# 题目

如果一个整数上的每一位数字与其相邻位上的数字的绝对差都是 1，那么这个数就是一个「步进数」。

例如，321 是一个步进数，而 421 不是。

给你两个整数，low 和 high，请你找出在 [low, high] 范围内的所有步进数，并返回 排序后 的结果。

示例：

输入：low = 0, high = 21

输出：[0,1,2,3,4,5,6,7,8,9,10,12,21]

提示：

0 <= low <= high <= 2 \* 10^9

# 分析

## 方法一：递归法/深度优先

解题思路

首先看题目，数组中每一位置都变动且和之前的某一位有关系（排列组合，括号生成），首先就想到的是回溯搜索系列，采用dfs来做

1、首先需要找到树向下生长的规律：

下一层 = 上一层的数字\*10 + 上一层的数字末尾加/减1

2、然后找到树生长的终止条件（剪枝）：

数字大于high时就停止生长

3、接下来考虑特殊情况：

当末尾是0时，不能进行减1操作

当末尾是9时，不能进行加1操作

4、最后，考虑算法本身的特殊处理（错误输入，溢出之类的）：

乘积可能会溢出，所以采用long

代码：

class Solution {

public:

vector<int> countSteppingNumbers(int low, int high) {

vector<int> res;

for (int i=1;i<10;i++)

{

dfs(res,i,i,high);

}

vector<int> result;

if (low<=0)

result.push\_back(0);

for (auto i:res)

{

if (i>=low && i<= high)

result.push\_back(i);

}

sort(result.begin(),result.end());

return result;

}

void dfs(vector<int>& v,long wholeNum,int lastNum,int high)

{

if (wholeNum > high)

return;

v.push\_back(wholeNum);

if (lastNum!=0)

dfs(v,wholeNum\*10+lastNum-1,lastNum-1,high);

if (lastNum!=9)

dfs(v,wholeNum\*10+lastNum+1,lastNum+1,high);

}

};

## 方法二：回溯/广度优先

思路和心得：

1、BFS

就像，一个点后面，最多有两个邻接点

BFS一直往外扩，先小后大

到了边界了，就停下

2.0要提前特判一下

因为当位数大于1位时，0不能作为首位

首位是1~9,省掉了排序

代码：

class Solution

{

public:

vector<int> countSteppingNumbers(int low, int high)

{

vector<int> res;

if (low == 0)

res.push\_back(0);

queue<long long> Q;

for (int x = 1; x < 10; x ++)

Q.push(x);

while (Q.size())

{

long long cur = Q.front(); Q.pop();

if (cur > high)

return res;

if (low <= cur)

res.push\_back(cur);

int last\_bit = cur % 10;

if (last\_bit > 0)

Q.push(cur \* 10 + (last\_bit - 1));

if (last\_bit < 9)

Q.push(cur \* 10 + (last\_bit + 1));

}

return res;

}

};