Instrucciones

Dado:

- Habitación de MxN espacios.
- Número de agentes.
- Porcentaje de celdas inicialmente sucias.
- Tiempo máximo de ejecución.

Realiza la siguiente simulación:

- Inicializa las celdas sucias (ubicaciones aleatorias).
- Todos los agentes empiezan en la celda [1,1].
- En cada paso de tiempo:
  - Si la celda está sucia, entonces aspira.
  - Si la celda está limpia, el agente elije una dirección aleatoria para moverse (unas de las 8 celdas vecinas) y elije la acción de movimiento (si no puede moverse allí, permanecerá en la misma celda).
- Se ejecuta el tiempo máximo establecido.

Deberás recopilar la siguiente información durante la ejecución:

- Tiempo necesario hasta que todas las celdas estén limpias (o se haya llegado al tiempo máximo).
- Porcentaje de celdas limpias después del termino de la simulación.
- Número de movimientos realizados por todos los agentes.

```
#Instalamos libreria mesa

%pip install mesa

#La clase Model para manejar los agentes y la clase Agent para un definir un agente.
from mesa import Agent, Model

# Debido a que necesitamos varios agentes por celda, elegimos ''MultiGrid''.
from mesa.space import MultiGrid

# Con ''RandomActivation'', hacemos que todos los agentes se activen ''al mismo tiempo''.
from mesa.time import RandomActivation

# Con ''DataCollector'', vamos a recuperar información del modelo.
from mesa.datacollection import DataCollector

# matplotlib lo vamos a utilizar para realizar una ''animación'' de mi modelo.
%matplotlib inline
import matplotlib
import matplotlib.pyplot as plt
import matplotlib.lines as mlines
import matplotlib.transforms as mtransforms
from matplotlib import colors
from matplotlib.ticker import PercentFormatter
import matplotlib.animation as animation
plt.rcParams["animation.html"] = "jshtml"
matplotlib.rcParams['animation.embed_limit'] = 2**128

import numpy as np
import pandas as pd

# Definimos otros paquete que vamos a usar para medir el tiempo de ejecución del modelo
import time
import datetime
from timeit import default_timer as timer

MAXVAL = 10000
```

```
Requirement already satisfied: backcall in /usr/local/lib/python3.10/dist-packages (from ipython>=4.0.0->ipywidgets->solara->mesa) (0
Requirement already satisfied: matplotlib-inline in /usr/local/lib/python3.10/dist-packages (from ipython>=4.0.0->ipywidgets->solara-
Requirement already satisfied: pexpect>4.3 in /usr/local/lib/python3.10/dist-packages (from ipython>=4.0.0->ipywidgets->solara->mesa)
Requirement already satisfied: attrs>=22.2.0 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=2.6->nbformat->solara->mesa
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=2.6-
Requirement already satisfied: referencing>=0.28.4 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=2.6->nbformat->solara
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=2.6->nbformat->solara->mes
Requirement already satisfied: platformdirs>=2.5 in /usr/local/lib/python3.10/dist-packages (from jupyter-core!=5.0.*,>=4.12->jupyter
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.10/dist-packages (from markdown-it-py>=2.2.0->rich->cookiecutter-
Requirement already satisfied: notebook>=4.4.1 in /usr/local/lib/python3.10/dist-packages (from widgetsnbextension~=3.6.0->ipywidgets
Requirement already satisfied: parso<0.9.0,>=0.8.3 in /usr/local/lib/python3.10/dist-packages (from jedi>=0.16->ipython>=4.0.0->ipywi
Requirement already satisfied: argon2-cffi in /usr/local/lib/python3.10/dist-packages (from notebook>=4.4.1->widgetsnbextension~=3.6.
Collecting jupyter-client>=7.0.0 (from solara->mesa)
  Downloading jupyter_client-7.4.9-py3-none-any.whl (133 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 133.5/133.5 kB 10.8 MB/s eta 0:00:00
Requirement already satisfied: nbconvert>=5 in /usr/local/lib/python3.10/dist-packages (from notebook>=4.4.1->widgetsnbextension~=3.6
Requirement already satisfied: nest-asyncio>=1.5 in /usr/local/lib/python3.10/dist-packages (from notebook>=4.4.1->widgetsnbextension
Requirement already satisfied: Send2Trash>=1.8.0 in /usr/local/lib/python3.10/dist-packages (from notebook>=4.4.1->widgetsnbextension
Requirement already satisfied: terminado>=0.8.3 in /usr/local/lib/python3.10/dist-packages (from notebook>=4.4.1->widgetsnbextension~
Requirement already satisfied: prometheus-client in /usr/local/lib/python3.10/dist-packages (from notebook>=4.4.1->widgetsnbextension
Requirement already satisfied: nbclassic>=0.4.7 in /usr/local/lib/python3.10/dist-packages (from notebook>=4.4.1->widgetsnbextension~
Requirement already satisfied: entrypoints in /usr/local/lib/python3.10/dist-packages (from jupyter-client>=7.0.0->solara->mesa) (0.4
Requirement already satisfied: ptyprocess>=0.5 in /usr/local/lib/python3.10/dist-packages (from pexpect>4.3->ipython>=4.0.0->ipywidge
Requirement already satisfied: wcwidth in /usr/local/lib/python3.10/dist-packages (from prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0-
Requirement already satisfied: jupyter-server>=1.8 in /usr/local/lib/python3.10/dist-packages (from nbclassic>=0.4.7->notebook>=4.4.1
Requirement already satisfied: notebook-shim>=0.2.3 in /usr/local/lib/python3.10/dist-packages (from nbclassic>=0.4.7->notebook>=4.4.
Requirement already satisfied: lxml in /usr/local/lib/python3.10/dist-packages (from nbconvert>=5->notebook>=4.4.1->widgetsnbextensio
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.10/dist-packages (from nbconvert>=5->notebook>=4.4.1->widgets
Requirement already satisfied: bleach in /usr/local/lib/python3.10/dist-packages (from nbconvert>=5->notebook>=4.4.1->widgetsnbextens
Requirement already satisfied: defusedxml in /usr/local/lib/python3.10/dist-packages (from nbconvert>=5->notebook>=4.4.1->widgetsnbex
Requirement already satisfied: jupyterlab-pygments in /usr/local/lib/python3.10/dist-packages (from nbconvert>=5->notebook>=4.4.1->wi
Requirement already satisfied: mistune<2,>=0.8.1 in /usr/local/lib/python3.10/dist-packages (from nbconvert>=5->notebook>=4.4.1->widg
Requirement already satisfied: nbclient>=0.5.0 in /usr/local/lib/python3.10/dist-packages (from nbconvert>=5->notebook>=4.4.1->widget
Requirement already satisfied: pandocfilters>=1.4.1 in /usr/local/lib/python3.10/dist-packages (from nbconvert>=5->notebook>=4.4.1->w
Requirement already satisfied: tinycss2 in /usr/local/lib/python3.10/dist-packages (from nbconvert>=5->notebook>=4.4.1->widgetsnbexte
Requirement already satisfied: argon2-cffi-bindings in /usr/local/lib/python3.10/dist-packages (from argon2-cffi->notebook>=4.4.1->wi
Requirement already satisfied: websocket-client in /usr/local/lib/python3.10/dist-packages (from jupyter-server>=1.8->nbclassic>=0.4.
Requirement already satisfied: cffi>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from argon2-cffi-bindings->argon2-cffi->notebo
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.10/dist-packages (from beautifulsoup4->nbconvert>=5->notebook>
Requirement already satisfied: webencodings in /usr/local/lib/python3.10/dist-packages (from bleach->nbconvert>=5->notebook>=4.4.1->w
Requirement already satisfied: pycparser in /usr/local/lib/python3.10/dist-packages (from cffi>=1.0.1->argon2-cffi-bindings->argon2-c
Installing collected packages: websockets, watchdog, types-python-dateutil, pymdown-extensions, mesa-viz-tornado, jedi, h11, binaryor
  Attempting uninstall: jupyter-client
    Found existing installation: jupyter-client 6.1.12
    Uninstalling jupyter-client-6.1.12:
      Successfully uninstalled jupyter-client-6.1.12
Successfully installed arrow-1.3.0 binaryornot-0.4.4 cookiecutter-2.6.0 h11-0.14.0 ipyvue-1.10.1 ipyvuetify-1.9.0 jedi-0.19.1 jupyter
```

```
class RobotLimpieza(Agent):
    def __init__(self, unique_id, model):
        super().__init__(unique_id, model)
        self.options = np.array([ [-1, -1], [-1,  0], [-1, +1],
                                  [ 0, -1],           [ 0, +1],
                                  [+1, -1], [+1,  0], [+1, +1] ])
        self.steps = 0

    def can_move(self, x, y):
        return (x >= 0 and x < self.model.grid.width and
                y >= 0 and y < self.model.grid.height)


    def step(self):
        if self.model.floor[self.pos[0]][self.pos[1]] == 1:
            self.model.floor[self.pos[0]][self.pos[1]] = 0
        else:
            i = int( np.random.rand() * MAXVAL ) % len(self.options)
            x = self.pos[0] + self.options[i][0]
            y = self.pos[1] + self.options[i][1]
            #Se toma como movimiento si se puede desplazar a alguna celda cercana (Vecina)
            if self.can_move(x, y):
                self.model.grid.move_agent(self, (x, y))
                self.model.step_counter += 1
                self.steps += 1

```

```
def get_grid(model):
    """ Esta función nos permite obtener el estado de los diferentes agentes.
        *param* model : Modelo del que se obtiene la información.
        *return* esto devuelve una matriz con la información del estado de cada uno de los agentes."""
    grid = np.zeros( (model.grid.width, model.grid.height) )
    for x in range (model.grid.width):
        for y in range (model.grid.height):
            if model.grid.is_cell_empty( (x, y) ) :
                grid[x][y] = model.floor[x][y] * 2
            else:
                grid[x][y] = 1

    return grid
```

```
class RobotCleanerModelo(Model):
    def __init__(self, width, height, num_agents, dirty_cells_percentage=0.5):
        super().__init__()
        self.num_agents = num_agents
        self.dirty_cells_percentage = dirty_cells_percentage
        self.grid = MultiGrid(width, height, True)
        self.schedule = RandomActivation(self)
        self.floor = np.zeros((width, height,))
        self.width = width
        self.height = height
        self.step_counter = 0
        self.timer = 0

        # Crear agentes
        for i in range(self.num_agents):
            a = RobotLimpieza(i, self)
            self.grid.place_agent(a, (0, 0))
            self.schedule.add(a)

        # Initializate the ''dirty'' cells
        amount = int((width * height) * dirty_cells_percentage)
        for i in range(amount):
            finished = False
            while not finished:
                x = int(np.random.rand() * MAXVAL) % width
                y = int(np.random.rand() * MAXVAL) % height
                if self.floor[x][y] == 0:
                    self.floor[x][y] = 1
                    finished = True

        self.datacollector = DataCollector(model_reporters={"Grid": get_grid})

    def is_all_clean(self):
        return np.all(self.floor == 0)

    def calculate_clean_cells(self):
        total_cells = self.width * self.height
        clean_cells = int((total_cells) - np.count_nonzero(self.floor == 1))
        clean_percentage = (clean_cells * 100) / total_cells

        return int(clean_percentage)


    def total_steps(self):
        return self.step_counter

    def total_time(self):
        return self.timer

    def step(self):
        """ Ejecuta un paso de la simulación."""
        start = timer()
        self.datacollector.collect(self)
        self.schedule.step()
        end = timer()
        delta_time = end - start
        self.timer += delta_time
```

```
# Parametros y definir grid
Ancho = 20
Altura = 20

# Definimos el número de celdas iniciales a estar sucias
DIRTY_CELLS_PERCENTAGE = 0.5

# Definimos el número máximo de generaciones a correr
MAX_GENERACIONES = 200


# Definimos el número de agentes que limpiaran
NUMERO_AGENTES = 45

NUM_AGENTS_3 = 100

# Registramos el tiempo de inicio
start_time = time.time()

model = RobotCleanerModelo(Ancho, Altura, NUMERO_AGENTES, DIRTY_CELLS_PERCENTAGE)


# Definimos que se siga iterando el modelo hasta que todo esté limpio o ya no se encuentren celdas sucias
num_of_agents = []
steps_of_agents = []
time_of_ejecution_arr = []

for k in range(25):
    model = RobotCleanerModelo(Ancho, Altura, NUMERO_AGENTES, DIRTY_CELLS_PERCENTAGE)
    i = 1
    final_time = 0
    while not model.is_all_clean():
        model.step()
        i += 1
        final_time += model.total_time()

    # Imprimimos el porcentaje de celdas limpias, total de pasos realizados
    # por los agentes y el tiempo de cada corrida

    print(f"Porcentaje de celdas limpias {model.calculate_clean_cells()}%")
    print(f"Pasos dados por todos los agentes: {model.step_counter}")
    print("Tiempo total de la ejecución: %.2f s" %final_time)
    print(f"Numero de agentes: {NUMERO_AGENTES}")
    print("================================================")
    num_of_agents.append(NUMERO_AGENTES)
    steps_of_agents.append(model.step_counter)
    time_of_ejecution_arr.append(float(f'{final_time:.2f}'))

    NUMERO_AGENTES += 10
```

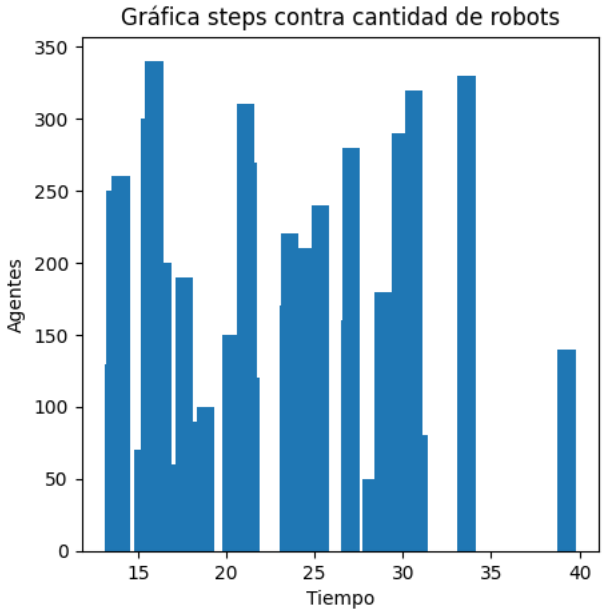
```
Tiempo total de la ejecución: 17.60 s
Numero de agentes: 235
================================================
Porcentaje de celdas limpias 100%
Pasos dados por todos los agentes: 30535
Tiempo total de la ejecución: 18.61 s
Numero de agentes: 245
================================================
Porcentaje de celdas limpias 100%
Pasos dados por todos los agentes: 35760
Tiempo total de la ejecución: 23.09 s
Numero de agentes: 255
================================================
Porcentaje de celdas limpias 100%
Pasos dados por todos los agentes: 29241
Tiempo total de la ejecución: 14.80 s
Numero de agentes: 265
================================================
Porcentaje de celdas limpias 100%
Pasos dados por todos los agentes: 32957
Tiempo total de la ejecución: 18.36 s
Numero de agentes: 275
================================================
Porcentaje de celdas limpias 100%
Pasos dados por todos los agentes: 30535
Tiempo total de la ejecución: 15.50 s
Numero de agentes: 285
================================================
```

```
# Obtenemos la información que almacenó el colector, lo cual no dara una grafica visual
all_grid = model.datacollector.get_model_vars_dataframe()

numAgents=(num_of_agents)
stepAgents=(steps_of_agents)
timeAgents=(time_of_ejecution_arr)
print(numAgents)
print(stepAgents)
print(timeAgents)


```

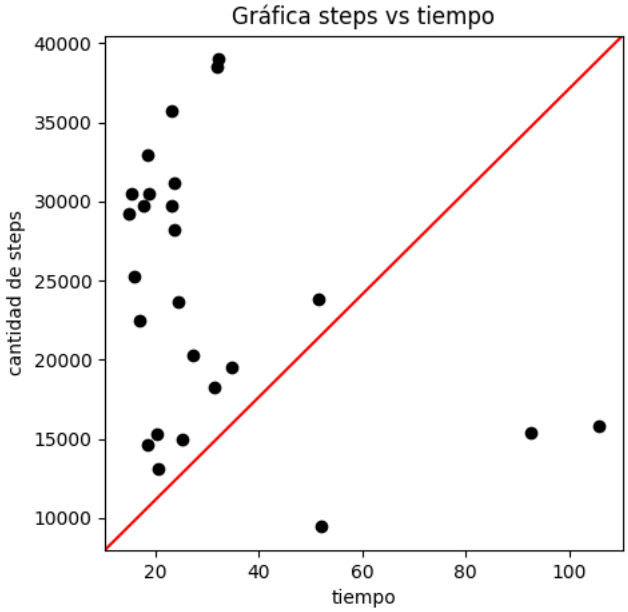
```
[45, 55, 65, 75, 85, 95, 105, 115, 125, 135, 145, 155, 165, 175, 185, 195, 205, 215, 225, 235, 245, 255, 265, 275, 285]
[9461, 15785, 15345, 18287, 14598, 13104, 14978, 19560, 15322, 23834, 20271, 23640, 22455, 28233, 29709, 25256, 31209, 38502, 38998, 297
[51.93, 105.79, 92.73, 31.36, 18.41, 20.43, 25.18, 34.75, 20.35, 51.5, 27.32, 24.34, 16.89, 23.53, 23.18, 15.84, 23.76, 31.85, 32.16, 17
```

```
#Informacion desplegada en grafica de barras
fig = plt.subplots(figsize=(5,5))
plt.title('Gráfica steps contra cantidad de robots')
plt.xlabel('Tiempo')
plt.ylabel('Agentes')

# Gráfica num agentes vs tiempos
plt.bar(timeAgents, numAgents, width=1, align='edge')
print(np.average(timeAgents))
print(np.average(numAgents))
```

```
21.844666666666665
195.0
```

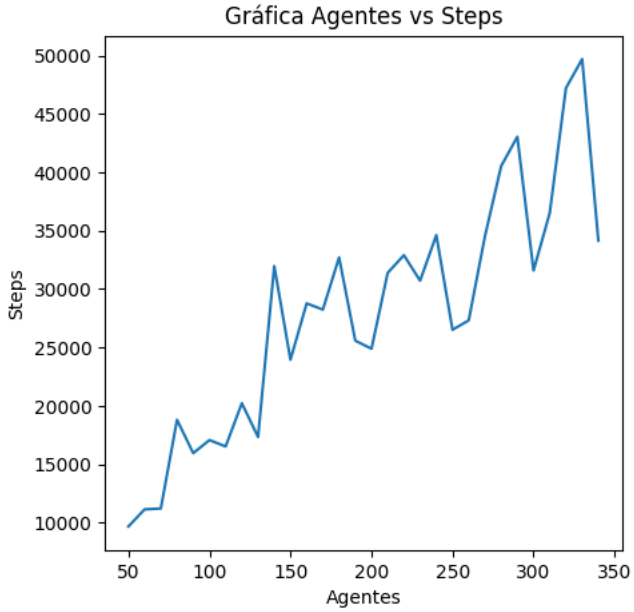



```
fig, ax = plt.subplots(figsize=(5,5))
plt.title('Gráfica steps vs tiempo')
plt.xlabel('tiempo')
plt.ylabel('cantidad de steps')

# Gráfica num steps vs tiempo


ax.scatter(timeAgents, stepAgents, c='black')
line = mlines.Line2D([0, 1], [0, 1], color='red')
transform = ax.transAxes
line.set_transform(transform)
ax.add_line(line)

plt.show()
```



```
#Grafica de lineas

fig = plt.subplots(figsize=(5,5))
plt.title('Gráfica Agentes vs Steps')
plt.xlabel('Agentes')
plt.ylabel('Steps')

# Gráfica num agentes vs steps
plt.plot(numAgents, stepAgents)
```

```
[<matplotlib.lines.Line2D at 0x7aea557747f0>]
```



Implementacion del modelo

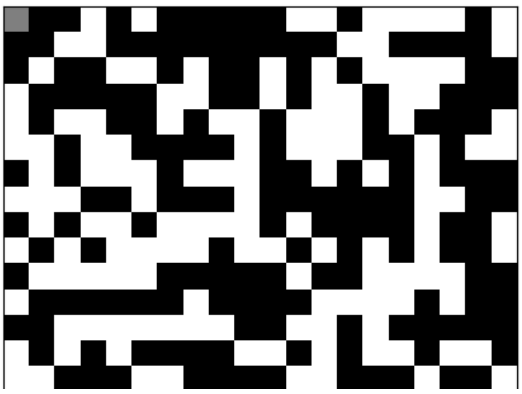
Aqui redactamos sobre como se comporta nuestro modelo de agentes

```
# Esto tiene formato de código
```

```
start_time = time.time()
model_2 = RobotCleanerModelo(20, 20, 50, 0.5)

i = 1
final_time = 0
while not model_2.is_all_clean():
    model_2.step()
    i += 1
    final_time += model_2.total_time()
```

```
#informacion de nuestro colector

all_grid = model_2.datacollector.get_model_vars_dataframe()
```

```
fig, axs = plt.subplots(figsize=(5,5))
axs.set_xticks([])
axs.set_yticks([])
patch = plt.imshow(all_grid.iloc[0][0], cmap=plt.cm.binary)

def animate(i):
    patch.set_data(all_grid.iloc[i][0])

anim = animation.FuncAnimation(fig, animate, frames=MAX_GENERACIONES)
```



```
1 anim
```