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;

}

}

}

}

};