САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
ФАКУЛЬТЕТ ИНФОКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по лабораторной работе №7
по курсу «Алгоритмы и структуры данных»
Тема: Динамическое программирование №1
Вариант 3

Выполнил:
Бай М.О.
К3141

Проверил:
Афанасьев А.В.

Санкт-Петербург
2024 г.

Содержание отчета

Оглавление

Содержание отчета 2

Задачи по варианту 3

Задача №1. Множество 3

Задача №2. Телефонная книга 5

Задача №4. Прошитый ассоциативный массив**Error! Bookmark not defined.**

Задача №5. Выборы в США **Error! Bookmark not defined.**

Вывод 8

Задачи по варианту

Задача №4. Наибольшая общая подпоследовательность двух последовательностей

Текст задачи:

Вычислить длину самой длинной общей подпоследовательности из двух последовательностей.

Даны две последовательности $A = (a_1, a_2, ..., a_n)$ и $B = (b_1, b_2, ..., b_m)$, найти длину их самой длинной общей подпоследовательности, т.е. наибольшее неотрицательное целое число p такое, что существуют индексы $1 \leq i_1 < i_2 < ... < i_p \leq n$ и $1 \leq j_1 < j_2 < ... < j_p \leq m$ такие, что $a_{i_1} = b_{j_1}, ..., a_{i_p} = b_{j_p}$.

Код программы:

```python
from Lab7.utils import read_input, write_output, decorate


def longest_common_subsequence(A, B):
    n = len(A)
    m = len(B)

    # Create DP table of size (n+1) x (m+1)
    dp = [[0] * (m + 1) for _ in range(n + 1)]

    for i in range(1, n + 1):
        for j in range(1, m + 1):
            if A[i - 1] == B[j - 1]:
                dp[i][j] = dp[i - 1][j - 1] + 1
            else:
                dp[i][j] = max(dp[i - 1][j], dp[i][j - 1])

    return dp[n][m]


def main():
    f = read_input(4)

    try:
        n = int(f[0].strip())
        A = list(map(int, f[1].strip().split()))
        m = int(f[2].strip())
        B = list(map(int, f[3].strip().split()))
    except ValueError as e:
        raise ValueError(f"Error parsing input: {e}")

    result = longest_common_subsequence(A, B)

    write_output(4, str(result) + "\n")
    print(str(result) + "\n")
    print()

if __name__ == "__main__":
    decorate(task=4, task_name='longest_common_subsequence')
```

Описание работы программы:

Основные Компоненты:
Импорты:
Импортируются функции read_input, write_output и decorate из модуля Lab7.utils, которые используются для работы с вводом/выводом и логированием.
Функция longest_common_subsequence(A, B):
Принимает две последовательности A и B.
Создает двумерный массив dp размером (n+1) x (m+1), где n и m — длины последовательностей A и B соответственно. Этот массив будет использоваться для хранения результатов подзадач.
Заполняет массив dp с помощью динамического программирования:
Если текущие элементы последовательностей равны (A[i-1] == B[j-1]), то значение в dp[i][j] будет равно dp[i-1][j-1] + 1.
Если не равны, то значение будет равно максимальному из dp[i-1][j] и dp[i][j-1].
Возвращает длину наибольшей общей подпоследовательности, которая находится в dp[n][m].
Функция main():
Считывает входные данные с помощью read_input(4).
Обрабатывает входные данные, извлекая количество элементов и сами последовательности A и B. В случае ошибки при парсинге данных выбрасывается исключение ValueError.
Вызывает функцию longest_common_subsequence, передавая ей последовательности A и B, и получает результат.
Записывает результат в выходной файл с помощью write_output(4, str(result) + "\n") и выводит его на экран.
Точка Входа:
Проверяет, запущен ли код как основной модуль, и вызывает функцию decorate, передавая параметры задачи.

Тесты:

```
import unittest
from Lab7.task4.src.longest_common_subsequence import longest_common_subsequence

class TestLongestCommonSubsequence(unittest.TestCase):

    def test_empty_sequences(self):
        self.assertEqual(longest_common_subsequence([], []), 0)

    def test_one_empty_sequence(self):
        self.assertEqual(longest_common_subsequence([], [1, 2, 3]), 0)
        self.assertEqual(longest_common_subsequence([1, 2, 3], []), 0)

    def test_no_common_elements(self):
```

```
        self.assertEqual(longest_common_subsequence([1, 2, 3], [4, 5, 6]), 0)

    def test_identical_sequences(self):
        self.assertEqual(longest_common_subsequence([1, 2, 3], [1, 2, 3]), 3)

    def test_partial_overlap(self):
        self.assertEqual(longest_common_subsequence([1, 2, 3, 4], [2, 4]), 2)
        self.assertEqual(longest_common_subsequence([1, 3, 4, 5], [1, 2, 3]), 2)

    def test_longer_sequences(self):
        self.assertEqual(longest_common_subsequence([1, 2, 3, 4, 5], [2, 3, 5, 6]), 3)
        self.assertEqual(longest_common_subsequence([1, 2, 3, 4, 5], [5, 4, 3, 2, 1]), 1)


if __name__ == '__main__':
    unittest.main()
```

Вывод по задаче:

Данный код реализует алгоритм для нахождения длины наибольшей общей подпоследовательности (Longest Common Subsequence, LCS) двух последовательностей. Он считывает входные данные, обрабатывает их и выводит результат.

Задача №7. Шаблоны

Текст задачи:

Многие операционные системы используют шаблоны для ссылки на группы объектов: файлов, пользователей, и т. д. Ваша задача – реализовать простейший алгоритм проверки шаблонов для имен файлов.

В этой задаче алфавит состоит из маленьких букв английского алфавита и точки («.»). Шаблоны могут содержать произвольные символы алфавита, а также два специальных символа: «?» и «*». Знак вопроса («?») соответствует ровно одному произвольному символу. Звездочка «+» соответствует подстроке произвольной длины (возможно, нулевой). Символы алфавита, встречающиеся в шаблоне, отображаются на ровно один такой же символ в проверяемой строчке. Строка считается подходящей под шаблон, если символы шаблона можно последовательно отобразить на символы строки таким образом, как описано выше. Например, строчки «ab», «aab» и «beda.» подходят под шаблон «*a?», а строчки «bebe», «a» и «ba» –нет.

Код программы:

```
import sys
import os

from Lab7.utils import read_input, write_output, decorate

sys.path.append(os.path.abspath(os.path.join(os.path.dirname(__file__), '../..')))

def template(temp: str, string: str) -> str:
    n, m = len(temp), len(string)
    array = [[False] * (m + 1) for _ in range(n + 1)]
```

```
    array[0][0] = True

    for i in range(1, n + 1):
        if temp[i - 1] == '*':
            array[i][0] = array[i - 1][0]
        else:
            break

    for i in range(1, n + 1):
        for j in range(1, m + 1):
            if temp[i - 1] == string[j - 1] or temp[i - 1] == '?':
                array[i][j] = array[i - 1][j - 1]
            elif temp[i - 1] == '*':
                array[i][j] = array[i - 1][j] or array[i][j - 1]

    return "YES" if array[n][m] else "NO"

def main():
    input_file = read_input(7)
    temp = input_file[0]
    string = input_file[1]

    res = 'YES' if template(temp, string) else 'NO'
    write_output(7, res)
    print(res)
    print()

if __name__ == '__main__':
    decorate(task=7, task_name='templates')
```

Описание работы программы:

Основные Компоненты:

Импорты:

Импортируются модули sys и os для работы с файловой системой.

Импортируются функции read_input, write_output и decorate из модуля Lab7.utils, которые используются для работы с вводом/выводом и логированием.

Функция template(temp: str, string: str) -> str:

Принимает на вход шаблон temp и строку string.

Создает двумерный массив array размером (n+1) x (m+1), где n и m — длины шаблона и строки соответственно. Этот массив будет использоваться для хранения результатов подзадач.

Инициализирует array[0][0] как True, так как пустой шаблон соответствует пустой строке.

Заполняет первую строку массива, обрабатывая случаи, когда шаблон начинается с *.

Заполняет массив array с помощью динамического программирования:

Если текущий символ шаблона равен текущему символу строки или символу ?, то значение в array[i][j] будет равно array[i-1][j-1].

Если текущий символ шаблона — *, то значение будет равно array[i-1][j] (игнорирование *) или array[i][j-1] (учет одного символа из строки).
Возвращает "YES", если array[n][m] равно True, и "NO" в противном случае.
Функция main():
Считывает входные данные с помощью read_input(7).
Извлекает шаблон и строку из входных данных.
Вызывает функцию template, передавая ей шаблон и строку, и получает результат.
Записывает результат в выходной файл с помощью write_output(7, res) и выводит его на экран.
Точка Входа:
Проверяет, запущен ли код как основной модуль, и вызывает функцию decorate, передавая параметры задачи.

Тесты:

```
import unittest
from Lab7.task7.src.templates import template

class TestTemplateMatching(unittest.TestCase):

    def test_exact_match(self):
        self.assertEqual(template("abc", "abc"), "YES")

    def test_question_mark_match(self):
        self.assertEqual(template("a?c", "abc"), "YES")
        self.assertEqual(template("a?c", "acc"), "YES")
        self.assertEqual(template("a?c", "abx"), "NO")

    def test_asterisk_match(self):
        self.assertEqual(template("*", "abc"), "YES")
        self.assertEqual(template("a*", "abc"), "YES")
        self.assertEqual(template("*c", "abc"), "YES")
        self.assertEqual(template("a*b", "aabb"), "YES")
        self.assertEqual(template("a*b", "ab"), "YES")
        self.assertEqual(template("a*b", "b"), "NO")

    def test_multiple_asterisks(self):
        self.assertEqual(template("*a*b*", "xxaabbxx"), "YES")
        self.assertEqual(template("*a*b*c*", "xxaabbccxx"), "YES")
        self.assertEqual(template("*a*b*c*", "xxaaxbbxx"), "NO")

    def test_no_match(self):
        self.assertEqual(template("abc", "def"), "NO")
        self.assertEqual(template("a?c", "abcx"), "NO")
        self.assertEqual(template("a*b", "xyz"), "NO")

    def test_empty_string_and_pattern(self):
        self.assertEqual(template("", ""), "YES")
        self.assertEqual(template("a", ""), "NO")
        self.assertEqual(template("", "a"), "NO")
        self.assertEqual(template("*", ""), "YES")
        self.assertEqual(template("a*", ""), "NO")
```

```
if __name__ == '__main__':
    unittest.main()
```

Вывод по задаче:

Данный код реализует алгоритм для проверки, соответствует ли заданная строка шаблону, который может содержать символы подстановки: ? (заменяет любой одиночный символ) и * (заменяет любую последовательность символов, включая пустую). Код считывает входные данные, обрабатывает их и выводит результат.

Вывод

В данной работе мы реализовали две задачи и тесты для них.