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:
Степанов Леонид Викторович
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_td(n):
    """
    Динамическое программирование назад.
    """
    if n <= 1:
        f[n] = n
    else:
        f[n] = fibonacci_td(n - 1) + fibonacci_td(n-2)
    return f[n]
```

Рисунок 1 – Фрагмент кода файла (fib.py)

Динамическое программирование вперед:

```
def fibonacci_bu(n):
    """
    Динамическое программирование вперед.
    """
    f = [-1] * (n+1)
    f[0] = 0
    f[1] = 1

    for i in range(2, n + 1):
        f[i] = f[i - 1] + f[i - 2]
    return f[n]
```

Рисунок 2 – Фрагмент кода файла (fib.py)

Уменьшение количества потребления памяти:

```
def fibonacci_bu_imroved(n):
    """
    Уменьшенным памяти.
    """
    if n <= 1:
        return n

    prev, curr = 0, 1

    for _ in range(n - 1):
        prev, curr = curr, prev + curr
    return curr
```

Рисунок 3 – Фрагмент кода файла (fib.py)

2. Нахождение в списка НВП и самой НВП:

Поиск длины наибольшей возрастающей подпоследовательности

```python
def list_bottom_up(a):
    """
    Поиск длины наибольшей возрастающей подпоследовательности.
    """
    n = len(a)
    D = []

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

    ans = max(D)

    return ans
```

Рисунок 4 – Фрагмент кода файла (list.py)

Восстановление НВП с помощью списка prev:

```python
def using_prev(prev, m_index):
    """
    Восстановление НВП с помощью списка prev
    """
    l = []
    while True:
        l.append(m_index)
        if prev[m_index] == -1:
            break
        m_index = prev[m_index]

    l.reverse()
    return l
```

Рисунок 5 – Фрагмент кода файла (list.py)

Восстановление НВП без помощи списка prev:

```python
def without_prev(d, ans, m_index):
    """
    Восстановление НВП без помощи списка prev
    """
    l = []
    while True:
        l.append(m_index)
        if ans == 1:
            break
        ans -= 1
        while True:
            m_index -= 1
            if d[m_index] == ans and a[m_index] < a[l[-1]]:
                break
    l.reverse()

    return l
```

Рисунок 6 – Фрагмент кода файла (list.py)

Поиск длины и самой НВП:

```python
def list_bottom_up_2(a):
    """
    Поиск длины и самой НВП.
    """
    n = len(a)
    d, prev = [], []
    for i in range(n):
        d.append(1)
        prev.append(-1)
        for j in range(i):
            if a[j] < a[i] and d[j] + 1 > d[i]:
                d[i] = d[j]+1
                prev[i] = j

    ans, max_index = 0, 0
    for i, item in enumerate(d):
        if ans < item:
            ans, max_index = item, i

    list_using_prev = using_prev(prev, max_index)
    list_without_prev = without_prev(d, ans, max_index)

    return ans, (list_using_prev, list_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, weight, cell):
        """
        Поиск максимальной стоимости предметов в рюкзаке.
        Предметы могут повторяться.
        """
        d = [0] * (W+1)
        for w in range(1, W+1):
            for weight_i, cell_i in zip(weight, cell):
                if weight_i <= w:
                    d[w] = max(d[w], d[w - weight_i] + cell_i)
        return d[W]
```

Рисунок 8 – Фрагмент кода файла (knapsack.py)

Предметы не могут повторяться:

```python
def knapsack_without_reps(W, weight, cell):
    """
    Поиск максимальной стоимости предметов в рюкзаке.
    Предметы не могут повторяться.
    """
    def restore(d, weight_rev, cell_rev):
        """
        Восстановление предметов в рюкзаке.
        """
        solution = []
        w = W
        elem = len(weight_rev)
        for weight_i, cell_i in zip(weight_rev, cell_rev):
            if d[w][elem] == d[w - weight_i][elem-1] + cell_i:
                solution.append(1)
                w -= weight_i
            else:
                solution.append(0)
            elem -= 1
        solution.reverse()
        return solution

    d = [[0] for _ in range(W+1)]
    d[0] = [0] * (len(weight) + 1)
    for weight_i, cell_i in zip(weight, cell):
        for w in range(1, W+1):
            d[w].append(d[w][-1])
            if weight_i <= w:
                d[w][-1] = max(d[w][-1], d[w - weight_i][-2] + cell_i)

    solution = restore(d, weight[::-1], cell[::-1])

    return d[W][-1], solution
```

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.