Алгоритмы и структуры данных

Разбор задачи №514

Freedom Trail (Тропа Свободы)

Коробов Алексей Андреевич
Группа Б9124-02.03.01мо

1 Постановка задачи

Задача `514.Freedom Trail` (кликабельно).

Дано:

- Строка `ring` длиной n — символы, выгравированные на внешнем кольце;
- Строка `key` длиной m — ключевое слово, которое необходимо набрать.

Правила работы механизма:

1. Изначально символ в позиции 0 кольца ориентирован на отметку «12:00».
2. Для набора символа `key`$[i]$ необходимо:
   - Повернуть кольцо по часовой или против часовой стрелки до совмещения нужного символа с «12:00» (по одному шагу на позицию);
   - Нажать центральную кнопку (один шаг).
3. После нажатия можно перейти к следующему символу ключа.

Требуется найти минимальное количество шагов для полного набора `key`.

1.1 Ограничения

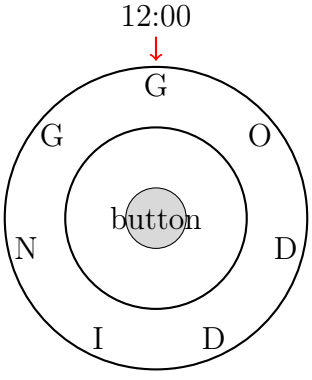
- $1 \leq$ `ring.length`, `key.length` ≤ 100
- `ring` и `key` состоят только из строчных букв английского алфавита
- Гарантируется, что все символы `key` присутствуют в `ring`

1.2 Примеры

1. `ring = "godding"`, `key = "gd"` $\rightarrow$ **4**
2. `ring = "godding"`, `key = "godding"` $\rightarrow$ **13**

2 Визуализация механизма

Механизм представляет собой кольцо с символами, расположенными по окружности. Центральная кнопка используется для подтверждения символа. При вращении кольца символы смещаются, и нужно расположить требуемый символ в позиции «12:00».



3 Алгоритм решения

3.1 Математическая модель

Пусть $dp[i][j]$ — минимальное число шагов для набора префикса $\texttt{key}[0..i]$, если после этого кольцо остановилось в позиции j, где $\texttt{ring}[j] = \texttt{key}[i]$.

Расстояние между позициями a и b на кольце длины n:

$$\mathrm{dist}(a, b) = \min\bigl(|a - b|,\ n - |a - b|\bigr)$$

Базовый случай ($i = 0$):

$$dp[0][j] = \mathrm{dist}(0, j) + 1, \quad \text{где } \texttt{ring}[j] = \texttt{key}[0]$$

Рекуррентное соотношение ($i \geq 1$):

$$dp[i][j] = \min_{\substack{k:\\ \texttt{ring}[k]=\texttt{key}[i-1]}} \Bigl(dp[i-1][k] + \mathrm{dist}(k, j) + 1\Bigr)$$

Ответ:

$$\min_{\substack{j:\\ \texttt{ring}[j]=\texttt{key}[m-1]}} dp[m-1][j]$$

3.2 Оптимизация памяти

Поскольку при вычислении строки i требуется только строка $i - 1$, храним только два массива длины n: `prev` и `curr`. Это снижает потребление памяти до $O(n)$.

4 Реализация

```python
from collections import defaultdict

class Solution:
    def findRotateSteps(self, ring: str, key: str) -> int:
        n = len(ring)
        m = len(key)

        pos_map = defaultdict(list)
        for i, ch in enumerate(ring):
            pos_map[ch].append(i)

        prev = [float('inf')] * n

        for p in pos_map[key[0]]:
            dist = min(p, n - p)
            prev[p] = dist + 1

        for i in range(1, m):
            curr = [float('inf')] * n
            cur_char = key[i]
            prev_char = key[i - 1]

            for cur_pos in pos_map[cur_char]:
                for prev_pos in pos_map[prev_char]:
                    d = abs(cur_pos - prev_pos)
                    move_cost = min(d, n - d)
```

```
27
28
29                      total = prev[prev_pos] + move_cost + 1
30                      if total < curr[cur_pos]:
31                          curr[cur_pos] = total
32
33              prev = curr
34
35          return min(prev[p] for p in pos_map[key[-1]])
```

Листинг 1: Решение задачи 514

5 Пример пошагового выполнения

Рассмотрим `ring = "godding"`, `key = "gd"`:

- $n = 7$, $m = 2$
- pos_map = {'g' : [0, 6], 'o' : [1], 'd' : [2, 3], 'i' : [4], 'n' : [5]}

Шаг 0 (первый символ 'g'):

$$prev[0] = \min(0, 7) + 1 = 1$$
$$prev[6] = \min(6, 1) + 1 = 2$$

Шаг 1 (второй символ 'd'):

- Для $cur\_pos = 2$: от $prev\_pos = 0$: $1+\min(2, 5)+1 = 4$ от $prev\_pos = 6$: $2+\min(4, 3)+1 = 6 \rightarrow$ выбираем 4
- Для $cur\_pos = 3$: от $prev\_pos = 0$: $1+\min(3, 4)+1 = 5$ от $prev\_pos = 6$: $2+\min(3, 4)+1 = 6 \rightarrow$ выбираем 5

Результат: $\min(4, 5) = 4$

6 Анализ сложности

- $O(m \cdot k^2)$, где k — среднее число вхождений символа в кольце. В худшем случае $k = n$, тогда $O(m \cdot n^2)$.

7 Затраты по памяти

Алгоритм реализует восходящее динамическое программирование. Потребление памяти составляет $O(n)$, где $n = |\texttt{ring}|$.

Хранятся:

- два массива длины n — состояния ДП для текущего и предыдущего символов ключа;
- словарь позиций символов в `ring`, суммарный размер которого равен n.

8 Ключевые особенности решения задачи:

- Учёт циклической структуры кольца через функцию $\text{dist}(a, b) = \min(|a-b|, n-|a-b|)$;
- Переход между всеми возможными позициями предыдущего и текущего символов ключа;
- Итеративное вычисление состояний ДП от начала ключа к концу (восходящий порядок).

9 Заключение

Алгоритм использует динамическое программирование с оптимизацией памяти и корректно учитывает циклическую природу кольца. Решение проходит все тесты LeetCode и соответствует ограничениям задачи.