

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:
Болуров Ислам Расулович
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, 2024 г.

Порядок выполнения работы:

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

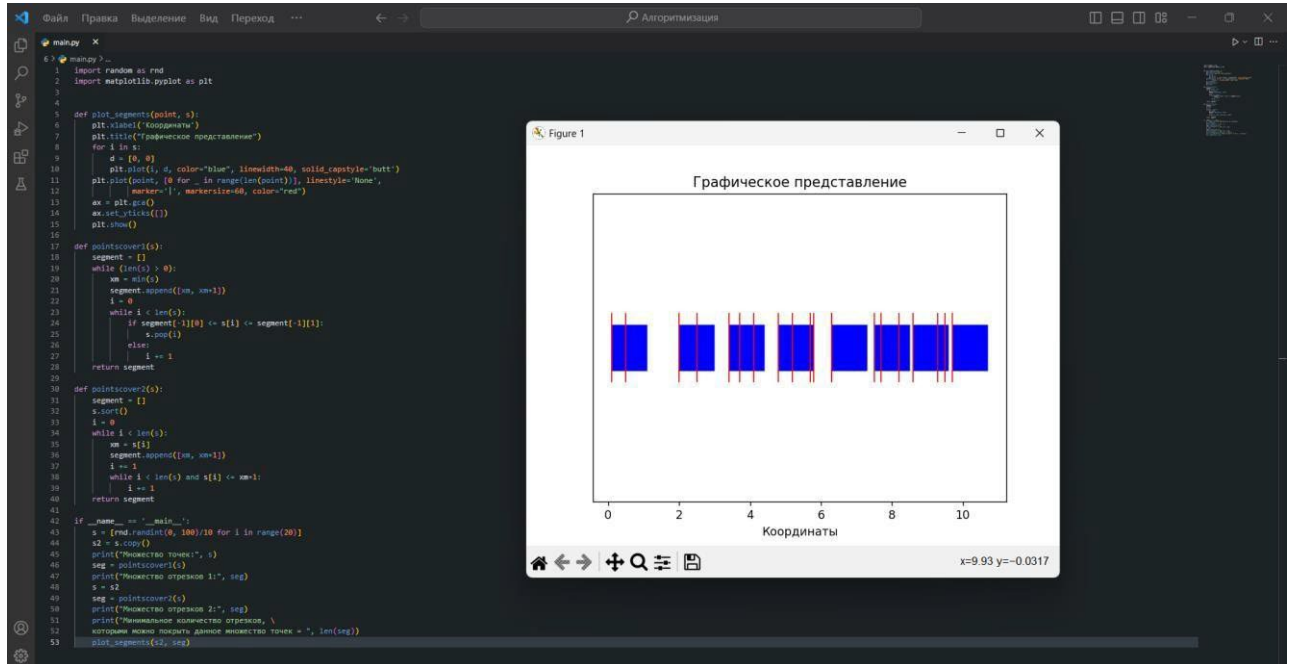


Рисунок 1. Код и результат работы программы main

```
PS C:\Users\dalam\OneDrive\Рабочий стол\projects\projects\Алгоритмизация> python -u "c:\Users\dalam\OneDrive\Рабочий стол\
Множество точек: [2.0, 0.5, 5.7, 4.1, 5.8, 7.5, 3.4, 5.2, 7.7, 0.1, 3.7, 2.5, 9.3, 6.3, 9.7, 8.2, 6.3, 9.5, 8.6, 4.8]
Множество отрезков 1: [[0.1, 1.1], [2.0, 3.0], [3.4, 4.4], [4.8, 5.8], [6.3, 7.3], [7.5, 8.5], [8.6, 9.6], [9.7, 10.7]]
Множество отрезков 2: [[0.1, 1.1], [2.0, 3.0], [3.4, 4.4], [4.8, 5.8], [6.3, 7.3], [7.5, 8.5], [8.6, 9.6], [9.7, 10.7]]
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.

```

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 actsel1(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)
46
47     plt.figure(1)
48     plt.title("Обычный алгоритм")
49     print("Массив отрезков: ", s)
50     sol = actsel1(s)
51     plot_segments(s, sol)
52     print("\n\nИтерационный алгоритм")
53     s = cps.copy()
54     plt.figure(2)
55     plt.title("Улучшенный алгоритм")
56     sol = actsel2(s)
57     print("Массив отсортированных отрезков: ", s)
58     print("Массив непересекающихся отсортированных отрезков: ", sol)
59     plot_segments(s, sol)
60     plt.show()

```

Рисунок 3. Код программы ActSel

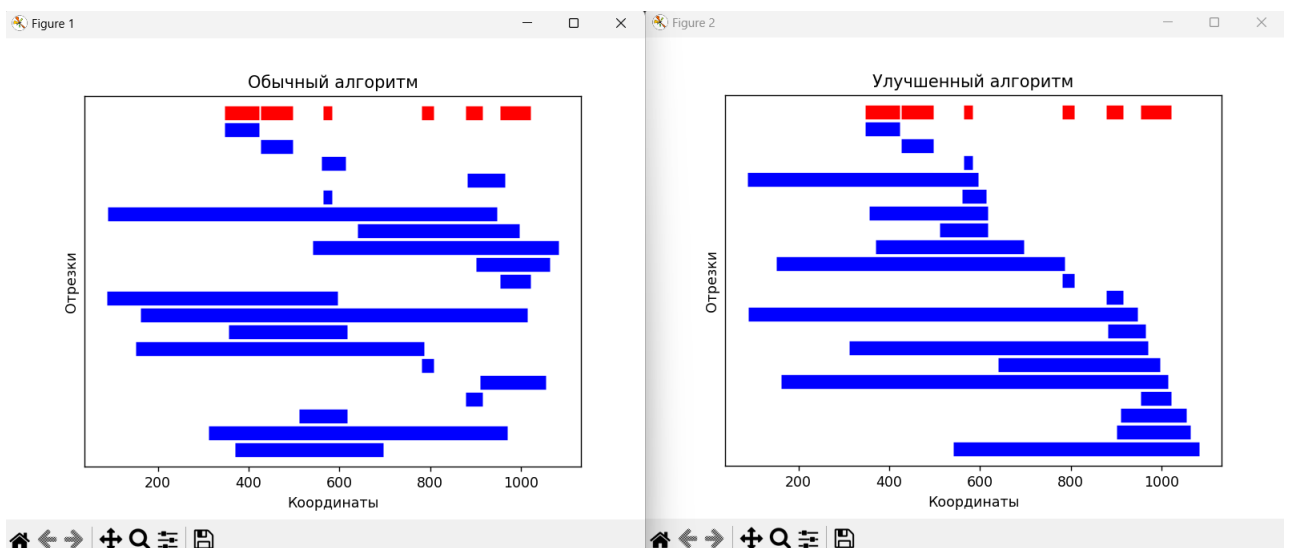


Рисунок 4. Графическое представление решения задачи обоих алгоритмов

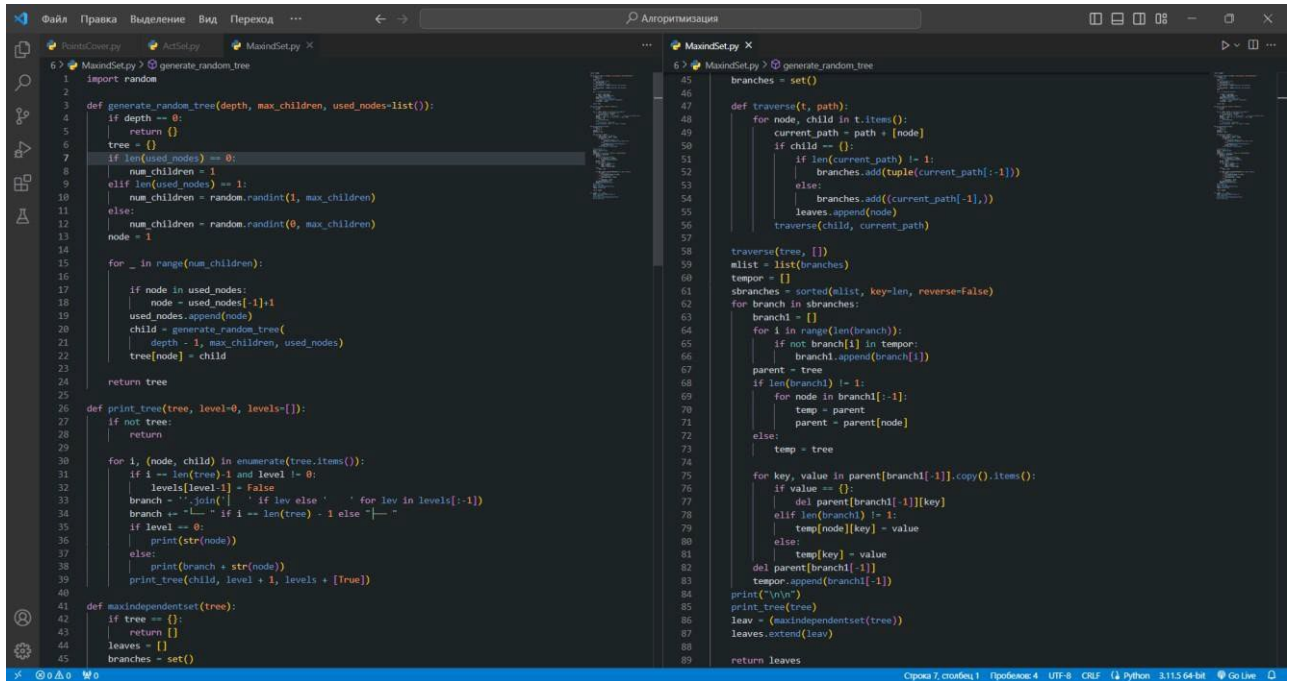
```

PS C:\Users\dalam\OneDrive\Рабочий стол\projects\projects\Алгоритмизация> python -u "c:\Users\dalam\OneDrive\Рабочий стол\projects\projects\Алгоритмизация\6\ActSel.py"
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 отсортированных 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 отсортированные отрезки: [[347, 423], [427, 498], [564, 583], [782, 808], [878, 915], [954, 1022]]

```

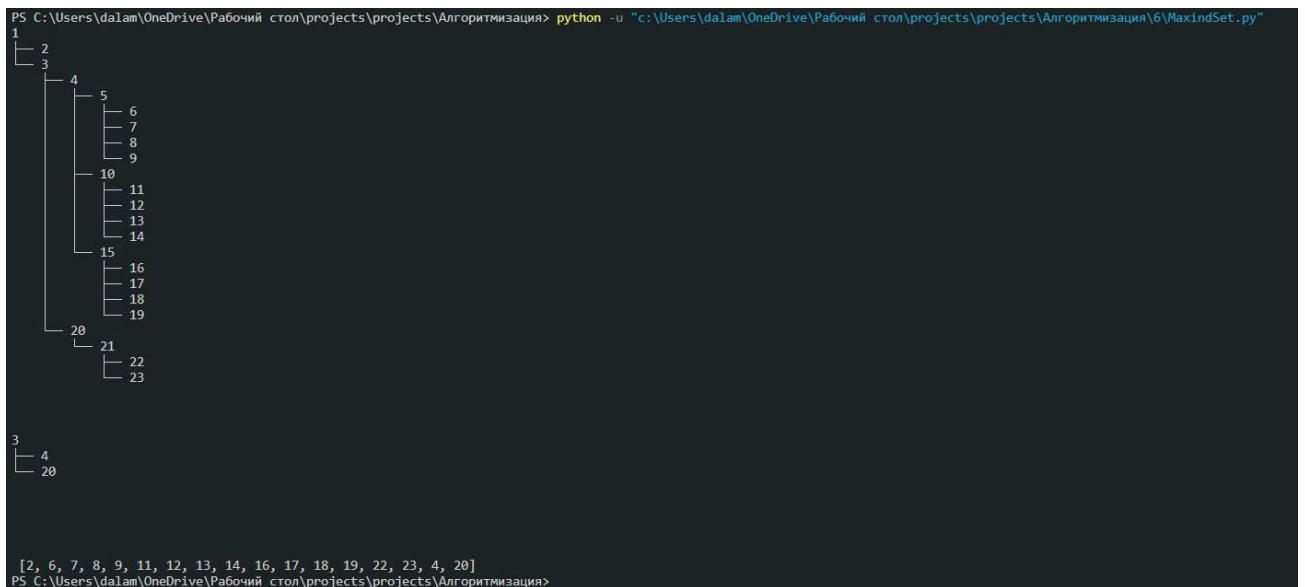
Рисунок 5. Вывод программы ActSel в терминал

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



```
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    for _ in range(num_children):
15        if node in used_nodes:
16            node = used_nodes[-1] + 1
17        used_nodes.append(node)
18        child = generate_random_tree(
19            depth - 1, max_children, used_nodes)
20        tree[node] = child
21    return tree
22
23 def print_tree(tree, level=0, levels=[]):
24     if not tree:
25         return
26     for i, (node, child) in enumerate(tree.items()):
27         if i == len(tree) - 1 and level != 0:
28             levels[level-1] = False
29             branch = ''.join(' ' if lev else ' ' for lev in levels[:level-1])
30             branch += '-|--' if i == len(tree) - 1 else '-|- '
31             if level == 0:
32                 print(str(node))
33             else:
34                 print(branch + str(node))
35             print_tree(child, level + 1, levels + [True])
36
37 def maxindependentset(tree):
38     if tree == {}:
39         return []
40     leaves = []
41     branches = set()
42
43     def traverse(t, path):
44         for node, child in t.items():
45             current_path = path + [node]
46             if child == {}:
47                 if len(current_path) != 1:
48                     branches.add(tuple(current_path[:-1]))
49             else:
50                 branches.add((current_path[-1],))
51                 leaves.append(node)
52                 traverse(child, current_path)
53
54     traverse(tree, [])
55     nlist = list(branches)
56     tempor = []
57     sbranches = sorted(nlist, key=len, reverse=False)
58     for branch in sbranches:
59         for i in range(len(branch)):
60             if not branch[i] in tempor:
61                 branch1.append(branch[i])
62                 parent = tree
63                 if len(branch1) != 1:
64                     for node in branch1[:i]:
65                         tempor = parent
66                         parent = parent[node]
67                 else:
68                     tempor = tree
69             for key, value in parent[branch1[-1]].copy().items():
70                 if value == {}:
71                     del parent[branch1[-1]][key]
72                 elif len(branch1) != 1:
73                     tempor[node][key] = value
74             else:
75                 tempor[key] = value
76             del parent[branch1[-1]]
77             tempor.append(branch1[-1])
78     print("\n\n")
79     print_tree(tree)
80     leav = (maxindependentset(tree))
81     leaves.extend(leav)
82     return leaves
```

Рисунок 6. Код программы MaxindSet



```
PS C:\Users\dalam\OneDrive\Рабочий стол\projects\projects\Алгоритмизация> python -u "c:\Users\dalam\OneDrive\Рабочий стол\projects\projects\Алгоритмизация\6\MaxindSet.py"
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
3
4
20
[2, 6, 7, 8, 9, 11, 12, 13, 14, 16, 17, 18, 19, 22, 23, 4, 20]
PS C:\Users\dalam\OneDrive\Рабочий стол\projects\projects\Алгоритмизация>
```

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.


```
PS C:\Users\dalam\OneDrive\Рабочий стол\projects\projects\Алгоритмизация> python -u "c:\Users\dalam\OneDrive\Рабочий стол\


| Элемент | Стоимость | Вес |
|---------|-----------|-----|
| 0       | 31        | 13  |
| 1       | 55        | 12  |
| 2       | 11        | 9   |
| 3       | 47        | 4   |
| 4       | 84        | 3   |
| 5       | 87        | 13  |
| 6       | 72        | 6   |
| 7       | 82        | 16  |
| 8       | 67        | 18  |
| 9       | 76        | 11  |


Решение:
Элемент 4 был взят весом 3
Элемент 6 был взят весом 6
Элемент 3 был взят весом 4
Элемент 9 был взят весом 11
Элемент 5 был взят весом 6
Стоимость рюкзака = 319.15384615384613
```

Рисунок 10. Вывод программы Knapsack в консоль

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.