МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Кафедра информационных систем управления

Отчет по лабораторной работе №4
Вариант 22

Бовта Тимофея Анатольевича
студента 3 курса
специальности «прикладная математика»

Преподаватель:

Д. Ю. Кваша

Минск, 2024 г.

Лабораторная работа №4

Постановка задачи.

Задача о рюкзаке (англ. Knapsack problem) — дано n предметов, предмет i имеет массу $w_i > 0$ и стоимость $p_i > 0$. Необходимо выбрать из этих предметов такой набор, чтобы суммарная масса не превосходила заданной величины W (вместимость рюкзака), а суммарная стоимость была максимальна.

Рассмотрим задачу **Неограниченный рюкзак** (англ. Unbounded Knapsack Problem), в которой любой предмет может быть выбран любое количество раз.

Формулировка Задачи
Каждый предмет может быть выбран любое число раз. Задача выбрать количество x_i предметов каждого типа так, чтобы максимизировать общую стоимость:

$$\sum_{i=1}^{n} p_i x_i,$$

выполнялось условие совместности:

$$\sum_{i=1}^{n} w_i x_i \leqslant W;$$

где $x_i \geqslant 0$ целое, для всех $i = 1, 2, \ldots, n$.

Условие задачи.

$$\max 4x_1 + 8x_2 + 15x_3;$$
$$s.t.\ 3x_1 + 4x_2 + 5x_3 \leqslant 10;$$
$$x_1, x_2, x_3 \in \mathbb{Z}_+$$

Программная реализация алгоритма.

```
1   import pandas as pd
2   import numpy as np
3
4   def knapsack(v, weights, W):
5       n = len(weights)
6       f = np.zeros((n + 1, W + 1))
7       p = np.zeros((n + 1, W + 1))
8       for i in range(1, n + 1):
9           for w in range(1, W + 1):
10              if w >= weights[i - 1]:
11                  f[i, w] = max([f[i - 1, w], f[i, w - weights[i - 1]] + v[i - 1]])
12              else:
13                  f[i, w] = f[i - 1, w]
14              p[i, w] = int(f[i, w] != f[i - 1, w])
15
16      table_list = []
17      for i in range(1, n+1):
```

```
18          table_list.append(pd.Series(f[i], name=f'f_{i}'))
19          table_list.append(pd.Series(p[i], name=f'p_{i}'))
20      table = pd.DataFrame(table_list).T
21
22      taken_list = np.zeros_like(weights)
23      w = W
24      i = n
25      while i != 0 and w != 0:
26          if p[i, w] == 1:
27              taken_list[i - 1] += 1
28              w -= weights[i - 1]
29          else:
30              i -= 1
31
32      return f[n, W], table, taken_list
33
34  if __name__ == "__main__":
35      w = [4, 8, 15]
36      v = [3, 4, 5]
37      W = 10
38      result, table, taken_list = knapsack(w, v, W)
39      print('Оптимальное решение задачи:', result)
40      print(table)
41      for i in range(len(w)):
42          print(f'x_{i} = {taken_list[i]}')
```

Результат выполнения программы.

```
Оптимальное решение задачи: 30.0
     f_1  p_1   f_2  p_2   f_3  p_3
0    0.0  0.0   0.0  0.0   0.0  0.0
1    0.0  0.0   0.0  0.0   0.0  0.0
2    0.0  0.0   0.0  0.0   0.0  0.0
3    4.0  1.0   4.0  0.0   4.0  0.0
4    4.0  1.0   8.0  1.0   8.0  0.0
5    4.0  1.0   8.0  1.0  15.0  1.0
6    8.0  1.0   8.0  0.0  15.0  1.0
7    8.0  1.0  12.0  1.0  15.0  1.0
8    8.0  1.0  16.0  1.0  19.0  1.0
9   12.0  1.0  16.0  1.0  23.0  1.0
10  12.0  1.0  16.0  1.0  30.0  1.0
x_0 = 0
x_1 = 0
x_2 = 2
```