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 ЛАБОРАТОРНОЙ РАБОТЕ №11
дисциплины «Алгоритмизация»

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

_____
(подпись)

Руководитель практики:
Воронкин Роман Александрович

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023 г.

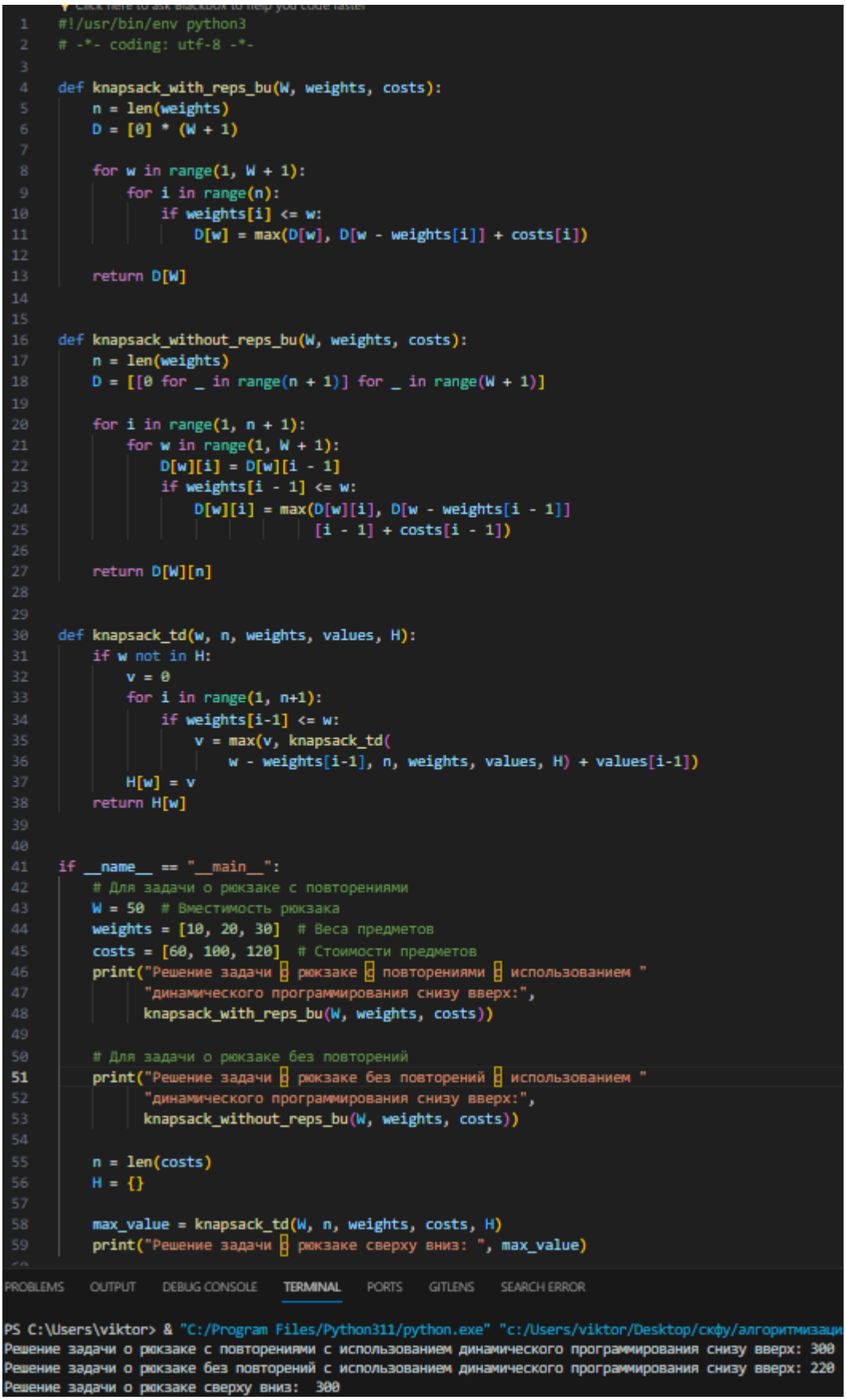
Порядок выполнения работы:

Написал программы вычисления числа фиббоначи, нахождения длины наибольшей возрастающей подпоследовательности и решения задачи о рюкзаке:

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

def fib_recursive(n):
    if n <= 1:
        return n
    else:
        return fib_recursive(n - 1) + fib_recursive(n - 2)


def fib_td(n, F):
    if F[n] == -1:
        if n <= 1:
            F[n] = n
        else:
            F[n] = fib_td(n - 1, F) + fib_td(n - 2, F)
    return F[n]


def fib_bu(n):
    F = [0, 1] + [0] * (n - 1)
    for i in range(2, n + 1):
        F[i] = F[i - 1] + F[i - 2]
    return F[n]


def fib_bu_improved(n):
    if n <= 1:
        return n
    prev, curr = 0, 1
    for _ in range(n - 1):
        next_val = prev + curr
        prev, curr = curr, next_val
    return curr


if __name__ == "__main__":
    # Для вычисления числа фиббоначи
    n = 10
    F = [-1] * (n + 1)
    print("Вычисление числа Фибоначчи рекурсивное:", fib_recursive(n))
    print("Вычисление числа Фибоначчи с использованием"
          " динамического программирования сверху вниз:", fib_td(n, F))
    print("Вычисление числа Фибоначчи с использованием"
          " динамического программирования снизу вверх:", fib_bu(n))
    print("Вычисление числа Фибоначчи с использованием улучшенной версии"
          " динамического программирования снизу вверх:", fib_bu_improved(n))
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS  SEARCH ERROR

PS C:\Users\viktor> & "C:/Program Files/Python311/python.exe" "c:/Users/viktor/Desktop/скфу/алгоритмизация/Al
Вычисление числа Фибоначчи рекурсивное: 55
Вычисление числа Фибоначчи с использованием динамического программирования сверху вниз: 55
Вычисление числа Фибоначчи с использованием динамического программирования снизу вверх: 55
Вычисление числа Фибоначчи с использованием улучшенной версии динамического программирования снизу вверх: 55
```

Рисунок 1. Код вычисления числа фиббоначи и его выполнение



```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

def knapsack_with_reps_bu(W, weights, costs):
    n = len(weights)
    D = [0] * (W + 1)

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

    return D[W]


def knapsack_without_reps_bu(W, weights, costs):
    n = len(weights)
    D = [[0 for _ in range(n + 1)] for _ in range(W + 1)]

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

    return D[W][n]


def knapsack_td(w, n, weights, values, H):
    if w not in H:
        v = 0
        for i in range(1, n+1):
            if weights[i-1] <= w:
                v = max(v, knapsack_td(
                    w - weights[i-1], n, weights, values, H) + values[i-1])
        H[w] = v
    return H[w]


if __name__ == "__main__":
    # Для задачи о рюкзаке с повторениями
    W = 50  # Вместимость рюкзака
    weights = [10, 20, 30]  # Веса предметов
    costs = [60, 100, 120]  # Стоимости предметов
    print("Решение задачи о рюкзаке с повторениями с использованием "
          "динамического программирования снизу вверх:",
          knapsack_with_reps_bu(W, weights, costs))

    # Для задачи о рюкзаке без повторений
    print("Решение задачи о рюкзаке без повторений с использованием "
          "динамического программирования снизу вверх:",
          knapsack_without_reps_bu(W, weights, costs))

    n = len(costs)
    H = {}

    max_value = knapsack_td(W, n, weights, costs, H)
    print("Решение задачи о рюкзаке сверху вниз: ", max_value)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS  SEARCH ERROR

PS C:\Users\viktor> & "C:/Program Files/Python311/python.exe" "c:/Users/viktor/Desktop/скфу/алгоритмизаци
Решение задачи о рюкзаке с повторениями с использованием динамического программирования снизу вверх: 300
Решение задачи о рюкзаке без повторений с использованием динамического программирования снизу вверх: 220
Решение задачи о рюкзаке сверху вниз:  300
```

Рисунок 2. Код нахождения длины наибольшей возрастающей подпоследовательности и его выполнение

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

def lis_bottom_up(A):
    n = len(A)
    D = [1] * n
    for i in range(n):
        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


def lis_bottom_up_2(A):
    n = len(A)
    D = [1] * n
    prev = [-1] * n
    for i in range(n):
        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(D)
    return ans, prev, D


def restore_answer(D, prev, ans):
    L = [0] * ans
    k = 1
    n = len(D)
    for i in range(2, n):
        if D[i] > D[k]:
            k = i
    j = ans
    while k > 0:
        L[j-1] = k
        j -= 1
        k = prev[k]
    return L


if __name__ == "__main__":
    # Для вычисления наибольшей возрастающей подпоследовательности
    A = [10, 22, 9, 33, 21, 50, 41, 60, 80]
    print("Вычисление наибольшей возрастающей подпоследовательности"
          " с использованием динамического программирования снизу вверх:",
          lis_bottom_up(A))
    ans, prev, D = lis_bottom_up_2(A)
    print("Вычисление наибольшей возрастающей подпоследовательности с "
          "использованием динамического программирования снизу вверх (версия 2):",
          ans, "\nвосстановленный ответ:",
          restore_answer(D, prev, ans))
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS  SEARCH ERROR

PS C:\Users\viktor> & "C:/Program Files/Python311/python.exe" "c:/Users/viktor/Desktop/скфу/алгоритмизация/AlgLab11/Main lis.py"
Вычисление наибольшей возрастающей подпоследовательности с использованием динамического программирования снизу вверх: 6
Вычисление наибольшей возрастающей подпоследовательности с использованием динамического программирования снизу вверх (версия 2): 6
Восстановленный ответ: [0, 1, 3, 5, 7, 8]
```

Рисунок 3. Код решения задачи о рюкзаке и его выполнение

Вывод: в результате выполнения лабораторной работы были изучены алгоритмы вычисления числа фиббоначи, нахождения длины наибольшей возрастающей подпоследовательности и решения задачи о рюкзаке.