Дискретні структури

ІПЗ-21-2
Губарєв Ростислав Вадимович

Лабораторна робота №12.2

Лабіринт задано так само, як у задачі 1. Задано список предметів, що знаходяться на карті. Гравцеві необхідно "зібрати" всі предмети і повернутися у вихідну точку. Знайдіть найкоротший маршрут гравця і візуалізуйте його на графі. Якщо таких маршрутів кілька виведіть будь-який із них. Якщо маршруту немає, виведіть повідомлення про це.

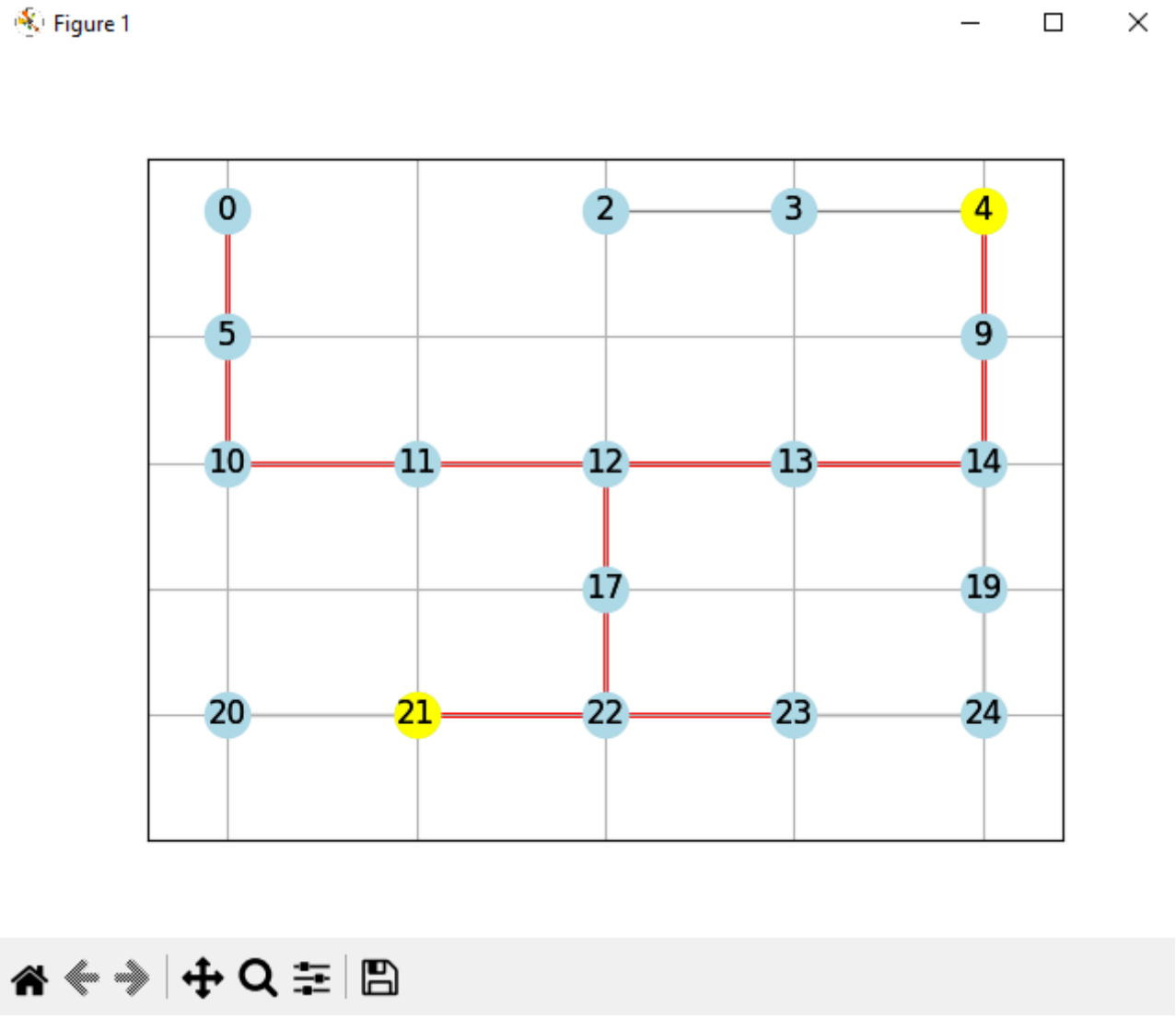
Код

```
import networkx as nx
import matplotlib.pyplot as plt

def draw_graph(graph, maze_matrix, paths=None, items=None):
    # Встановлення позицій вузлів графа відповідно до їхніх координат у матриці
    pos = dict()
    rows = len(maze_matrix)
    cols = len(maze_matrix[0])
    for i in range(rows):
        for j in range(cols):
            node = i * cols + j
            pos[node] = (j, -i)  # Використання від'ємних значень для правильного
відображення графа
    # Візуалізація вузлів та ребер графа
    nx.draw_networkx_nodes(graph, pos, node_color='lightblue')
    nx.draw_networkx_labels(graph, pos)
    nx.draw_networkx_edges(graph, pos, edge_color='gray')
    if paths:
        for path in paths:
            edges = list(zip(path, path[1:]))
            nx.draw_networkx_edges(graph, pos, edgelist=edges, edge_color='red',
width=2.0)
    if items:
        item_nodes = [node for node in graph.nodes if 'item' in graph.nodes[node]]
        nx.draw_networkx_nodes(graph, pos, nodelist=item_nodes, node_color='yellow')
        item_labels = {node: maze_matrix[node // cols][node % cols] for node in
item_nodes}
        nx.draw_networkx_labels(graph, pos)
    plt.xticks(range(cols))
    plt.yticks(range(-rows, 0))
    plt.grid(visible=True)
    plt.show()

def find_shortest_path(graph, start, end):
    # Алгоритм пошуку в ширину для знаходження найкоротшого шляху
    queue = [(start, [start])]
    while queue:
        (vertex, path) = queue.pop(0)
        for next_vertex in graph[vertex]:
            if next_vertex == end:
                return path + [next_vertex]
            else:
                queue.append((next_vertex, path + [next_vertex]))
    return None

def create_graph_from_maze(maze):
    rows = len(maze)
    cols = len(maze[0])
    graph = nx.Graph()
    for i in range(rows):
```

```
        for j in range(cols):
            if maze[i][j] != 1:
                node = i * cols + j
                graph.add_node(node)
                if maze[i][j] == 10:
                    end_node = node
                if maze[i][j] == 2 or maze[i][j] == 3:
                    graph.nodes[node]['item'] = maze[i][j]
                if i > 0 and maze[i - 1][j] != 1:
                    graph.add_edge(node, (i - 1) * cols + j)
                if i < rows - 1 and maze[i + 1][j] != 1:
                    graph.add_edge(node, (i + 1) * cols + j)
                if j > 0 and maze[i][j - 1] != 1:
                    graph.add_edge(node, i * cols + (j - 1))
                if j < cols - 1 and maze[i][j + 1] != 1:
                    graph.add_edge(node, i * cols + (j + 1))
    return graph, end_node

# Приклад лабіринту у вигляді матриці
maze_matrix = [
    [0, 1, 0, 0, 3],
    [0, 1, 1, 1, 0],
    [0, 0, 0, 0, 0],
    [1, 1, 0, 1, 0],
    [0, 2, 0, 10, 0]
]

# Створення графа з матриці лабіринту
maze_graph, exit_node = create_graph_from_maze(maze_matrix)

# Знаходження координат предметів
item_nodes = [node for node in maze_graph.nodes if 'item' in maze_graph.nodes[node]]
item_coords = [(node % len(maze_matrix[0]), -node // len(maze_matrix[0])) for node in
item_nodes]

# Визначення шляху до кожного предмета та до виходу
start_node = 0
paths_to_items = []
for item_node in item_nodes:
    shortest_path_to_item = find_shortest_path(maze_graph, start_node, item_node)
    if shortest_path_to_item:
        paths_to_items.append(shortest_path_to_item)
        start_node = item_node

# Визначення шляху до виходу
shortest_path_to_exit = find_shortest_path(maze_graph, start_node, exit_node)

# Візуалізація графа лабіринту з підказками шляху до предметів та до виходу
if shortest_path_to_exit:
    paths = paths_to_items + [shortest_path_to_exit]
    print(f"Найкоротші шляхи до предметів та до виходу:")
    for i, path in enumerate(paths):
        print(f"Шлях {i + 1}: {path}")
    draw_graph(maze_graph, maze_matrix, paths, item_coords)
else:
    print("Шляху до виходу немає.")
```

Результат

```
Найкоротші шляхи до предметів та до виходу:
Шлях 1: [0, 5, 10, 11, 12, 13, 14, 9, 4]
Шлях 2: [4, 9, 14, 13, 12, 17, 22, 21]
Шлях 3: [21, 22, 23]
```



```
Найкоротші шляхи до предметів та до виходу:
Шлях 1: [0, 5, 10, 11, 12, 13, 14, 9, 4]
Шлях 2: [4, 9, 14, 19, 24]
Шлях 3: [24, 23, 22, 21, 20]
```