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»

Выполнила:
Беседина Инга Олеговна
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и вычислительная техника (профиль) «Программное обеспечение средств вычислительной техники и автоматизированных систем», очная форма обучения

_____
(подпись)

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

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023 г.

Порядок выполнения работы:

Алгоритм 1. Покрытие точек отрезками

```
import numpy as np
import matplotlib.pyplot as plt


def pointsCover(points):
    lines = []
    points = sorted(points)
    n = len(points)
    i = 0
    while i < n:
        l = points[i]
        r = points[i] + 1
        i += 1
        while i < n and points[i] <= r + 1:
            r = points[i]
            i += 1
        lines.append([l, r])
    return lines


dots = np.random.choice(15, 8, replace=False)
print(sorted(dots))
print(pointsCover(dots))
```

```
[0, 2, 3, 5, 7, 8, 10, 11]
[[0, 3], [5, 8], [10, 11]]
```

Рисунок 1. Точки и отрезки

Алгоритм 2. Поиск максимального количества попарно непересекающихся отрезков

```
def act_sel(segments):
    sorted_segments = sorted(segments, key=lambda x: x[1])
    n = len(sorted_segments)
    solution = []
    for i in range(n):
        if not solution or sorted_segments[i][0] > solution[-1][1]:
            solution.append(sorted_segments[i])
    return solution


lines = [[5, 10],  [20, 25], [7, 11], [32,37], [9, 16], [1, 2], [14, 17],
[18, 21]]
print("Неотсортированные отрезки:", lines)
sorted_lines = act_sel(lines)
print("Отсортированные отрезки:",sorted_lines)
```

```
Неотсортированные отрезки: [[5, 10], [20, 25], [7, 11], [32, 37], [9, 16], [1, 2], [14, 17], [18, 21]]
Отсортированные отрезки: [[1, 2], [5, 10], [14, 17], [18, 21], [32, 37]]
```

Рисунок 2. Результат работы алгоритма

Алгоритм 3. Максимального размера множество не соединенных друг с другом вершин

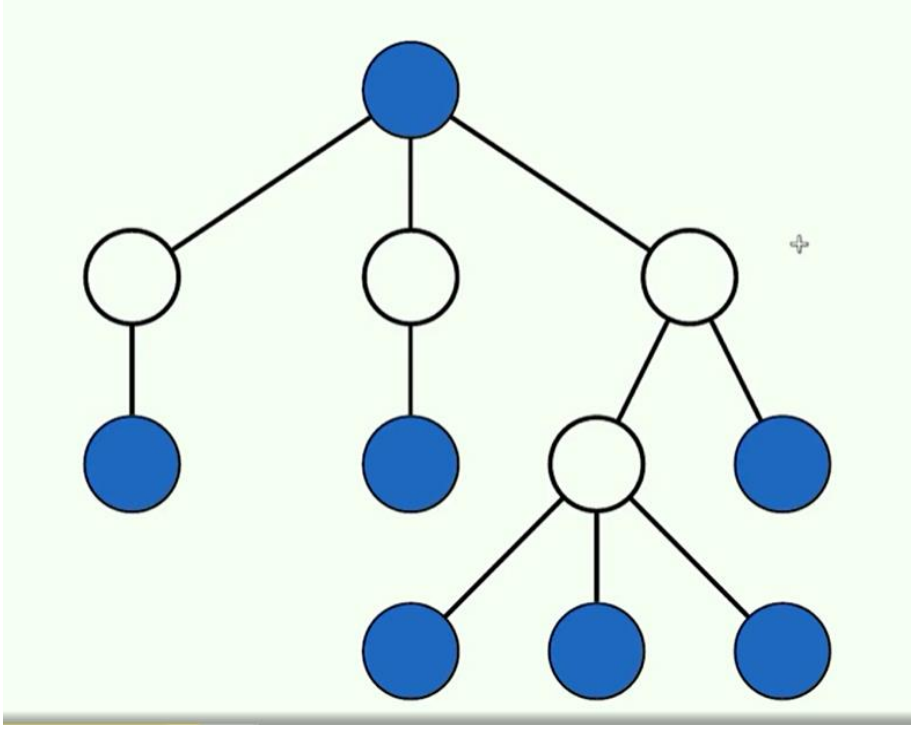

Рисунок 3. Дерево 1

```
import numpy as np


def max_set(T):
    sol = []
    leaves = [node for node in T if T[node]['children_id'] == []]

    for leaf in leaves:
        for node in T:
            if (T[leaf]['lvl'] - T[node]['lvl'] == 2) and (node not in sol)
and (node not in leaves):
                sol.append(node)
    for leaf in leaves:
        for s in sol:
            if T[leaf]['id'] in T[s]['children_id']:
                sol.remove(s)

    sol = np.hstack([sol, leaves])
    return sol


T = {
    'A': {'id': 0, 'lvl': 1, 'parent_id': None, 'children_id': [1, 2, 3]},
    'B': {'id': 1, 'lvl': 2, 'parent_id': 0, 'children_id': [4]},
    'C': {'id': 2, 'lvl': 2, 'parent_id': 0, 'children_id': [5]},
    'D': {'id': 3, 'lvl': 2, 'parent_id': 0, 'children_id': [6, 7]},
    'E': {'id': 4, 'lvl': 3, 'parent_id': 1, 'children_id': []},
    'F': {'id': 5, 'lvl': 3, 'parent_id': 2, 'children_id': []},
    'G': {'id': 6, 'lvl': 3, 'parent_id': 3, 'children_id': [8, 9, 10]},
    'H': {'id': 7, 'lvl': 3, 'parent_id': 3, 'children_id': []},
    'I': {'id': 8, 'lvl': 4, 'parent_id': 6, 'children_id': []},
    'J': {'id': 9, 'lvl': 4, 'parent_id': 6, 'children_id': []},
    'K': {'id': 10, 'lvl': 4, 'parent_id': 6, 'children_id': []},
```

```
}

print(max_set(T))
```

```
['A' 'E' 'F' 'H' 'I' 'J' 'K']
```

Рисунок 4. Результат работы алгоритма для дерева 1

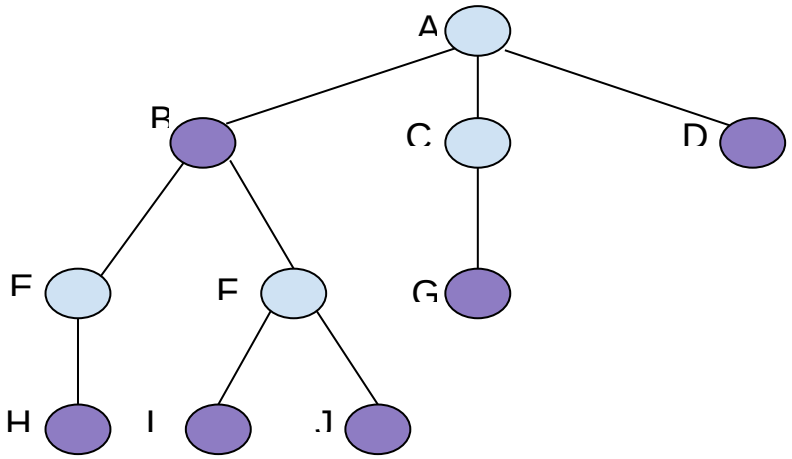

Рисунок 5. Дерево 2

```
['B' 'D' 'G' 'H' 'I' 'J']
```

Рисунок 6. Результат работы алгоритма для дерева 2

Алгоритм 4. Максимальная стоимость частей предметов суммарного веса не более w

```
def knapsack(items):
    items.sort(key=lambda x: x[0]/x[1], reverse=True)

    wmax = 7
    cmax = 0

    for c, w in items:
        for i in range(w, 0, -1):
            if wmax >= i:
                wmax -= i
                cmax += c/w*i
                break
            else:
                continue
    return cmax


print(knapsack([[18, 3], [14, 2], [20, 4]]))
print(knapsack([[35, 5], [16, 2], [14, 7], [27, 3]]))
```

```
42.0
57.0
```

Рисунок 7. Результат работы алгоритма