Báo cáo LAB_02

Thực hiện:

Trần Gia Nguyễn

Huỳnh Khôi Nguyên

Nguyễn Anh Khoa

LAB_02_01: AGENT

TASK 01: TRIỂN KHAI MÔI TRƯỜNG MÔ PHỎNG

```python
def simulation_environment(agent_function, n=5, p=0.2, max_steps=1000, verbose=False):
    # Initialize room: True = dirty, False = clean
    room = np.random.rand(n, n) < p
    # Agent position
    x, y = random.randint(0, n-1), random.randint(0, n-1)
    actions_taken = 0

    for step in range(max_steps):
        # Sensors
        bumpers = {
            "north": y == 0,
            "south": y == n-1,
            "west": x == 0,
            "east": x == n-1
        }
        dirty = room[y, x]
        # Agent decides action
        action = agent_function(bumpers, dirty)
        actions_taken += 1
        if verbose:
            print(f"Step {step}: Pos=({x},{y}) Action={action} Dirty={dirty}")
```

Cài đặt môi trường

LAB_02_01: AGENT

TASK 01: TRIỂN KHAI MÔI TRƯỜNG MÔ PHỎNG

```
        # React to action
        if action == "suck":
            room[y, x] = False
        elif action == "north" and not bumpers["north"]:
            y -= 1
        elif action == "south" and not bumpers["south"]:
            y += 1
        elif action == "west" and not bumpers["west"]:
            x -= 1
        elif action == "east" and not bumpers["east"]:
            x += 1
        # Check if all clean
        if not room.any():
            break
    return actions_taken
```

Cài đặt môi trường

LAB_02_01: AGENT

TASK 01: TRIỂN KHAI MÔI TRƯỜNG MÔ PHỎNG

Chạy kiểm chứng và kết quả

```
steps = simulation_environment(simple_randomized_agent, n=5, p=0.2, max_steps=1000, verbose=True)
print(f"Actions taken to clean the room: {steps}")
```

```
Step 0: Pos=(2,3) Action=suck Dirty=False
Step 1: Pos=(2,3) Action=east Dirty=False
Step 2: Pos=(3,3) Action=east Dirty=False
Step 3: Pos=(4,3) Action=west Dirty=True
Step 4: Pos=(3,3) Action=west Dirty=False
Step 5: Pos=(2,3) Action=north Dirty=False
Step 6: Pos=(2,2) Action=suck Dirty=False
Step 7: Pos=(2,2) Action=suck Dirty=False
Step 8: Pos=(2,2) Action=west Dirty=False
Step 9: Pos=(1,2) Action=west Dirty=False
Step 10: Pos=(0,2) Action=south Dirty=True
Step 11: Pos=(0,3) Action=west Dirty=False
Step 12: Pos=(0,3) Action=north Dirty=False
Step 13: Pos=(0,2) Action=south Dirty=True
Step 14: Pos=(0,3) Action=suck Dirty=False
Step 15: Pos=(0,3) Action=east Dirty=False
Step 16: Pos=(1,3) Action=south Dirty=False
Step 17: Pos=(1,4) Action=east Dirty=True
Step 18: Pos=(2,4) Action=south Dirty=False
Step 19: Pos=(2,4) Action=south Dirty=False
Step 20: Pos=(2,4) Action=suck Dirty=False
Step 21: Pos=(2,4) Action=west Dirty=False
Step 22: Pos=(1,4) Action=north Dirty=True
Step 23: Pos=(1,3) Action=west Dirty=False
Step 24: Pos=(0,3) Action=north Dirty=False
...
Step 398: Pos=(4,4) Action=south Dirty=True
Step 399: Pos=(4,4) Action=south Dirty=True
Step 400: Pos=(4,4) Action=suck Dirty=True
Actions taken to clean the room: 401
```

LAB_02_01: AGENT

TASK 02: AGENT PHẢN XẠ ĐƠN GIẢN

Cài đặt

```
def simple_reflex_agent(bumpers, dirty):
    if dirty:
        return "suck"
    # Prefer moving in available directions
    for direction in ["north", "east", "south", "west"]:
        if not bumpers[direction]:
            return direction
    return "suck"  # fallback
```

LAB_02_01: AGENT

TASK 02: AGENT PHẢN XẠ ĐƠN GIẢN

Chạy kiểm chứng và kết quả

```
steps = simulation_environment(simple_reflex_agent, n=5, p=0.2, max_steps=1000, verbose=True)
print(f"Actions taken to clean the room: {steps}")
```

```
Step 0: Pos=(2,2) Action=north Dirty=False
Step 1: Pos=(2,1) Action=north Dirty=False
Step 2: Pos=(2,0) Action=east Dirty=False
Step 3: Pos=(3,0) Action=east Dirty=False
Step 4: Pos=(4,0) Action=suck Dirty=True
Step 5: Pos=(4,0) Action=south Dirty=False
Step 6: Pos=(4,1) Action=suck Dirty=True
Step 7: Pos=(4,1) Action=north Dirty=False
Step 8: Pos=(4,0) Action=south Dirty=False
Step 9: Pos=(4,1) Action=north Dirty=False
Step 10: Pos=(4,0) Action=south Dirty=False
Step 11: Pos=(4,1) Action=north Dirty=False
Step 12: Pos=(4,0) Action=south Dirty=False
Step 13: Pos=(4,1) Action=north Dirty=False
Step 14: Pos=(4,0) Action=south Dirty=False
Step 15: Pos=(4,1) Action=north Dirty=False
Step 16: Pos=(4,0) Action=south Dirty=False
Step 17: Pos=(4,1) Action=north Dirty=False
Step 18: Pos=(4,0) Action=south Dirty=False
Step 19: Pos=(4,1) Action=north Dirty=False
Step 20: Pos=(4,0) Action=south Dirty=False
Step 21: Pos=(4,1) Action=north Dirty=False
Step 22: Pos=(4,0) Action=south Dirty=False
Step 23: Pos=(4,1) Action=north Dirty=False
Step 24: Pos=(4,0) Action=south Dirty=False
...
Step 997: Pos=(4,1) Action=north Dirty=False
Step 998: Pos=(4,0) Action=south Dirty=False
Step 999: Pos=(4,1) Action=north Dirty=False
Actions taken to clean the room: 1000
```

LAB_02_01: AGENT

TASK 03: AGENT PHẢN XẠ DỰA TRÊN MÔ HÌNH

Cài đặt

```
class ModelBasedAgent:
    def __init__(self, n):
        self.n = n
        self.x = 0
        self.y = 0
        self.direction = "east"  # start moving east
        self.visited = set()
    def agent_function(self, bumpers, dirty):
        pos = (self.x, self.y)
        self.visited.add(pos)
        if dirty:
            return "suck"
```

LAB_02_01: AGENT

TASK 03: AGENT PHẢN XẠ DỰA TRÊN MÔ HÌNH

Cài đặt

```
# Zig-zag pattern: move east until wall, then south, then west, then south, repeat
if self.direction == "east":
    if not bumpers["east"]:
        self.x += 1
        return "east"
    elif not bumpers["south"]:
        self.y += 1
        self.direction = "west"
        return "south"
elif self.direction == "west":
    if not bumpers["west"]:
        self.x -= 1
        return "west"
    elif not bumpers["south"]:
        self.y += 1
        self.direction = "east"
        return "south"
```

LAB_02_01: AGENT

TASK 03: AGENT PHẢN XẠ DỰA TRÊN MÔ HÌNH

Cài đặt

```
# If stuck, try any available move
for d in ["north", "east", "south", "west"]:
    if not bumpers[d]:
        if d == "north": self.y -= 1
        if d == "south": self.y += 1
        if d == "east": self.x += 1
        if d == "west": self.x -= 1
        return d
return "suck"
```

LAB_02_01: AGENT

TASK 03: AGENT PHẢN XẠ DỰA TRÊN MÔ HÌNH

Chạy kiểm chứng và kết quả

```
agent = ModelBasedAgent(n=5)
def agent_func(bumpers, dirty):
    return agent.agent_function(bumpers, dirty)
steps = simulation_environment(agent_func, n=5, p=0.2, max_steps=1000, verbose=True)
print(f"Actions taken to clean the room: {steps}")
```

```
Step 0: Pos=(4,3) Action=south Dirty=False
Step 1: Pos=(4,4) Action=suck Dirty=True
Step 2: Pos=(4,4) Action=west Dirty=False
Step 3: Pos=(3,4) Action=suck Dirty=True
Step 4: Pos=(3,4) Action=west Dirty=False
Step 5: Pos=(2,4) Action=west Dirty=False
Step 6: Pos=(1,4) Action=west Dirty=False
Step 7: Pos=(0,4) Action=north Dirty=False
Step 8: Pos=(0,3) Action=south Dirty=False
Step 9: Pos=(0,4) Action=east Dirty=False
Step 10: Pos=(1,4) Action=east Dirty=False
Step 11: Pos=(2,4) Action=east Dirty=False
Step 12: Pos=(3,4) Action=east Dirty=False
Step 13: Pos=(4,4) Action=north Dirty=False
Step 14: Pos=(4,3) Action=south Dirty=False
Step 15: Pos=(4,4) Action=west Dirty=False
Step 16: Pos=(3,4) Action=west Dirty=False
Step 17: Pos=(2,4) Action=west Dirty=False
Step 18: Pos=(1,4) Action=west Dirty=False
Step 19: Pos=(0,4) Action=north Dirty=False
Step 20: Pos=(0,3) Action=south Dirty=False
Step 21: Pos=(0,4) Action=east Dirty=False
Step 22: Pos=(1,4) Action=east Dirty=False
Step 23: Pos=(2,4) Action=east Dirty=False
Step 24: Pos=(3,4) Action=east Dirty=False
...
Step 997: Pos=(4,4) Action=north Dirty=False
Step 998: Pos=(4,3) Action=south Dirty=False
Step 999: Pos=(4,4) Action=west Dirty=False
Actions taken to clean the room: 1000
```

LAB_02_01: AGENT

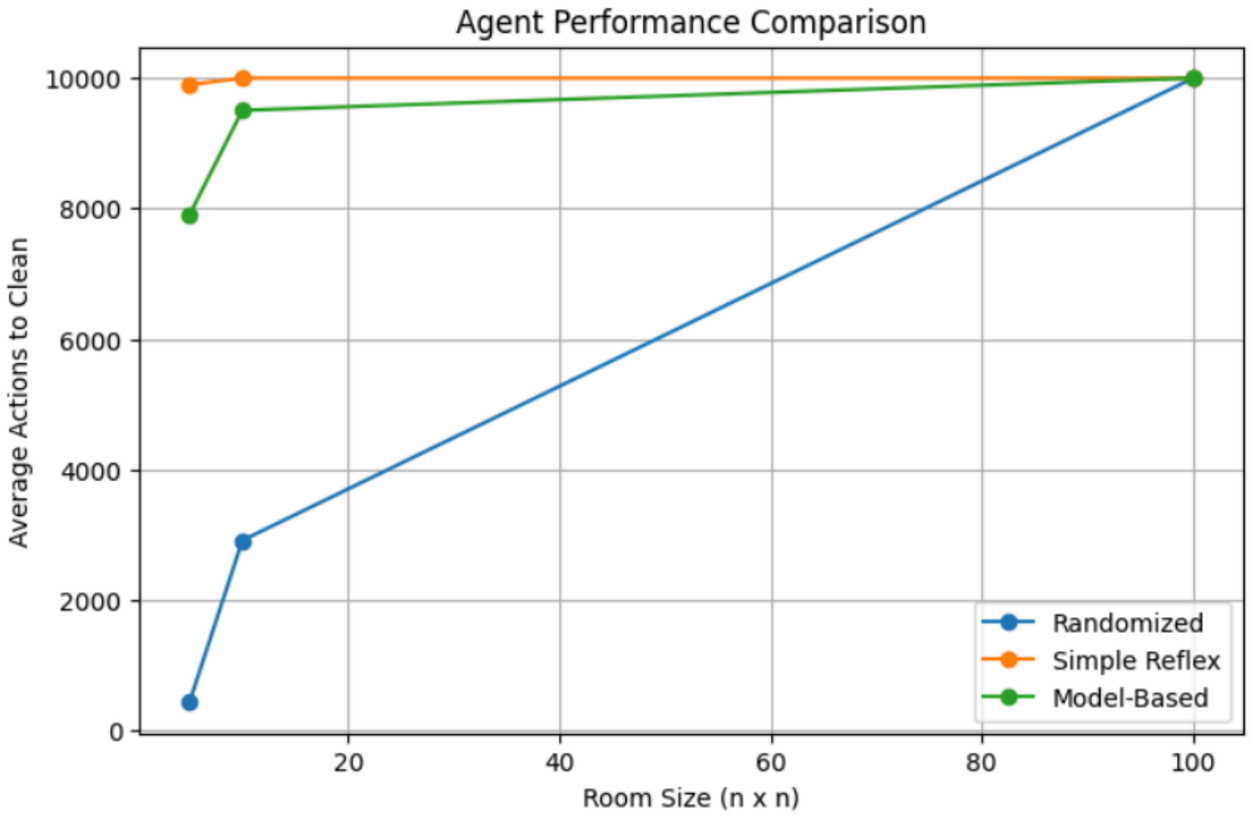
TASK 04: SO SÁNH HIỆU NĂNG

```python
def evaluate_agent(agent_func, n, p=0.2, runs=100):
    results = [simulation_environment(agent_func, n=n, p=p, max_steps=10000) for _ in range(runs)]
    return np.mean(results)

sizes = [5, 10, 100]
results = {"Randomized": [], "Simple Reflex": [], "Model-Based": []}

for size in sizes:
    # Randomized agent
    results["Randomized"].append(evaluate_agent(simple_randomized_agent, size))
    # Simple reflex agent
    results["Simple Reflex"].append(evaluate_agent(simple_reflex_agent, size))
    # Model-based agent
    agent = ModelBasedAgent(n=size)
    def agent_func(bumpers, dirty):
        return agent.agent_function(bumpers, dirty)
    results["Model-Based"].append(evaluate_agent(agent_func, size))

# Create DataFrame for table
df = pd.DataFrame(results, index=[f"{s}x{s}" for s in sizes])
print(df)
```

LAB_02_01: AGENT

TASK 04: SO SÁNH HIỆU NĂNG

	Randomized	Simple Reflex	Model-Based
5x5	431.43	9900.01	7903.94
10x10	2910.60	10000.00	9505.57
100x100	10000.00	10000.00	10000.00



LAB_02_01: AGENT

TASK 05: ĐỘ BỀN VỮNG

Giải thích về độ bền vững của các agent

1. Nếu phòng có kích thước không xác định:

- Agent ngẫu nhiên và phản xạ đơn giản vẫn hoạt động nhưng hiệu quả thấp, dễ bỏ sót hoặc lặp lại các bước.
- Agent dựa trên mô hình có thể khám phá phòng, ghi nhớ các vị trí đã đi qua, xây dựng bản đồ dần dần nên thích nghi tốt hơn.

2. Nếu khu vực làm sạch có hình dạng bất thường:

- Agent ngẫu nhiên và phản xạ đơn giản dễ bỏ sót các khu vực xa hoặc lãng phí hành động.
- Agent dựa trên mô hình nếu ghi nhớ các ô đã đi qua sẽ thích nghi tốt, đảm bảo làm sạch toàn bộ khu vực.

3. Nếu phòng có vật cản:

- Tất cả agent cần xử lý cảm biến va chạm. Agent ngẫu nhiên và phản xạ đơn giản có thể bị kẹt hoặc va chạm lặp lại.
- Agent dựa trên mô hình có thể ghi nhớ vị trí vật cản để tránh lặp lại va chạm, di chuyển tối ưu hơn.

4. Nếu cảm biến bụi không hoàn hảo (10% sai lệch):

- Agent có thể bỏ sót ô bẩn hoặc làm sạch ô đã sạch. Agent ngẫu nhiên và phản xạ đơn giản không có cơ chế kiểm tra lại.
- Agent dựa trên mô hình có thể kiểm tra lại các ô nghi ngờ nhiều lần để giảm số ô còn bẩn.

5. Nếu cảm biến va chạm không hoàn hảo (10% sai lệch):

- Agent có thể thử di chuyển vào vị trí không hợp lệ, dễ bị kẹt hoặc lãng phí hành động.
- Agent dựa trên mô hình có thể học từ các lần di chuyển thất bại, ghi nhớ vị trí thực tế của tường/vật cản để tránh lặp lại lỗi.