Виконавець: Штонда Єлизавета Олександрівна

Група: К-13/2

Варіант: 140

Викладач практичних занять: Єфремов Микола Сергійович

Пункт 2:

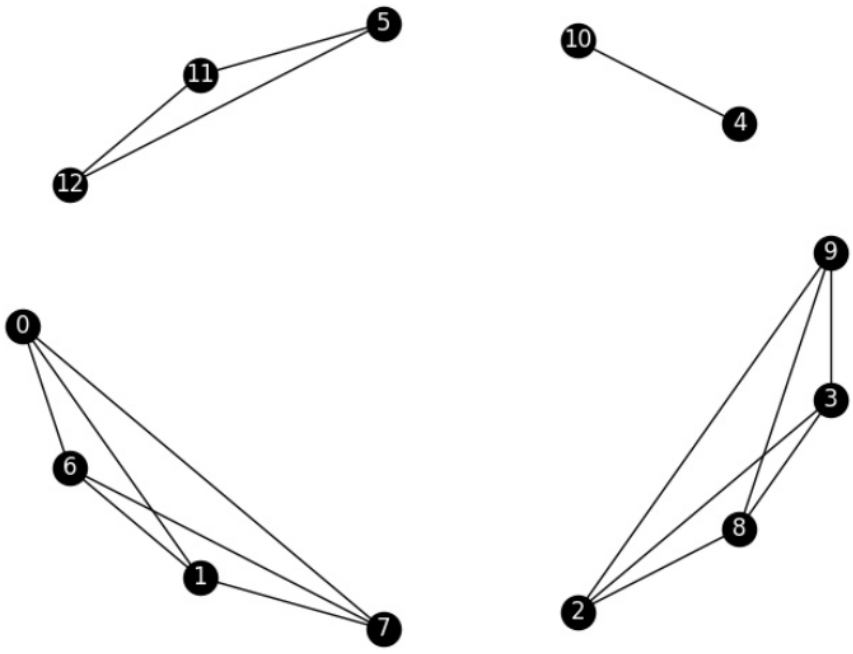
Побудувати графічний файл із зображенням графа. Граф має завантажуватись з файлу, побудованому на попередньому кроці. Використати бібліотечні засоби розташування вершин.

Програма завантажує з текстового файлу список суміжності графа за допомогою методу nx.read_adjlist(). В якості бібліотечного засобу використано метод nx.draw_shell():

```
def automatic_graph(path_from, path_to):
    g = read_adjlist(path_from)
    nx.draw_shell(g, with_labels=True, font_color='white', node_color='black')
    save_graph(path_to)
    return g


def save_graph(path):
    plt.savefig(path, format='jpg')
```

Результат:



Пункт 3:

Побудувати графічний файл із зображенням графа. Налаштувати розташувавши вершини графа так, щоб компоненти зв'язності розташовувались послідовно, а ребра мали якомога менше перетинів. Автоматизація розташування вершин не вимагається. При цьому заборонено використовувати бібліотечні засоби розташування вершин. Усі подальші графічні зображення будувати за допомогою цього розташування вершин.

Розташування вершин налаштовано програмно:

```
pos = {i: (i, 2) for i in range(6)}
pos.update({i: (i-6, 0) for i in range(6, 13)})
```

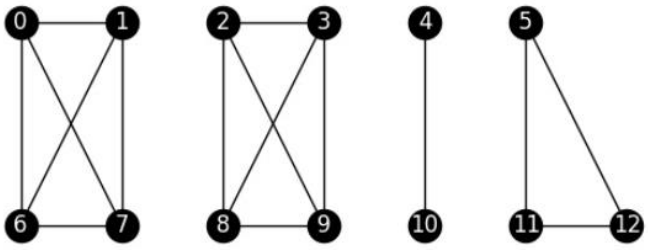
Для побудови графіка використано метод nx.draw():

```
def manual_graph(pos, path_to):
    draw_graph(make_graph(), pos)
    save_graph(path_to)


def draw_graph(graph, pos):
    plt.axes().set_aspect('equal', adjustable='datalim')
    nx.draw(graph, pos=pos, with_labels=True, font_color='white', node_color='black')


def save_graph(path):
    plt.savefig(path, format='jpg')
```

Результат:



Пункт 4:

Для графа програмно знайти і вивести в текстовому режимі по кожній його компоненті зв’язності вивести:

Кількість вершин та ребер, степені та ексцентриситети вершин, радіус та діаметр.

Знаходження всіх величин виконувалось за допомогою бібліотечних засобів:

```python
def general_information(g):
    h = 0
    components = ['First', 'Second', 'Third', 'Fourth']
    for i in nx.connected_components(g):
        print(f"{components[h]} component: ", end='\n')
        name = g.subgraph(i)
        h += 1
        print('\tnumber of nodes: ', end='')
        print(len(nx.nodes(name)))
        print('\tnumber of edges: ', end='')
        print(len(nx.edges(name)))
        print('\tdegree and eccentricity of nodes: \n', end='')
        for j in i:
            print('\t', j, ': ', name.degree(j), '; ', nx.eccentricity(name, j), sep='')
        print('\tradius: ', nx.radius(name), sep='')
        print('\tdiameter: ', nx.diameter(name), sep='', end='\n\n')
```

Результат:

```
First component:
    number of nodes: 4
    number of edges: 6
    degree and eccentricity of nodes:
    0: 3; 1
    1: 3; 1
    6: 3; 1
    7: 3; 1
    radius: 1
    diameter: 1

Second component:
    number of nodes: 4
    number of edges: 6
    degree and eccentricity of nodes:
    8: 3; 1
    9: 3; 1
    2: 3; 1
    3: 3; 1
    radius: 1
    diameter: 1
```

```
Third component:
    number of nodes: 2
    number of edges: 1
    degree and eccentricity of nodes:
    10: 1; 1
    4: 1; 1
    radius: 1
    diameter: 1

Fourth component:
    number of nodes: 3
    number of edges: 3
    degree and eccentricity of nodes:
    11: 2; 1
    12: 2; 1
    5: 2; 1
    radius: 1
    diameter: 1
```

Пункт 5:

Для кожної нетривіальної компоненти зв’язності програмно знайти хоча б один діаметр. Побудувати графічне зображення , де ребра та вершини відмічені кольором.

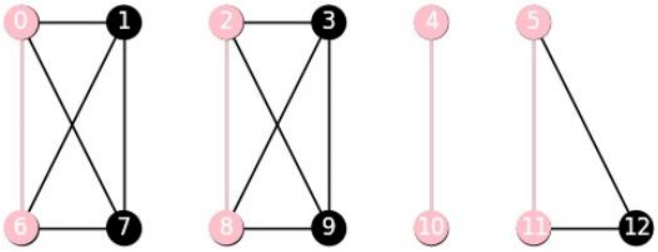
Діаметр знайдено за допомогою обходу графа вширину бібліотечним методом nx.bfs_edges():

```python
def diameter_bfs(g, color, path):
    for i in nx.connected_components(g):
        g1 = g.subgraph(i)
        diam = nx.diameter(g1)
        lst = []
        for v, w in nx.bfs_edges(g1, list(i)[0]):
            lst.append([v, w])
        lst = lst[:diam]
        v = lst[0][0]
        w = lst[0][1]
        g1[v][w]['color'] = color
        edges_colors = [color for a, b, color in g1.edges.data(data='color', default='black')]
        nodes_colors = [color for v, color in g1.nodes.data(data='color', default='black')]
        for j in (v, w):
            if j in g1.nodes:
                ind = list(g1.nodes).index(j)
                nodes_colors[ind] = color
        draw_diameter(g1, pos, edges_colors, nodes_colors)
    save_graph(path)


def draw_diameter(graph, pos,  edges_colors, nodes_colors):
    nx.draw(graph, pos=pos, with_labels=True, font_color='white', node_color=nodes_colors, edge_color=edges_colors)


def save_graph(path):
    plt.savefig(path, format='jpg')
```

Результат:



Пункт 6:

Програмно побудувати глибинний кістяковий ліс графа. Побудувати зображення графа, на якому ребра знайденого лісу виділено кольором.

Глибинний кістяковий ліс граф побудовано в результаті обходу графу вглибину за допомогою бібліотечного методу nx.dfs_edges():

```
def forest_graph(g, color):
    g1 = g.copy()
    for v, w in nx.dfs_edges(g1):
        g1[v][w]['color'] = color
    edges_colors = [color for a, b, color in g1.edges.data(data='color', default='black')]
    draw_forest_graph(g1, pos, edges_colors)
    save_graph('forest.jpg')


def draw_forest_graph(graph, pos, edges_color):
    nx.draw(graph, pos=pos, with_labels=True, font_color='white', node_color='black', edge_color=edges_color)


def save_graph(path):
    plt.savefig(path, format='jpg')
```

Результат: