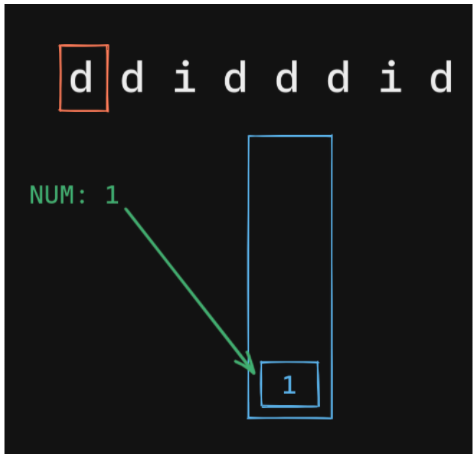
**1. Problem Discussion -:**

You are given a pattern of up to 8 lengths containing characters 'i' and 'd'. 'd' stands for decreasing and 'i' stands for increasing. You have to print the smallest number, using the digits 1 to 9 only without repetition, such that the digit decreases following a 'd' and increases following 'i'. d -> 21 i -> 12 ddd -> 4321 iii -> 1234 dddiddd -> 43218765 iiddd -> 126543

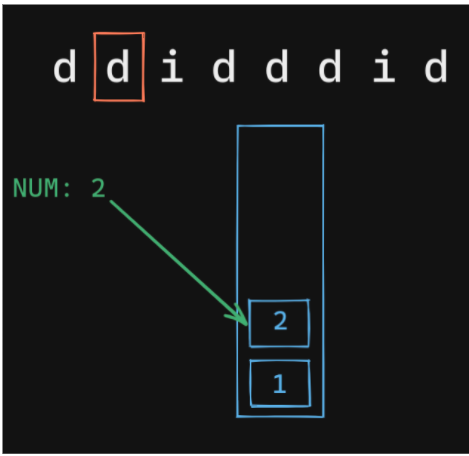
**2. Approach :-**

Let's first understand some of the patterns through examples: Example 1: only d => Our answer would be 21 and we can clearly see that 2 is decreasing to 1. Example 2: only i => Our answer would be 12, 1 increasing to 2 and 11 because 11 has repetition digit 1 and that is against the rules. Example 3: iii => Our answer is 1234 where 1<2<3<4 are in increasing order and this is the smallest number for this particular pattern. Example 4: ddd => Our output here is 4321 which is in decreasing order and is the smallest number for this particular pattern. Example 5: ddidd => Our output here is 321654 as we can see all the rules are followed where 3>2>1<6>5>4 and this is the smallest number for this pattern because if we analyze then according to first "dd" the smallest decreasing number is 321 and for second "dd" it is 654 as repetition is not allowed and i is splitting the two d's. This thing will work and will provide us the smallest number for any particular pattern because the most significant place has the smallest numbers followed by other places with no repetition. Also when we increase we have to check the sequence of d's and then the least possible increment is done whereas while decreasing we always decrease by 1. Based on the pattern we can say that our logic will work this way: Whenever we come across "d" we will push the number into the stack and then we will increment the number by 1. When we get "i" in the pattern we will first push into the stack then we will increment the number and then we will pop from the stack till the stack is empty and will print the result. When we reach the end of the pattern we will push the last increment number into the stack and then will pop from the stack till it is empty and will print the result. If you notice every time we get a 'd' we are pushing and not printing anything. But when we get the 'i' we do the operation of pop and print for all the stack items. Hence these points correspond to the 'd's.

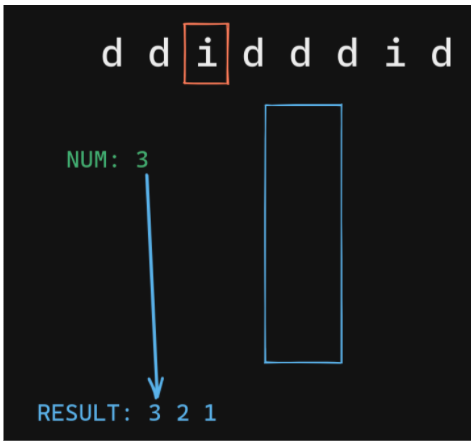
Now let's take an example, Pattern: "ddidddid"

****

First, we come across d so we insert num into the stack and increment it.

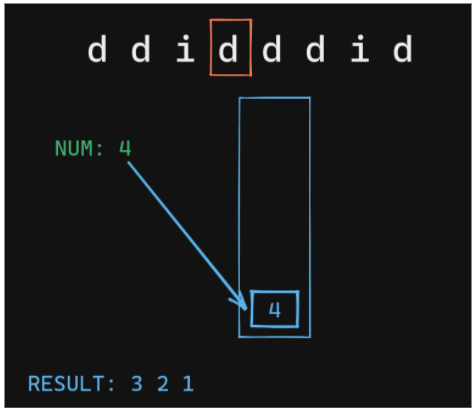
****

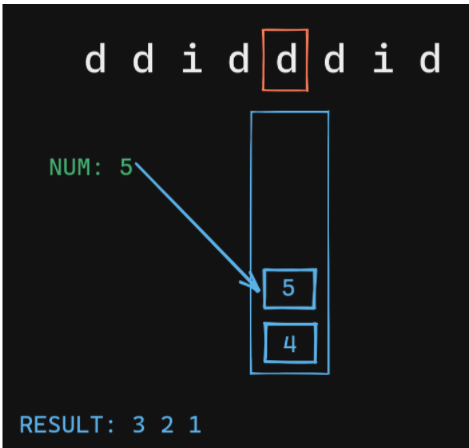
Now another time "d" is present so we insert num once more, and increment it.

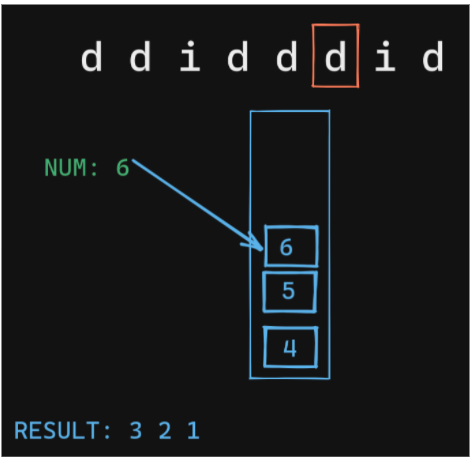
****

Now since we reach "i" we will print num and all the elements in the stack. Also, keep incrementing num.

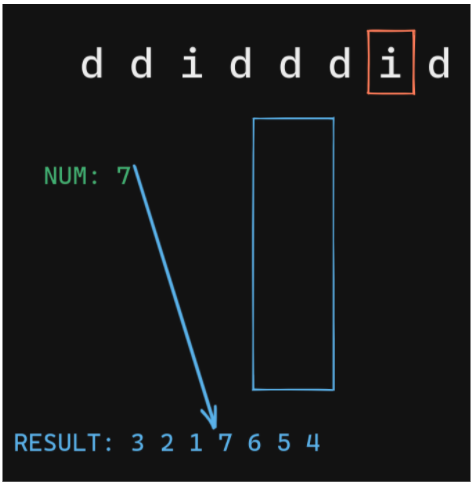
For the next 3 'd' s we keep doing the same thing, we push num and increment it.

****

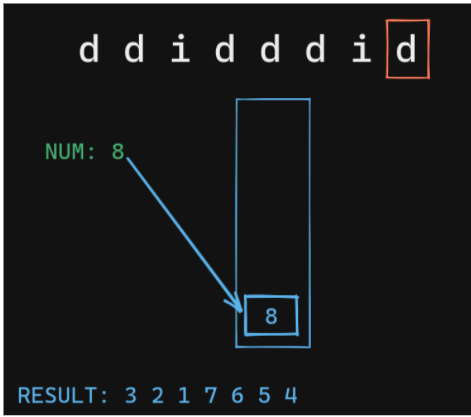
****

****

Since we got 'i' we will print num and pop all the items in the stack, and increment num.

****

This is our last operation.

****

If after this something is present in the stack we will simply print and pop it. So our final smallest number for this pattern is 321765498.

**3. CODE -:**

ConsoleJava

import java.io.\*;

import java.util.\*;

public class Main {

public static void main(String[] args) throws Exception {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String str = br.readLine();

// code

Stack< Integer> st = new Stack< >();

int num = 1;

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

char ch = str.charAt(i);

if (ch == 'd') { // when we encounter d

st.push(num);

num++;

} else { // when we encounter i

st.push(num);

num++;

while (st.size() > 0) {

System.out.print(st.pop());

}

}

}

st.push(num); // for last number

while (st.size() > 0) {

System.out.print(st.pop());

}

}

}

**4. CODE DISCUSSION -:**

Create a new stack Iterate through the string. If the char is d push the num to the stack and increment num.

Else push num and increment it but also run a loop until the stack is empty and in each iteration print the data. For the last number insert num and then run a loop until the stack is empty and in each iteration print the data.

**5. Analysis -:**

Time Complexity : O(n)

The Time Complexity is O(n)

Space Complexity : O(n)

The space complexity is O(n).

That was easy. Wasn't it? Our desire to make you learn will remain unsatisfactory if you still have doubts. We strongly recommend you to watch our video lecture on Smallest Number Following Pattern for clearing any type of doubts. Suggestions and feedback are always welcomed. You can contact us via our website. All the best for a bright future! Happy Coding!