When people send messages on their phones, they sometimes modify word spelling by adding extra letters for emphasis. For example, if you want to emphasize hello you might write it “hellloooooooo”. Let's call the “ls” and the “os” word extensions. Regular text contains 2 or fewer of the same character in a row, while word extensions have 3 or more of the same character in a row. Given an input string representing one word, write a method that returns the start and end indices of all extensions in the word.

“Hellllooo” -> [[2,4], [5,7]]

“Hhhellloloo” -> [[0,2], [4,6]]

```
List<List<Integer>> res = new LinkedList<>();
Public List<List<Integer>> count(String input)  {

        char preChar = ‘ ’;
        int curStart = 0;
        int curCount = 0;

        for(int i = 0; i< input.size(); i++){
                char cur = input.charAt(i);

                if(i == input.size()-1 && cur == preChar) count++; // last position

                if(cur != preChar || i == input.size()-1 ){
                        if(curCount > 2){//add to result
                                LinkedList<Integer> startEnd = new LinkedList<>();
                                startEnd.add(curStart);
                                startEnd.add(i-1);
                                res.add(startEnd);
                        }

                        //reset progress
                        curCount = 1;
                        curStart = i;
                        preChar = cur;
                } else {
                        curCount++;
                }
        }

        return res;
}
```

Now we want to spell-check extended words. You are given a dictionary of words. Implement method isExtendedDictionaryWord that will return:

- true if it is a dictionary word.
- true if you collapse the extensions in the word and it is a dictionary word.
- false otherwise.

[‘hello’, ‘hi’]
‘Hellllloooo’ -> ‘hello’

Worst: lllllloooohhhhh (5l, 2 * 2 * 2 = 8)
llloohhh = (2*2*2=8)
n size of String, (2 to power N/3)

```
String input;
HashSet<String> dict;
boolean res = false;
List<List<Integer>> extensions;
public boolean isValidWord(HashSet<String> dict, String input){
	this.input = input;
	this.dict = dict;
	this.extensions =  count(String input);
	if(extensions.size()==0) return dict.contains(input);

	backTrack(0, “”);
	return res;
}

public void backTrack(int curIdx, String curPath){
	if(res) return;
	if(curIdx >= extensions.size() && dict.contains(curPath)){
		res = true;
		return;
	}

	for(int i=curIdx; i < extensions.size(); i++){
		LinkedList<Integer> ext = extensions.get(i);
		char cur = input.getCharAt(i.get(0));
		backTrack(i+1, curPath + cur);
		backTrack(i+1, curPath + cur + cur);
	}
}
```