【模拟】-靠谱的车

题目描述与示例

题目描述

程序员小明打了一辆出租车去上班。出于职业敏感，他注意到这辆出租车的计费表有点问题，总是偏大。出租车司机解释说他不喜欢数字 `4` ，所以改装了计费表，任何数字位置遇到数字 `4` 就直接跳过，其余功能都正常。

比如：

- `23` 再多一块钱就变为 `25`
- `39` 再多一块钱就变为 `50`
- `399` 再多一块钱就变为 `500`

小明识破了司机的伎俩，准备利用自己的学识打败司机的阴谋，给出计费表的表面读数，返回实际产生的费用。

输入描述

只有一行，数字 `N` ，表示里程表的读数。

`(1 <= N <= 888888888)`

输出描述

一个数字，表示实际产生的费用。

示例一

输入

```
1 5
```

输出

```
4
```

示例二

输入

```
25
```

输出

```
22
```

示例三

输入

```
100
```

输出

```
81
```

解题思路

从A进制到九进制

假设这种特殊的进制规则（逢 `4` 跳过）我们将其称之为A进制。

容易发现，在A进制中，每一个数位实际上只有 `9` 个数字，即 `0-3` 和 `5-9` 。

对于九进制而言，我们每一个数位也只有 `9` 个数字，即 `0-8` 。

另外，A进制是逢 `9` 进位（但逢 `4` 跳过），九进制是逢 `8` 进位。

显然，A进制和九进制之间映射关系是非常明显的：

- A进制的 `0-3` 对应着九进制的 `0-3`
- A进制的 `5-9` 对应着九进制的 `4-8`

因此，如果我们想令A进制转化为九进制的话，只需要枚举每一个数位 `digit` ，当这个数位 `digit` 大于等于 `5` 时，将其 `-1` 即可变成九进制的数位。即

```
for i in range(len(num_str)):
    digit = int(num_str[i])
    # 如果这位数字digit超过5，那么在九进制中实际上为digit-1
    if digit >= 5:
        digit -= 1
    # 经过上述语句后，digit即为九进制中的数位结果了
    pass
```

从九进制到十进制

> 2024/01/20 真题讲解 中对于题目 【模拟】2024D-来自异国的客人，我对进制转换做了比较详细的介绍。

当我们已经得到了该数字在九进制下的结果后，再将其转化为十进制就非常轻松了。

根据 算法题中常用数学概念、公式、方法汇总 中的相关概念，我们知道如果要将 `m` 进制转化为十进制，需要将 `m` 进制下**给定的数字按权展开，每一位乘以对应位置的权重**。比如将九进制下的 `1230` 转化为十进制的 `918`：

$$1230_9 = 1 \times 9^3 + 2 \times 9^2 + 3 \times 9^1 + 0 \times 9^0 = 918_{10}$$

由于权重的幂是从低位开始从 `0` 不断增加，所以我们可以先对原九进制的数字进行反转再正序遍历，就实现了从低位遍历了。即

```python
num_str = num_str[::-1]
ans = 0

for i in range(len(num_str)):
    digit = int(num_str[i])
    if digit >= 5:
        digit -= 1
    ans += digit * 9 ** i
```

最终输出 `ans` 即可。

代码

Python

```python
# 题目：【模拟】2024D-靠谱的车
# 分值：100
# 作者：许老师-闭着眼睛学数理化
# 算法：数学/模拟
# 代码看不懂的地方，请直接在群上提问


# 直接输入字符串，不需要转成数字，方便处理
num_str = input()

# 由于需要从低位开始考虑
# 因此对num_str进行反转
num_str = num_str[::-1]

# 初始化答案
ans = 0

# 遍历整个数字字符串，从低位开始考虑
for i in range(len(num_str)):
    # 获得该位数字
    digit = int(num_str[i])
    # 如果这位数字digit超过5，那么在九进制中实际上为digit-1
    if digit >= 5:
        digit -= 1
    # 根据digit的结果，将其转换为十进制结果，递增入ans中
    ans += digit * 9 ** i

# 输出答案
print(ans)
```

Java

```java
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // 直接输入字符串，不需要转成数字，方便处理
        String numStr = scanner.nextLine();

        // 由于需要从低位开始考虑
        // 因此对numStr进行反转
        numStr = new StringBuilder(numStr).reverse().toString();

```

```java
        // 初始化答案
        long ans = 0;

        // 遍历整个数字字符串，从低位开始考虑
        for (int i = 0; i < numStr.length(); i++) {
            // 获得该位数字
            int digit = Character.getNumericValue(numStr.charAt(i));
            // 如果这位数字digit超过5，那么在九进制中实际上为digit-1
            if (digit >= 5) {
                digit -= 1;
            }
            // 根据digit的结果，将其转换为十进制结果，递增入ans中
            ans += digit * Math.pow(9, i);
        }

        // 输出答案
        System.out.println(ans);
    }
}
```

C++

```cpp
#include <iostream>
#include <string>
#include <algorithm>
#include <cmath>

using namespace std;

int main() {
    string numStr;
    cin >> numStr;

    // 由于需要从低位开始考虑
    // 因此对numStr进行反转
    reverse(numStr.begin(), numStr.end());

    // 初始化答案
    long long ans = 0;

    // 遍历整个数字字符串，从低位开始考虑
    for (size_t i = 0; i < numStr.length(); ++i) {
        // 获得该位数字
        int digit = numStr[i] - '0';
        // 如果这位数字digit超过5，那么在九进制中实际上为digit-1
```

```
        if (digit >= 5) {
            digit -= 1;
        }
        // 根据digit的结果，将其转换为十进制结果，递增入ans中
        ans += digit * pow(9, i);
    }

    // 输出答案
    cout << ans << endl;

    return 0;
}
```

时空复杂度

时间复杂度：`O(n)`。仅需循环 `n` 次，其中 `n` 为A进制下的数字 `num` 的数位数，或者说字符串 `num_str` 的长度。

空间复杂度：`O(1)`。除了输入的字符串，仅需若干常数变量维护遍历过程。