МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования «Северо-Кавказский федеральный университет»

Кафедра инфокоммуникаций

Отчет по лабораторной работе №11 по дисциплине «Алгоритмизация»

Выполнил студент группы ИВТ-б-о-22-1
Пушкин Н.С. « »_____20__г.
Подпись студента_____
Работа защищена « »_____20__г.
Проверил Воронкин Р.А. _____
(подпись)

Ставрополь 2023

Порядок выполнения работы:

Динамическое программирование

Решение Фибоначчи из примера в видео

```python
def main():
    N = 10
    print(f"\nФибоначчи({N}) разными функциями:")
    print(f"fib_td({N}) = {fibonacci(N, func: 'TD')}")
    print(f"fib_bu({N}) = {fibonacci(N, func: 'BU')}")
    print(f"fib_bu_improved({N}) = {fibonacci(N, func: 'BU_I')}")

3 usages
def fibonacci(n, func='TD'):
    f = [0] * (n + 1)

    def fib_td(k):
        if k <= 1:
            f[k] = k
        else:
            f[k] = fib_td(k - 1) + fib_td(k - 2)
        return f[k]

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

    def fib_bu_improved(k):
        if k <= 1:
            return k
        prev, curr = 0, 1
        for _ in range(1, k):
            temp = curr
            curr = prev + curr
            prev = temp
        return curr

    if func == 'TD':
        f = [0] * (n + 1)
        return fib_td(n)
    elif func == 'BU':
        return fib_bu(n)
    elif func == 'BU_I':
```

Результат:

```
Фибоначчи(10) разными функциями:
fib_td(10) = 55
fib_bu(10) = 55
fib_bu_improved(10) = 55

Process finished with exit code 0
```

Решения задачи о рюкзаке в двух случаях: когда предметов неограниченное количество, и когда каждый предмет может быть использован только один раз

```python
def main():
    W = 10
    weight = [6, 3, 4, 2]
    cell = [30, 14, 16, 9]

    result = knapsack_bu(W, weight, cell)
    print(f"С повторениями: {result[0]}\nБез повторений: {result[1][0]}")
    print("Реконструированное решение: [" + ", ".join(map(str, result[1][1])) + "]")

1 usage
def knapsack_bu(W, weight, cell):
    def knapsack_with_reps(w, weight, cell):
        d = [0] * (W + 1)
        for i in range(1, W + 1):
            for j in range(len(weight)):
                if weight[j] <= i:
                    d[i] = max(d[i], d[i - weight[j]] + cell[j])
        return d[W]

    def knapsack_without_reps(W, weight, cell):
        d = [[0] * (len(weight) + 1) for _ in range(W + 1)]
        solution = [[0] * (len(weight) + 1) for _ in range(W + 1)]

        for i in range(1, W + 1):
            for j in range(1, len(weight) + 1):
                d[i][j] = d[i][j - 1]
                if weight[j - 1] <= i:
                    new_value = d[i - weight[j - 1]][j - 1] + cell[j - 1]
                    if new_value > d[i][j]:
                        d[i][j] = new_value
                        solution[i][j] = 1

        reconstructed_solution = [0] * len(weight)
        W_remaining = W
        for j in range(len(weight), 0, -1):
            if solution[W_remaining][j] == 1:
                reconstructed_solution[j - 1] = 1
                W_remaining -= weight[j - 1]

        return (d[W][len(weight)], reconstructed_solution)
```

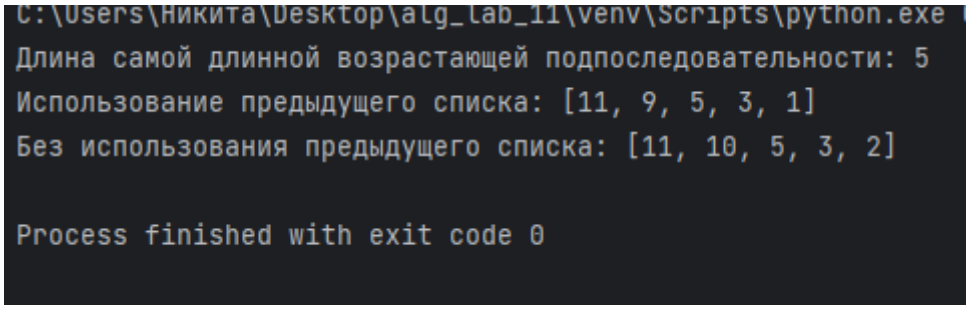
Результат:

```
C:\Users\Никита\Desktop\alg_lab_11\venv\
С повторениями: 48
Без повторений: 46
Реконструированное решение: [1, 0, 1, 0]

Process finished with exit code 0
```

Нахождения длины НВП в списке

```python
def main():
    a = [7, 2, 1, 3, 8, 4, 9, 1, 2, 6, 5, 9, 3, 8, 1]
    result = list_bottom_up_2(a)
    print(f"Длина самой длинной возрастающей подпоследовательности: {result[0]}")
    print(f"Использование предыдущего списка: {result[1][0]}")
    print(f"Без использования предыдущего списка: {result[1][1]}")

1 usage
def list_bottom_up_2(a):
    d = [1] * len(a)
    prev = [-1] * len(a)

    for i in range(len(a)):
        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 in range(len(d)):
        if ans < d[i]:
            ans = d[i]
            max_index = i

    list_using_prev = restore_using_prev(prev, max_index)
    list_without_prev = restore_without_prev(ans, max_index, d, a)

    return (ans, (list_using_prev, list_without_prev))

1 usage
def restore_using_prev(prev, max_index):
    result = []
    while True:
        result.append(max_index)
        if prev[max_index] == -1:
            break
        max_index = prev[max_index]
    return result

def restore_without_prev(ans, max_index, d, a):
    result = []
    while True:
        result.append(max_index)
        if ans == 1:
            break
        ans -= 1
        while True:
            max_index -= 1
            if max_index < 0:
                break
            if d[max_index] == ans and a[max_index] < a[result[-1]]:
                break
    return result

if __name__ == "__main__":
    main()
```

Результат:


C:\Users\Никита\Desktop\alg_lab_11\venv\Scripts\python.exe
Длина самой длинной возрастающей подпоследовательности: 5
Использование предыдущего списка: [11, 9, 5, 3, 1]
Без использования предыдущего списка: [11, 10, 5, 3, 2]
Process finished with exit code 0