فاز دوم پروژه شاور: پیاده‌سازی کنش‌گرهای هوشمند

در فاز اول پروژه، محیط شبیه‌ساز بازی شاور را به همراه یک کنش‌گر غیرهوشمند (تصادفی) پیاده‌سازی کردیم.

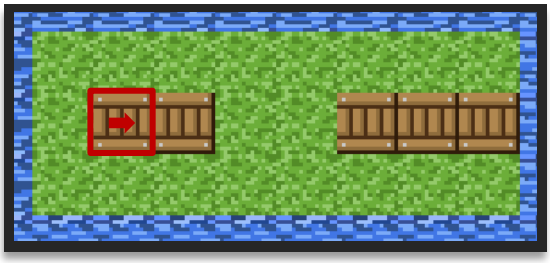
در فاز دوم پروژه، محیط شبیه‌ساز شده یکسانی از پیش به همه دانشجویان داده می‌شود. (این فایل در سامانه درس‌افزار و گروه پیام‌رسان درس، قابل دسترسی است.)

**** قوانین محاسبه هزینه نسبت به شرح مسئله در فاز اول، شامل دو تغییرکوچک اما مهم زیر بوده است:**

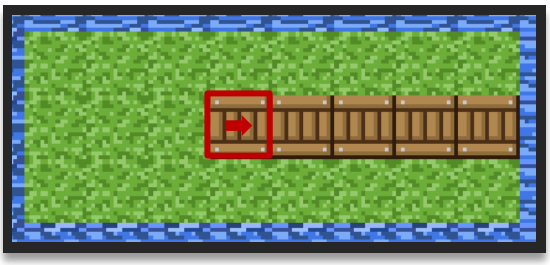
۱. هزینه اضافی حرکت طبق شرایط از پیش توضیح داده شده، از (۱+) به (۴+) تغییر پیدا کرده است.

۲. اگر طول قطاری از جعبه‌ها که در حرکت قبل جابجا شده‌اند a در حرکت جدید به b جعبه دیگر ملحق شوند تنها در صورتی هزینه نیروی اولیه همچنان صفر در نظر گرفته می‌شود که $a \geq b$ باشد.

** به مثال زیر توجه کنید:*



(۱)



(۲)

به جعبه با کادر قرمز توجه کنید (۱). اگر جعبه را برای بار اول به سمت راست هل دهیم، ۲+۴ هزینه خواهیم داشت. اگر همان جعبه را دوباره در جهت قبلی هل دهیم، هزینه ۲+۰ خواهد بود. حال به وضعیت شکل (۲) رسیده‌ایم. اگر همان جعبه مشخص شده در (۲) را دوباره در راستای قبلی هل دهیم، دوباره هزینه اضافی اعمال خواهد شد و در مجموع ۵+۴ هزینه صرف می‌شود، چرا که قطار جعبه روبه‌روی آن بزرگتر از قطاری که هل دادیم بود.

در ادامه، به توضیحات مربوط به کار با فایل پروژه و قوانین برنامه‌نویسی کنش‌گر خواهیم پرداخت که لازم است به آن‌ها توجه شود.

* قوانین و نحوه کار با فایل پروژه:

- زبان پیاده‌سازی پروژه پایتون است و طبعا کنش‌گر دانشجو در همان زبان نوشته خواهد شد.
- برای اجرای بازی نیاز به نصب یا بروزرسانی به آخرین نسخه کتابخانه pygame دارید.
- برای اجرای بازی، فایل main.py بایستی اجرا شود.

- پروژه شامل قسمت‌های مختلفی است، اما **دانشجو تنها مجاز به اعمال تغییرات در فایل *ai.py* است**. یعنی نه تنها در برنامه‌نویسی دیگر فایل‌ها نباید تغییری ایجاد شود، بلکه import **از هیچ‌کدام از فایل‌های دیگر پروژه نیز مجاز نیست**.
- ورودی الگوریتم یک لیست ارسال شده از سمت شبیه‌ساز از وضعیت فعلی آن است، که با متد ()perceive در کلاس Agent قابل دریافت است.
- خروجی الگوریتم می‌تواند یک **تک‌کنش** یا **دنباله‌ای از کنش‌ها** باشد. (کنش در برنامه یک شئ است، که شامل سه ویژگی مختصات x، y و جهت هل دادن می‌باشد. به طبع آن، برای ایجاد شئ جدید کنش لازم است‌که Action(x, y, direction) فراخوانی شود.)
- متد ()act در کلاس Agent، وظیفه‌اش ارسال یک کنش به محیط شبیه‌ساز در ازای درخواست محیط است و از پیش، جهت سهولت کار، به طور کامل پیاده‌سازی شده است: اگر خروجی الگوریتم دنباله‌ای از کنش‌ها باشد، در زمان فراخوانی ()agent.act در main، کنش‌ها یکی یکی پاپ و اجرا می‌شوند. اگر خروجی الگوریتم تک‌کنش باشد، متد ()act آن را به محیط ارسال می‌کند و در صورت درخواست دوباره محیط، الگوریتم برای ارسال کنش جدید دوباره اجرا می‌شود.
- می‌توان برای قواعد بازی مورد نیاز در کلاس State در فایل *ai.py*، از کد داخل فایل *env.py* الهام گرفت و یا بدون نیاز به تغییری، رونوشت و استفاده کرد. اما باز هم تاکید می‌شود به هیچ‌وجه import از فایل‌های دیگر پروژه در فایل *ai.py* صورت نگیرد.
- فایل *ai.py* قابلیت اجرای الگوریتم‌های متفاوت به تعداد دلخواه دارد. برای انتخاب نوع کنش‌گر، کافی است در کلاس Agent، در متد __init__، agent_type را با توجه به کنش‌گرهای موجود در دیکشنری agent_type_dict انتخاب شود. (واضح است **در صورت تمایل به اضافه کردن الگوریتم‌های دیگر**، لازم است مشابه الگوی استفاده شده در agent_type_dict، تابع الگوریتم جدید به لیست آن اضافه گردد.)
- در فایل other_agents.py، دیگر الگوریتم‌هایی برای حل مسئله پیاده‌سازی شده‌اند که دانشجویان می‌توانند آن‌ها را اجرا کنند و عملکرد آن‌ها را با جستجوهای درختی کلاسیک مقایسه کنند.

❋ وظایف دانشجو در انجام پروژه:

- در گام اول، پیاده‌سازی الگوریتم‌های IDS، A* و RBFS با رعایت ساختاربندی مناسب، برای **حل مسئله‌های 4x4 و 5x5** که ۵۰ درصد مساحت زمین جعبه، ۵ درصد مانع و بدون پرتگاه اضافی هست، پیاده‌سازی و تست کنید. (پروژه در همین حد قابل تحویل است و نمره کامل خواهد گرفت.)
  - نکته لازم توجه این است که در یک زمین با ابعاد m و n، 4mn کنش قابل انجام است و این ضریب انشعاب بسیار بزرگی برای یک درخت جستجو است. در نتیجه، الگوریتم‌ها در زمینی با ابعاد بزرگ‌تر، حالت‌های بسیار زیادی را باید بررسی کنند، که این باعث طولانی شدن زمان رسیدن به پاسخ می‌شود.
  - هیوریستیک استفاده شده در A* و RBFS بایستی رسیدن به حالت بهینه را در شرایط مذکور تضمین کند. (❋ هر چه هیوریستیک وابستگی کمتری به شرایط بالا، در عین حفظ بهینگی راه حل، داشته باشد، بهتر است.)
- ❋ در گام بعدی، برنامه‌هایی که قادر به حل زمین‌های بزرگتر (حدود 7x7 تا 11x11 با همان نسبت ۵۰ درصد جعبه، ۵ درصد مانع و بدون پرتگاه اضافی) با کاربست یک planning هیوریستیک بر روی یکی از سه الگوریتم فوق باشد، نمره دوبرابر خواهند گرفت.
- ❋ دانشجو همچنین می‌تواند نقشه‌های چالشی را به کمک رابط کاربری محیط گرافیکی طراحی و ذخیره کند. به چالشی‌ترین نقشه‌ها با ابعاد مناسب، امتیاز خوبی تعلق می‌گیرد.

با آرزوی موفقیت