

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»**  
**Вариант 29**

Выполнил:  
Саенко Андрей Максимович  
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., канд. технических  
наук, доцент кафедры  
инфокоммуникаций

---

(подпись)

Отчет защищен с оценкой \_\_\_\_\_ Дата защиты \_\_\_\_\_  
Ставрополь, 2023 г.

Тема: Динамическое программирование

Порядок выполнения работы:

### 1. Вычисление чисел Фибоначчи

Использованы следующие методы динамического программирования:

динамическое программирование сверху вниз и динамическое программирование снизу вверх.

Код программы:

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

```
def fib(n, variant=0):
    def fib_td(n):
        if n <= 1:
            f[n] = n
        else:
            f[n] = fib_td(n - 1) + fib_td(n-2)
        return f[n]
```

```
def fib_bu(n):
    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]
```

```
def fib_bu_improved(n):
    if n <= 1:
        return n

    first, second = 0, 1

    for _ in range(n - 1):
        first, second = second, first + second

    return second
```

```
match variant:
    case 0:
        f = [-1]*(n+1)
        return fib_td(n)
    case 1:
```

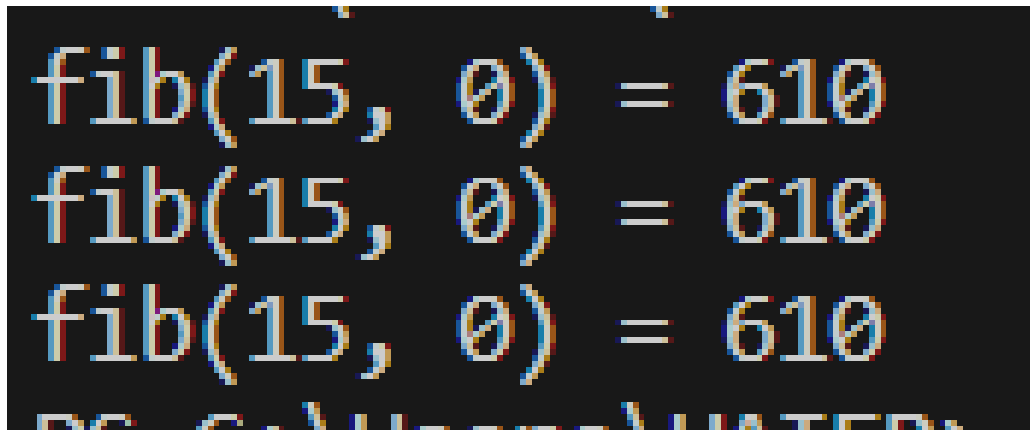
```

        return fib_bu(n)
    case 2:
        return fib_bu_improved(n)
    case _:
        print(f"Ошибка")

if __name__ == '__main__':
    print(f"fib(15, 0) = {fib(15, 0)}")
    print(f"fib(15, 0) = {fib(15, 1)}")
    print(f"fib(15, 0) = {fib(15, 2)}")

```

Результат работы программы:



```

fib(15, 0) = 610
fib(15, 0) = 610
fib(15, 0) = 610

```

Рисунок 1 – Результат работы программы

## 2. Нахождение НВП и её длины в списке.

Код программы:

```

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

def list_bottom_up(a):
    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

```

```

def using_prev(prev, m_index):
    result = []

    while True:
        result.append(m_index)

        if prev[m_index] == -1:
            break

        m_index = prev[m_index]

    result.reverse()
    return result

```

```

def without_prev(d, ans, m_index):
    result = []

    while True:
        result.append(m_index)

        if ans == 1:
            break
        ans -= 1

        while True:
            m_index -= 1
            if d[m_index] == ans and a[m_index] < a[result[-1]]:
                break

    result.reverse()
    return result

```

```

def list_bottom_up_2(a):
    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 = 0
    max_index = 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)

if __name__ == '__main__':
    a = [4, 2, 1, 6, 8, 0, 8, 6, 3, 5, 6, 7, 4, 2, 5]
    print(list_bottom_up(a))
    print(list_bottom_up_2(a))

```

Результат работы программы:

```

5
(5, ([1, 8, 9, 10, 11], [5, 8, 9, 10, 11]))

```

Рисунок 2 – Результат работы программы

### 3. Решение задачи о рюкзаке

Код программы:

```

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

def knapsack_bu(W, weight, cell):
    def knapsack_with_reps(W, weight, cell):
        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]

    def knapsack_without_reps(W, weight, cell):
        def restore(d, weight_rev, cell_rev):
            result = []
            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:
                    result.append(1)
                    w -= weight_i

```

```

        else:
            result.append(0)

        elem -= 1
    result.reverse()
    return result

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)

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

return d[W][-1], result

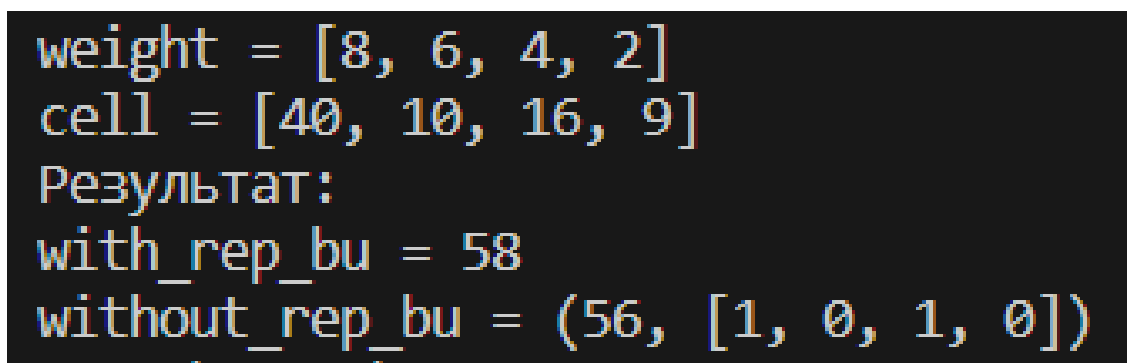
with_rep = knapsack_with_reps(W, weight, cell)
without_rep = knapsack_without_reps(W, weight, cell)

return with_rep, without_rep

if __name__ == '__main__':
    W = 12
    weight = [8, 6, 4, 2]
    cell = [40, 10, 16, 9]
    with_rep_bu, without_rep_bu = knapsack_bu(W, weight, cell)
    print(f'weight = {weight}\ncell = {cell}\nРезультат:')
    print(f'with_rep_bu = {with_rep_bu}\nwithout_rep_bu = {without_rep_bu}')

```

Результат работы программы:



```

weight = [8, 6, 4, 2]
cell = [40, 10, 16, 9]
Результат:
with_rep_bu = 58
without_rep_bu = (56, [1, 0, 1, 0])

```

Рисунок 3 – Результат работы программы

## Вывод

В ходе выполнения работы были изучены алгоритмы, в основе которых лежит концепция динамического программирования. С использованием динамического программирования были написаны программы для нахождения чисел Фибоначчи, НВП и решения задачи о рюкзаке.