Дискретні структури ІПЗ-21-2

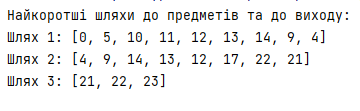
Губарєв Ростислав Вадимович

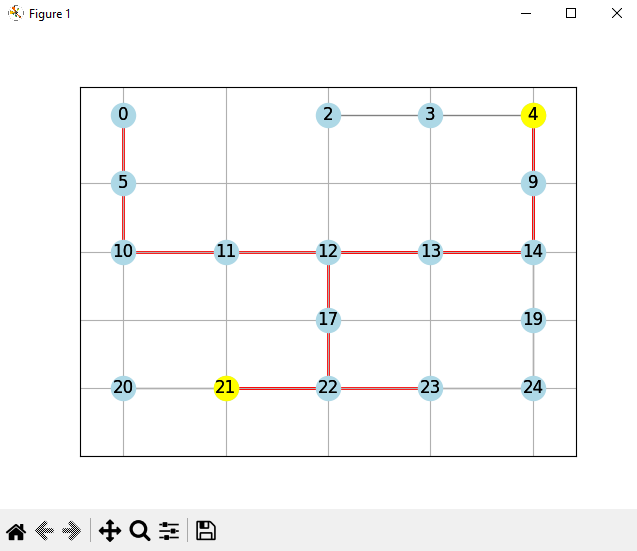
**Лабораторна робота №12.2**

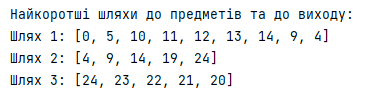
Лабіринт задано так само, як у задачі 1. Задано список предметів, що знаходяться на карті. Гравцеві необхідно "зібрати" всі предмети і повернутися у вихідну точку. Знайдіть найкоротший маршрут гравця і візуалізуйте його на графі. Якщо таких маршрутів кілька виведіть будь-який із них. Якщо маршруту немає,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

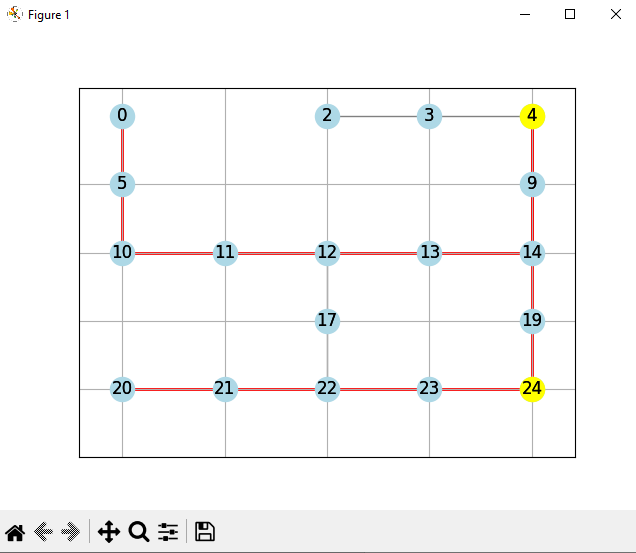
|  |
| --- |
| **Код**  import networkx as nx import matplotlib.pyplot as plt  def draw\_graph(graph, maze\_matrix, paths=None, items=None):  *# Встановлення позицій вузлів графа відповідно до їхніх координат у матриці* pos = dict()  rows = len(maze\_matrix)  cols = len(maze\_matrix[0])  for i in range(rows):  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**

****

****

****

****