【模拟-一种字符串压缩表示的解压

题目描述与示例

题目描述

有一种简易压缩算法: 针对全部由小写英文字母组成的字符串，将其中**连续超过两个相同字母的部分压缩为连续个数加该字母，其他部分保持原样不变**。

例如: 字符串 `"aaabbccccd"` 经过压缩成为字符串 `"3abb4cd"` 。

请您编写解压函数，根据输入的字符串，判断其是否为合法压缩过的字符串若输入合法则输出解压缩后的字符串，否则输出字符串 `"!error"` 来报告错误。

输入描述

输入一行，为一个 `ASCII` 字符串，长度不会超过 `100` 字符，用例保证输出的字符串长度也不会超过 `100` 字符。

输出描述

若判断输入为合法的经过压缩后的字符串，则输出压缩前的字符串若输入不合法，则输出字符串 `"!error"`

示例一

输入

```
4dff
```

输出

```
1 ddddff
```

示例二

输入

```
1 2dff
```

输出

```
1 !error
```

说明

两个 `d` 不需要压缩，故输入不合法。

示例三

输入

```
1 4d@A
```

输出

```
1 !error
```

说明

全部由小写英文字母组成的字符串压缩后不会出现特殊字符 `@` 和大写字母 `A` ，故输入不合法

解题思路

题意理解

本题的难点在于，**关于异常的表示并没有在题意中讲解的非常清楚**。

实际上，以下情况都应该属于异常

- 情况一：出现非数字和非小写字符，比如示例三的 `4d@A` 中出现了 `@` 和 `A`
- 情况二：数字出现了单个的 `0` 、 `1` 、 `2` 的情况，譬如示例二的 `2dff` ，或者例子 `0x1y`
- 情况三：出现了连续 `3` 个或 `3` 个以上的字符，没有在原字符串中被压缩，比如例子 `3dfff` ， `3dd`
- 情况四：原本应该被压缩在一起的字符并没有压缩在一起，比如例子 `a4a` ， `3b4b` ， `cc5c` 等等
- 情况五：对于数字而言，出现了先导 `0` ，比如例子 `04a` ， `a05b` 等等
- 情况六：原字符串的末尾出现了数字

分类讨论

首先可以先设置一个标记 `isError` 来表示遍历过程中是否出现异常。

一旦出现异常则将 `isError` 修改为 `True` ，且直接退出循环即可。

另外，我们还需要一个数字 `num` 来记录遍历过程中得到的数字，以及一个字符串 `ans` 来储存解压后的结果。

情况六是最容易判断的，我们可以在遍历结束后判断 `num` 是否为 `0` ，或者s的最后一个字符 `s[-1]` 是否为数字就可以判断出来。

```
# 在最末尾写上该判断
# 或者判断s[-1].isdigit()
if num != 0:
    isError = True
```

在理清楚题意之后，剩余内容就是进行分类讨论了。遍历字符串过程中，最容易分类的一共是3种情况。

- `ch` 为小写字符
- `ch` 为数字
- `ch` 为其他无关字符

当ch为无关字符

显然当 `ch` 为其他无关字符时，对应异常情况一。因此可以构建出整体代码框架为

```
for i, ch in enumerate(s):
    # 如果ch是小写字符
    if ch.islower():
        pass
    # 如果ch是数字
    elif ch.isdigit():
        pass
    # 情况一：
    # 遇到非小写字符、非数字，出现异常
    # 修改isError标记为True，表示出现异常，直接退出循环
    else:
        isError = True
        break
```

紧接着我们需要分别讨论 `ch` 为小写字符和 `ch` 为数字的情况。

当ch为数字

当 `ch` 为数字的时候，有可能发生在此处的错误是情况五，出现先导 `0` 。

这在代码上也是非常好处理的，先导 `0` 需要满足以下条件：

1. 该数字是 `0`
2. 该数字前面是一个非数字或前面没有字符
3. 该数字后面是一个数字

由于在本题中，单独的 `0` 也不能作为压缩后的数字出现（情况二）。

因此我们在判断先导 `0` 的过程中无需判断该数字后面是否是一个数字。

一旦出现某个数字 `0` 其前面的字符是一个非数字或者没有字符，则直接说明出现异常。对应的代码为

```
for i, ch in enumerate(s):
    # 如果ch是小写字符
    if ch.islower():
        pass
    # 如果ch是数字
    elif ch.isdigit():
        if len(ans) > 0 and ans[-1] == ch:
            isError = True
            break
        pass
    # 如果ch是其他无关字符
    else:
         pass
```

同时， `ch` 如果只是一个普通的数字，那么需要将 `ch` 记录在 `num` 中，因为此时我们还不确定这个数字的位数。

对应的代码为

```
for i, ch in enumerate(s):
    # 如果ch是小写字符
    if ch.islower():
        pass
    # 如果ch是数字
    elif ch.isdigit():
        if len(ans) > 0 and ans[-1] == ch:
            isError = True
            break
        num = num * 10 + int(ch)
    # 如果ch是其他无关字符
    else:
         pass
```

当ch为字母

当 `ch` 为字母时，这种情况最为复杂，包含了 `3` 种异常情况。

获得一个字母的时候，其前一个字符的情况只可能是两种：

- 前一个字符是数字
- 前一个字符是字母

如果前一个字符是字母，则说明这个连续的字母必须是以不超过连续两个的情况出现的。

此时对应的 `num` 的值是初始化值 `0` ，我们需要判断情况三是否出现。

如果通过了情况三的检查，则这个字符 `ch` 可以加入到 `ans` 中。

故对应的代码为

```
for i, ch in enumerate(s):
    # 如果ch是小写字符
    if ch.islower():
        if num == 0:
            if len(ans) >= 2 and ans[-1] == ch and ans[-2] == ch:
                isError = True
                break
            else:
                ans += ch
        pass
    # 如果ch是数字
    elif ch.isdigit():
        pass
    # 如果ch是其他无关字符
    else:
         pass
```

如果前一个字符是数字，那么还需要继续分类讨论。

如果这个数字小于等于 `2` ，则说明情况二出现。

```
for i, ch in enumerate(s):
    # 如果ch是小写字符
    if ch.islower():
        if num == 0:
            pass
        elif num <= 2:
            isError = True
            break
```

```
        else:
            pass
    # 如果ch是数字
    elif ch.isdigit():
        pass
    # 如果ch是其他无关字符
    else:
         pass
```

如果这个数字大于 `2` ，则可以进行解压操作。但同时还需要排除情况四的出现。

只有当所有的异常情况都没有出现时，才可以进行解压操作，并且将解压结果加入到 `ans` 末尾。

对应代码为

```
for i, ch in enumerate(s):
    # 如果ch是小写字符
    if ch.islower():
        if num == 0:
            pass
        elif num <= 2:
            pass
        else:
            if len(ans) > 0 and ans[-1] == ch:
                isError = True
                break
            ans += ch * num
            num = 0
    # 如果ch是数字
    elif ch.isdigit():
        pass
    # 如果ch是其他无关字符
    else:
         pass
```

把上述繁冗的分类讨论组织在一起，就构成了最终的代码。

代码

Python

```python
# 题目：2024E-一种字符串压缩表示的解压
# 分值：100
# 作者：许老师-闭着眼睛学数理化
# 算法：模拟
# 代码看不懂的地方，请直接在群上提问


s = input()
ans = str()
# 用于储存解压数字的变量num，初始化为0
num = 0
# 标记是否出现异常的变量
isError = False

# 遍历原字符串s中的所有字符
for i, ch in enumerate(s):
    # 如果ch是小写字符
    if ch.islower():
        # 如果num为0，说明ch无需解压，直接往ans后面延长ch即可
        if num == 0:
            # 情况三：
            # 在往ans后面延长之前，
            # 需要判断ans的倒数两个字符是否是ch
            # 如果出现连续三个字符相等，
            # 则此处的ch不应该作为单个字符出现
            # 即ch对应的num不应该是0，而应该是一个不小于3的数
            # 出现这种情况，说明原字符串的压缩不合理
            if len(ans) >= 2 and ans[-1] == ch and ans[-2] == ch:
                isError = True
                break
            else:
                ans += ch
        # 情况二：
        # 如果num为小于等于2的数字（即1或2），
        # 这是不合法的操作，因为压缩的重复字符数目至少为3
        # 修改isError标记为True，表示出现异常，直接退出循环
        elif num <= 2:
            isError = True
            break
        # 如果num为≥ 3 的数字，说明字符ch需要解压，
        # 将ch重复num次后，加入在ans后面
```

```python
        # 需要注意，使用完num后要将其重置为0
        else:
            # 情况四：
            # 如果此时答案中最后一个元素和当前解压元素相等
            # 说明原本应该被压缩为在一起的字符并没有压缩在一起
            # 修改isError标记为True，表示出现异常，直接退出循环
            if len(ans) > 0 and ans[-1] == ch:
                isError = True
                break
            ans += ch * num
            num = 0
    # 如果遇到数字
    elif ch.isdigit():
        # 情况五：
        # 首先判断先导0的情况
        # 若ch是0且其前一个元素是非数字或者该0是s的第一个字符
        # 则说明这个0是一个先导0（包括数字0在内）
        # 修改isError标记为True，表示出现异常，直接退出循环
        if int(ch) == 0:
            if (i > 0 and not s[i-1].isdigit()) or i == 0:
                isError = True
                break
        # 如果不是先导0，则需要将num扩大10倍后加上int(ch)
        # 用于解决遇到数字位数大于1的情况
        num = num * 10 + int(ch)
    # 情况一：
    # 遇到非小写字符、非数字，出现异常
    # 修改isError标记为True，表示出现异常，直接退出循环
    else:
        isError = True
        break

# 情况六：
# 如果退出循环时，num不为0，说明原字符串s的末尾是数字，属于不合法输入
# 此处用s[-1].isdigit()进行判断也可以
# 修改isError标记为True，表示出现异常，直接退出循环
if num != 0:
    isError = True

# 如果出现异常，则输出"!error"
# 否则输出解压后的字符串ans
print("!error") if isError else print(ans)
```

Java

```java
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String s = scanner.nextLine();
        StringBuilder ans = new StringBuilder();
        // 用于储存解压数字的变量num，初始化为0
        int num = 0;
        // 标记是否出现异常的变量
        boolean isError = false;

        // 遍历原字符串s中的所有字符
        for (int i = 0; i < s.length(); i++) {
            char ch = s.charAt(i);
            // 如果ch是小写字符
            if (Character.isLowerCase(ch)) {
                // 如果num为0，说明ch无需解压，直接往ans后面延长ch即可
                if (num == 0) {
                    // 情况三：
                    // 在往ans后面延长之前，
                    // 需要判断ans的倒数两个字符是否是ch
                    // 如果出现连续三个字符相等，
                    // 则此处的ch不应该作为单个字符出现
                    // 即ch对应的num不应该是0，而应该是一个不小于3的数
                    // 出现这种情况，说明原字符串的压缩不合理
                    if (ans.length() >= 2 && ans.charAt(ans.length() - 1) ==
ch && ans.charAt(ans.length() - 2) == ch) {
                        isError = true;
                        break;
                    } else {
                        ans.append(ch);
                    }
                }
                // 情况二：
                // 如果num为小于等于2的数字（即1或2），
                // 这是不合法的操作，因为压缩的重复字符数目至少为3
                // 修改isError标记为True，表示出现异常，直接退出循环
                else if (num <= 2) {
                    isError = true;
                    break;
                }
                // 如果num为≥ 3 的数字，说明字符ch需要解压，
                // 将ch重复num次后，加入在ans后面
                // 需要注意，使用完num后要将其重置为0
                else {
```

```
47                      // 情况四:
48                      // 如果此时答案中最后一个元素和当前解压元素相等
49                      // 说明原本应该被压缩为在一起的字符并没有压缩在一起
50                      // 修改isError标记为True，表示出现异常，直接退出循环
51                      if (ans.length() > 0 && ans.charAt(ans.length() - 1) ==
    ch) {
52                          isError = true;
53                          break;
54                      }
55                      for (int j = 0; j < num; j++) {
56                          ans.append(ch);
57                      }
58                      num = 0;
59                  }
60              }
61              // 如果遇到数字
62              else if (Character.isDigit(ch)) {
63                  // 情况五:
64                  // 首先判断先导0的情况
65                  // 若ch是0且其前一个元素是非数字或者该0是s的第一个字符
66                  // 则说明这个0是一个先导0（包括数字0在内）
67                  // 修改isError标记为True，表示出现异常，直接退出循环
68                  if (ch == '0') {
69                      if ((i > 0 && !Character.isDigit(s.charAt(i - 1))) || i ==
    0) {
70                          isError = true;
71                          break;
72                      }
73                  }
74                  // 如果不是先导0，则需要将num扩大10倍后加上int(ch)
75                  // 用于解决遇到数字位数大于1的情况
76                  num = num * 10 + (ch - '0');
77              }
78              // 情况一:
79              // 遇到非小写字符、非数字，出现异常
80              // 修改isError标记为True，表示出现异常，直接退出循环
81              else {
82                  isError = true;
83                  break;
84              }
85          }
86
87          // 情况六:
88          // 如果退出循环时，num不为0，说明原字符串s的末尾是数字，属于不合法输入
89          // 此处用s.charAt(s.length() - 1)进行判断也可以
90          // 修改isError标记为True，表示出现异常，直接退出循环
91          if (num != 0) {
```

```
            isError = true;
        }

        // 如果出现异常，则输出"!error"
        // 否则输出解压后的字符串ans
        if (isError) {
            System.out.println("!error");
        } else {
            System.out.println(ans.toString());
        }

        scanner.close();
    }
}

```

C++

```
#include <iostream>
#include <string>

using namespace std;

int main() {
    string s;
    cin >> s;
    string ans = "";
    // 用于储存解压数字的变量num，初始化为0
    int num = 0;
    // 标记是否出现异常的变量
    bool isError = false;

    // 遍历原字符串s中的所有字符
    for (int i = 0; i < s.length(); i++) {
        char ch = s[i];

        // 如果ch是小写字符
        if (islower(ch)) {
            // 如果num为0，说明ch无需解压，直接往ans后面延长ch即可
            if (num == 0) {
                // 情况三：
                // 在往ans后面延长之前，
                // 需要判断ans的倒数两个字符是否是ch
```

```
                // 如果出现连续三个字符相等，
                // 则此处的ch不应该作为单个字符出现
                // 即ch对应的num不应该是0，而应该是一个不小于3的数
                // 出现这种情况，说明原字符串的压缩不合理
                if (ans.length() >= 2 && ans[ans.length() - 1] == ch &&
ans[ans.length() - 2] == ch) {
                    isError = true;
                    break;
                } else {
                    ans += ch;
                }
            }
            // 情况二：
            // 如果num为小于等于2的数字（即1或2），
            // 这是不合法的操作，因为压缩的重复字符数目至少为3
            // 修改isError标记为True，表示出现异常，直接退出循环
            else if (num <= 2) {
                isError = true;
                break;
            }
            // 如果num为≥ 3 的数字，说明字符ch需要解压，
            // 将ch重复num次后，加入在ans后面
            // 需要注意，使用完num后要将其重置为0
            else {
                // 情况四：
                // 如果此时答案中最后一个元素和当前解压元素相等
                // 说明原本应该被压缩为在一起的字符并没有压缩在一起
                // 修改isError标记为True，表示出现异常，直接退出循环
                if (!ans.empty() && ans.back() == ch) {
                    isError = true;
                    break;
                }
                ans.append(num, ch);
                num = 0;
            }
        }
        // 如果遇到数字
        else if (isdigit(ch)) {
            // 情况五：
            // 首先判断先导0的情况
            // 若ch是0且其前一个元素是非数字或者该0是s的第一个字符
            // 则说明这个0是一个先导0（包括数字0在内）
            // 修改isError标记为True，表示出现异常，直接退出循环
            if (ch == '0') {
                if ((i > 0 && !isdigit(s[i - 1])) || i == 0) {
                    isError = true;
                    break;
```

```
                }
            }
            // 如果不是先导0，则需要将num扩大10倍后加上int(ch)
            // 用于解决遇到数字位数大于1的情况
            num = num * 10 + (ch - '0');
        }
        // 情况一:
        // 遇到非小写字符、非数字，出现异常
        // 修改isError标记为True，表示出现异常，直接退出循环
        else {
            isError = true;
            break;
        }
    }

    // 情况六:
    // 如果退出循环时，num不为0，说明原字符串s的末尾是数字，属于不合法输入
    // 此处用isdigit(s.back())进行判断也可以
    // 修改isError标记为True，表示出现异常
    if (num != 0) {
        isError = true;
    }

    // 如果出现异常，则输出"!error"
    // 否则输出解压后的字符串ans
    if (isError) {
        cout << "!error" << endl;
    } else {
        cout << ans << endl;
    }

    return 0;
}

```

时空复杂度

时间复杂度： `O(N)` 。需要从头到尾一次遍历原字符串 `s` 。

空间复杂度： `O(1)` 。不考虑 `s` 和 `ans` ，仅需若干常数变量。