为2种可能性，因此算法的时间复杂度为。

同时，递归函数在每一层递归的过程中会需要递归工作栈，因此空间复杂度也相应的为。

# 运行截图

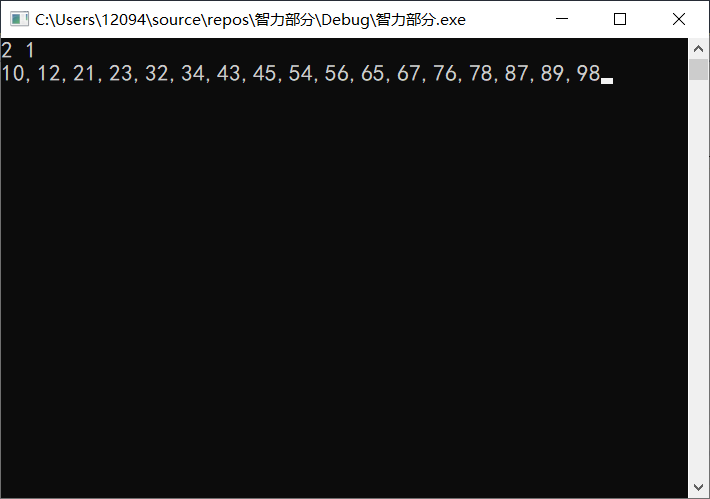
## 函数测试

**测试用例1**

输入：n = 2, k = 1

预期输出：[10,12,21,23,32,34,43,45,54,56,65,67,76,78,87,89,98]

实际输出：

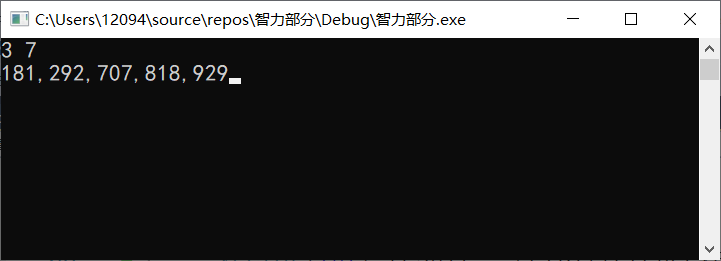


**测试用例2**

输入： n = 3, k = 7

预期输出： [181,292,707,818,929]

实际输出：



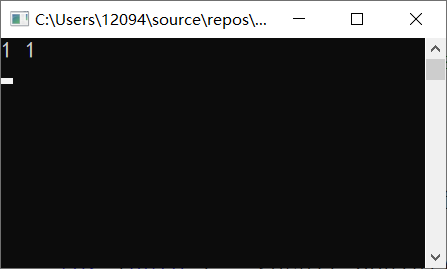
## 边界测试

**测试用例1**

输入： n = 1, k = any value

预期输出： []

实际输出：

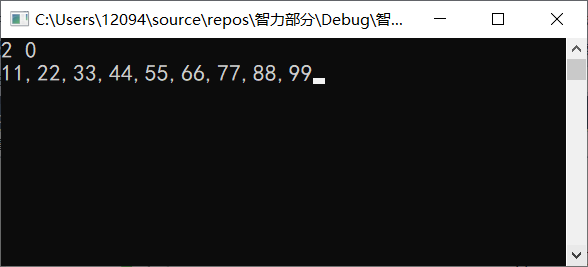


**测试用例2**

输入： n = 2, k = 0

预期输出：[11,22,33,44,55,66,77,88,99]

实际输出：

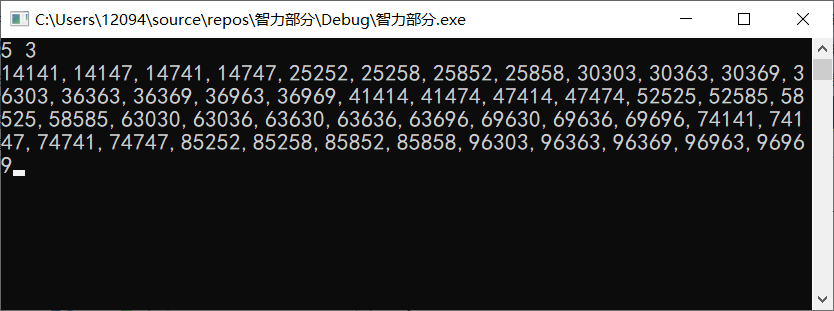


## 性能测试

**测试用例1**

输入： n = 5 , k = 3

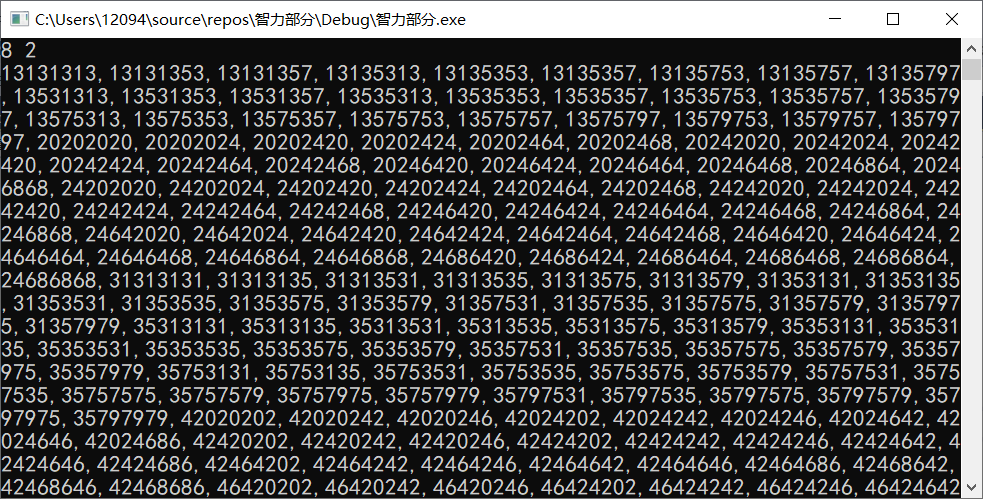
实际输出：



**测试用例2**

输入： n = 8 , k = 2

实际输出：



（详见1851055\_perf\_case\_2.txt）