

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:
Кожуховский Виктор Андреевич
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 Роман Александрович

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

Порядок выполнения работы:

Написал программы для четырех жадных алгоритмов:

```
C:\Users\viktor\Desktop\сффу\алгоритмизация\Алгоритмы\1
1  #!/usr/bin/env python3
2  #coding: utf-8 -*-
3
4  import random
5
6
7  a = []
8  arr = []
9  solution = []
10 segLength = 5
11
12
13 def fillArr(numOfEl):
14     arr.clear()
15     for i in range(numOfEl):
16         arr.append(random.randint(0, 100))
17
18
19 def pointscover1(a):
20     sol = []
21     while len(a) > 0:
22         xmin = min(a)
23         i = 0
24         sol.append([xmin, xmin + segLength])
25         while i < len(a):
26             if sol[-1][0] <= a[i] <= sol[-1][1]:
27                 a.pop(i)
28             else:
29                 i += 1
30     return sol
31
32
33 def pointscover2(a):
34     a.sort()
35     i = 0
36     sol = []
37     while i < len(a):
38         sol.append([a[i], a[i] + segLength])
39         b = a[i]
40         i += 1
41         while i < len(a) and a[i] <= b + segLength:
42             i += 1
43     return sol
44
45
46 if __name__ == '__main__':
47     fillArr(20)
48     print("Точки: ", sorted(arr))
49     arr2 = arr.copy()
50     print("Отрезки func 1", pointscover1(arr))
51     print("Отрезки func 2", pointscover2(arr2))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

```
PS C:\Users\viktor> & "C:/Program Files/Python311/python.exe" c:/Users/viktor/Desktop/сффу/алгоритмизация/Ал
• Точки: [21, 21, 21, 38, 41, 43, 46, 50, 52, 59, 61, 61, 72, 73, 75, 77, 78, 89, 91, 98]
Отрезки func 1 [[21, 26], [38, 43], [46, 51], [52, 57], [59, 64], [72, 77], [78, 83], [89, 94], [98, 103]]
Отрезки func 2 [[21, 26], [38, 43], [46, 51], [52, 57], [59, 64], [72, 77], [78, 83], [89, 94], [98, 103]]
```

Рисунок 1. Алгоритмы pointscover и pointscover улучшенный

```

1  #!/usr/bin/env python3
2  #coding: utf-8 -*-
3
4  import random
5
6
7  a = []
8  arr = []
9  solution = []
10 seglength = 5
11
12
13 def fillArr(numOfEl):
14     arr.clear()
15     for _ in range(numOfEl):
16         first = random.randint(0, 100)
17         second = first + random.randint(1, 40) # Генерируем правый конец отрезка
18         arr.append((first, second))
19
20
21 def actsel(s):
22     solution = []
23     while len(s) > 0:
24         lmin, rmin = s[0][0], s[0][1]
25         for seg in s:
26             if seg[1] < rmin:
27                 lmin, rmin = seg[0], seg[1]
28
29         solution.append([lmin, rmin])
30
31         s = [x for x in s if x[0] > rmin or x[1] < lmin]
32     return solution
33
34
35 def actsell(s):
36     s.sort(key=lambda x: x[1])
37     sol = []
38     while len(s) > 0:
39         lmin, rmin = s[0]
40         sol.append([lmin, rmin])
41         # Удаление отрезков, пересекающихся с [lmin, rmin]
42         s = [x for x in s if x[0] > rmin]
43     return sol
44
45 if __name__ == '__main__':
46     fillArr(20)
47     print("Отрезки: ", sorted(arr))
48     arr1 = arr.copy()
49     print("Отрезки непересекающиеся", actsel(arr))
50     print("Отрезки непересекающиеся", actsell(arr1))

```

PS C:\Users\viktor> "C:\Program Files\Python311\python.exe" c:\Users\viktor\Desktop\сфд\алгоритмизация\AlgLab6\2.py
Отрезки: [(6, 35), (13, 30), (14, 26), (14, 39), (16, 44), (22, 25), (23, 24), (24, 37), (42, 74), (57, 58), (59, 78), (60, 61), (65, 98), (66, 104), (67, 72), (67, 76), (69, 78), (87, 124), (98, 112), (99, 120)]
Отрезки непересекающиеся [[23, 24], [57, 58], [60, 61], [67, 72], [98, 112]]
Отрезки непересекающиеся [[23, 24], [57, 58], [60, 61], [67, 72], [98, 112]]

Рисунок 2. Алгоритмы ActSel и ActSel улучшенный

```

8
9 num_of_nodes = 90
10 solution = []
11 something = [i for i in range(1, num_of_nodes + 1)]
12
13 def MaxIndependentSet(T):
14     independent_set = set()
15
16     while T:
17         leaves = set()
18         parents = set()
19
20         for node, parent in T:
21             if all(other_parent != node for _, other_parent in T) and node not in independent_set:
22                 leaves.add(node)
23                 parents.add(parent)
24
25         independent_set.update(leaves)
26
27         T = [(node, parent) for node, parent in T if node not in leaves and parent not in leaves]
28     solution = [elem for elem in something if elem not in independent_set]
29
30     return solution
31
32
33 def create_random_tree(n):
34     tree = []
35     for i in range(2, n + 1):
36         parent = random.randint(1, i - 1)
37         tree.append((parent, i))
38     return tree
39
40
41 def display_tree(tree):
42     G = nx.DiGraph()
43     G.add_edges_from(tree)
44     pos = graphviz_layout(G, prog='dot', root=1)
45
46     plt.figure(figsize=(8, 8))
47     nx.draw_networkx(G, pos, with_labels=True, node_size=90, node_color='lightblue', font_size=8, font_color='black')
48     plt.tight_layout()
49     plt.show()
50
51
52 if __name__ == '__main__':
53     random.seed(4)
54     random_tree = create_random_tree(num_of_nodes)
55
56     independent_set = MaxIndependentSet(random_tree)
57     print("Максимальное независимое множество:", independent_set)
58
59     display_tree(random_tree)

```

PS C:\Users\viktor> "C:\Program Files\Python311\python.exe" c:\Users\viktor\Desktop\сфд\алгоритмизация\AlgLab6\3.py
Максимальное независимое множество: [3, 10, 14, 23, 30, 32, 35, 38, 41, 43, 45, 47, 48, 50, 51, 52, 54, 55, 58, 59, 62, 63, 64, 65, 66, 68, 70, 71, 73, 74, 75, 77, 78, 79, 80, 81, 83, 84, 85, 86, 87, 88, 89, 90]

Рисунок 3. Алгоритм MaxIndependentSet

```
6
7 a = []
8 arr = []
9 solution = []
10 totalcost = []
11
12
13 def fillArr(numOfEl):
14     arr.clear()
15     for _ in range(numOfEl):
16         w = random.randint(10, 500)
17         c = random.randint(1, 200) # Генерируем правый конец отрезка
18         arr.append((w, c))
19
20
21 def knapsack(a):
22     a.sort(key=lambda x: x[1]/x[0])
23     for i in range(len(a)):
24         freeSpace = 5000
25         cnt, tc = 0, 0
26         while freeSpace > (0 + a[i][0]):
27             freeSpace -= a[i][0]
28             cnt += 1
29             tc += a[i][1]
30         solution.append(cnt)
31         totalcost.append(tc)
32     return solution, totalcost
33
34
35 if __name__ == '__main__':
36     fillArr(12)
37     print("Предметы (вес, стоимость): ", sorted(arr, key=lambda x: x[1] / x[0]))
38     sol, total = knapsack(arr)
39     arr.sort(key=lambda x: x[1]/x[0])
40     print("\nМесто в рюкзаке = 5000 грамм")
41     print("Вес (грамм)\tКоличество помещившихся предметов\tЦена")
42     for i in range(len(sol)):
43         print(f"{arr[i][0]}\t\t\t{sol[i]}\t\t\t{total[i]}")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

PS C:\Users\viktor> & "C:/Program Files/Python311/python.exe" c:/Users/viktor/Desktop/скфу/алгоритмизация/AlgLab6/4.py

Предметы (вес, стоимость): [(386, 2), (385, 7), (377, 27), (73, 19), (303, 106), (482, 185), (258, 114), (100, 45), (106, 54), (157, 144), (86, 89), (17, 180)]

Место в рюкзаке = 5000 грамм

Вес (грамм)	Количество помещившихся предметов	Цена
386	12	24
385	12	84
377	13	351
73	68	1292
303	16	1696
482	10	1850
258	19	2166
100	49	2205
106	47	2538
157	31	4464
86	58	5162
17	294	52920

Рисунок 4. Алгоритм KnapSack

Вывод: в результате выполнения лабораторной работы были изучены четыре жадных алгоритма.