Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Лабораторна робота №3

“Агенти на основі знань”

З дисципліни

“Вступ до Штучного Інтелекту”

Виконав:
Студент групи ІП-05
Амелін В. О.

Перевірив:
Таран В. І.

Київ - 2022

Мета роботи: удосконалити інтелектуального агента з попередньої лабораторної роботи, щоб агент створював та використовував власну базу знань для орієнтування та побудови маршруту рухаючись по графу-дорозі. Отримати практичні навички роботи з базами знань.

Завдання:

- Отримати навички роботи з базами знань.
- Удосконалити агента-автомобіля з попередньої лабораторної роботи

Опис алгоритму генерації дороги

Сам граф спочатку генерується за допомогою методу бібліотеки networkx - grid_2d_graph.

Лабораторна робота заснована на першій роботі, де ми створювали зв'язний граф і видалили в ньому ребра. Алгоритм працює за таким принципом:

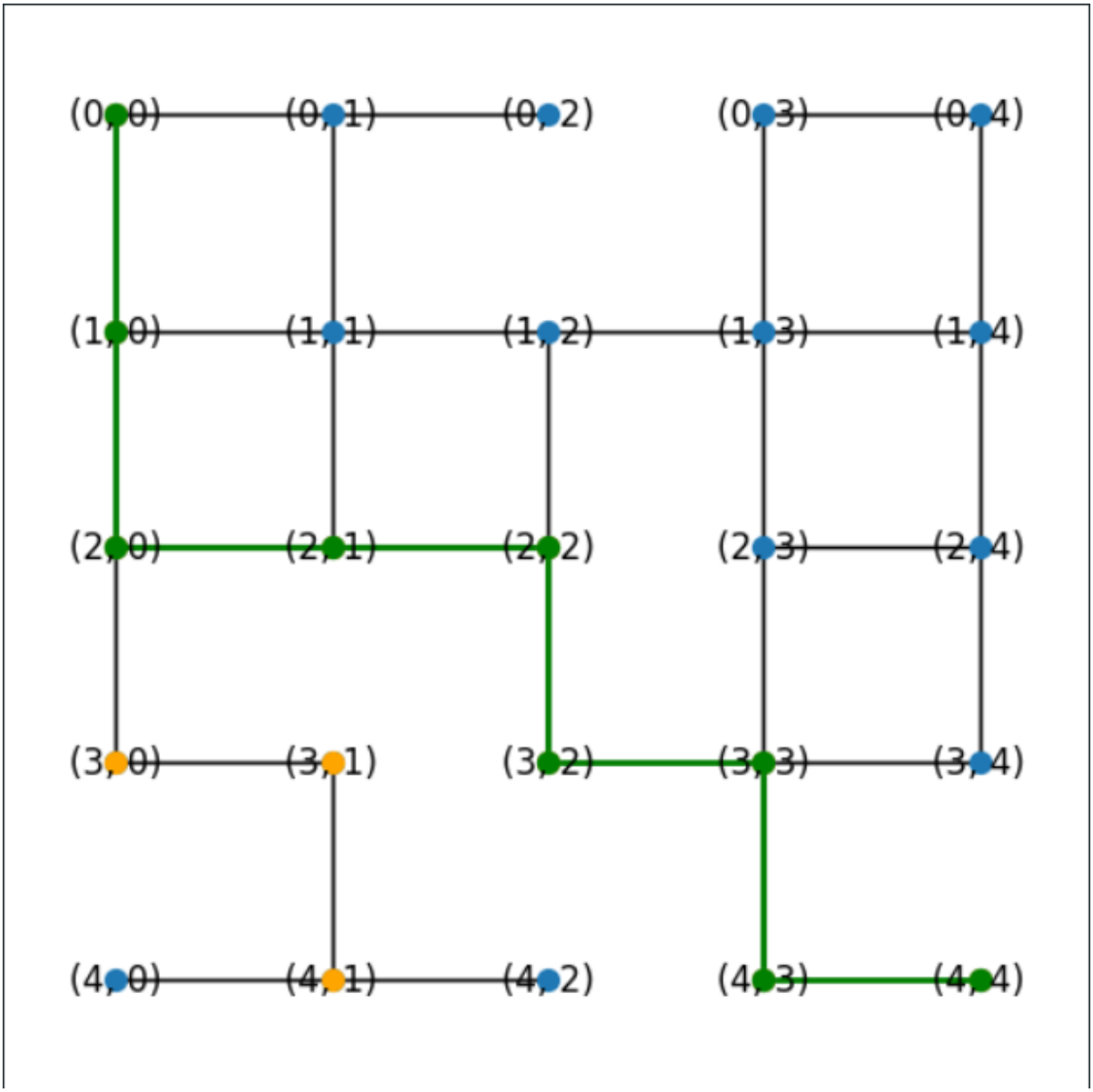
1. Ми перевіряємо коректність введених точок.
2. Додаємо в масив відвіданих точок і масив шляху.
3. Йдемо в наступну вершину
4. Перевіряємо, чи є ця нода і суміжні до них виходом.
5. Перевіряємо коректність нової вершини.
6. Перевіряємо суміжні вершини.
7. У вдалому випадку переходимо на наступну вершину
8. У негативному випадку, ми видаляємо останнє значення в масиві шляху.
9. У випадку, коли поточна координата дорівнює кінцевій, виходимо з циклу.

10. **Програмний код генерації дороги:**

```
start_cord = (0, 0)
end_cord = (4, 4)
if((size, size) <= end_cord and (size, size) <= start_cord):
  raise ValueError(f"Should be less {size}")

if(end_cord > (0, 0) and start_cord > (0, 0)):
  raise ValueError("Must be greater than zero")
def agent(G, start_cord, end_cord):
  visited = (start_cord, )
  path = [start_cord]
  memory = {}
  last_node = path[-1]

  while last_node != end_cord:
    memory = get_tell(memory, last_node, end_cord)
    best = get_ask(memory, visited)

    if best not in G[last_node]:

        movements = get_passage_options(G[last_node], visited)

        while len(movements) == 0:
            path.pop()
            last_node = path[-1]
            movements = get_passage_options(G[last_node], visited)

        best = min(movements, key=lambda node: get_approach(node, best))

    visited += (best,)
    path.append(best)
    last_node = path[-1]
    print(*path, sep=' => ')

  return path, visited

def get_tell(memory, last_node, end_cord):
    for near_node in G[last_node]:
```

```
        if not get_locked(near_node):
            if near_node not in memory:
                memory[near_node] = get_approach(near_node, end_cord)

            for far_node in G[near_node]:
                if far_node not in memory:
                    memory[far_node] = get_approach(far_node, end_cord)
    return memory

def get_ask(memory, visited):
    return min(get_passage_options(memory, visited), key=lambda node:
memory[node])

def get_passage_options(node_list, visited):
    return list(filter(lambda node: not get_locked(node), filter(lambda
node: node not in visited, node_list)))

def get_approach(node, end_cord):
    return abs(node[0] - end_cord[0]) + abs(node[1] - end_cord[1])

def get_locked(node):
    return len(G[node]) == 1 and node != end_cord

path, visited = agent(G, start_cord, end_cord)


plt.figure(figsize=(5,5))
pos = {(x,y):(y,-x) for x,y in G.nodes()}
nx.draw(G, pos, with_labels=(pos, pos), node_size=50)
nx.draw(G, pos, nodelist=visited, node_color='orange', node_size=50,)
nx.draw(G, pos, nodelist=path, node_color='g', node_size=50,
edgelist=[(path[i], path[i+1]) for i in range(len(path)-1)],
edge_color='g', width=2)

plt.show()
```

Результат:



Висновок: під час виконання лабораторної роботи, я удосконалити інтелектуального агента з попередньої лабораторної роботи. Створив агента який використовує власну базу знань для орієнтування та побудови маршруту рухаючись по графу-дорозі. Отримав практичні навички роботи з базами знань.