

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»
Вариант 5.

Выполнила:
Михеева Елена Александровна
2 курс, группа ИВТ-б-з-20-1,
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....

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

Порядок выполнения работы.

1. Были написана программа, которая на вход принимает множество точек, а выводит минимальное количество отрезков единичной длины, которые покрывают все точки.

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  # Покрывтие точек отрезками
5  # Наивный алгоритм
6
7  def PointsCover(S):
8      result = []
9      while len(S) > 0:
10         Xm = min(S)
11         result.append([Xm, Xm + 1])
12         while Xm in S:
13             S.remove(Xm)
14         while Xm+1 in S:
15             S.remove(Xm+1)
16     return result
17
18
19 if __name__ == '__main__':
20     S = [3, 8, 2, 9, 1, 2, 5, 5, 6]
21     print(PointsCover(S))
22
```

ПРОБЛЕМЫ ТЕРМИНАЛ ... + ∨ Python Debug Console [] [] ... ^

(base) mikheeva@MacBook-Pro-Elena Algorithm_6 % python3 Ex_1.1.py
[[1, 2], [3, 4], [5, 6], [8, 9]]

Рисунок 1. Покрывтие точек отрезками (наивный алгоритм)

2. Написана программа для покрытия точек единичными отрезками по улучшенному алгоритму.

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  # Покрывтие точек отрезками
5  # Улучшенный алгоритм
6
7
8  def PointsCover(S):
9      S.sort()
10     result = []
11     i = 0
12     while i < len(S):
13         Xm = S[i]
14         result.append([S[i], S[i] + 1])
15         i += 1
16         while i < len(S) and S[i] <= Xm + 1:
17             i += 1
18     return result
19
20
21 if __name__ == '__main__':
22     S = [3, 8, 2, 9, 1, 2, 5, 5, 6]
23     print(PointsCover(S))
24
```

ПРОБЛЕМЫ ТЕРМИНАЛ ... + ∨ Python Debug Console [] [] ... ^

(base) mikheeva@MacBook-Pro-Elena Algorithm_6 % python3 Ex_1.2.py
[[1, 2], [3, 4], [5, 6], [8, 9]]

Рисунок 2. Покрывтие точек отрезками (улучшенный алгоритм)

3. Была написана программа, которая на вход получает множество отрезков на прямой, а выводит максимальное количество попарно не пересекающихся отрезков.

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  # Задача о выборе заявок
5  # Наивный алгоритм
6
7
8  def ActSel(S):
9      result = []
10     while len(S) != 0:
11         min_right_end = min(S, key=lambda x: x[1])
12         result.append(min_right_end)
13         S = [segment for segment in S if segment[0] > min_right_end[1]]
14
15     return result
16
17
18 if __name__ == '__main__':
19     S = [(2, 3), (2, 4), (3, 5), (4, 6), (1, 2)]
20     print(ActSel(S))
21
```

ПРОБЛЕМЫ ТЕРМИНАЛ ... + - Python Debug Console [] [X] ... ^

(base) mikheev@MacBook-Pro-Elena Algorithm_6 % python3 Ex_2.1.py
[(1, 2), (3, 5)]

Рисунок 3. Задача о выборе заявок (наивный алгоритм)

4. Написали программу для «выбора заявок» по улучшенному алгоритму.

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  # Задача о выборе заявок
5  # Улучшенный алгоритм
6
7
8  def ActSel(S):
9      S.sort(key=lambda x: x[1])
10     result = []
11     last_end = float('-inf')
12     for segment in S:
13         if segment[0] > last_end:
14             result.append(segment)
15             last_end = segment[1]
16     return result
17
18
19 if __name__ == '__main__':
20     S = [(2, 3), (2, 4), (3, 5), (4, 6), (1, 2)]
21     print(ActSel(S))
22
```

ПРОБЛЕМЫ ТЕРМИНАЛ ... + - Python Debug Console [] [X] ... ^

python3: error: SDK "macosx" cannot be located
[(1, 2), (3, 5)]

Рисунок 4. Задача о выборе заявок (улучшенный алгоритм)

5. Была реализована программа, которая на вход получает дерево, а на выходе максимальное независимое множество вершин. Для этого была написана функция `maxIndependentSet()`, которая пока граф не пустой делит его на 2 множества вершин: родителей и детей. С помощью разности этих множеств ищет листья графа. Затем удаляет из графа листья и их родителей.

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  # Планирование вечеринки в компании
5
6
7  def delete(T, leave, parents):
8      result = []
9      fordelete = leave | parents
10
11     for segment in T:
12         if segment[1] not in fordelete:
13             result.append(segment)
14
15     return result
16
17
18  def maxIndependentSet(T):
19      result = set()
20      vertices = set(range(1, len(T)+1))
21
22      while len(T) != 0:
23          # Определила 2 множества с родителями и детьми
24          par = [segment[0] for segment in T]
25          child = [segment[1] for segment in T]
26
27          # Множество листьев
28          leave = set(child).difference(set(par))
29          result = result | leave
30
31          # Определила родителей для листьев
32          leave_parents = set()
33          for segment in T:
34              if segment[1] in leave:
35                  leave_parents.add(segment[0])
36
37          # Из множества вершин удалила листья и их родителей
38          vertices -= (leave | leave_parents)
39          T = delete(T, leave, leave_parents)
40
41          if len(vertices) == 1 and len(T) == 0:
42              result |= vertices
43
44      return result
45
46
47  if __name__ == '__main__':
48      graf = ([1, 2], [1, 3], [1, 4], [2, 5], [3, 6],
49             [4, 7], [4, 8], [7, 9], [7, 10], [7, 11])
50
51      print("Ответ: ", maxIndependentSet(graf))
52

```

ПРОБЛЕМЫ ВЫХОДНЫЕ ДАННЫЕ КОНСОЛЬ ОТЛАДКИ **ТЕРМИНАЛ** ПОРТЫ + Python Debug Console

(base) mikheeva@MacBook-Pro-Elena Algorithm_6 % python3 Ex_3.py
 Ответ: {1, 5, 6, 8, 9, 10, 11}

Рисунок 5. Задача «Планирование вечеринки в компании»

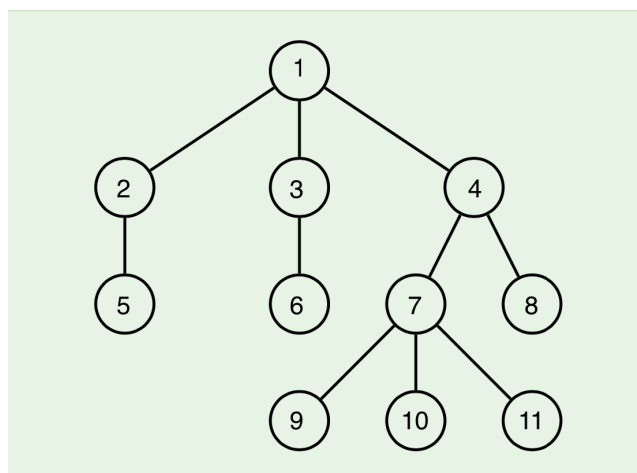


Рисунок 6. Исходный граф

6. Написали программу, которая на вход получает двумерный массив с весом и ценой предметов и вместимость рюкзака. Функция Knapsack() ищет суммарную стоимость частей предметов суммарного веса не более веса рюкзака.

```
1  #!/usr/bin/env python3
2  #-*- coding: utf-8 -*-
3
4  : Непрерывный рюкзак
5
6
7  def Knapsack(P, w):
8      result = []
9      P.sort(key=lambda x: (x[1]/x[0]), reverse=True)
10     bagw = 0
11     cost = 0
12
13     for item in P:
14         if bagw < w:
15             result.append(item)
16             bagw += item[0]
17             cost += item[1]
18
19     if bagw > w:
20         cost -= result[-1][1]
21         result[-1][1] -= (result[-1][1] / result[-1][0] * (bagw-w))
22         result[-1][0] -= (bagw - w)
23         cost += result[-1][1]
24
25     print(f"Итоговая стоимость: {cost}")
26
27     return result
28
29
30 .f __name__ == '__main__':
31     P = [[1, 3], [1, 4], [2, 5], [3, 6], [4, 7], [10, 9], [7, 10], [5,
32     W = 10
33
```

ПРОБЛЕМЫ ТЕРМИНАЛ ... + ∨ Python Debug Console □ □

Итоговая стоимость: 25.0
[[1, 4], [1, 3], [2, 5], [5, 11], [1, 2.0]]

Рисунок 7. Задача «непрерывный рюкзак»