Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский институт ИТМО»

Факультет МР и П

Алгоритмы и структуры данных

Лабораторная работа №7.

«Динамическое программирование No1»

Вариант «9»

Выполнил студент:

Розметов Джалолиддин

Группа № D3210

Преподаватель: Артамонова Валерия Евгеньевна

г. Санкт-Петербург

2024

## Оглавление



# Задание 1

Дана последовательность, требуется найти ее наибольшую возрастающую под последовательность.

• Формат ввода / входного файла (input.txt). В первой строке входных данных задано целое число n – длина последовательности (1 ≤ n ≤ 300000). Во второй строке задается сама последовательность. Числа разделяются пробелом. Элементы последовательности – целые числа, не превосходящие по модулю $10^9$.

– Подзадача 1 (полегче). n ≤ 5000.

– Общая подзадача. n ≤ 300000.

• Формат вывода / выходного файла (output.txt). В первой строке выведите длину наибольшей возрастающей подпоследовательности, а во второй строке выведите через пробел саму наибольшую возрастающаю подпоследовательность данной последовательности. Если ответов несколько – выведите любой.

• Ограничение по времени. 2 сек.

• Ограничение по памяти. 256 мб.

• Пример:

| input.txt | output.txt |
|---|---|
| 6<br>3 29 5 5 28 6 | 3<br>3 5 28 |

## Код

```
def longest_increasing_subsequence(n, sequence):
    dp = [1] * n
        prev = [-1] * n

    for i in range(1, n):
        for j in range(i):
            if sequence[j] < sequence[i] and dp[j] + 1 > dp[i]:
                dp[i] = dp[j] + 1
                prev[i] = j
```

```
    max_length = max(dp)

    max_index = dp.index(max_length)


    lis = []
    while max_index != -1:
        lis.append(sequence[max_index])
        max_index = prev[max_index]


    lis.reverse()

    return max_length, lis

def main():
    with open('input.txt', 'r') as f:
        n = int(f.readline().strip())
        sequence = list(map(int, f.readline().strip().split()))

    length, lis = longest_increasing_subsequence(n, sequence)

    with open('output.txt', 'w') as f:
        f.write(f"{length}\n")
        f.write(" ".join(map(str, lis)))

if __name__ == "__main__":
    main()
```

Вводимые данные:
6
3 29 5 5 28 6
Вывод данных:
3
3 5 28

Описание кода:

1. Функция longest_increasing_subsequence: Определяет и возвращает длину наибольшей возрастающей подпоследовательности (НВП) и элементы этой подпоследовательности в заданном списке чисел.
2. Функция main: Читает данные из файла input.txt, содержащего список чисел. Вызывает longest_increasing_subsequence для нахождения НВП. Записывает длину и элементы НВП в файл output.txt.

Описание проведенных тестов.
Тестирование кода включает проверки на корректность алгоритма вычисления наибольшей возрастающей подпоследовательности для различных типов последовательностей и проверку правильности чтения из файла и записи результатов в файл.

Вывод по работе кода:
Код успешно находит наибольшую возрастающую подпоследовательность чисел с помощью динамического программирования и осуществляет ввод-вывод данных из файла, обеспечивая удобство использования.

## Задание 2

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.

• Формат ввода / входного файла (input.txt). Первая строка входного файла определяет шаблон . Вторая строка S состоит только из символов алфавита. Ее необходимо проверить на соответствие шаблону. Длины обеих строк не превосходят 10 000. Строки могут быть пустыми – будьте внимательны!

• Формат вывода / выходного файла (output.txt). Если данная строка под-

ходит под шаблон, выведите YES. Иначе выведите NO.

• Ограничение по времени. 2 сек.

• Ограничение по памяти. 256 мб.

• Пример:

| input.txt | output.txt | input.txt | output.txt |
|---|---|---|---|
| k?t*n<br>kitten | YES | k?t?n<br>kitten | NO |

### Код

```
import re

def pattern_matching(pattern, s):
    regex = pattern.replace(".", r"\.").replace("?", ".").replace("*", ".*")
    regex = "^" + regex + "$"

    if re.match(regex, s):
```

```
        return "YES"
    else:
        return "NO"

def main():
    with open('input.txt', 'r') as f:
        pattern = f.readline().strip()
        s = f.readline().strip()

    result = pattern_matching(pattern, s)

    with open('output.txt', 'w') as f:
        f.write(result)

if __name__ == "__main__":
    main()
```

Вводимые данные:
k?t?n
kitten

Вывод кода:
NO

Описание код:
Преобразование шаблона:

- Замена символов . на \. для точного совпадения.
- Замена символов ? на . для совпадения с одним символом.
- Замена символов * на .* для совпадения с нулем или более повторениями любых символов.

Сопоставление с помощью регулярных выражений:

- Сопоставление строки с регулярным выражением, созданным на основе шаблона.
- Возвращение "YES", если совпадение найдено, и "NO" в противном случае.

Основная функция main:

- Чтение входных данных из файла input.txt.
- Вызов функции pattern_matching с полученными данными.
- Запись результата в файл output.txt.

Описание проведенных измерений:

В тестах проверяется, что функция корректно обрабатывает различные случаи сопоставления шаблона со строкой, включая точные совпадения, частичные совпадения и случаи без совпадений. Также проверяется, что функция правильно читает входные данные из файла и записывает результаты в файл вывода.

Выводы по работе:

Код эффективно решает задачу сопоставления шаблона с заданной строкой с помощью регулярных выражений. Он обеспечивает правильную обработку различных типов шаблонов и строк, а также корректное чтение и запись данных из файлов.

## Вывод

Лабораторная работа позволила изучить и применить методику динамического программирования для решения задачи нахождения наибольшей возрастающей подпоследовательности. Мы разработали функцию, эффективно находящую длину НВП и саму подпоследовательность, а также реализовали ввод-вывод данных из файлов. Этот опыт помог лучше понять принципы и преимущества динамического программирования в решении задач оптимизации.