

Национальный исследовательский Университет ИТМО
Мегафакультет информационных и трансляционных технологий
Факультет инфокоммуникационных технологий

Алгоритмы и структуры данных

Лабораторная работа №7

Работу

выполнил:

Рудникова В.О.

Группа: К3125

Преподаватель:

Артамонова В.Е.

Санкт-Петербург
2022

Содержание

Задачи по варианту	3
Задача 4. Наибольшая общая подпоследовательность двух последовательностей	3
Условие задачи	3
Листинг кода	3
Текстовое объяснение решения	3
Результаты работы кода	4
Время выполнения и затраты памяти	5
Вывод по задаче	5
Дополнительные задачи	6
Задача 5. Наибольшая общая подпоследовательность трёх последовательностей	6
Условие задачи	6
Листинг кода	7
Текстовое объяснение решения	7
Результаты работы кода	8
Время выполнения и затраты памяти	9
Вывод по задаче	9
Вывод	10
Список использованных источников	11

Задачи по варианту

Задача 4. Наибольшая общая подпоследовательность двух последовательностей

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

- Формат ввода / входного файла (input.txt).
 - Первая строка: n - длина первой последовательности.
 - Вторая строка: a_1, a_2, \dots, a_n через пробел.
 - Третья строка: m - длина второй последовательности.
 - Четвертая строка: b_1, b_2, \dots, b_m через пробел.
- Ограничения: $1 \leq n, m \leq 100$; $-10^9 < a_i, b_i < 10^9$.
- Формат вывода / выходного файла (output.txt). Выведите число p .
- Ограничение по времени. 1 сек.

```
1  from time import process_time
2  from tracemalloc import start, get_traced_memory
3
4
5  def lcs(arr1, arr2):
6      n = len(arr1)
7      m = len(arr2)
8      dp = [[0] * (m + 1) for i in range(n + 1)]
9      for i in range(1, n + 1):
10         for j in range(1, m + 1):
11             if arr1[i - 1] == arr2[j - 1]:
12                 dp[i][j] = dp[i - 1][j - 1] + 1
13             else:
14                 dp[i][j] = max(dp[i - 1][j], dp[i][j - 1])
15     return dp[n][m]
16
17
18  if __name__ == '__main__':
19      start()
20      with open('input.txt', 'r') as f:
21          a = int(f.readline())
22          seq1 = [int(i) for i in f.readline().split()]
23          b = int(f.readline())
24          seq2 = [int(i) for i in f.readline().split()]
25      with open('output.txt', 'w') as g:
26          g.write(str(lcs(seq1, seq2)))
27
28      print('Time:', str(process_time()), 'sec')
29      print('Memory usage:', str(get_traced_memory()[1] / 1024), 'KB')
```

Длина общих подпоследовательностей хранится в двумерном списке, каждая мера которого отвечает за одну из данных последовательностей. Длина общей подпоследовательности для каждой ячейки списка определяется его предыдущими ячейками.

Результат работы кода на примерах из текста задачи:??

```
input.txt x
1 3
2 2 7 5
3 2
4 2 5

output.txt x
1 2
```

Результат работы кода на минимальных и максимальных значениях:??

```
input.txt x
1 1000
2 999 998 997 996 995 994 993 992 991 990
3 1000
4 999 998 997 996 995 994 993 992 991 990
5

output.txt x
1 1000
```

```
input.txt x
1 2
2 1 2
3 2
4 2 1

output.txt x
1 1
```

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.078375 sec	17.244140625 KB
Пример из задачи	0.078375 sec	17.244140625 KB
Верхняя граница диапазона значений входных данных из текста задачи	0.078375 sec	17.244140625 KB

Вывод по задаче:

В задаче я реализовала алгоритм поиска длины наибольшей общей подпоследовательности для двух последовательностей. [1]

Дополнительные задачи

Задача 5. Наибольшая общая подпоследовательность трёх последовательностей

Вычислить длину самой длинной общей подпоследовательности из двух последовательностей. Даны три последовательности: $A = (a_1, a_2, \dots, a_n)$, $B = (b_1, b_2, \dots, b_m)$ и $C = (c_1, c_2, \dots, c_k)$ найти длину их самой длинной общей подпоследовательности, т.е. наибольшее неотрицательное целое число p такое, что существуют индексы $1 \leq i_1 < i_2 < \dots < i_p \leq n$, $1 \leq j_1 < j_2 < \dots < j_p \leq m$ и $1 \leq k_1 < k_2 < \dots < k_p \leq k$ такие, что $a_{i_1} = b_{j_1} = c_{k_1}, \dots, a_{i_p} = b_{j_p} = c_{k_p}$.

- Формат ввода / входного файла (input.txt).
 - Первая строка: n - длина первой последовательности.
 - Вторая строка: a_1, a_2, \dots, a_n через пробел.
 - Третья строка: m - длина второй последовательности.
 - Четвертая строка: b_1, b_2, \dots, b_m через пробел.
 - Пятая строка: k - длина третьей последовательности.
 - Шестая строка: c_1, c_2, \dots, c_k через пробел.
- Ограничения: $1 \leq n, m \leq 100$; $-10^9 < a_i, b_i < 10^9$.
- Формат вывода / выходного файла (output.txt). Выведите число p .
- Ограничение по времени. 1 сек. (1)

$$y = \frac{x^2 - 3}{x - 2}. \quad (1)$$

```

1  from time import process_time
2  from tracemalloc import start, get_traced_memory
3
4
5  def lcs(arr1, arr2, arr3):
6      n = len(arr1)
7      m = len(arr2)
8      l = len(arr3)
9      dp = [[[0] * (l + 1) for _ in range(m + 1)] for _ in range(n + 1)]
10     for i in range(1, n + 1):
11         for j in range(1, m + 1):
12             for k in range(1, l + 1):
13                 if arr1[i - 1] == arr2[j - 1] and arr3[k - 1] == arr2[j -
14                     ↪ 1]:
15                     dp[i][j][k] = dp[i - 1][j - 1][k - 1] + 1
16                 else:
17                     dp[i][j][k] = max(dp[i - 1][j][k], dp[i][j - 1][k],
18                     ↪ dp[i][j][k - 1])
19
20     return dp[n][m][l]
21
22 if __name__ == '__main__':
23     start()
24     with open('input.txt', 'r') as f:
25         a = int(f.readline())
26         seq1 = [int(i) for i in f.readline().split()]
27         b = int(f.readline())
28         seq2 = [int(i) for i in f.readline().split()]
29         c = int(f.readline())
30         seq3 = [int(i) for i in f.readline().split()]
31     with open('output.txt', 'w') as g:
32         g.write(str(lcs(seq1, seq2, seq3)))
33
34     print('Time:', str(process_time()), 'sec')
35     print('Memory usage:', str(get_traced_memory()[1] / 1024), 'KB')

```

Длина общих подпоследовательностей хранится в трёхмерном списке, каждая мера которого отвечает за одну из данных последовательностей. Длина общей подпоследовательности для каждой ячейки списка определяется его предыдущими ячейками.

Результат работы кода на примерах из текста задачи:??

input.txt	
1	5
2	8 3 2 1 7
3	7
4	8 2 1 3 8 10 7
5	6
6	6 8 3 1 4 7
output.txt	
1	3

Результат работы кода на минимальных и максимальных значениях:??

input.txt	
1	1000
2	999 998 997 996 995 994 993 992 991 990
3	1000
4	999 998 997 996 995 994 993 992 991 990
5	1000
6	999 998 997 996 995 994 993 992 991 990
output.txt	
1	1000

input.txt	
1	2
2	1 2
3	2
4	2 1
5	2
6	1 3
output.txt	
1	1

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.078375 sec	17.244140625 KB
Пример из задачи	0.078375 sec	17.244140625 KB
Верхняя граница диапазона значений входных данных из текста задачи	0.078375 sec	17.244140625 KB

Вывод по задаче:

В задаче я реализовала алгоритм поиска длины наибольшей общей подпоследовательности для трёх последовательностей.

Вывод:

В работе я вспомнила принципы динамического программирования и их применение к задачам про общие подпоследовательности.

Список использованных источников

1. Викиконспекты [Электронный ресурс] (дата обращения: 23.12.2022). — URL: https://neerc.ifmo.ru/wiki/index.php?title=%D0%97%D0%B0%D0%B4%D0%B0%D1%87%D0%B0_%D0%BE_%D0%BD%D0%B0%D0%B8%D0%B1%D0%BE%D0%BB%D1%8C%D1%88%D0%B5%D0%B9_%D0%BE%D0%B1%D1%89%D0%B5%D0%B9_%D0%BF%D0%BE%D0%B4%D0%BF%D0%BE%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D0%B8.