Java-数组排序字符串-单词接龙

题目描述：单词接龙的规则是：可用于接龙的单词首字母必须要前一个单词的尾字母相同；当存在多个首字母相同的单词时，取长度最长的单词，如果长度也相等，则取字 典序最小的单词；已经参与接龙的单词不能重复使用。 现给定一组全部由小写字母组成单词数组，并指定其中的一个单词作为起始单词，进行单词接龙，请输出最长的单词串，单词串是单词拼接而成，中间没有空 格。

输入描述：输入的第一行为一个非负整数，表示起始单词在数组中的索引 K，0 <= K < N ； 输入的第二行为一个非负整数，表示单词的个数 N； 接下来的 N 行，分别表示单词数组中的单词。

输出描述：输出一个字符串，表示最终拼接的单词串。

补充说明：单词个数 N 的取值范围为[1, 20]； 单个单词的长度的取值范围为[1, 30]；

```java
import java.util.*;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int start = scanner.nextInt();
        scanner.nextLine();
        int N = scanner.nextInt();
        scanner.nextLine();
        HashMap<Character, LinkedList<String>> map = new HashMap<>();
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < N; i++) {
            String s = scanner.nextLine();
            if (i==start){
                sb.append(s);
            }else {
                char c = s.charAt(0);
                LinkedList<String> strings = map.get(c);
                if (strings!=null){
                    strings.add(s);
                }else {
                    LinkedList<String> list = new LinkedList<>();
                    list.add(s);
                    map.put(c,list);
                }
//                List<String> list = map.getOrDefault(s.charAt(0), new LinkedList<>());
//                list.add(s);
            }
        }
        for (LinkedList<String> value : map.values()) {
//            System.out.println(value);
            value.sort((a,b)->{
                if (a.length()==b.length()){
                    return a.compareTo(b);
                }
                return b.length()-a.length();
            });
        }
//        for (Map.Entry<Character, LinkedList<String>> entry : map.entrySet()) {
//            System.out.println(entry.getKey()+" "+entry.getValue());
//        }
        while (true){
            char c = sb.charAt(sb.length()-1);
            LinkedList<String> strings = map.get(c);
            if (strings!=null&&strings.size()>0){
                sb.append(strings.getFirst());
                strings.removeFirst();
            }else {
                break;
            }
        }
        System.out.println(sb);
    }
}
```