题目难度：MEDIUM

给定一个数字字符串 S，比如 S = "123456579"，我们可以将它分成斐波那契式的序列 [123, 456, 579]。

形式上，斐波那契式序列是一个非负整数列表 F，且满足：

0 <= F[i] <= 2^31 - 1，

（也就是说，每个整数都符合 32 位有符号整数类型）；

F.length >= 3；

对于所有的0 <= i < F.length - 2，都有 F[i] + F[i+1] = F[i+2] 成立。

另外，请注意，将字符串拆分成小块时，每个块的数字一定不要以零开头，除非这个块是数字 0 本身。

返回从 S 拆分出来的任意一组斐波那契式的序列块，如果不能拆分则返回 []。

示例 1：

输入："123456579"

输出：[123,456,579]

示例 2：

输入: "11235813"

输出: [1,1,2,3,5,8,13]

示例 3：

输入: "112358130"

输出: []

解释: 这项任务无法完成。

示例 4：

输入："0123"

输出：[]

解释：每个块的数字不能以零开头，因此 "01"，"2"，"3" 不是有效答案。

示例 5：

输入: "1101111"

输出: [110, 1, 111]

解释: 输出 [11,0,11,11] 也同样被接受。

提示：

1 <= S.length <= 200

字符串 S 中只含有数字。

思路：Backtracking+剪枝

接下来上题解区题解：

无

时间击败65%（时间一般）

空间击败27%（空间不咋地）

上代码：

class Solution {

public:

vector<int> res;//全局

vector<int> splitIntoFibonacci(string S) {

if (S.empty()) return {};

vector<int> vec;

dfs(S, 0, vec, 0);

return res;

}

void dfs(string s, int str\_index, vector<int>& vec, int vec\_index) {

int size = s.size();

if (!res.empty()) return;

if (str\_index == size && vec.size() >= 3) {//递归返回条件

res = vec;

return;

}

for (int i = 1; i <= size - str\_index && i <= 10; ++i) {//截取长度

string sub = s.substr(str\_index, i);

long cur = \_atoi(sub);

if (cur >= INT\_MAX) return; // 剪枝1：拆分出的数必须符合 32 位有符号整数类型，即每个数必须在[0, 2^31−1] 的范围内

if (vec\_index < 2 && cur != -1) {//直接下一层

if (sub.size() > size - str\_index - i) return;//剪枝

vec.push\_back(cur);

dfs(s, str\_index + i, vec, vec\_index + 1);

vec.pop\_back();

}

else if (cur != -1) {

long last2 = vec[vec\_index - 2];//防止大数

long last1 = vec[vec\_index - 1];//防止大数

long sum = last1 + last2; //防止大数

if (sum > INT\_MAX) return;

if (cur == last1 + last2) {

vec.push\_back(cur);

dfs(s, str\_index + i, vec, vec\_index + 1);

vec.pop\_back();

}

else if (cur > last1 + last2) return;//剪枝2： 如果列表中至少有 22 个数，并且拆分出的数已经大于最后 22 个数的和，就不需要继续尝试拆分了。

else continue;

}

else return;// 剪枝3 ：0开头直接返回

}

}

int \_atoi(string str) {

long sum = 0; //防止大数

int size = str.size();

if (size > 1 && str[0] == '0') return -1;//0开头不符合条件

for (int i = 0; i < str.size(); ++i) {

sum \*= 10;

sum += (str[i] - '0');

}

return sum;

}

};