Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ПРАКТИЧЕСКОЙ РАБОТЕ №11
дисциплины «Алгоритмизация»

Выполнил:
Говоров Егор Юрьевич
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и вычислительная техника»,
направленность (профиль) «Программное обеспечение средств вычислительной техники и автоматизирование систем», очная форма обучения

_____
(подпись)

Руководитель практики:
Воронкин Р.А., канд. техн. наук, доцент, доцент кафедры инфокоммуникаций

_____
(подпись)

Отчет защищен с оценкой_____Дата защиты_____

Ставрополь, 2023 г.

Тема: Динамическое программирование

Порядок выполнения работы:

1. Нахождение числа фиббоначи:

Динамическое программирование назад:

```
def fibonacci(n, calculation_method=0):
    def fibonacci_td(n, memo={}):
        if n in memo:
            return memo[n]
        if n <= 1:
            return n
        memo[n] = fibonacci_td(n - 1) + fibonacci_td(n - 2)
        return memo[n]
```

Рисунок 1 – Фрагмент кода файла (fib.py)

Динамическое программирование вперед:

```
def fibonacci_bu(n):
    if n <= 1:
        return n
    dp = [0] * (n + 1)
    dp[1] = 1
    for i in range(2, n + 1):
        dp[i] = dp[i - 1] + dp[i - 2]
    return dp[n]
```

Рисунок 2 – Фрагмент кода файла (fib.py)

Уменьшение количества потребления памяти:

```
def fibonacci_bu_improved(n):
    if n <= 1:
        return n
    a, b = 0, 1
    for _ in range(2, n + 1):
        a, b = b, a + b
    return b
```

Рисунок 3 – Фрагмент кода файла (fib.py)

2. Нахождение в списка НВП и самой НВП:

Поиск длины наибольшей возрастающей подпоследовательности

```
def list_bottom_up(a):
    n = len(a)
    D = [1] * n

    for i in range(1, n):
        for j in range(i):
            if a[i] > a[j] and D[i] < D[j] + 1:
                D[i] = D[j] + 1

    return max(D)
```

Рисунок 4 – Фрагмент кода файла (list.py)

Восстановление НВП с помощью списка prev:

```
def using_prev(prev, m_index):
    result = []
    while m_index is not None:
        result.insert(0, m_index)
        m_index = prev[m_index]
    return result
```

Рисунок 5 – Фрагмент кода файла (list.py)

Восстановление НВП без помощи списка prev:

```
def without_prev(d, ans, m_index):
    result = []
    while d[m_index] != 1:
        result.insert(0, ans[m_index])
        m_index -= 1
    result.insert(0, ans[m_index])
    return result
```

Рисунок 6 – Фрагмент кода файла (list.py)

Поиск длины и самой НВП:

```python
def list_bottom_up_2(a):
    n = len(a)
    D = [1] * n
    prev = [None] * n
    m_index = 0

    for i in range(1, n):
        for j in range(i):
            if a[i] > a[j] and D[i] < D[j] + 1:
                D[i] = D[j] + 1
                prev[i] = j
                if D[i] > D[m_index]:
                    m_index = i

    result_using_prev = using_prev(prev, m_index)
    result_without_prev = without_prev(D, a, m_index)

    return D[m_index], result_using_prev, result_without_prev
```

Рисунок 7 – Фрагмент кода файла (list.py)

3. Поиск максимальной стоимости предметов в рюкзаке

Предметы могут повторяться:

```python
def knapsack_with_reps(W, weights, values):
    n = len(weights)
    cell = [0] * (W + 1)

    for w in range(1, W + 1):
        for i in range(n):
            if weights[i] <= w:
                cell[w] = max(cell[w], cell[w - weights[i]] + values[i])

    return cell[W]
```

Рисунок 8 – Фрагмент кода файла (knapsack.py)

Предметы не могут повторяться:

```python
def knapsack_without_reps(W, weights, values):
    n = len(weights)
    cell = [[0] * (W + 1) for _ in range(n + 1)]

    for i in range(1, n + 1):
        for w in range(1, W + 1):
            if weights[i - 1] <= w:
                cell[i][w] = max(cell[i - 1][w], cell[i - 1][w - weights[i - 1]] + values[i - 1])
            else:
                cell[i][w] = cell[i - 1][w]

    w, i = W, n
    selected_items = [0] * n
    while i > 0 and w > 0:
        if cell[i][w] != cell[i - 1][w]:
            selected_items[i - 1] = 1
            w -= weights[i - 1]
        i -= 1

    return cell[n][W], selected_items
```

Рисунок 9 – Фрагмент кода файла (knapsack.py)

Вывод: в ходе выполнения работы мы познакомились с алгоритмами динамического программирования такими как нахождение числа Фиббоначи, нахождение НВП и алгоритм расчёта максимальной стоимости предметов в рюкзаке.