Hybrid A* algorithm

Main idea behind hybrid A* algorithm is to use more than a path planning algorithm (as Dijkstra algorithm and A* algorithm) and combine them to get a result of the shortest path.

It discovers paths that contain curves not just straight lines which is more suitable for autonomous vehicles. It's also more suitable in real-world environments while A* is suitable in grid-based applications of games and maze-solving robots.

Maze game algorithm

```python
import random
from collections import deque
import networkx as nx
from colorama import Fore


def genaret_random_grid(rows_num: int, cols_num: int, start: tuple, end: tuple):···


def print_grid(grid: list):···


def grid_is_solvable(grid: list):···


def find_path(grid: list):···


def draw_path(grid: list, path: list):···


rows = 8
cols = 8
start = (5, 4)
end = (2, 7)

grid = genaret_random_grid(rows, cols, start, end)
while not grid_is_solvable(grid):
    grid = genaret_random_grid(rows, cols, start, end)

draw_path(grid, find_path(grid))
```

the algorithm implementation consists of 4 main steps:

1. Generating random grid
2. Ensuring grid is solvable
3. Figuring out the path
4. Printing the grid with the path

Step 1: used random library of python to generate a grid with the specified dimensions and put obstacles in random positions, referring to empty spaces with dots(.) and obstacles with letter (o). this step is implemented in generate_random_grid() function. It takes the grid's width & height and the starting & ending positions as parameters.

Step 2: applying DFS algorithm to ensure that the grid is solvable. Using a deque to store all available indices and a list to store all visited indices. This step is implemented in grid_is_solvable() function. . It takes the grid's width & height as parameters.

Step 3: networkX library was a perfect choice to help me implement the A-star algorithm. But its A-star function takes the grid to search in as a graph. So, it was required to turn the grid into a graph. It is implemented in the first part of find_path() function which takes the grid as a parameter.

The second part of this function is to use the library's A-star function which takes grid in graph form, starting index and end index.

In this step we get a list of indices that represents the path found by A-star function. This list is used in the next step.

Step 4: in this step the grid with the solution is printed using the list of indices from previous step and colorama library to highlight the path with different color.

Sample runs: