

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 №6
дисциплины «Алгоритмизация»**

Выполнил:
Кенесбаев Хилол Куат улы
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:

1. Написал программу по задаче покрытия точек отрезками единичной длины двумя способами: обычным (pointscover1) и улучшенным (pointscover2) алгоритмами.



Рисунок 1 Код и результат работы программы *main*

```
C:\Users\HP-PC\AppData\Local\Programs\Python\Python312\python.exe "C:\Users\HP-PC\Desktop\study\Алгоритмизация\алгоритмизация 6\algorotm
Множество точек: [1.1, 1.4, 8.3, 7.1, 5.7, 1.4, 0.3, 6.3, 8.7, 3.6, 2.3, 2.8, 2.3, 4.2, 7.0, 6.0, 9.7, 10.0, 9.6, 6.2]
Множество отрезков 1: [[0.3, 1.3], [1.4, 2.4], [2.8, 3.8], [4.2, 5.2], [5.7, 6.7], [7.0, 8.0], [8.3, 9.3], [9.6, 10.6]]
Множество отрезков 2: [[0.3, 1.3], [1.4, 2.4], [2.8, 3.8], [4.2, 5.2], [5.7, 6.7], [7.0, 8.0], [8.3, 9.3], [9.6, 10.6]]
Минимальное количество отрезков, которыми можно покрыть данное множество точек = 8
```

Рисунок 2 Вывод программы *main* в терминал

2. Написал программу по задаче о выборе заявок, в которой требуется найти максимальное количество попарно не пересекающихся отрезков двумя способами: обычным (actsel1) и улучшенным (actsel2) алгоритмами.

```

main.py  actsel.py  X
> actsel.py > ...
1  from random import randint
2  import matplotlib.pyplot as plt
3
4  def plot_segments(s, sol):
5      plt.xlabel('Координаты')
6      plt.ylabel('Отрезки')
7      k = 40
8      for i in s:
9          d1 = [k, k]
10         plt.plot(i, d1, color="blue", linewidth=10, solid_capstyle='butt')
11         k -= 10
12     d2 = [50, 50]
13     for k in range(len(sol)):
14         plt.plot(sol[k], d2, color="red", linewidth=10, solid_capstyle='butt')
15     ax = plt.gca()
16     ax.set_yticks([])
17
18 def actsell(s):
19     solution = []
20     while len(s) > 0:
21         m = min(x[1] for x in s)
22         minindex = next(i for i in range(len(s)) if s[i][1] == m)
23         d = s[minindex]
24         solution.append(d)
25         i = 0
26         while i < len(s):
27             if s[i][0] <= d[1]:
28                 s.pop(i)
29             else:
30                 i += 1
31     return solution
32
33 def actsel2(s):
34     s.sort(key=lambda x: x[1])
35     solution = [s[0]]
36     for i in s:
37         if i[0] > solution[-1][1]:
38             solution.append(i)
39     return solution
40
41 if __name__ == '__main__':
42     s = [[a, randint(a+1, 1100)]
43          for a in (randint(0, 1000) for _ in range(20))]
44     cps = s.copy()
45     plt.figure(1)

```

Рисунок 3 Код программы ActSel

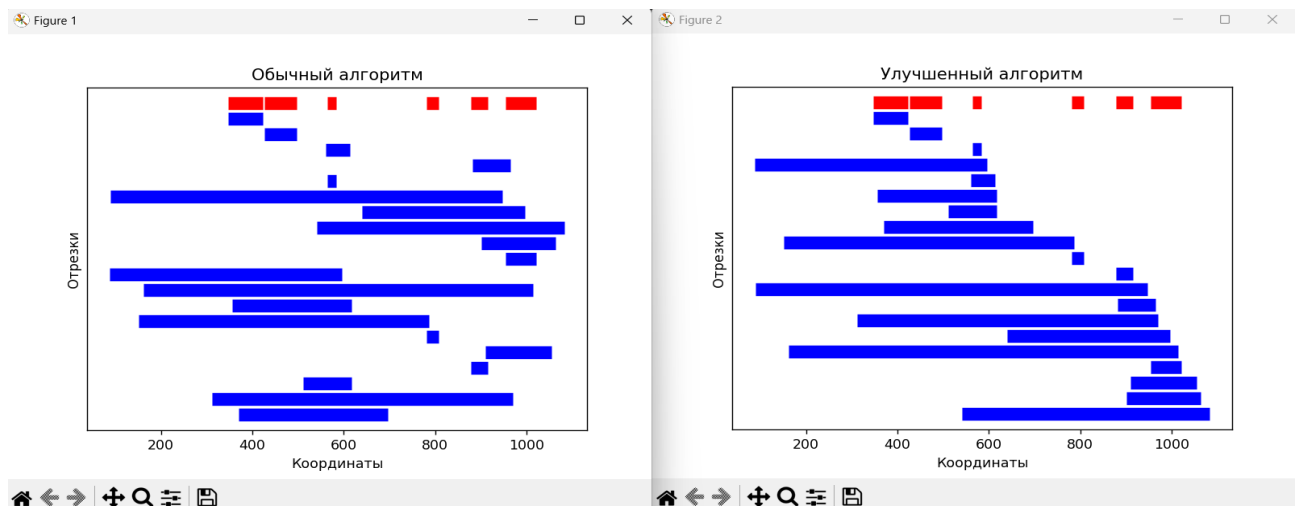


Рисунок 4 Графическое представление решения задачи обоих алгоритмов

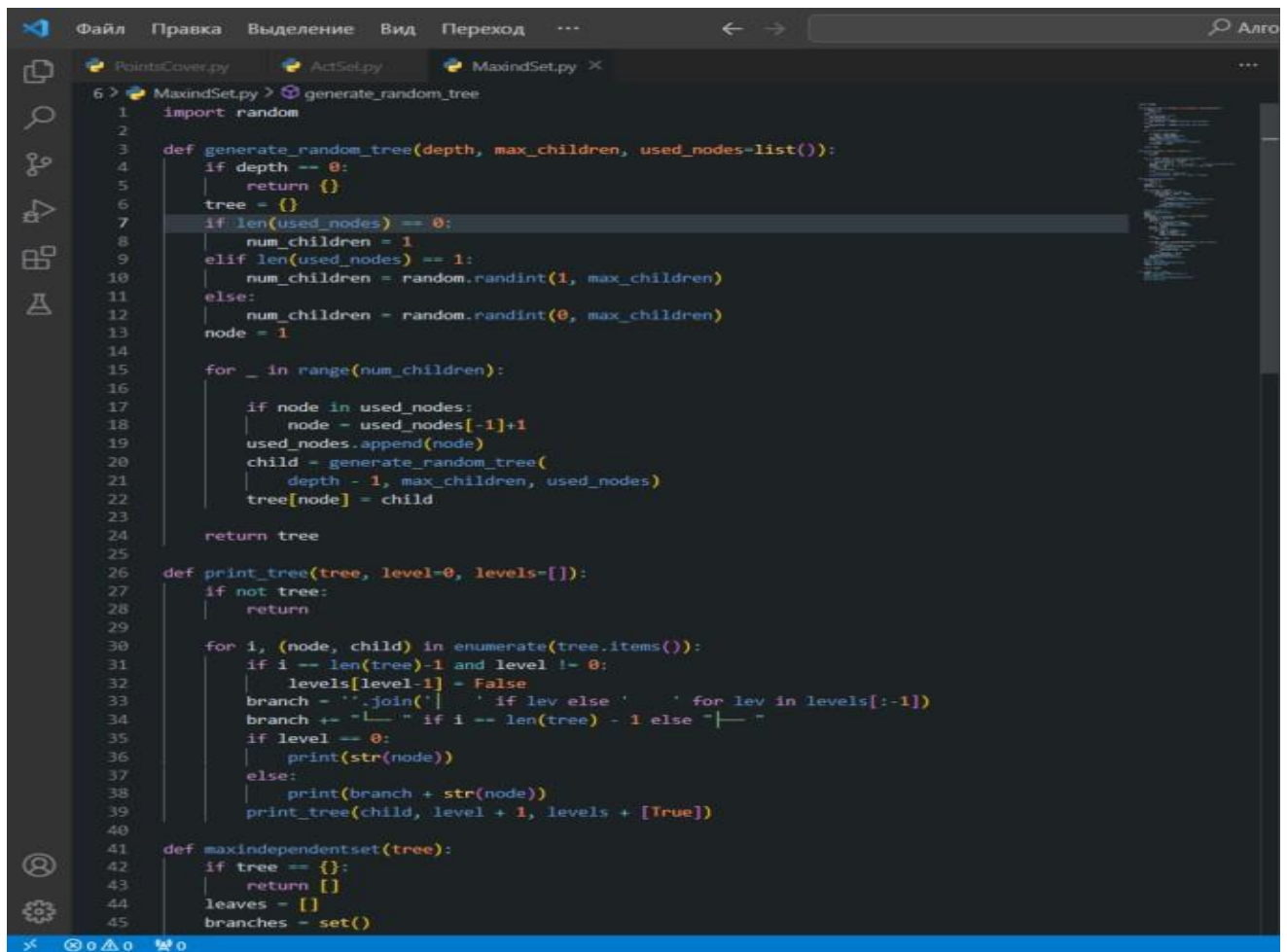
```

Массив отрезков: [[347, 423], [427, 498], [560, 613], [882, 965], [564, 583], [90, 948], [640, 997], [542, 1083], [901, 1064], [954, 1022], [88, 596], [162, 1014], [356, 617], [12, 618], [311, 970], [371, 696]]
Непересекающиеся отрезки: [[347, 423], [427, 498], [564, 583], [782, 808], [878, 915], [954, 1022]]
-----
Массив сортированных отрезков: [[347, 423], [427, 498], [564, 583], [88, 596], [560, 613], [356, 617], [512, 618], [371, 696], [151, 787], [782, 808], [878, 915], [90, 948], [1022], [911, 1055], [901, 1064], [542, 1083]]
Непересекающиеся сортированные отрезки: [[347, 423], [427, 498], [564, 583], [782, 808], [878, 915], [954, 1022]]

```

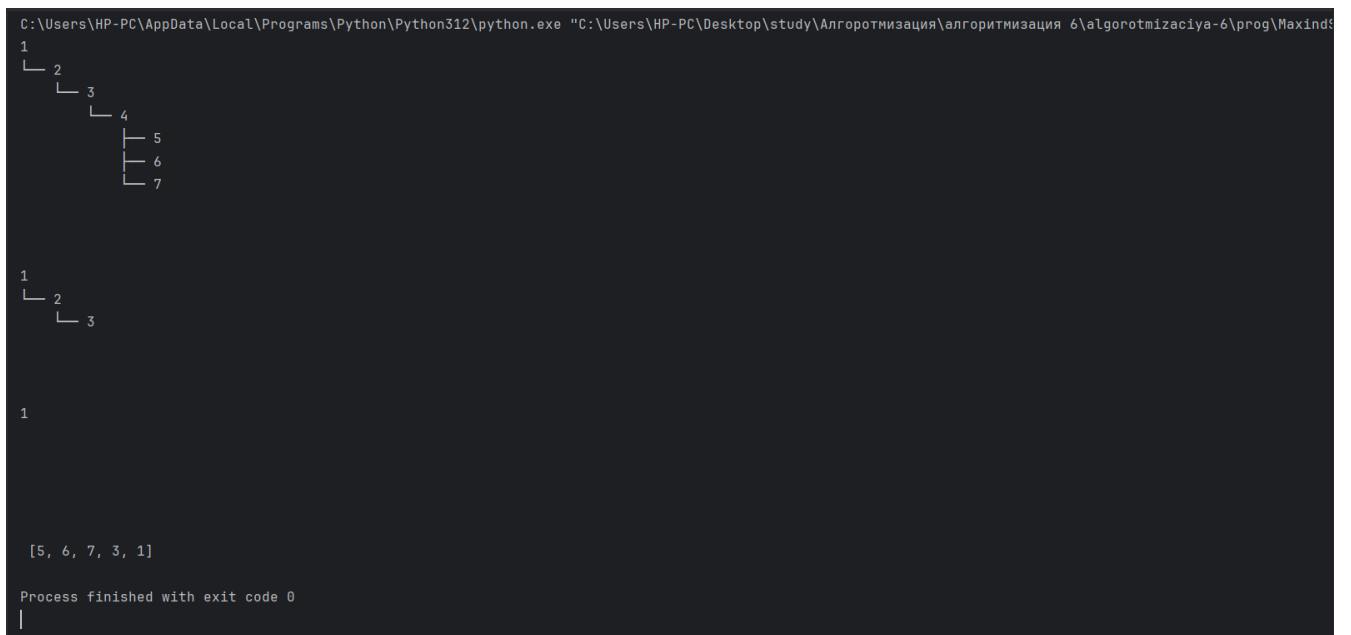
Рисунок 5 Вывод программы ActSel в терминал

3. Написал программу по задаче планирования вечеринки в компании, в которой требуется по заданному дереву определить независимое множество (множество не соединённых друг с другом вершин) максимального размера.



```
File Edit Selection View Transition ...
PointsCover.py ActSet.py MaxindSet.py X
6 > MaxindSet.py > generate_random_tree
1 import random
2
3 def generate_random_tree(depth, max_children, used_nodes=list()):
4     if depth == 0:
5         return {}
6     tree = {}
7     if len(used_nodes) == 0:
8         num_children = 1
9     elif len(used_nodes) == 1:
10        num_children = random.randint(1, max_children)
11    else:
12        num_children = random.randint(0, max_children)
13    node = 1
14
15    for _ in range(num_children):
16
17        if node in used_nodes:
18            node = used_nodes[-1]+1
19        used_nodes.append(node)
20        child = generate_random_tree(
21            depth - 1, max_children, used_nodes)
22        tree[node] = child
23
24    return tree
25
26 def print_tree(tree, level=0, levels=[]):
27     if not tree:
28         return
29
30     for i, (node, child) in enumerate(tree.items()):
31         if i == len(tree)-1 and level != 0:
32             levels[level-1] = False
33             branch = ''.join(' | ' if lev else ' ' for lev in levels[:level-1])
34             branch += "└─" if i == len(tree) - 1 else "├─"
35             if level == 0:
36                 print(str(node))
37             else:
38                 print(branch + str(node))
39             print_tree(child, level + 1, levels + [True])
40
41 def maxindependentset(tree):
42     if tree == {}:
43         return []
44     leaves = []
45     branches = set()
```

Рисунок 6 Код программы MaxindSet



```
C:\Users\HP-PC\AppData\Local\Programs\Python\Python312\python.exe "C:\Users\HP-PC\Desktop\study\Алгоритмизация\алгоритмизация_6\algorotmizaciya-6\prog\MaxindS
1
├─ 2
│  └─ 3
│     └─ 4
│        └─ 5
│           └─ 6
│              └─ 7
└─ 1

1
├─ 2
│  └─ 3
└─ 1

[5, 6, 7, 3, 1]

Process finished with exit code 0
```

Рисунок 7 Вывод программы MaxindSet в терминал

4. Написал программу по задаче о непрерывном рюкзаке, в которой требуется частями предметов с весами w_i , стоимостями c_i набрать рюкзак фиксированного размера на максимальную стоимость.

```
6 > KnapSack.py > ...
1 import random
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import matplotlib.patches as patches
5
6 def fractional_knapsack(items, capacity):
7     items.sort(key=lambda x: x[1]/x[2], reverse=True)
8     solution_mas = {}
9     solution_money = 0
10    total_weight = 0
11
12    for item in items:
13        t = capacity - total_weight
14        if (t) / item[2] > 1:
15            solution_mas[item[0]] = item[2]
16            total_weight += item[2]
17            solution_money += item[1]
18        else:
19            solution_mas[item[0]] = t
20            solution_money += item[1]/item[2]*t
21            break
22
23    return solution_mas, solution_money
24
25 if __name__ == '__main__':
26     n = 10
27     items_mas = [[i, random.randint(1, 100), random.randint(3, 20)]
28                  for i in range(n)]
29     r = 10
30     print("{} | {} | {}".format(
31         'Элемент', 'Стоимость', 'Вес', r=r))
32     for i, j, k in items_mas:
33         print("{} | {} | {}".format(i, j, k, r=r+2))
34         print("{} | {} | {}".format(
35             i, j, k, r=r))
36
37     capacity = 30
38     solution, money = fractional_knapsack(items_mas, capacity)
39
40     print("\nРешение:")
41     for item in solution:
42         print("Элемент", item, "был взят весом", solution[item])
43     print("Стоимость рюкзака = ", money)
44     fig, ax = plt.subplots()
45     ax.add_patch(patches.Rectangle((0, -0.5), capacity, 1, facecolor='gray'))
```

Рисунок 8 Код программы KnapSack

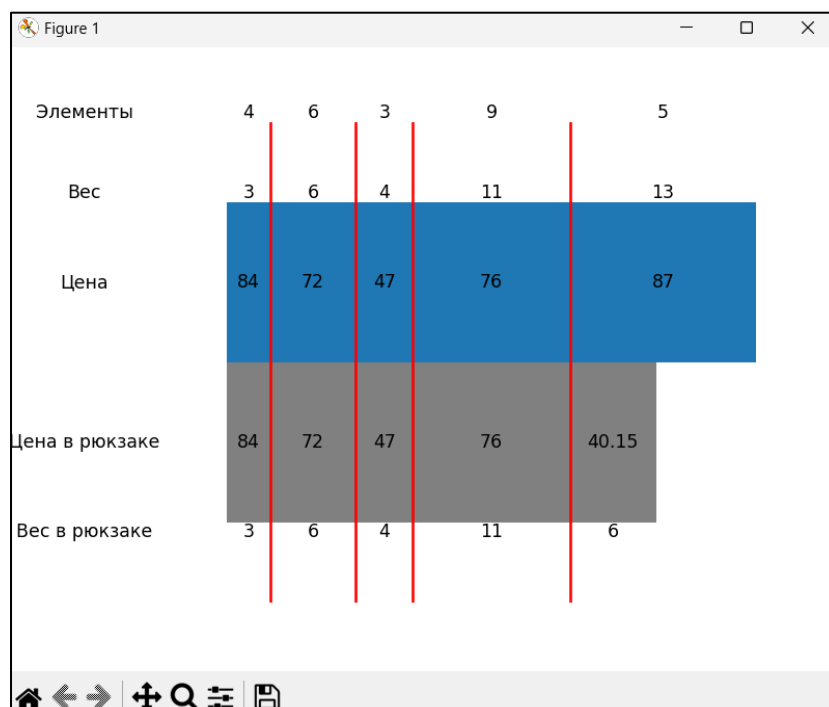


Рисунок 9 Графическое представление решения

```
C:\Users\HP-PC\AppData\Local\Programs\Python\Python312\python.exe "C:\Users\HP-PC\Desktop\study\Алгоритмизация\алгоритмизация 6\algorotimizaciya-6\prog\Knapsack.py"
```

Элемент	Стоимость	Вес
0	89	14
1	93	20
2	53	7
3	67	8
4	82	16
5	71	19
6	41	19
7	67	9
8	49	9
9	70	11

```
Решение:  
Элемент 3 был взят весом 8  
Элемент 2 был взят весом 7  
Элемент 7 был взят весом 9  
Элемент 9 был взят весом 6  
Стоимость рюкзака = 225.1818181818182
```

Рисунок 10 Вывод программы Knapsack в консоль

Вывод: В ходе выполнения лабораторной работы были исследованы некоторые из примеров жадных алгоритмов, решающих такие задачи как: задача о покрытии точек минимальным количеством отрезков, задача о нахождении максимального количества попарно непересекающихся отрезков и др.