شبیه‌ساز آسانسور

نام نویسندگان:

محمدجواد محمدی

استاد:

دکتر باقرپور

تابستان 1402

مقدمه

در این پروژه ما به وسلیه یکی از الگوریم های هوش مصنوعی به نام A* که نوعی الگوریتم search است مجموعه ای از آسانسور ها را طراحی کرده ایم که در هر مرحله با سنجیدن وضعیت بهترین حرکت را انجام میدهند. الویت با زمان انتظار کمتر برای استفاده کنندگان و بعد از ان مصرف انرژی کمتر میباشد. ایده ای که در این پروژه استفاده شده است میتواند تجاری نیز شود، چرا که در اکثر برج ها و ساختمان هایی که از چندین اسانسور استفاده میکنند، هر اسانسور یک کلید دارد اکه این باعث نا هماهنگی و غیر بهینه بودن اسانسور ها میشود. اگر سیستم مرکزی اسانسور وجود داشته باشد میتواند بهترین و نزدیک ترین اسانسور را به طبقه درخواست کننده برساند.

ساختار فایل‌ها

state

فایل State: در این فایل کلاس State قرار دارد که مشخص کننده یکی از وضعیت های ممکن ساختمان میباشد. (نحوه قرار گیری آسانسورها و افرادی که منتظر آسانسور هستند.)

در این کلاس متد های مختلفی وجود دارد که در ادامه به شرح آن ها میپردازیم:

- __init__: در این متد مقادیر اولیه تنظیم میشوند. این متغییر ها شامل تعداد طبقات، تعداد آسانسور ها هستند. اگر وضعیت از حالت اولیه شروع نشده باشد، متغییر های waiting_list و elv_loc برای مشخص کردن وضعیت افراد منتظر و مکان اسانسور ها داده میشود.
- add_floor: وقتی این تابع فراخوانی شود، یک طبقه به لیست انتظار اضافه میشود، اگر قبلا در ان طبقه منتظر بوده باشند، زمان صفر نمیشود در غیر این صورت زمان صفر برای این طبقه ذخیره میشود.

- doit: برای اعمال یک action روی یک وضعیت استفاده میشود. به عنوان مثال action میتواند دستورات UP، DOWN و یا STOP برای هر اسانسور باشد. اگر اسانسور در طبقه ای STOP کند، آن طبقه خالی شده و از لیست انتظار حذف میشود.
- transition: با گرفتن یک لیست از actions ها یک وضعیت جدید میسازد و آن ها را روی آن اجرا کرده و خروجی میدهد.
- cost: تابع ای که یک مقدار به اسم هزینه را برای یک وضعیت مشخص برمیگرداند. این مقدار حاصل جمع تمام زمان هایی است که اشخاص در طبقات مختلف منتظر مانده اند به علاوه فاصله هر طبقه در حال انتظار از نزدیک ترین آسانسور به آن. هرچقدر این عدد کوچک تر باشد یعنی وضعیت بهتری داریم.
- actions: تابع ای که میتواند تمام action هایی که روی یک وضعیت قابل پیاده سازی هستند را خروجی دهد. اگر اسانسوری در همکف باشد پایین نمیتواند برود و اگر اسانسوری در طبقه اخر باشد نمیتواند بالا برود.
- decision: تابع ای که بر اساس توابع قبلی بهترین حرکت را روی وضعیت اعمال میکند. کارکرد ان به این شکل است که در ابتدا با تابع actions تمام action ها را روی یک وضعیت پیاده سازی میکند و cost آن ها را در دیکشنری ذخیره میکند. بعد از ان لیست تصمیم را میسازد. در ابتدا ان را به صورتی مرتب (sort) میکند که action هایی که STOP بیشتری دارند اول لیست قرار بگیرند. (سکون بر حرکت مقدم است و انرژی کمتری مصرف میشود.) سپس مرتب سازی را بر اساس cost های ان ها انجام میدهد.

این که مرتب سازی اول بر اساس تعداد STOP ها است باعث میشود اگر دو حالت cost یکسانی داشته باشند، حالتی انتخاب شود که STOP بیشتری دارد.

در نهایت عضو اول این لیست روی حالت با تابع doit اعمال میشود.

UI

در این فایل گرافیک یا GUI مربوط به پروژه زده شده است. از انجا که هدف اصلی پروژه الگوریتم بوده است نه کار با گرافیک، فایل اصلی همان فایل State میباشد و از شرح دادن فایل UI صرف نظر میکنیم.

نحوه کارکردن با نرم افزار

ران کردن کد

برای استفاده از نرم افزار فایل UI را اجرا کنید. در ابتدا از شما خواسته میشود که تعداد طبقات و تعداد اسانسور ها را مشخص کنید. به علت محدودیت صفحه نمایش تعداد طبقات نمیتواند بیشتر از 14 باشد. همچنین تعداد اسانسور ها نمیتوانند بیشتر از ۵ باشد. برای تغییر این محدودیت میتوانید از فایل UI در ابتدای ان متغیر MAX_FLR برای تعداد طبقات و MAX_ELV برای تعداد اسانسور ها را تغییر دهید.

با زدن گزینه ثبت وارد شبیه ساز میشویم. با کلیک کردن روی هر کدام از دکمه های مربوط به طبقات میتوانیم مشخص کنیم که اسانسور باید به ان طبقه برود. و با زدن دکمه next step میتوانیم یک مرحله حرکت اسانسور ها را ببینیم. با زدن دکمه return به صفحه قبل باز میگردیم.

استفاده از فایل اجرایی

با باز کردن فایل ElevatorSimiulation.exe میتوانیم به طور مستقیم و حتی بدون نصب داشتن پایتون با نرم افزار کار کنیم.