گزارش پروژه

رضیه کرمی- ۴۰۱۳۸۸۲۳

این کد برای شبیه‌سازی حرکت یک عامل در محیطی ماتریسی نوشته شده است که از فایل ورودی (`input.txt`)بارگذاری می‌شود. عامل با استفاده از الگوریتم جستجوی عمقی بازگشتی و جستجوی عمقی تکرارشونده (Iterative Deepening Search) یا (IDS) در این محیط پیمایش می‌کند تا به هدف مشخصی برسد. در ادامه، جزئیات عملکرد هر بخش کد توضیح داده می‌شود.

---

۱ .بارگذاری محیط و تنظیمات اولیه

ابتدای برنامه شامل تنظیمات اولیه و بارگذاری اطلاعات ورودی است:

- `file = open('input.txt', 'r')` و `lines = file.readlines()`:

این دستورات فایل `input.txt`را باز کرده و تمام خطوط آن را می‌خوانند. فایل باید حاوی اطلاعاتی شامل اندازه محیط، موقعیت اولیه عامل، و ساختار محیط باشد.

- `dim = [int(i) for i in lines[0].split(',')]`:

این خط اندازه محیط را به صورت آرایه‌ای از دو عدد (طول و عرض) دریافت می‌کند و در متغیر `dim`ذخیره می‌کند.

- `agent_pos = [int(i) for i in lines[0].split(',')]`:

موقعیت اولیه عامل در محیط را از فایل ورودی استخراج و در متغیر `agent_pos`ذخیره می‌کند.

- `env = []`:

ماتریس دوبعدی محیط که نمایانگر ساختار محیط است. با حلقه‌ای که خطوط باقی‌مانده فایل را می‌خواند، پر می‌شود.

۲ .توابع تعریف‌شده

`environment(action, pos)`

این تابع موقعیت جدید عامل را بر اساس جهت حرکت مشخص شده (`UP, DOWN, LEFT, RIGHT`) محاسبه می‌کند و موقعیت جدید را بازمی‌گرداند.

- اگر حرکت انتخابی خارج از مرزهای محیط باشد، تابع `None`بازمی‌گرداند.
- اگر حرکت درون مرزها باشد، موقعیت جدید و محتوای آن نقطه از محیط بازگردانده می‌شود.

`pos_to_str(node)`

این تابع موقعیت داده‌شده را به رشته‌ای تبدیل می‌کند. این کار برای ردیابی موقعیت‌ها در مجموعه‌ای از موقعیت‌های بازدید شده مفید است.

`recursive_dfs(pos, path, directions, limit, move_count)`

این تابع جستجوی عمقی بازگشتی را برای حرکت در محیط پیاده‌سازی می‌کند:

- موقعیت فعلی عامل و محتوای آن در محیط را در متغیر `perception_history`ذخیره می‌کند.
- اگر عامل به هدف (علامت `f`)برسد، مسیر پیمایش‌شده (`path`) و تعداد حرکت‌ها (`move_count`)بازگردانده می‌شود.
- اگر طول مسیر به حد مشخص‌شده `limit`برسد، تابع بدون بازگشت مسیر، خاتمه می‌یابد.
- در غیر این صورت، با اجرای عملگر `agent(percept)`, اقدام مناسبی برای حرکت عامل انتخاب می‌شود و سپس این حرکت اجرا می‌شود.
- اگر حرکت فعلی به نتیجه نرسد، تابع برای حرکات دیگر (`UP, RIGHT, DOWN, LEFT`) نیز امتحان می‌شود تا به مسیر مناسب دست یابد.

`agent(percept)`

این تابع تصمیم‌گیری عامل است که اقدامات ممکن (`RIGHT, DOWN, UP, LEFT`) را بررسی می‌کند تا مسیری بدون مانع (*) بیابد و آن مسیر را انتخاب کند.

ids(max_depth=20)

این تابع جستجوی عمقی تکرارشونده را با افزایش تدریجی عمق جستجو انجام می‌دهد. در هر مرحله، یک مجموعه جدید از موقعیت‌های بازدید شده (visited) ایجاد می‌کند و تابع recursive_dfsرا با محدودیت جدیدی فراخوانی می‌کند. اگر مسیری به هدف پیدا شود، آن مسیر و تعداد حرکت‌ها را بازمی‌گرداند.

---

۳ .اجرای برنامه

در بخش نهایی، برنامه بررسی می‌کند که موقعیت اولیه عامل قابل دسترسی باشد (if env[agent_pos[0]][agent_pos[1]] != '*'). اگر عامل در موقعیت غیرقابل دسترسی باشد، پیام خطایی نمایش می‌دهد.

در غیر این صورت:

- جستجوی عمقی تکرارشونده (ids) فراخوانی می‌شود.
- اگر مسیری یافت شود، تعداد حرکات و مراحل حرکت چاپ می‌شود.
- اگر مسیری پیدا نشود، پیام "No path found" نمایش داده می‌شود.

جمع‌بندی

این کد با استفاده از ترکیبی از جستجوی عمقی بازگشتی وIDS ، محیطی را پیمایش می‌کند و عامل را از موقعیت شروع تا هدف مشخص هدایت می‌کند.

```
Number of Moves: 4
Moves Steps:
[3,2]--RIGHT-->[3,3]--RIGHT-->[3,4]--RIGHT-->[3,5]--RIGHT-->[3,6]
PS D:\Uni\project 1-2>
```

input.txt

```
6,10
3,2

**********
*--------*
*--***---*
*-a---f--*
*-*--*---*
**********
```

```
Number of Moves: 16
Moves Steps:
[4,8]--UP-->[3,8]--LEFT-->[3,7]--LEFT-->[3,6]
PS D:\Uni\project 1-2>
```

input.txt

```
6,10
4,8

**********
*--------*
*--***---*
*-----f--*
*-*--*--a*
**********
```

```
Agent is not in a valid position.
PS D:\Uni\project 1-2>
```

input.txt

```
6,10
2,0

**********
*--------*
*--***---*
*-----f--*
*-*--*---*
**********
```

```
Number of Moves: 0
Moves Steps:
[3,6]
PS D:\Uni\project 1-2>
```

input.txt

```
6,10
3,6

**********
*--------*
*--***---*
*-----f--*
*-*--*---*
**********
```