

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 ПРАКТИЧЕСКОЙ РАБОТЕ №6**  
**дисциплины «Алгоритмизация»**  
**Вариант 8**

Выполнил:  
Данилецкий Дмитрий Витальевич  
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.

## Ход работы

1. В соответствии с приведённым ниже псевдокодом написал программу на Python, которая на вход принимает множество точек, а выводит минимальное количество отрезков единичной длины, которыми можно покрыть все точки. Алгоритм заключается в том, что пока размер входной массив данных не пуст: мы находим в нём минимальное значение и добавляем к решению отрезок  $[x_{\min}, x_{\min} + 1]$ , а затем удаляем все точки, которые входят в данный отрезок.

Функция  $\text{POINTS COVER}(x_1, \dots, x_n)$

$S \leftarrow \{x_1, \dots, x_n\}$

пока  $S$  не пусто:

$x_m \leftarrow$  минимальная точка  $S$

    добавить к решению отрезок  $[\ell, r] = [x_m, x_m + 1]$

    выкинуть из  $S$  точки, покрытые отрезком  $[\ell, r]$

вернуть построенное решение

Рисунок 1. Алгоритм PointsCover

```
C: > Users > slime > AlgoritmlAB6 > progr > PointsCover.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  if __name__ == '__main__':
5      S = [float(x) for x in input("Введите список чисел через пробел: ").split()]
6      result = []
7
8      while S:
9          x = float(min(S))
10         result.append([x, x + 1])
11
12         S = [num for num in S if num < x or num > x + 1]
13
14     print(result)
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

```
PS C:\Users\slime> & C:/msys64/mingw64/bin/python.exe c:/Users/slime/AlgoritmlAB6/progr/PointsCover.py
Введите список чисел через пробел: 1.2 3.3 2.2 1.7 5.5 1.6
[[1.2, 2.2], [3.3, 4.3], [5.5, 6.5]]
PS C:\Users\slime>
```

Рисунок 2. Результат работы программы PointsCover

2. В соответствии с приведённым ниже псевдокодом написал программу на Python, которая на вход принимает множество точек, а выводит минимальное количество отрезков единичной длины, которыми можно покрыть все точки. Алгоритм заключается в том, что сначала массив сортируется, далее пока  $i$  меньше размера массива добавляем отрезок  $[S[i], S[i]+1]$  и пока  $S[i]$  меньше предыдущего значения конца отрезка добавляем 1 к  $i$ .

### Функция POINTSCOVER( $x_1, \dots, x_n$ )

```

 $x_1, \dots, x_n \leftarrow \text{SORT}(x_1, \dots, x_n)$ 
 $i \leftarrow 1$ 
пока  $i \leq n$ :
    добавить к решению отрезок  $[\ell, r] = [x_i, x_i + 1]$ 
     $i \leftarrow i + 1$ 
    пока  $i \leq n$  и  $x_i \leq r$ :
         $i \leftarrow i + 1$ 
вернуть построенное решение

```

Рисунок 3. Улучшенный алгоритм PointsCover

```

C: > Users > slime > AlgoritmlAB6 > progr > PointsCover.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  if __name__ == '__main__':
5      S = [float(x) for x in input("Введите список чисел через пробел: ").split()]
6      result = []
7
8      while S:
9          x = float(min(S))
10         result.append([x, x + 1])
11
12         S = [num for num in S if num < x or num > x + 1]
13
14     print(result)

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS C:\Users\slime> & C:\msys64\mingw64\bin\python.exe c:/Users/slime/AlgoritmlAB6/progr/PointsCover.py
Введите список чисел через пробел: 1.2 3.3 2.2 1.7 5.5 1.6
[[1.2, 2.2], [3.3, 4.3], [5.5, 6.5]]
PS C:\Users\slime>

```

Рисунок 4. Результат работы программы PointsCover2

3. В соответствии с приведённым ниже псевдокодом, написал программу для решения задачи о выборе заявок, в которой требуется найти максимальное количество попарно не пересекающихся отрезков.

Функция  $\text{ACTSEL}(\ell_1, r_1, \dots, \ell_n, r_n)$

$S \leftarrow \{[\ell_1, r_1], \dots, [\ell_n, r_n]\}$

пока  $S$  не пусто:

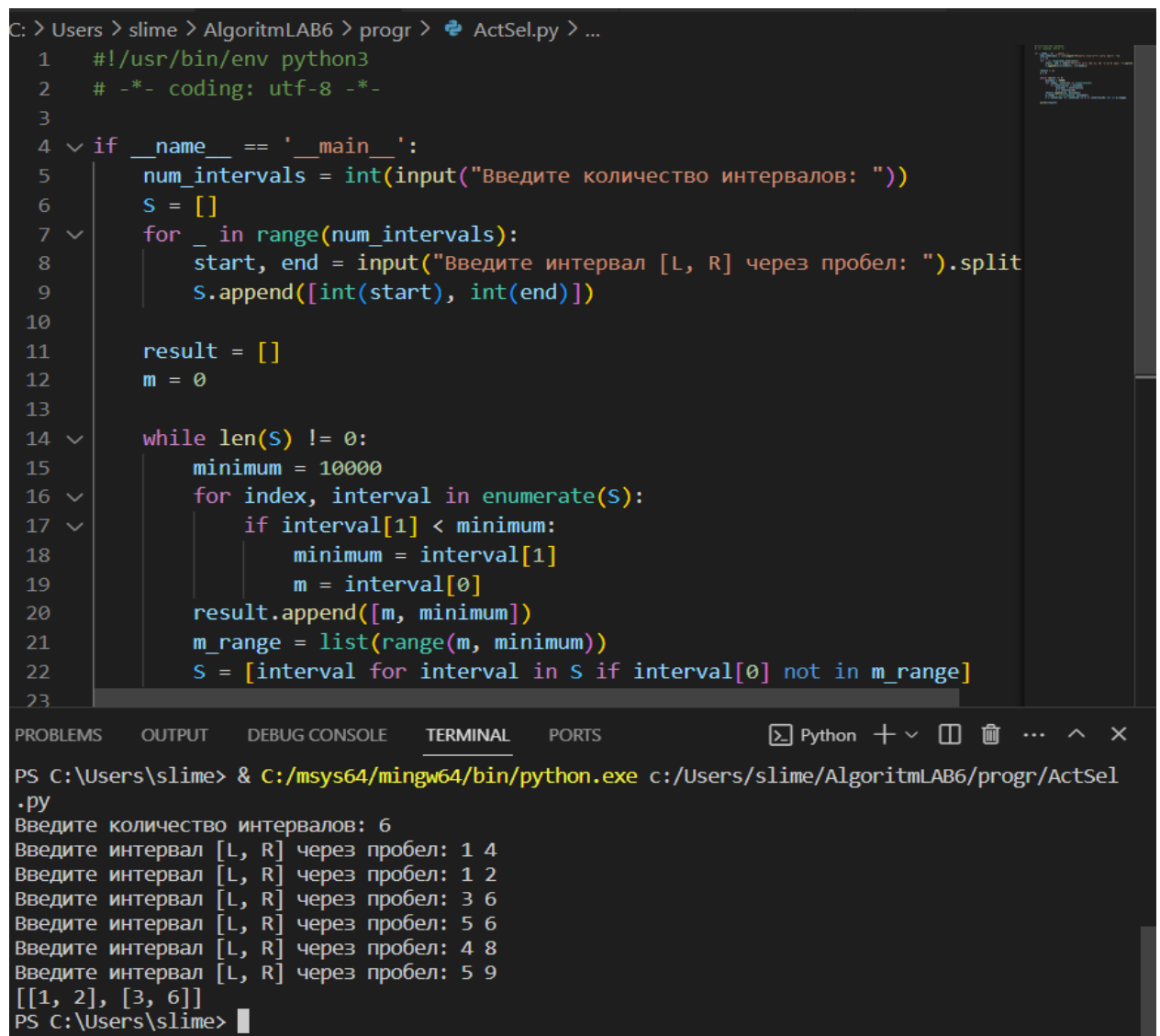
$[\ell_m, r_m] \leftarrow$  отрезок из  $S$  с мин. правым концом

    добавить  $[\ell_m, r_m]$  к решению

    выкинуть из  $S$  отрезки, пересекающиеся с  $[\ell_m, r_m]$

вернуть построенное решение

Рисунок 5. Алгоритм ActSel



```
C: > Users > slime > AlgoritmLAB6 > progr > ActSel.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  if __name__ == '__main__':
5      num_intervals = int(input("Введите количество интервалов: "))
6      S = []
7      for _ in range(num_intervals):
8          start, end = input("Введите интервал [L, R] через пробел: ").split()
9          S.append([int(start), int(end)])
10
11     result = []
12     m = 0
13
14     while len(S) != 0:
15         minimum = 10000
16         for index, interval in enumerate(S):
17             if interval[1] < minimum:
18                 minimum = interval[1]
19                 m = interval[0]
20         result.append([m, minimum])
21         m_range = list(range(m, minimum))
22         S = [interval for interval in S if interval[0] not in m_range]
23
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
Python + - [ ] [ ] ... ^ x
PS C:\Users\slime> & C:/msys64/mingw64/bin/python.exe c:/Users/slme/AlgoritmLAB6/progr/ActSel
.py
Введите количество интервалов: 6
Введите интервал [L, R] через пробел: 1 4
Введите интервал [L, R] через пробел: 1 2
Введите интервал [L, R] через пробел: 3 6
Введите интервал [L, R] через пробел: 5 6
Введите интервал [L, R] через пробел: 4 8
Введите интервал [L, R] через пробел: 5 9
[[1, 2], [3, 6]]
PS C:\Users\slime>
```

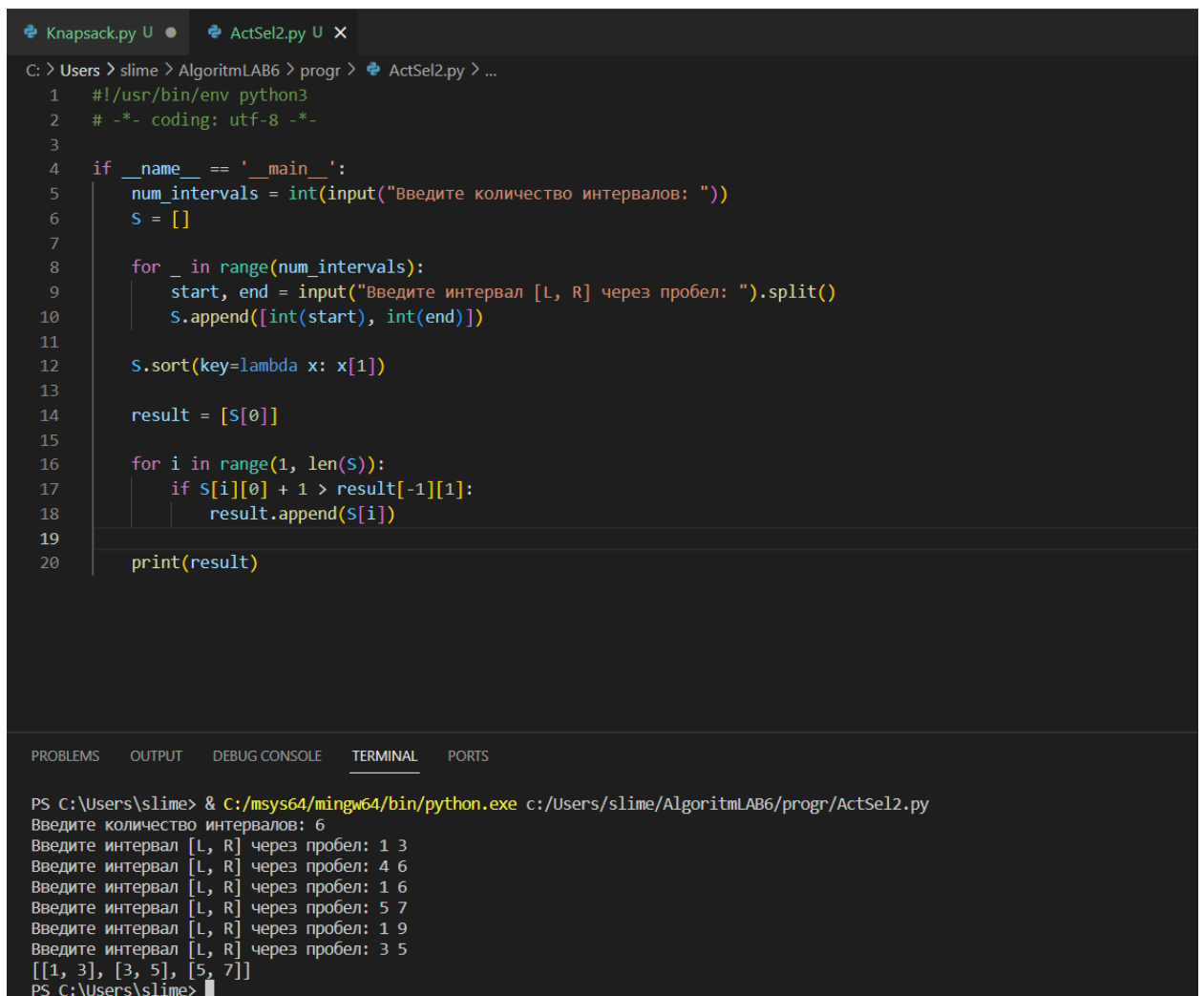
Рисунок 6. Результат работы программы ActSel

4. В соответствии с приведённым ниже псевдокодом, написал улучшенную программу для решения задачи о выборе заявок, в которой требуется найти максимальное количество попарно не пересекающихся отрезков.

### Функция $\text{ACTSEL}(\ell_1, r_1, \dots, \ell_n, r_n)$

отсортировать  $n$  отрезков по правым концам  
для всех отрезков в полученном порядке:  
    если текущий отрезок не пересекает  
        последний добавленный:  
        взять его в решение  
вернуть построенное решение

Рисунок 7. Улучшенный алгоритм ActSel



```
Knapsack.py U • ActSel2.py U X
C: > Users > slime > AlgoritmLAB6 > progr > ActSel2.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  if __name__ == '__main__':
5      num_intervals = int(input("Введите количество интервалов: "))
6      S = []
7
8      for _ in range(num_intervals):
9          start, end = input("Введите интервал [L, R] через пробел: ").split()
10         S.append([int(start), int(end)])
11
12     S.sort(key=lambda x: x[1])
13
14     result = [S[0]]
15
16     for i in range(1, len(S)):
17         if S[i][0] + 1 > result[-1][1]:
18             result.append(S[i])
19
20     print(result)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\slime> & C:/msys64/mingw64/bin/python.exe c:/Users/slme/AlgoritmLAB6/progr/ActSel2.py
Введите количество интервалов: 6
Введите интервал [L, R] через пробел: 1 3
Введите интервал [L, R] через пробел: 4 6
Введите интервал [L, R] через пробел: 1 6
Введите интервал [L, R] через пробел: 5 7
Введите интервал [L, R] через пробел: 1 9
Введите интервал [L, R] через пробел: 3 5
[[1, 3], [3, 5], [5, 7]]
PS C:\Users\slime>
```

Рисунок 8. Результат работы программы ActSel2

5. В соответствии с приведённым ниже псевдокодом написал программу, которая получает на вход дерево, а на выходе независимое множество. Сначала находится максимальное число в массиве состоящем из ребер графа, потом пока этот массив не пуст, создаётся множество локальных решений, в который добавляются элементы графа которые имеют 1 связь, а эта связь проверяется, которая возвращает количество элементов соответствующих значению num. Если вершина имеет 1 связь – это значит, что лист найдет и он добавляется в массив локальных решений. Те ребра вершин, которые находятся в локальном решении удаляются из входного массива и цикл проходит до того момента, пока число элементов входного массива не станет равно 0.

### Функция `MAXINDEPENDENTSET( $T$ )`

пока  $T$  не пусто:

    взять в решение все листья

    выкинуть их и их родителей из  $T$

вернуть построенное решение

Рисунок 9.Алгоритм MaxIndependentSet

```

11
12     result = []
13
14     max = 0
15     for index, item in enumerate(T):
16         for j in range(2):
17             if item[j] > max:
18                 max = item[j]
19
20     while T:
21         res = []
22         for i in range(max, 0, -1):
23             count = 0
24             for index, element in enumerate(T):
25                 if element[0] == i or element[1] == i:
26                     count += 1
27             if count == 1:
28                 res.append(i)
29             i = 0
30         while i != len(T):
31             if T[i][1] in res:
32                 T.pop(i)
33                 i = i - 1
34             i = i + 1
35         result.append(res)

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS C:\Users\slime> & c:/msys64/mingw64/bin/python.exe c:/Users\slime/AlgoritmlAB6/progr/MaxIndependentSet.py
Введите количество ребер дерева: 8
Введите [Род. Дочерн.] через пробел: 1 2
Введите [Род. Дочерн.] через пробел: 1 3
Введите [Род. Дочерн.] через пробел: 2 4
Введите [Род. Дочерн.] через пробел: 2 5
Введите [Род. Дочерн.] через пробел: 3 6
Введите [Род. Дочерн.] через пробел: 5 7
Введите [Род. Дочерн.] через пробел: 5 8
Введите [Род. Дочерн.] через пробел: 6 9
[[9, 8, 7, 4], [6, 5], [3, 2]]
PS C:\Users\slime>

```

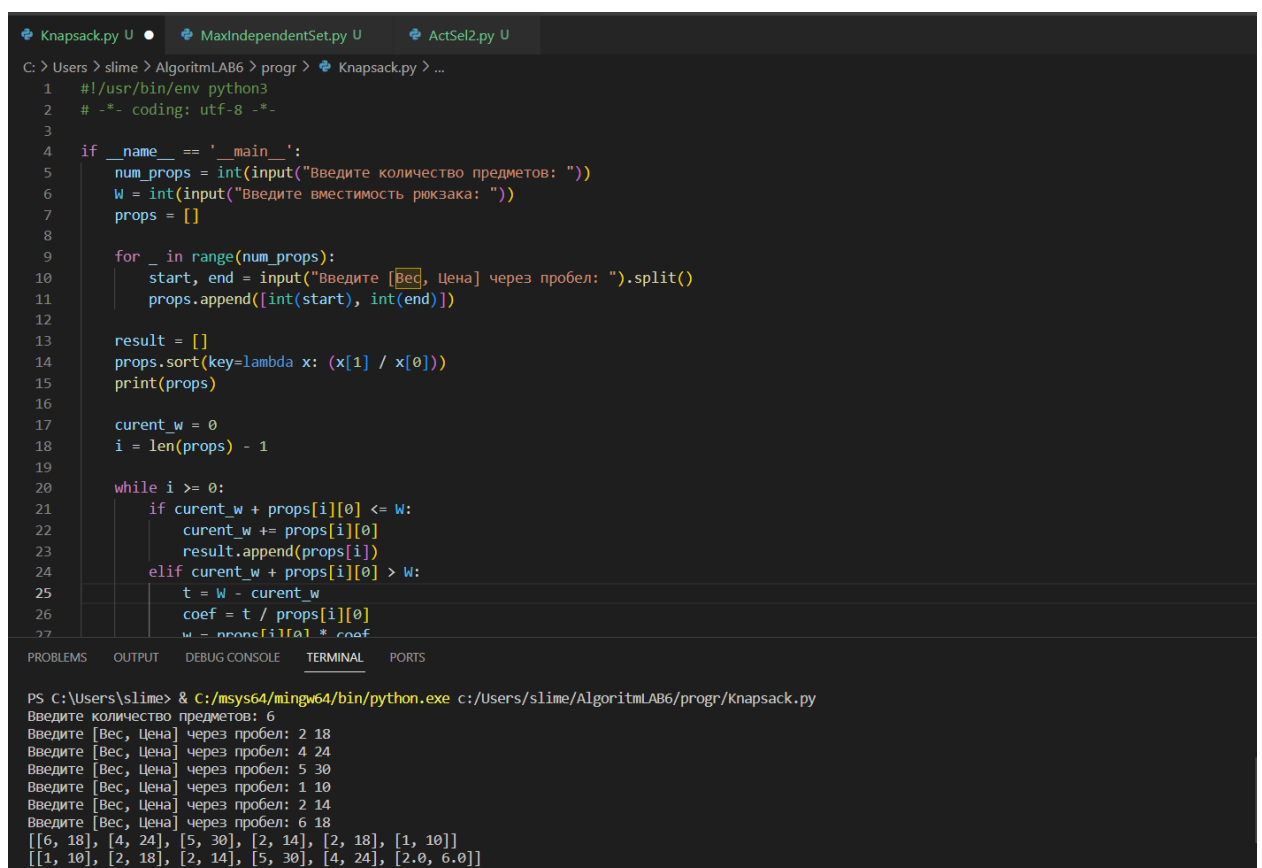
Рисунок 8. Результат работы программы MaxIndependentSet

6. В соответствии с приведённым ниже псевдокодом написал программу по задаче о непрерывном рюкзаке, в которой требуется частями предметов с весами и их стоимостью заполнить рюкзак определённого размера так, чтобы стоимость помещённых в него предметов была максимальной.

Функция  $\text{KNAPSACK}(w_1, c_1, \dots, w_n, c_n)$

отсортировать предметы по убыванию  $c/w$   
для всех предметов в полученном порядке:  
взять по максимуму текущего предмета  
вернуть построенное решение

Рисунок 11. Алгоритм Knapsack



```
Knapsack.py U • MaxIndependentSet.py U ActSel2.py U
C: > Users > slime > AlgoritmlAB6 > progr > Knapsack.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  if __name__ == '__main__':
5      num_props = int(input("Введите количество предметов: "))
6      W = int(input("Введите вместимость рюкзака: "))
7      props = []
8
9      for _ in range(num_props):
10         start, end = input("Введите [Вес, Цена] через пробел: ").split()
11         props.append([int(start), int(end)])
12
13     result = []
14     props.sort(key=lambda x: (x[1] / x[0]))
15     print(props)
16
17     current_w = 0
18     i = len(props) - 1
19
20     while i >= 0:
21         if current_w + props[i][0] <= W:
22             current_w += props[i][0]
23             result.append(props[i])
24         elif current_w + props[i][0] > W:
25             t = W - current_w
26             coef = t / props[i][0]
27             w = props[i][0] * coef
28
29     print(result)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\slime> & C:/msys64/mingw64/bin/python.exe c:/Users/slime/AlgoritmlAB6/progr/Knapsack.py
Введите количество предметов: 6
Введите [Вес, Цена] через пробел: 2 18
Введите [Вес, Цена] через пробел: 4 24
Введите [Вес, Цена] через пробел: 5 30
Введите [Вес, Цена] через пробел: 1 10
Введите [Вес, Цена] через пробел: 2 14
Введите [Вес, Цена] через пробел: 6 18
[[6, 18], [4, 24], [5, 30], [2, 14], [2, 18], [1, 10]]
[[1, 10], [2, 18], [2, 14], [5, 30], [4, 24], [2.0, 6.0]]
```

Рисунок 12. Результат работы программы Knapsack

Вывод: в ходе выполнения лабораторной работы были исследованы некоторые примеры жадных алгоритмов, решающих различные задачи. На основании этих примеров можно сказать, что жадные алгоритмы

действительно строят оптимальное решение благодаря понятиям надёжного шага и оптимальности подзадач.