

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»**

Выполнил:  
Степанов Леонид Викторович  
2 курс, группа ИВТ-б-о-22-1,  
09.03.01 «Информатика и  
вычислительная техника»,  
направленность (профиль)  
«Программное обеспечение  
средств вычислительной  
техники и автоматизирование  
систем», очная форма обучения

---

(подпись)

Руководитель практики:  
Воронкин Р.А., канд. техн. наук,  
доцент, доцент кафедры  
инфокоммуникаций

---

(подпись)

Отчет защищен с оценкой \_\_\_\_\_ Дата защиты \_\_\_\_\_

Ставрополь, 2023 г.

Тема: жадные алгоритмы

Порядок выполнения работы:

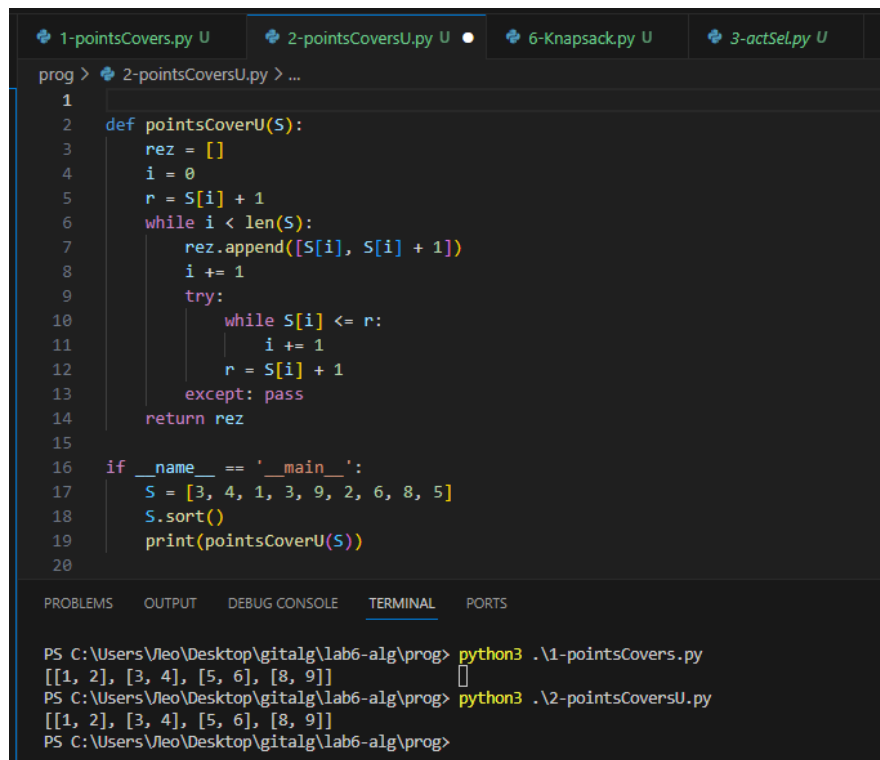
1. Написал программу (1-pointsCovers.py), которая на вход принимает множество точек, а выводит минимальное количество отрезков единичной длины, которыми можно покрыть все точки. Алгоритм на Python заключается в том, что пока размер входного массива данных не равен нулю: мы находим минимальное значение  $X_m$ , добавляем к решению отрезок  $[X_m, X_m+1]$  и удаляем все точки, которые пересекает данный отрезок в массиве.

```
1-pointsCovers.py U X
prog > 1-pointsCovers.py > ...
1
2 def removeS(S, x):
3     while x in S:
4         S.remove(x)
5
6 def pointsCover(S) -> list:
7     rez = []
8     while len(S) != 0:
9         Xm = int(min(S))
10        rez.append([Xm, Xm + 1])
11        removeS(S, Xm)
12        removeS(S, Xm + 1)
13    return rez
14
15 if __name__ == '__main__':
16     S = [3, 4, 1, 3, 1, 2, 6, 8, 5]
17     print(pointsCover(S))

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\Leo\Desktop\gitalg\lab6-alg\prog> python3 .\1-pointsCovers.py
[[1, 2], [3, 4], [5, 6], [8, 9]]
PS C:\Users\Leo\Desktop\gitalg\lab6-alg\prog> 
```

Рисунок 1 – Результат выполнения программы 1-pointsCovers.py

2. Написал программу (2-pointsCoversU.py), которая на вход принимает множество точек, а выводит минимальное количество отрезков единичной длины, которыми можно покрыть все точки. Алгоритм на Python 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  $i$ .



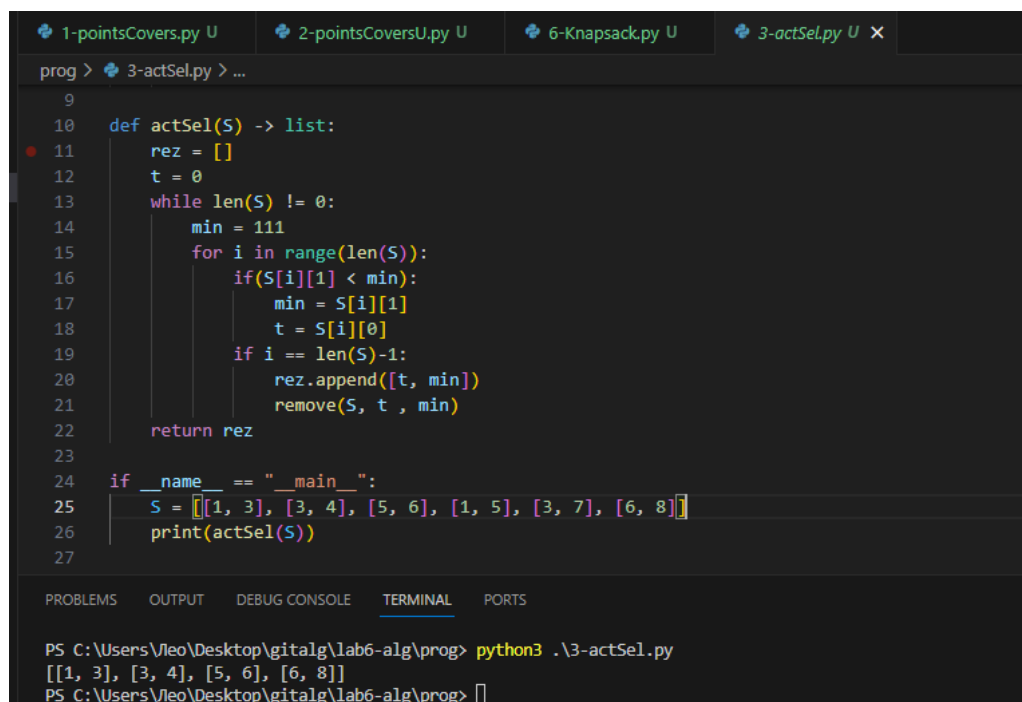
```
1-pointsCovers.py U 2-pointsCoversU.py U 6-Knapsack.py U 3-actSel.py U
prog > 2-pointsCoversU.py > ...
1
2 def pointsCoverU(S):
3     rez = []
4     i = 0
5     r = S[i] + 1
6     while i < len(S):
7         rez.append([S[i], S[i] + 1])
8         i += 1
9         try:
10             while S[i] <= r:
11                 i += 1
12                 r = S[i] + 1
13         except: pass
14     return rez
15
16 if __name__ == '__main__':
17     S = [3, 4, 1, 3, 9, 2, 6, 8, 5]
18     S.sort()
19     print(pointsCoverU(S))
20

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\leo\Desktop\gitalg\lab6-alg\prog> python3 .\1-pointsCovers.py
[[1, 2], [3, 4], [5, 6], [8, 9]]
PS C:\Users\leo\Desktop\gitalg\lab6-alg\prog> python3 .\2-pointsCoversU.py
[[1, 2], [3, 4], [5, 6], [8, 9]]
PS C:\Users\leo\Desktop\gitalg\lab6-alg\prog>
```

Рисунок 2 – Результат выполнения программы 2-pointsCoversU.py

3. Написал программу (3-actsel.py), которая получает на вход множество отрезков на прямой, а выводит максимальное количество попарно не пересекающихся отрезков. Алгоритм на Python заключается в том, что находится минимальный отрезок по правому концу, он добавляется в решение, те отрезки, которые его пересекают выкидываются из множества отрезков.



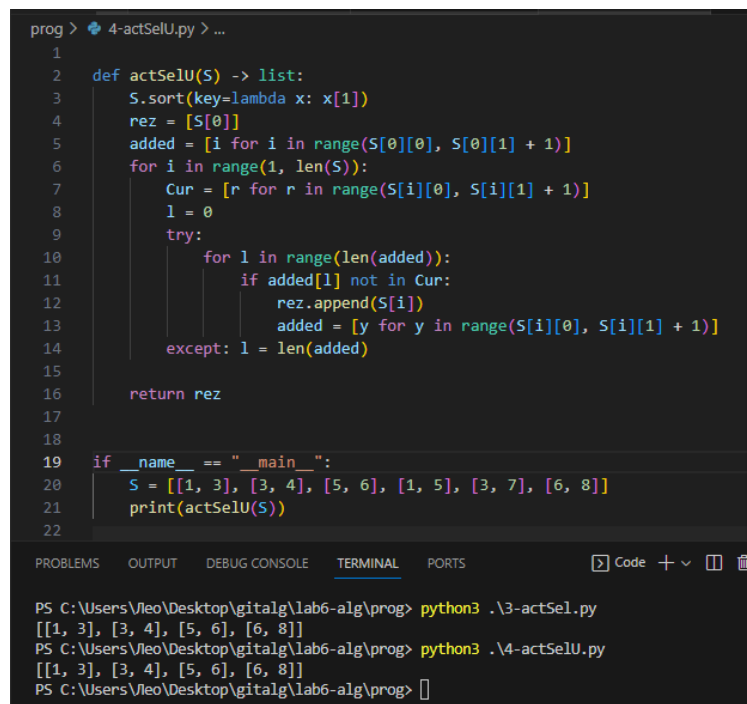
```
1-pointsCovers.py U 2-pointsCoversU.py U 6-Knapsack.py U 3-actSel.py U X
prog > 3-actSel.py > ...
9
10 def actSel(S) -> list:
11     rez = []
12     t = 0
13     while len(S) != 0:
14         min = 111
15         for i in range(len(S)):
16             if S[i][1] < min:
17                 min = S[i][1]
18                 t = S[i][0]
19             if i == len(S)-1:
20                 rez.append([t, min])
21                 remove(S, t, min)
22     return rez
23
24 if __name__ == '__main__':
25     S = [[1, 3], [3, 4], [5, 6], [1, 5], [3, 7], [6, 8]]
26     print(actSel(S))
27

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\leo\Desktop\gitalg\lab6-alg\prog> python3 .\3-actSel.py
[[1, 3], [3, 4], [5, 6], [6, 8]]
PS C:\Users\leo\Desktop\gitalg\lab6-alg\prog>
```

Рисунок 3 – Результат выполнения программы 3-actSel.py

4. Написал программу (4-actSelU.py), которая получает на вход множество отрезков на прямой, а выводит максимальное количество попарно не пересекающихся отрезков. Алгоритм на Python заключается в том, что сначала сортируется массив по второму элементу: концу отрезков, далее добавляется в результат и в «последний добавленный» самый первый элемент отсортированного массива, далее происходит перебор всех элементов в массиве, в котором проверяется пересекается ли последний добавленный отрезок с текущим.



```
prog > 4-actSelU.py > ...
1
2 def actSelU(S) -> list:
3     S.sort(key=lambda x: x[1])
4     rez = [S[0]]
5     added = [i for i in range(S[0][0], S[0][1] + 1)]
6     for i in range(1, len(S)):
7         Cur = [r for r in range(S[i][0], S[i][1] + 1)]
8         l = 0
9         try:
10             for l in range(len(added)):
11                 if added[l] not in Cur:
12                     rez.append(S[i])
13                     added = [y for y in range(S[i][0], S[i][1] + 1)]
14             except: l = len(added)
15
16     return rez
17
18
19 if __name__ == "__main__":
20     S = [[1, 3], [3, 4], [5, 6], [1, 5], [3, 7], [6, 8]]
21     print(actSelU(S))
22
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Code + -

```
PS C:\Users\Ileo\Desktop\gitalg\lab6-alg\prog> python3 .\3-actSel.py
[[1, 3], [3, 4], [5, 6], [6, 8]]
PS C:\Users\Ileo\Desktop\gitalg\lab6-alg\prog> python3 .\4-actSelU.py
[[1, 3], [3, 4], [5, 6], [6, 8]]
PS C:\Users\Ileo\Desktop\gitalg\lab6-alg\prog>
```

Рисунок 4 – Результат выполнения программы 4-actSelU.py

5. Написал программу (5-maxIndependentSet.py), которая получает на вход дерево, а на выходе независимое множество. Алгоритм на Python заключается в том, что сначала находится максимальное число в массиве состоящем из ребер графа, потом пока этот массив не пуст, создаётся массив локальных решений, в который добавляются элементы графа которые имеют 1 связь, а эта связь проверяется функцией countElements(T, num), 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.

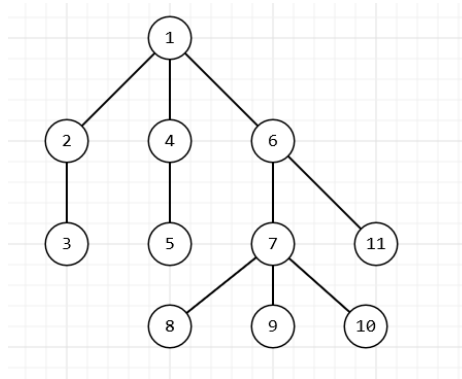


Рисунок 5 – Граф

```

prog > 5-maxIndependentSet.py > ...
1 def delete(T, r):
2     i = 0
3     while i != len(T):
4         if T[i][1] in r:
5             T.pop(i)
6             i = i - 1
7         i = i + 1
8
9 def maximum(T) -> int:
10    max = 0
11    for i in range(len(T)):
12        for j in range(2):
13            if T[i][j] > max:
14                max = T[i][j]
15    return max
16
17 def countElements(T, num) -> int:
18    count = 0
19    for i in range(len(T)):
20        if T[i][0] == num or T[i][1] == num:
21            count += 1
22    return count
23
24 def maxIndependentSet(T) -> list:
25    rez = []
26    a = maximum(T)
27    while len(T) != 0:
28        r = []
29        for i in range(a, 0, -1):
30            t = countElements(T, i)
31            if t == 1:
32                r.append(i)
33            delete(T, r)
34            rez.append(r)
35    return rez
36
37
38 if __name__ == "__main__":
39     T = [[1, 2], [2, 3], [1, 4], [4, 5], [1, 6], [6, 7], [7, 8], [7, 9], [7, 10], [6, 11]]
40     print(maxIndependentSet(T))
  
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS


```

PS C:\Users\leo\Desktop\gitalg\lab6-alg\prog> python3 .\5-maxIndependentSet.py
[[11, 10, 9, 8, 5, 3], [7, 4, 2], [6, 1]]
PS C:\Users\leo\Desktop\gitalg\lab6-alg\prog>
  
```

Рисунок 6 – Результат выполнения программы 5-maxIndependentSet.py

6. Написал программу (6-Knapsack.py), которая на вход принимает двумерный массив, состоящий из веса и ценности предмета, максимальный вес предметов, а на выход самый эффективную по стоимости комбинацию предметов. Алгоритм на Python заключается в том, что данный массив

сортируется по возрастанию по отношению ценности к весу предмета. Далее каждый элемент массива с конца добавляется к массиву решения предварительно проверяя текущий вес, если текущий вес равен максимальному – цикл прекращается, если текущий вес больше максимального, то ищется избыток веса, и находится коэффициент уменьшения веса и ценности. Он будет равняться весу элемента массива, при котором общая масса больше максимальной минус избыток веса в минус первой степени и к решению добавляется значения текущего элемента, умноженные на коэффициент уменьшения. Если оба условия не выполняются, то текущий элемент добавляется к решению.



```
1-pointsCovers.py U 2-pointsCoversU.py U 6-Knapsack.py U x 5
prog > 6-Knapsack.py > Knapsack
1
2 def Knapsack(T, W) -> list:
3     rez = []
4     curW = 0
5     T.sort(key=lambda x: (x[1] / x[0]))
6     print(T)
7     i = len(T) - 1
8     while i >= 0:
9         curW += T[i][0]
10        if curW == W:
11            break
12        if curW > W:
13            t = 0
14            for j in range(curW, W, -1):
15                t += 1
16            coef = 1/(T[i][0] - t)
17            w = T[i][0]*coef
18            p = T[i][1]*coef
19            rez.append([w, p])
20        else:
21            rez.append(T[i])
22            i = i - 1
23    return rez
24
25 if __name__ == "__main__":
26     T = [[2, 14], [4, 20], [3, 18], [5, 30]]
27     W = 12
28     print(Knapsack(T, W))
29
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS [x] Code +
PS C:\Users\leo\Desktop\gitalg\lab6-alg\prog> python3 .\6-Knapsack.py
[[4, 20], [3, 18], [5, 30], [2, 14]]
[[2, 14], [5, 30], [3, 18], [2.0, 10.0]]
PS C:\Users\leo\Desktop\gitalg\lab6-alg\prog>
```

Рисунок 7- Результат выполнения программы 6-Knapsack.py