Question: https://leetcode.com/problems/remove-invalid-parentheses/

We have 2 choices to include or exclude a closing or opening parenthesis based on how many invalid opening or closing brackets we have till that position, and add the valid ones to the set.

So we use recursion and backtrack to all possible solution and including them in set gives us unique solutions.

Code:

class Solution {

int invalidopen=0, invalidclose=0;

public List<String> removeInvalidParentheses(String s) {

validity(s, true);

Set<String> res = new HashSet<>();

helper(s, 0, invalidopen, invalidclose, res);

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

li.addAll(res);

return li;

}

public void helper(String s, int start, int invalidopen, int invalidclose, Set<String> res){

if(invalidopen == 0 && invalidclose == 0 && validity(s, false)) res.add(s);

int l = s.length();

for(int i=start; i<l; i++){

char c=s.charAt(i);

if(c=='(' && invalidopen>0){

helper((s.substring(0,i)+s.substring(i+1,l)), i, invalidopen-1, invalidclose, res);

}

else if(c==')' && invalidclose>0){

helper((s.substring(0,i)+s.substring(i+1,l)), i, invalidopen, invalidclose-1, res);

}

}

}

public boolean validity(String s, boolean update){

char carr[] = s.toCharArray();

int open=0, close=0, iopen=0, iclose=0;

for(char c : carr){

if(c=='(') open++;

else if(c==')') close++;

if(open<close){

iclose++;

close--;

}

}

if(open>close) iopen=open-close;

if(iopen==0 && iclose==0) return true;

else if(update){

this.invalidopen=iopen;

this.invalidclose=iclose;

}

return false;

}

}

Github Link :<https://lnkd.in/ecwtJeaz>