

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:
Гайчук Дарья Дмитриевна
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

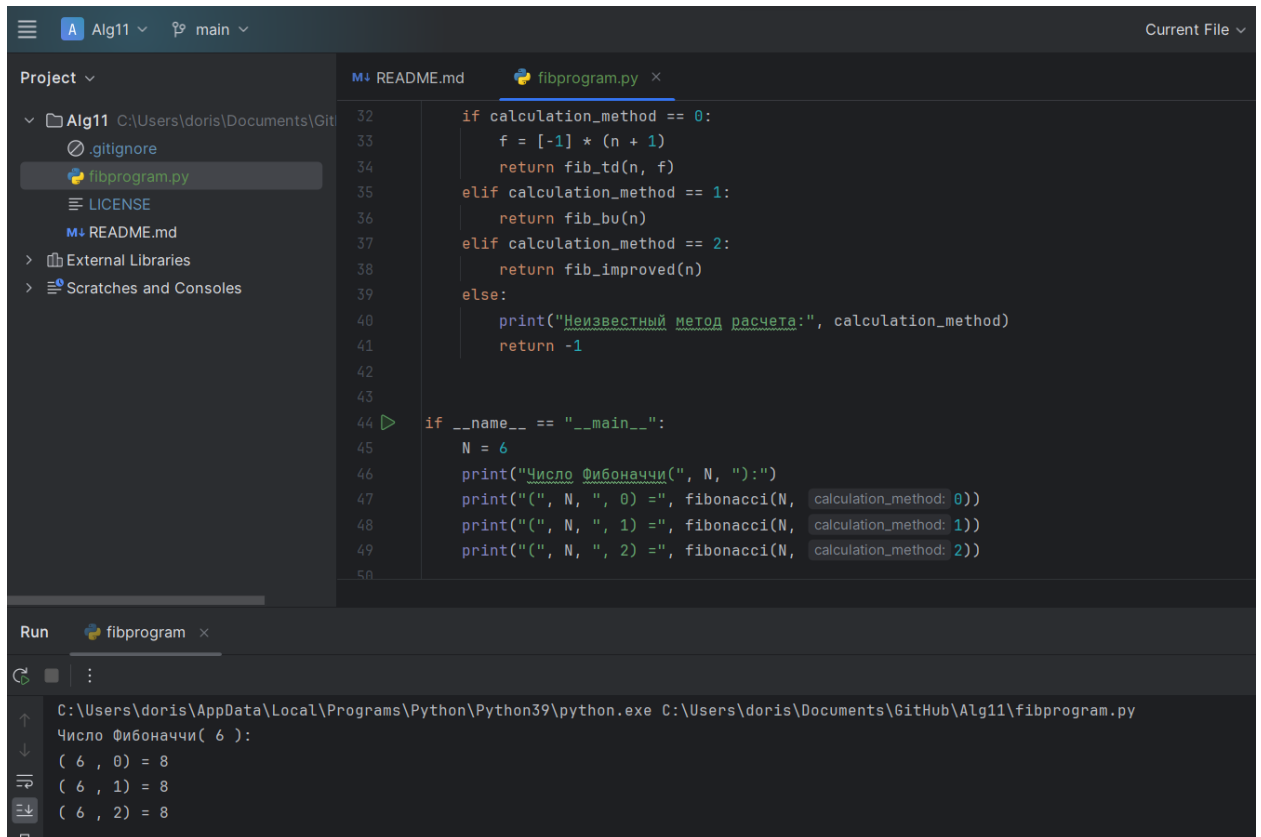
(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

Порядок выполнения работы:

Задание 1. Написала программу по примерам из видео «07 - Алгоритмы. Динамическое программирование» для нахождения числа Фибоначчи.



The screenshot shows a code editor with a project named 'Alg11'. The file 'fibprogram.py' is open, showing a function 'fibonacci' that takes 'n' and 'calculation_method' as arguments. The function uses a dictionary to select between three methods: 'fib_td' (top-down), 'fib_bu' (bottom-up), and 'fib_improved' (improved). The 'if __name__ == "__main__":' block sets 'N = 6' and prints the results for each method. The output window shows the execution of the program, displaying the Fibonacci number for N=6 and the results for each method.

```
32     if calculation_method == 0:
33         f = [-1] * (n + 1)
34         return fib_td(n, f)
35     elif calculation_method == 1:
36         return fib_bu(n)
37     elif calculation_method == 2:
38         return fib_improved(n)
39     else:
40         print("Неизвестный метод расчета:", calculation_method)
41         return -1
42
43
44 if __name__ == "__main__":
45     N = 6
46     print("Число Фибоначчи(", N, "):")
47     print("(", N, ", 0) =", fibonacci(N, calculation_method: 0))
48     print("(", N, ", 1) =", fibonacci(N, calculation_method: 1))
49     print("(", N, ", 2) =", fibonacci(N, calculation_method: 2))
50
```

Run fibprogram x

C:\Users\doris\AppData\Local\Programs\Python\Python39\python.exe C:\Users\doris\Documents\GitHub\Alg11\fibprogram.py

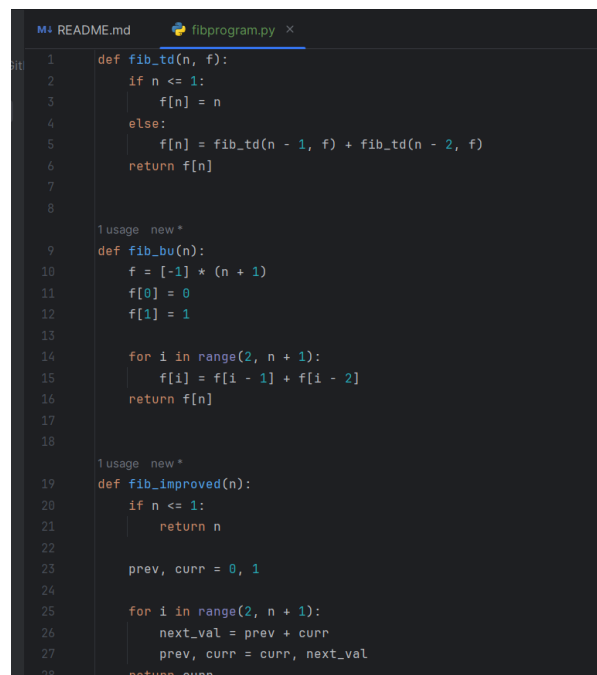
Число Фибоначчи(6):

(6 , 0) = 8

(6 , 1) = 8

(6 , 2) = 8

Рисунок 1. Программа и результат выполнения

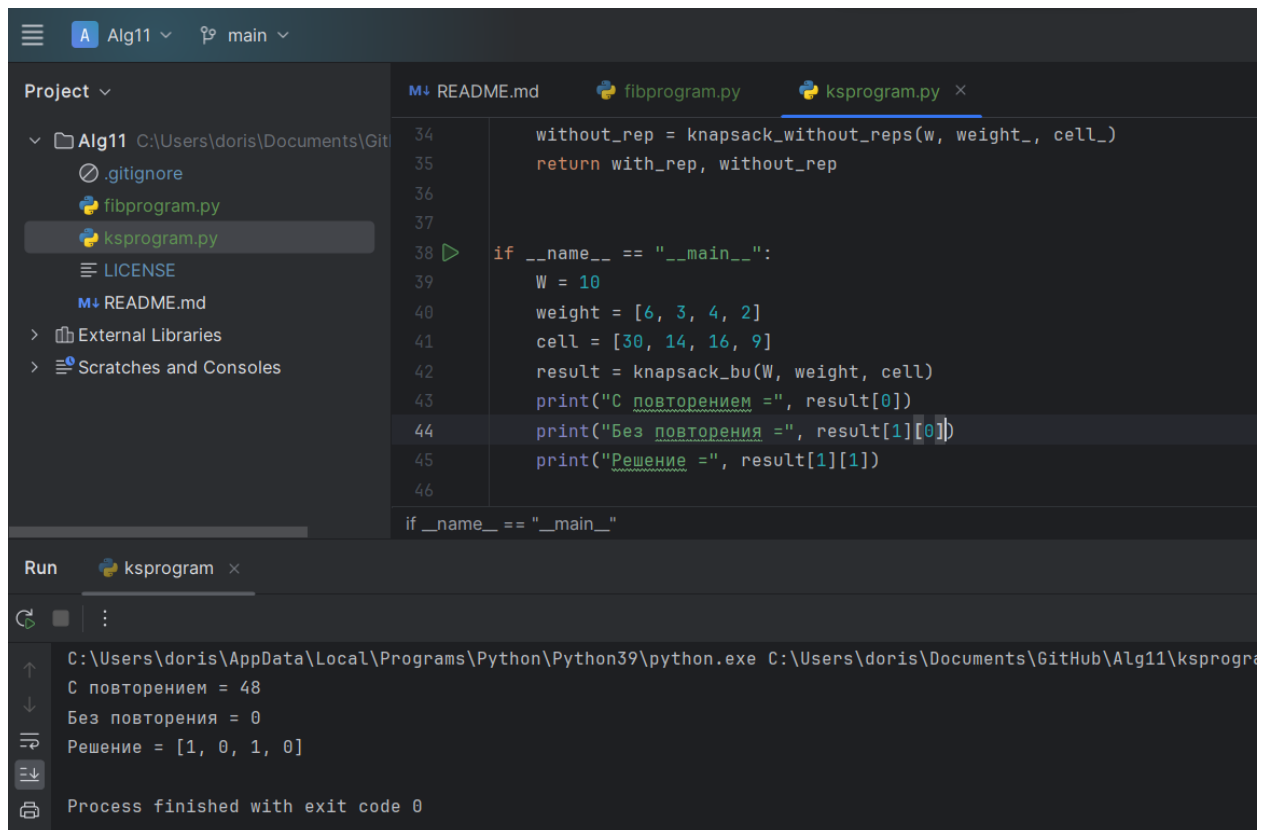


The screenshot shows the implementation of three Fibonacci algorithms in 'fibprogram.py'. The 'fib_td' function uses a dictionary for memoization. The 'fib_bu' function uses a list to store the Fibonacci sequence. The 'fib_improved' function uses a sliding window of two variables to calculate the next value.

```
1 def fib_td(n, f):
2     if n <= 1:
3         f[n] = n
4     else:
5         f[n] = fib_td(n - 1, f) + fib_td(n - 2, f)
6     return f[n]
7
8
9 usage new*
10 def fib_bu(n):
11     f = [-1] * (n + 1)
12     f[0] = 0
13     f[1] = 1
14     for i in range(2, n + 1):
15         f[i] = f[i - 1] + f[i - 2]
16     return f[n]
17
18
19 usage new*
20 def fib_improved(n):
21     if n <= 1:
22         return n
23     prev, curr = 0, 1
24     for i in range(2, n + 1):
25         next_val = prev + curr
26         prev, curr = curr, next_val
27     return curr
28
```

Рисунок 2. Реализация трех алгоритмов

Задание 2. Написала программу по примерам из видео «07 - Алгоритмы. Динамическое программирование» для поиска максимальной стоимости предметов в рюкзаке.



The screenshot shows an IDE with a project named 'Alg11'. The file explorer on the left lists files: .gitignore, fibprogram.py, ksprogram.py, LICENSE, README.md, External Libraries, and Scratches and Consoles. The main editor displays the code for 'ksprogram.py'. The code defines a 'knapsack_without_reps' function and a main block. The main block sets W = 10, weight = [6, 3, 4, 2], and cell = [30, 14, 16, 9]. It then calls 'knapsack_bu(W, weight, cell)' and prints the results. The Run console shows the output: 'С повторением = 48', 'Без повторения = 0', and 'Решение = [1, 0, 1, 0]'. The process finished with exit code 0.

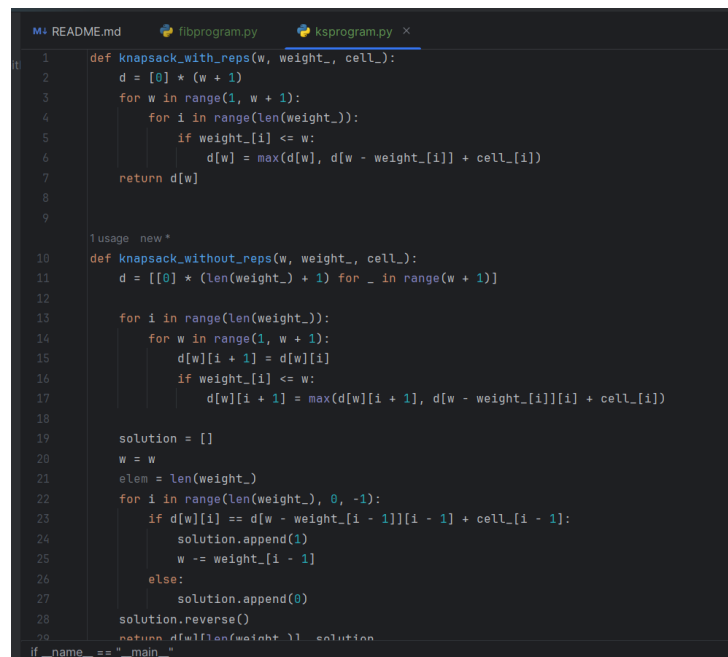
```
34 without_rep = knapsack_without_reps(w, weight_, cell_)
35 return with_rep, without_rep
36
37
38 if __name__ == "__main__":
39     W = 10
40     weight = [6, 3, 4, 2]
41     cell = [30, 14, 16, 9]
42     result = knapsack_bu(W, weight, cell)
43     print("С повторением =", result[0])
44     print("Без повторения =", result[1][0])
45     print("Решение =", result[1][1])
46
if __name__ == "__main__"
```

Run ksprogram x

C:\Users\doris\AppData\Local\Programs\Python\Python39\python.exe C:\Users\doris\Documents\GitHub\Alg11\ksprogram.py

С повторением = 48
Без повторения = 0
Решение = [1, 0, 1, 0]
Process finished with exit code 0

Рисунок 3. Программа и результат выполнения

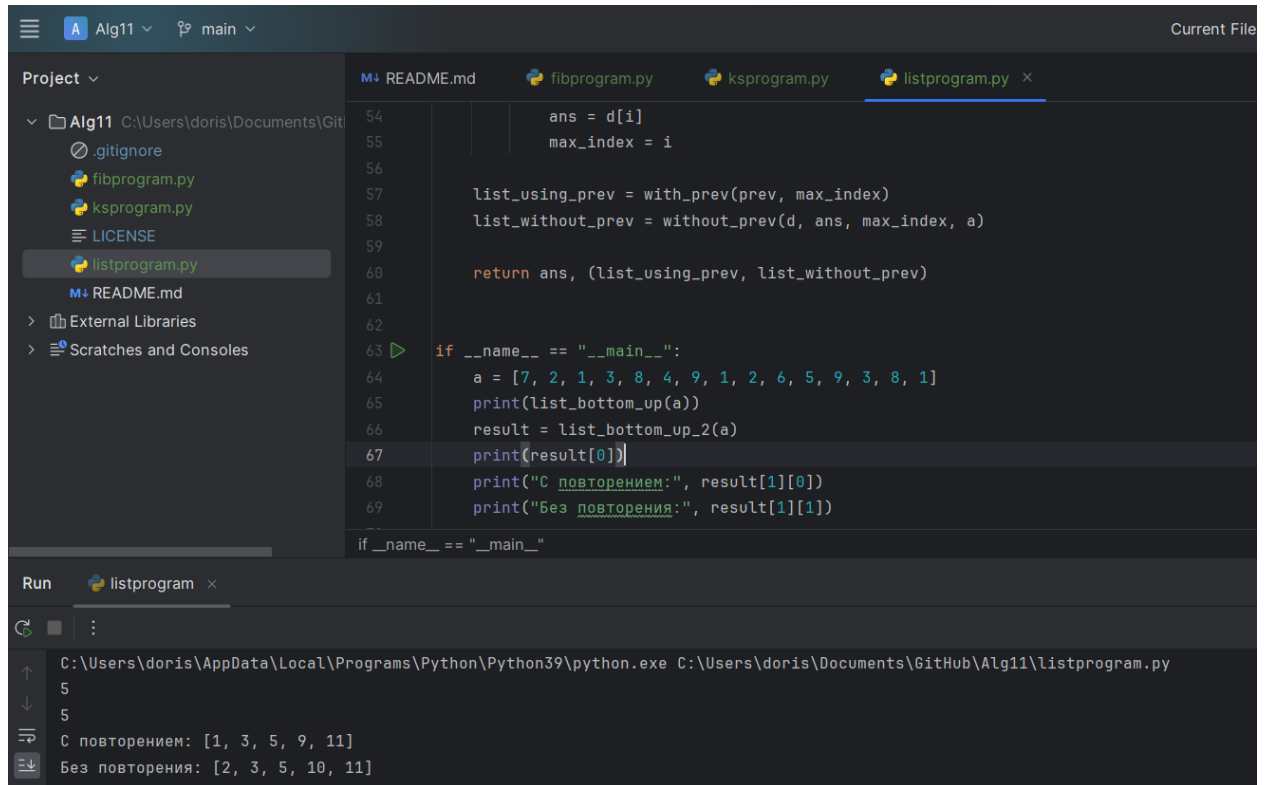


The screenshot shows the implementation of the knapsack algorithms in 'ksprogram.py'. It includes functions for 'knapsack_with_reps' and 'knapsack_without_reps', and a main block that calculates the maximum value and the corresponding solution for the knapsack problem.

```
1 def knapsack_with_reps(w, weight_, cell_):
2     d = [0] * (w + 1)
3     for w in range(1, w + 1):
4         for i in range(len(weight_)):
5             if weight_[i] <= w:
6                 d[w] = max(d[w], d[w - weight_[i]] + cell_[i])
7     return d[w]
8
9
10 usage new *
11 def knapsack_without_reps(w, weight_, cell_):
12     d = [[0] * (len(weight_) + 1) for _ in range(w + 1)]
13
14     for i in range(len(weight_)):
15         for w in range(1, w + 1):
16             d[w][i + 1] = d[w][i]
17             if weight_[i] <= w:
18                 d[w][i + 1] = max(d[w][i + 1], d[w - weight_[i]][i] + cell_[i])
19
20     solution = []
21     w = w
22     elem = len(weight_)
23     for i in range(len(weight_), 0, -1):
24         if d[w][i] == d[w - weight_[i - 1]][i - 1] + cell_[i - 1]:
25             solution.append(1)
26             w -= weight_[i - 1]
27         else:
28             solution.append(0)
29     solution.reverse()
30     return d[w][len(weight_)] , solution
31
if __name__ == "__main__"
```

Рисунок 4. Реализация алгоритмов

Задание 3. Написала программу по примерам из видео «07 - Алгоритмы. Динамическое программирование» для нахождения НВП и его длины.

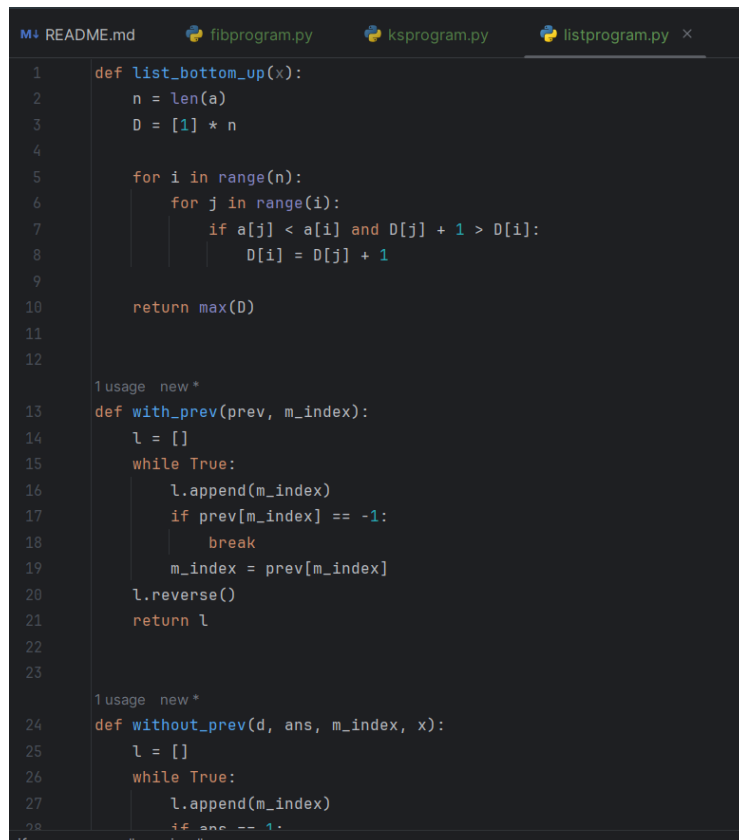


```
54     ans = d[i]
55     max_index = i
56
57     list_using_prev = with_prev(prev, max_index)
58     list_without_prev = without_prev(d, ans, max_index, a)
59
60     return ans, (list_using_prev, list_without_prev)
61
62
63 if __name__ == "__main__":
64     a = [7, 2, 1, 3, 8, 4, 9, 1, 2, 6, 5, 9, 3, 8, 1]
65     print(list_bottom_up(a))
66     result = list_bottom_up_2(a)
67     print(result[0])
68     print("С повторением:", result[1][0])
69     print("Без повторения:", result[1][1])
70
71 if __name__ == "__main__":
```

Run listprogram x

```
C:\Users\doris\AppData\Local\Programs\Python\Python39\python.exe C:\Users\doris\Documents\GitHub\Alg11\listprogram.py
5
5
С повторением: [1, 3, 5, 9, 11]
Без повторения: [2, 3, 5, 10, 11]
```

Рисунок 5. Программа и результат выполнения



```
1 def list_bottom_up(x):
2     n = len(a)
3     D = [1] * n
4
5     for i in range(n):
6         for j in range(i):
7             if a[j] < a[i] and D[j] + 1 > D[i]:
8                 D[i] = D[j] + 1
9
10    return max(D)
11
12
13 1 usage new *
14 def with_prev(prev, m_index):
15     l = []
16     while True:
17         l.append(m_index)
18         if prev[m_index] == -1:
19             break
20         m_index = prev[m_index]
21     l.reverse()
22     return l
23
24 1 usage new *
25 def without_prev(d, ans, m_index, x):
26     l = []
27     while True:
28         l.append(m_index)
29         if ans == -1:
30             break
31         m_index = ans
```

Рисунок 6. Реализация алгоритмов

Вывод: в ходе выполнения лабораторной работы были изучены три различные задачи, для решения которых был использован метод динамического программирования. Этот метод эффективен в случаях, когда задача имеет рекуррентную структуру, при которой каждое решение состоит из решений более мелких подзадач. Кроме того, в зависимости от конкретной задачи может потребоваться использование различных подходов динамического программирования, таких как вперед, назад, сверху вниз или снизу вверх.