166. Fraction to Recurring Decimal

We will use a dict to record numerator and mod as key, index as value to detect repeat cycle.

```
class Solution:
    def fractionToDecimal(self, numerator: int, denominator: int) -> str:
        result = ''
        memo = {}
        mark = 1
        if numerator * denominator< 0:
            mark = -1
            numerator = abs(numerator)
            denominator = abs(denominator)
        rest, mod = divmod(numerator, denominator)
        numerator = mod * 10
        fraction = []
        count = 0
        result += str(rest)
        while numerator:
            rest, mod = divmod(numerator, denominator)
            if (numerator, mod) in memo:
                fraction.append(')')
                break;
            memo[(numerator, mod)] = count
            numerator = mod
            mod *= 10
            numerator = mod
            fraction.append(str(rest))
            count += 1
        if numerator != 0:
            fraction.insert(memo[(numerator, mod)], '(')
        if fraction:
            result += '.' + ''.join(fraction)

        return '-' + result if mark == -1 else result
```

739. Daily Temperatures

We will use a stack to store all (value, idx) pairs if its value is less than latter one, or we will empty this list and append this this pair. We will go through all elements and get index difference with previous one. TC is O(n)

```
from collections import deque
class Solution:
    def dailyTemperatures(self, T: List[int]) -> List[int]:
        length = len(T)
        stack = []
        result = []
        for i in range(length - 1, -1, -1):
```

```
            while stack and T[i] >= stack[-1][0]:
                stack.pop()
            if not stack:
                result.append(0)
            else:
                result.append(stack[-1][1] - i)
            stack.append((T[i], i))
        return result[::-1]
```

244. Shortest Word Distance II
We will use a dict to record all words index. Then we will compare two sorted list and get shortest difference of two elements in two lists. TC is O(m + n)

```
from collections import defaultdict
from bisect import *
class WordDistance:

    def __init__(self, words: List[str]):
        self.memo = defaultdict(list)
        for idx, word in enumerate(words):
            self.memo[word].append(idx)

    def shortest(self, word1: str, word2: str) -> int:
        min_length = float('inf')
        for i in self.memo[word1]:
            left = bisect_left(self.memo[word2], i)
            if left - 1 >= 0:
                min_length = min(min_length, abs(self.memo[word2][left - 1] - i))
            if left < len(self.memo[word2]):
                min_length = min(min_length, abs(self.memo[word2][left] - i))
        return min_length
```

957. Prison Cells After N Days
We will find pattern and rule. TC is O(1)

```
class Solution:
    def prisonAfterNDays(self, cells: List[int], N: int) -> List[int]:
        N = (N - 1) % 14 + 1
        for i in range(N):
            result = [0]
            for i in range(1, 7):
                result.append(1 if cells[i - 1] == cells[i + 1] else 0)
            result.append(0)
            cells = result
        return result
```

541. Reverse String II

We will find reverse first k elements for every 2k elements. TC is O(n)

```
class Solution:
    def reverseStr(self, s: str, k: int) -> str:
        result = ''
        for i in range(0, len(s), 2 * k):
            result += s[i:i + k][::-1] + s[i + k:i + 2 * k]
        return result
```