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：

1. 23再多一块钱就变为25；
2. 39再多一块钱变为50；
3. 399再多一块钱变为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。以回车结束。

用例

输入	5
输出	4
说明	5表示计费表的表面读数。 4表示实际产生的费用其实只有4块钱。

输入	17
输出	15
说明	17表示计费表的表面读数。 15表示实际产生的费用其实只有15块钱。

输入	100
输出	81
说明	100表示计费表的表面读数。 81表示实际产生的费用其实只有81块钱。

题目解析

看到题目提示输入数字N的取值范围1<=N<=888888888，我陷入了沉思。这是要我们设计一个O(1)时间复杂度的算法呀，因为就算是O(n)也遭不住9个8的数量级啊。

原型题应该是：

本题的解题思路类似，但是不是八进制，而是九进制

	实际费用	计费表费用	
	1	1	1*9^0
	2	2	2*9^0
	3	3	3*9^0
	4	5	(5-1)*9^0
	5	6	(6-1)*9^0
	6	7	(7-1)*9^0
	7	8	(8-1)*9^0
	8	9	(9-1)*9^0
逢9进1	9	10	1*9^1 + 0*9^0
	10	11	1*9^1 + 1*9^0
	11	12	1*9^1 + 2*9^0
	12	13	1*9^1 + 3*9^0
计费表工作原理相当于九进制计算器	13	15	1*9^1 + (5-1)*9^0
	14	16	1*9^1 + (6-1)*9^0
	15	17	1*9^1 + (7-1)*9^0
	16	18	1*9^1 + (8-1)*9^0
	17	19	1*9^1 + (9-1)*9^0
	18	20	2*9^1 + 0*9^0

Java算法源码

```java
import java.util.Arrays;
import java.util.Scanner;

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

    int[] arr = Arrays.stream(sc.nextLine().split("")).mapToInt(Integer::parseInt).toArray();

    System.out.println(getResult(arr));
  }

  public static int getResult(int[] arr) {
    int correct = 0;

    for (int i = 0; i < arr.length; i++) {
      int fault = arr[i];
      if (fault > 4) fault--;

      for (int j = arr.length - i - 1; j > 0; j--) {
        fault *= 9;
      }

      correct += fault;
    }

    return correct;
  }
}
```

JS算法源码

```javascript
/* JavaScript Node ACM模式 控制台输入获取 */
const readline = require("readline");

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout,
});

rl.on("line", (line) => {
  const arr = line.split("").map((ele) => parseInt(ele));

  let correct = 0;
  for (let i = 0; i < arr.length; i++) {
    // 遍历输入数的每一位
    let fault = arr[i];
    if (fault > 4) {
      // 如果本位数比4大，则相当于跳过1次，前置将本位数-1
      fault--;
    }

    for (let j = arr.length - i - 1; j > 0; j--) {
      // 将本位转成十进制
      fault *= 9;
    }

    correct += fault; // 累加各位对应十进制数
  }
  console.log(correct);
});
```

Python算法源码

```python
# 输入获取
arr = list(map(int, list(input())))


# 算法入口
def getResult():
    correct = 0
    for i in range(len(arr)):
        fault = arr[i]
        if fault > 4:
            fault -= 1

        for j in range(len(arr) - 1 - i, 0, -1):
            fault *= 9

        correct += fault
    return correct


# 调用算法
print(getResult())
```