241. Different Ways to Add Parentheses

給出一個字符串，找出所有加括號情況下的得數，只考慮+-\*

Input: "2-1-1".

((2-1)-1) = 0

(2-(1-1)) = 2

Output: [0, 2]

public class Solution {

public List<Integer> diffWaysToCompute(String input) {

List<Integer> res = new ArrayList<Integer>();

for (int i = 0; i < input.length(); i++) {

char c = input.charAt(i);

if (c == '-' || c == '+' || c == '\*') {

String a = input.substring(0, i);

String b = input.substring(i + 1);

List<Integer> al = diffWaysToCompute(a);

List<Integer> bl = diffWaysToCompute(b);

for (int x : al) {

for (int y : bl) {

if (c == '-') {

res.add(x - y);

} else if (c == '+') {

res.add(x + y);

} else if (c == '\*') {

res.add(x \* y);

}

}

}

}

}

if (res.size() == 0) res.add(Integer.valueOf(input));

return res;

}

}

[[LeetCode] Permutations II 全排列之二](http://www.cnblogs.com/grandyang/p/4359825.html)

思路： 排除重複方法

例如3334，在選取這一層的數時，只有當前面相同數字在上一層中選過方可選。

例如第1個3 在上一層選了，這一層可以選第二個3，但是第三和第四個選不了

class Solution {

public:

vector<vector<int> > permuteUnique(vector<int> &num) {

vector<vector<int> > res;

vector<int> out;

vector<int> visited(num.size(), 0);

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

permuteUniqueDFS(num, 0, visited, out, res);

return res;

}

void permuteUniqueDFS(vector<int> &num, int level, vector<int> &visited, vector<int> &out, vector<vector<int> > &res) {

if (level >= num.size()) res.push\_back(out);

else {

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

if (visited[i] == 0) {

if (i > 0 && num[i] == num[i - 1] && visited[i - 1] == 0) continue;

visited[i] = 1;

out.push\_back(num[i]);

permuteUniqueDFS(num, level + 1, visited, out, res);

out.pop\_back();

visited[i] = 0;

}

}

}

}

};

282. Expression Add Operators

Given a string that contains only digits 0-9 and a target value, return all possibilities to add **binary** operators (not unary) +, -, or \*between the digits so they evaluate to the target value.

Examples:

"123", 6 -> ["1+2+3", "1\*2\*3"]

"232", 8 -> ["2\*3+2", "2+3\*2"]

"105", 5 -> ["1\*0+5","10-5"]

"00", 0 -> ["0+0", "0-0", "0\*0"]

"3456237490", 9191 -> []

解釋：並非每一個數字都是一位數，可以是多位數。

思路：multed記錄最右邊非乘法運算，之後連乘的結果，例如1+2\*3\*4 到了第五位5，如果要乘5，必須把eval+multed+multed\*5.

如果是加法和減法，不影響運算順序，加法的話multed設為 +current, 減法的話設為-current.

由於並不是每個數字都是一位數，所以每一層都要循環

加入這層循環第一個index對應是0, 到了第二次循環就break，因為01 ，02這類並不是合法數字

public class Solution {

public List<String> addOperators(String num, int target) {

List<String> rst = new ArrayList<String>();

if(num == null || num.length() == 0) return rst;

helper(rst, "", num, target, 0, 0, 0);

return rst;

}

public void helper(List<String> rst, String path, String num, int target, int pos, long eval, long multed){

if(pos == num.length()){

if(target == eval)

rst.add(path);

return;

}

for(int i = pos; i < num.length(); i++){

if(i != pos && num.charAt(pos) == '0') break;

long cur = Long.parseLong(num.substring(pos, i + 1));

if(pos == 0){

helper(rst, path + cur, num, target, i + 1, cur, cur);

}

else{

helper(rst, path + "+" + cur, num, target, i + 1, eval + cur , cur);

helper(rst, path + "-" + cur, num, target, i + 1, eval -cur, -cur);

helper(rst, path + "\*" + cur, num, target, i + 1, eval - multed + multed \* cur, multed \* cur );

}

}

}

}