CS 480 – Assignment 1: Vacuum Robot Planner

CS 480 Spring 25 Duran

Due: Monday, January 27 at 11:00 PM

Submission Instructions

Submit a link to your **public GitHub repository** containing your source code.

Overview

You will write a program to solve planning problems in *Vacuum World*, a grid-based environment in which some cells are dirty. Your robot must return a plan—a sequence of actions—that moves it around the grid and vacuums all dirty cells. The robot can move in the four cardinal directions (North, South, East, West) and can vacuum when on a dirty cell. All actions have uniform cost.

You must implement two search algorithms:

- Uniform-Cost Search (UCS)
- Depth-First Search (DFS)

You may use **any programming language**, but you may **only use standard libraries**. Do not import or include any third-party packages.

Input Format

Your program must be able to read a `.txt` file containing a grid world. The format is:

```
4
3
_*__
__#*
_@*#
```

- Line 1: number of columns
- Line 2: number of rows
- Remaining lines: one row per line from top to bottom

Each character represents a cell:

- `_` = empty cell
- `#` = blocked cell
- `*` = dirty cell
- `@` = robot starting location

Program Usage

Your program should be run from the command line with two arguments:

```
python3 planner.py [algorithm] [world-file]
```

- `[algorithm]` should be either `uniform-cost` or `depth-first`
- `[world-file]` should be the path to a `.txt` file as described above

Example:

```
python3 planner.py uniform-cost tiny-1.txt
```

Output Format

Your program must print:

1. One action per line: `N`, `S`, `E`, `W`, or `V`
2. One line with the number of nodes generated
3. One line with the number of nodes expanded

Example output:

```
E
V
W
N
N
V
E
E
S
V
85 nodes generated
39 nodes expanded
```

World Generator

A helper script is provided to generate random test worlds. It takes arguments for grid size, obstacle density, and number of dirty cells. You are encouraged to use it while developing and debugging your planner.

Requirements

- Implement both UCS and DFS as specified.
- DFS must avoid infinite loops or cycles.
- UCS must produce an optimal (minimum-cost) plan.
- Your state must encode both the robot's location and the positions of remaining dirty cells.
- Use only standard libraries from your chosen language.

Example Workflow

Below is a sample usage demonstrating how to generate a random vacuum world and run your planner on it using either uniform-cost or depth-first search.

1. Make the world generator script executable and run it

```
chmod +x make_vacuum_world.py
./make_vacuum_world.py 5 7 0.15 3 > sample-5x7.txt
```

This generates a 5-row by 7-column world with approximately 15% blocked cells and 3 dirty cells, and saves it to `sample-5x7.txt`.

2. Run your planner with the generated world

To run your solution using `uniform-cost` search:

```
python3 planner.py uniform-cost sample-5x7.txt
```

To run your solution using `depth-first` search:

```
python3 planner.py depth-first sample-5x7.txt
```

Your output should be a sequence of one-letter actions (one per line), followed by the number of nodes generated and expanded during the search.

Evaluation (20 points total)

0–4 pts	No or non-working code
5–10 pts	One working search algorithm
11–16 pts	Both algorithms work; code is clear
17–20 pts	Code is clean, correct, efficient

Reminders

- Your program must read the world file and produce output to `stdout` only.
- Remove any debugging or extraneous print statements.
- Test thoroughly using small maps before scaling up.