

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:
Кенесбаев Хилол Куат улы
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 Р.А., доцент кафедры
инфокоммуникаций

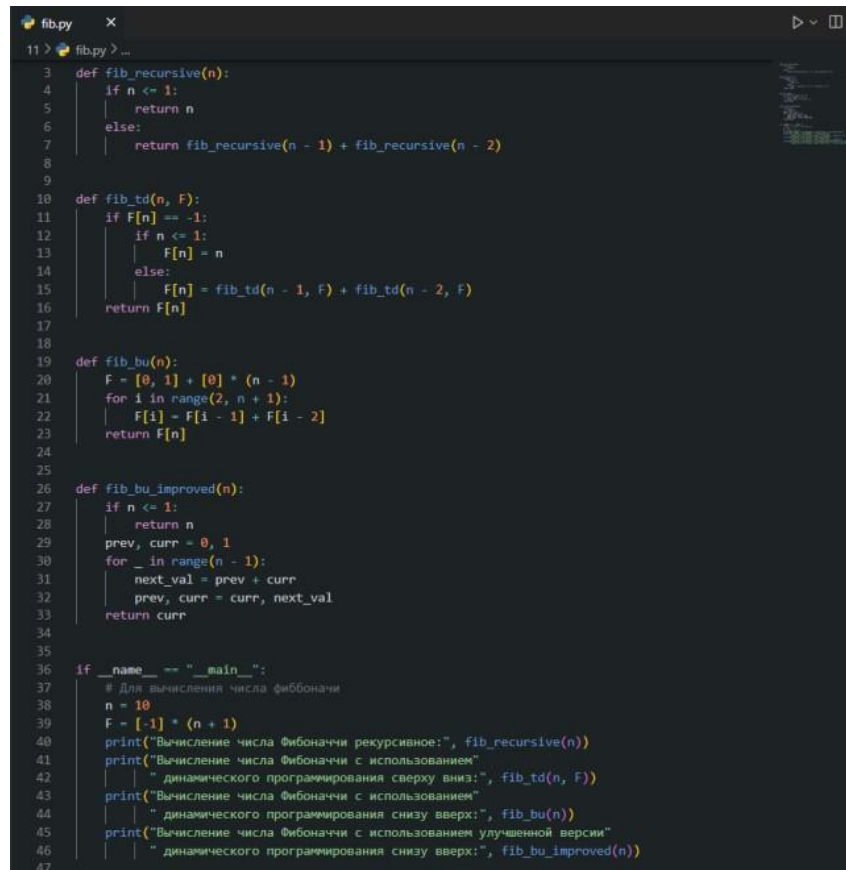
(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

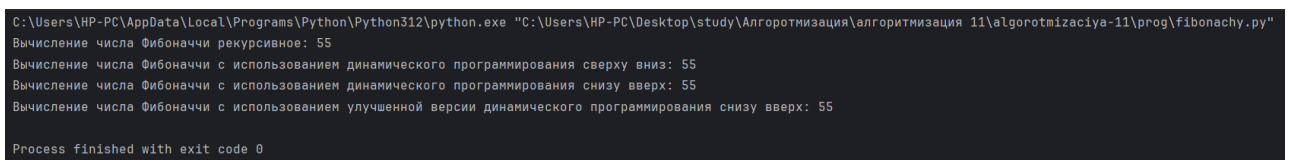
Порядок выполнения работы:

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



```
11 > fib.py > ...
3 def fib_recursive(n):
4     if n <= 1:
5         return n
6     else:
7         return fib_recursive(n - 1) + fib_recursive(n - 2)
8
9
10 def fib_td(n, F):
11     if F[n] == -1:
12         if n <= 1:
13             F[n] = n
14         else:
15             F[n] = fib_td(n - 1, F) + fib_td(n - 2, F)
16     return F[n]
17
18
19 def fib_bu(n):
20     F = [0, 1] + [0] * (n - 1)
21     for i in range(2, n + 1):
22         F[i] = F[i - 1] + F[i - 2]
23     return F[n]
24
25
26 def fib_bu_improved(n):
27     if n <= 1:
28         return n
29     prev, curr = 0, 1
30     for _ in range(n - 1):
31         next_val = prev + curr
32         prev, curr = curr, next_val
33     return curr
34
35
36 if __name__ == "__main__":
37     # Для вычисления числа фибоначи
38     n = 10
39     F = [-1] * (n + 1)
40     print("Вычисление числа Фибоначи рекурсивное:", fib_recursive(n))
41     print("Вычисление числа Фибоначи с использованием")
42     | " динамического программирования сверху вниз:", fib_td(n, F)
43     print("Вычисление числа Фибоначи с использованием")
44     | " динамического программирования снизу вверх:", fib_bu(n))
45     print("Вычисление числа Фибоначи с использованием улучшенной версии")
46     | " динамического программирования снизу вверх:", fib_bu_improved(n))
47
```

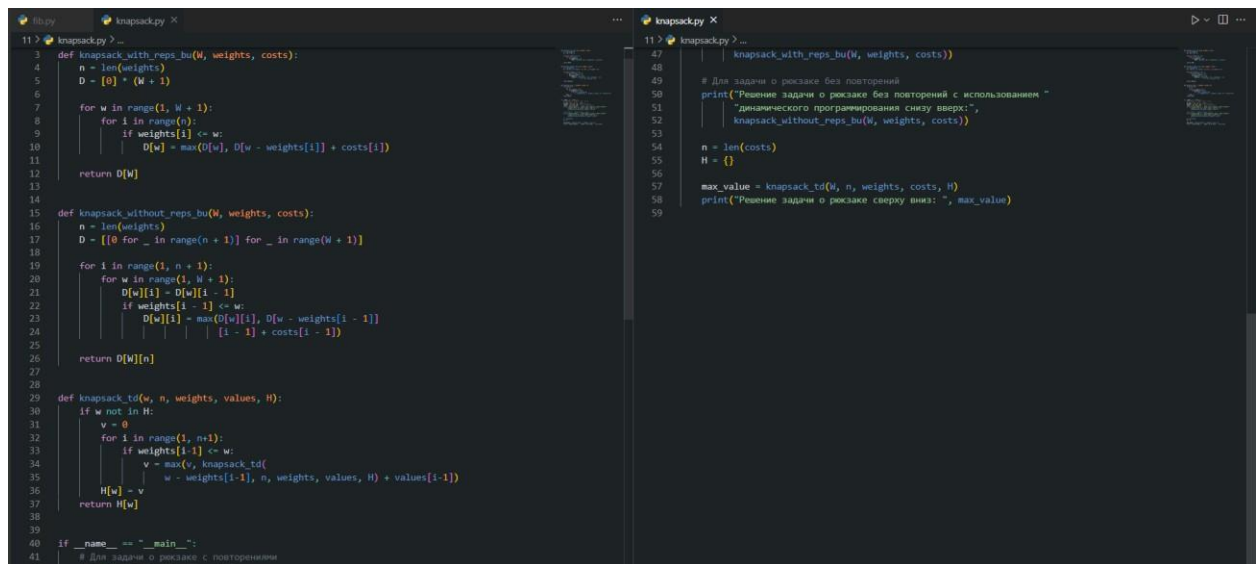
Рисунок 1 Код вычисления числа фиббоначи



```
C:\Users\HP-PC\AppData\Local\Programs\Python\Python312\python.exe "C:\Users\HP-PC\Desktop\study\Алгоритмизация\алгоритмизация 11\algoritimizaciya-11\prog\fibonachy.py"
Вычисление числа Фибоначи рекурсивное: 55
Вычисление числа Фибоначи с использованием динамического программирования сверху вниз: 55
Вычисление числа Фибоначи с использованием динамического программирования снизу вверх: 55
Вычисление числа Фибоначи с использованием улучшенной версии динамического программирования снизу вверх: 55

Process finished with exit code 0
```

Рисунок 2 Результат работы программы



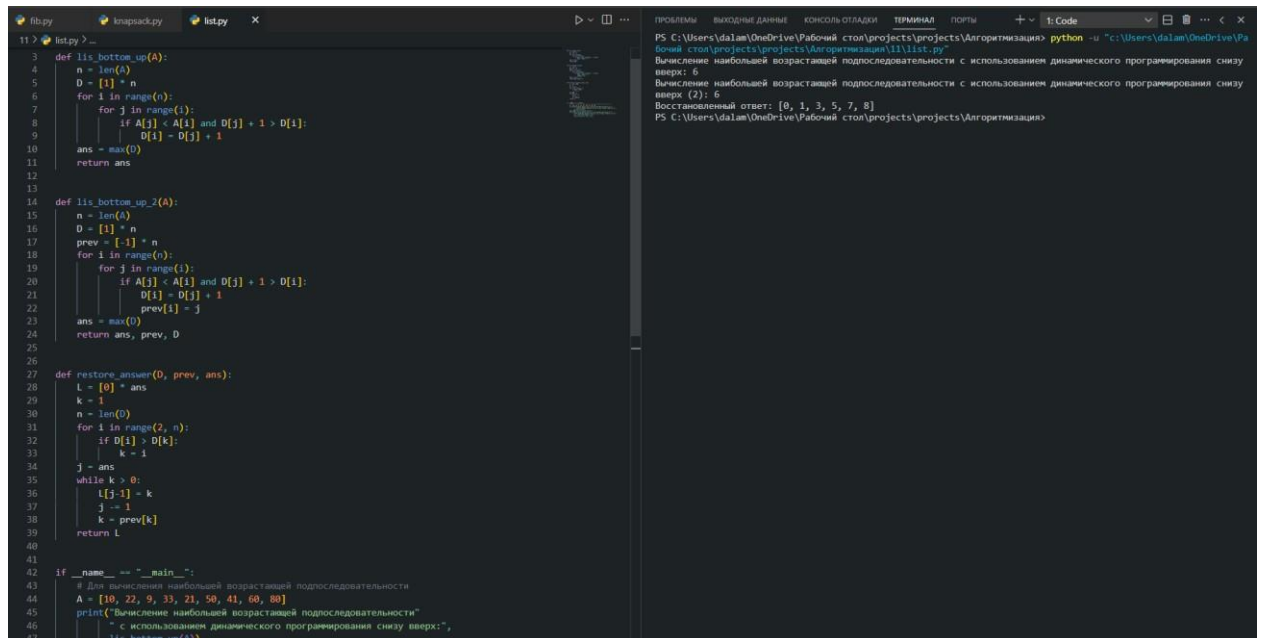
```
11 > knapsack.py > ...
3 def knapsack_with_reps(W, weights, costs):
4     n = len(weights)
5     D = [0] * (W + 1)
6
7     for w in range(1, W + 1):
8         for i in range(1, n + 1):
9             if weights[i] <= w:
10                 D[w] = max(D[w], D[w - weights[i]] + costs[i])
11
12     return D[W]
13
14
15 def knapsack_without_reps_bu(W, weights, costs):
16     n = len(weights)
17     D = [[0 for _ in range(n + 1)] for _ in range(W + 1)]
18
19     for i in range(1, n + 1):
20         for w in range(1, W + 1):
21             D[w][i] = D[w][i - 1]
22             if weights[i] <= w:
23                 D[w][i] = max(D[w][i], D[w - weights[i]][i - 1] + costs[i - 1])
24
25     return D[W][n]
26
27
28
29 def knapsack_td(w, n, weights, values, H):
30     if w not in H:
31         v = 0
32         for i in range(1, n + 1):
33             if weights[i] <= w:
34                 v = max(v, knapsack_td(w - weights[i], n, weights, values, H) + values[i - 1])
35         H[w] = v
36     return H[w]
37
38
39
40 if __name__ == "__main__":
41     # Для задачи о рюкзаке с повторениями
42     W = 50
43     # Для задачи о рюкзаке без повторений
44     W = 50
45     n = len(weights)
46     n = len(costs)
47     H = {}
48     max_value = knapsack_td(W, n, weights, costs, H)
49     print("Решение задачи о рюкзаке сверху вниз: ", max_value)
50
51     # Для задачи о рюкзаке без повторений
52     print("Решение задачи о рюкзаке без повторений с использованием")
53     | " динамического программирования снизу вверх:",
54     | knapsack_without_reps_bu(W, weights, costs))
55
56
57
58
59
```

Рисунок 3 Knapsak.py

```
C:\Users\HP-PC\AppData\Local\Programs\Python\Python312\python.exe "C:\Users\HP-PC\Desktop\study\Алгоритмизация\алгоритмизация 11\algorotmizaciya-11\prog\knapsack.py"
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

Process finished with exit code 0
```

Рисунок 4 Результат выполнения программы нахождения длины НВП



```
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)
```

PS C:\Users\dalame\OneDrive\Рабочий стол\projects\projects\Алгоритмизация> python -u "c:\Users\dalame\OneDrive\Рабочий стол\projects\projects\Алгоритмизация\11\list.py"

Вычисление наибольшей возрастающей подпоследовательности с использованием динамического программирования снизу вверх: 6

Вычисление наибольшей возрастающей подпоследовательности с использованием динамического программирования снизу вверх (2): 6

Восстановленный ответ: [0, 1, 3, 5, 7, 8]

PS C:\Users\dalame\OneDrive\Рабочий стол\projects\projects\Алгоритмизация>

Рисунок 5 Код решения задачи о рюкзаке и результат выполнения

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