

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:
Болуров Ислам Расулович
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

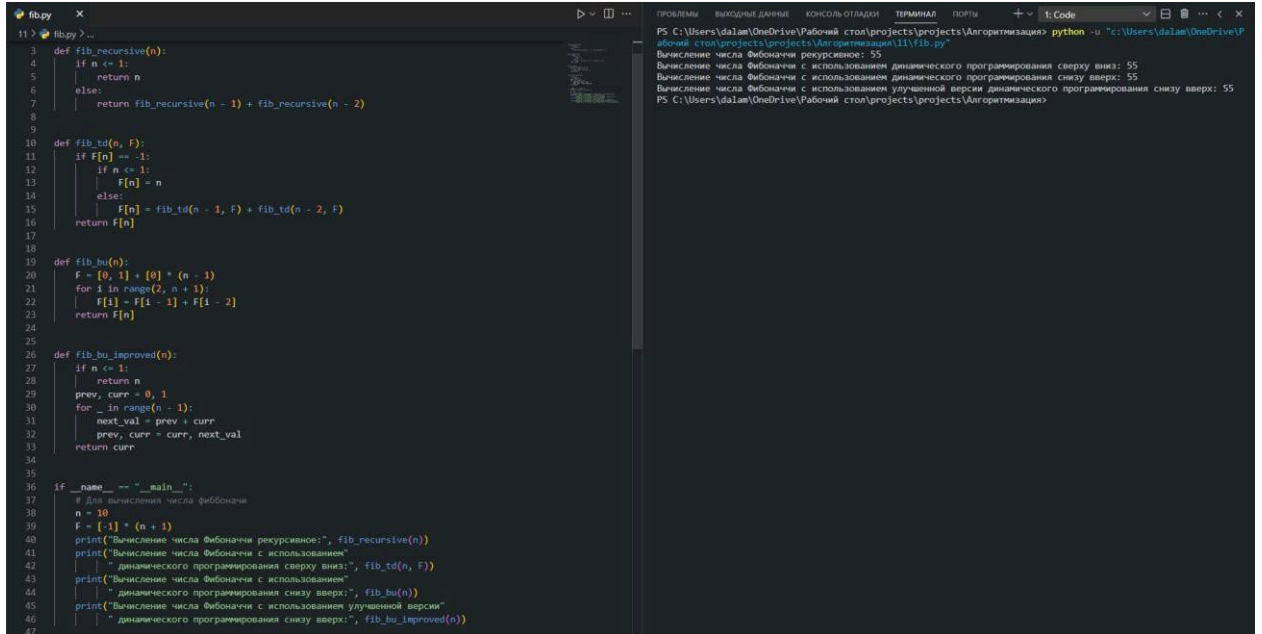
(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2024 г.

Порядок выполнения работы:

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



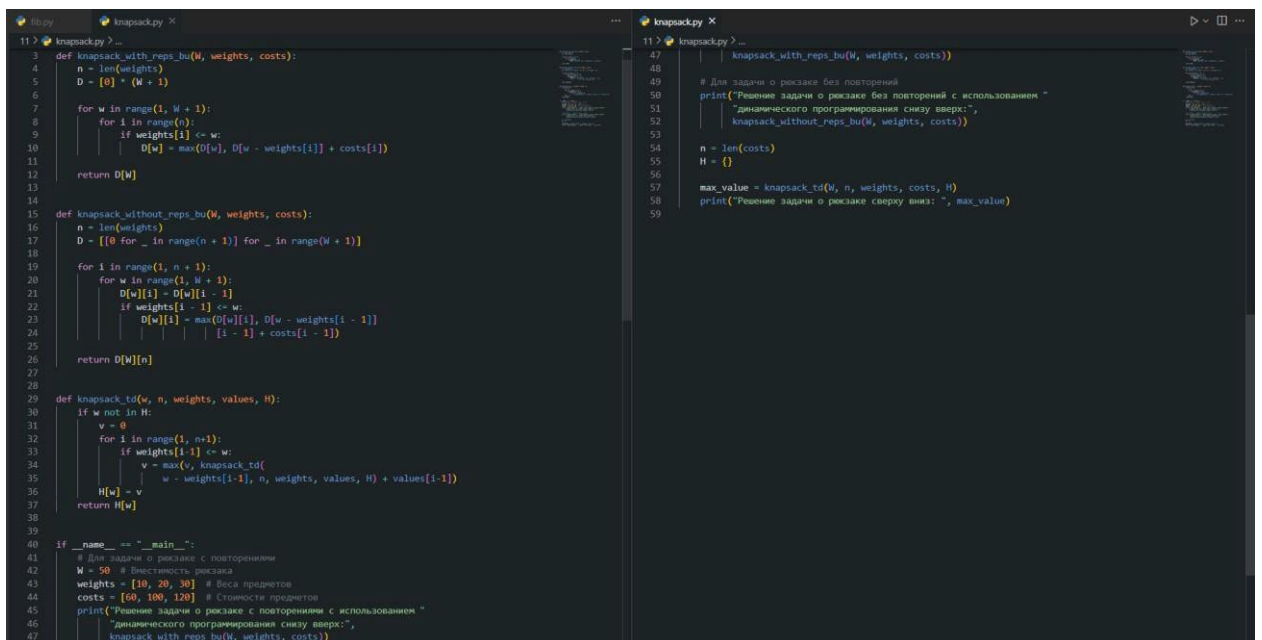
```
11 > fibpy > ...
3 def fib_recursive(n):
4     if n <= 1:
5         return n
6     else:
7         return fib_recursive(n - 1) + fib_recursive(n - 2)
8
9
10 def fib_td(n, F):
11     if F[n] == -1:
12         if n <= 1:
13             F[n] = n
14         else:
15             F[n] = fib_td(n - 1, F) + 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
```

PS C:\Users\dalaa\OneDrive\Рабочий стол\projects\projects\Алгоритмизация> python -u "C:\Users\dalaa\OneDrive\Рабочий стол\projects\projects\Алгоритмизация\11\fib.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

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

Рисунок 1. Код вычисления числа фиббоначи и результат выполнения



```
11 > knapsackpy > ...
3 def knapsack_with_reps_bu(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(n):
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 - 1] <= w:
23                 D[w][i] = max(D[w][i], D[w - weights[i - 1]][i - 1] + costs[i - 1])
24
25     return D[W][n]
26
27
28 def knapsack_td(W, n, weights, values, H):
29     if W not in H:
30         v = 0
31         for i in range(1, n + 1):
32             if weights[i - 1] <= W:
33                 v = max(v, knapsack_td(W - weights[i - 1], n, weights, values, H) + values[i - 1])
34         H[W] = v
35     return H[W]
36
37
38
39
40 if __name__ == "__main__":
41     # Для задачи о рюкзаке с повторениями
42     W = 50 # Вместимость рюкзака
43     weights = [10, 20, 30] # Веса предметов
44     costs = [60, 100, 120] # Стоимости предметов
45     print("Решение задачи о рюкзаке с повторениями с использованием "
46           "динамического программирования снизу вверх:",
47           knapsack_with_reps_bu(W, weights, costs))
48
49
50 # Для задачи о рюкзаке без повторений
51 print("Решение задачи о рюкзаке без повторений с использованием "
52       "динамического программирования снизу вверх:",
53       knapsack_without_reps_bu(W, weights, costs))
54
55 n = len(costs)
56 H = {}
57 max_value = knapsack_td(W, n, weights, costs, H)
58 print("Решение задачи о рюкзаке сверху вниз:", max_value)
59
```

Рисунок 2. Код нахождения длины наибольшей возрастающей последовательности

```

PS C:\Users\dalam\OneDrive\Рабочий стол\projects\projects\Алгоритмизация> python -u "c:\Users\dalam\OneDrive\Рабочий стол\projects\projects\Алгоритмизация\11\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
PS C:\Users\dalam\OneDrive\Рабочий стол\projects\projects\Алгоритмизация>

```

Рисунок 3. Результат выполнения программы нахождения длины НВП

The image shows a code editor with a Python script for a knapsack problem. The script defines three functions: `lis_bottom_up`, `lis_bottom_up_2`, and `restore_answer`. It then runs the `lis_bottom_up` function on a list `A = [10, 22, 9, 33, 21, 50, 41, 60, 80]` and prints the result. The terminal window on the right shows the output of the program, which includes the maximum value of the increasing subsequence (300) and the reconstructed subsequence `[0, 1, 3, 5, 7, 8]`.

```

11 > listpy > ...
3 def lis_bottom_up(A):
4     n = len(A)
5     D = [1] * n
6     for i in range(n):
7         for j in range(i):
8             if A[j] < A[i] and D[j] + 1 > D[i]:
9                 D[i] = D[j] + 1
10    ans = max(D)
11    return ans
12
13
14 def lis_bottom_up_2(A):
15     n = len(A)
16     D = [1] * n
17     prev = [-1] * n
18     for i in range(n):
19         for j in range(i):
20             if A[j] < A[i] and D[j] + 1 > D[i]:
21                 D[i] = D[j] + 1
22                 prev[i] = j
23    ans = max(D)
24    return ans, prev, D
25
26
27 def restore_answer(D, prev, ans):
28     L = [0] * ans
29     k = 1
30     n = len(D)
31     for i in range(2, n):
32         if D[i] > D[k]:
33             k = i
34     j = ans
35     while k > 0:
36         L[j-1] = k
37         j -= 1
38         k = prev[k]
39     return L
40
41
42 if __name__ == "__main__":
43     # Для вычисления наибольшей возрастающей подпоследовательности
44     A = [10, 22, 9, 33, 21, 50, 41, 60, 80]
45     print("Вычисление наибольшей возрастающей подпоследовательности"
46           " с использованием динамического программирования снизу вверх:")
47     lis_bottom_up(A)

```

ПРОБЛЕМЫ Выходные данные КОНСОЛЬ ОТЛАДКИ ТЕРМИНАЛ ПОРТЫ + Code

```

PS C:\Users\dalam\OneDrive\Рабочий стол\projects\projects\Алгоритмизация> python -u "c:\Users\dalam\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\dalam\OneDrive\Рабочий стол\projects\projects\Алгоритмизация>

```

Рисунок 4. 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.