به نام خداوند بخشنده‌ی مهربان

پاسخ تمرین عملی اول هوش مصنوعی

نیم‌سال اول ۱۴۰۰-۱۴۰۱

سید پارسا نشایی - ۹۸۱۰۶۱۳۴

دانشگاه صنعتی شریف
دانشکده‌ی مهندسی کامپیوتر

مقدمه

در این تمرین، فرآیند یافتن و رسیدن به هدف (منبع تغذیه یا باتری) را توسط یک ربات شبیه‌سازی کرده، به گونه‌ای که ربات بتواند با طی کردن حداقل گام‌های ممکن و قبل از آن‌که انرژی‌اش تمام شود، به منبع تغذیه برسد. برای حل مسئله از روش‌های گوناگون جست‌وجو که درس خوانده‌ایم استفاده شده و نتایج آن‌ها با یک‌دیگر مقایسه شده است. مکان ربات و باتری و نیز ابعاد محیط و موانع موجود در آن، در قالب فایل XML به برنامه داده می‌شوند. جزئیات بیش‌تر از چگونگی عملکرد برنامه، در ادامه آمده است.

نحوه‌ی اجرا

برنامه به زبان Python نوشته شده و روی نسخه‌ی ۳ این زبان تست شده است. در صورت اجرا نشدن برنامه، ممکن است مشکل از تفاوت کتاب‌خانه‌ی مربوط به دستور print در نسخه‌های مختلف پایتون باشد که با حذف خط ۱ برنامه (که import یک کتاب‌خانه‌ی داخلی پایتون و مربوط به تابع print است) این مشکل حل خواهد شد. دیگر کتاب‌خانه‌های

import شده، کتاب‌خانه‌هایی داخلی برای تولید اعداد تصادفی و نیز parse کردن XML اند؛ برای حل مسئله، از هیچ ابزار، کتاب‌خانه یا toolkit آماده در راستای حل مسئله استفاده نشده و الگوریتم به شکل دستی نوشته شده است. برای اجرای برنامه کافیست فایل‌های XML با شماره‌های یک تا شش داده شده، در پوشه‌ی شامل فایل code.py قرار گرفته و سپس code.py اجرا شود.

با توجه به آن‌چه در کلاس گفته شد، به سبب تلاش برای اختصار مطالب گفته شده در این گزارش، از شرح کد اجتناب شده است، گرچه اسامی متغیرها به گونه‌ای واضح قرار داده شده‌اند که کد را self-explanatory کرده‌اند و نیاز به توضیح آن نیست. ضمنا، در خصوص ابهامی که بخشی از کلاس صبح دوشنبه ۱ آذر ۱۴۰۰ به رفع آن و صحبت دانش‌جویان درس در خصوص آن گذشت، بیان شد که می‌توان فرضیاتی را در بیان راه‌حل برای سوال در نظر گرفته و طبق آن فرضیات – که در این‌جا فرض آن است که می‌توان هیوریستیک تا مقصد را حساب کرد – سوال را حل کرد.

برای هر ورودی و نیز هر الگوریتم، ابتدا هر گام طی شده به شکل گرافیکی در یک grid چاپ می‌شود (ستاره بیانگر مکان کنونی ربات، # بیانگر مانع و B بیانگر باتری است) و سپس طول مسیر پیدا شده توسط الگوریتم (با حساب راس مبدا)، تعداد چک‌های صورت گرفته (یعنی تعداد بارهایی که «راس»ی را بررسی کرده‌ایم) با حساب راس مبدا، و سپس مسافتی (واقعی) که خود ربات می‌بایسته طی کند تا به خانه‌هایی که در الگوریتم Explore می‌شوند برسد چاپ شده و سپس در یک grid، مسیر پیدا شده به عنوان پاسخ توسط الگوریتم نمایش داده می‌شود. در مسیر نهایی چاپ شده، S به معنی مکان اولیه‌ی ربات، ستاره بیانگر رئوسی که در مسیر پاسخ واقع‌اند، # بیانگر مانع و B بیانگر باتری است. سپس به تفکیک مختصات خانه‌های در مسیر پاسخ – از مبدا حرکت ربات تا باتری – نیز چاپ می‌شوند.

راه‌حل

در جریان راه‌حل زیر، به تمامی سوالات پرسیده شده در فایل تمرین، پاسخ داده شده است. الگوریتم‌های استفاده شده برای بررسی، عبارت‌اند از:

- الگوریتم *A با تابع هیوریستیک جمع فاصله‌ی افقی و عمودی تا هدف (فاصله‌ی منهتن)
  - این تابع قابل قبول است، زیرا در سریع‌ترین حالت رسیدن به هدف، مانعی جلوی ما نیست و مستقیم تا هم‌خط‌افقی‌شدن تا هدف جابه‌جا می‌شویم (به شکل عمودی) و سپس به شکل افقی مستقیم به سمت مبدا می‌رویم، پس این تابع همواره کران پایینی از هزینه‌ی واقعی است.
  - این تابع سازگار نیز هست، زیرا $h(n')$ در کم‌ترین حالت خود، می‌تواند یک واحد از $h(n)$ کم‌تر باشد (زیرا با هر جابه‌جایی افقی یا عمودی، تنها یک خانه در یک بعد جابه‌جا می‌شویم و ماکسیمم یکی از فاصله‌ی منهتن کم

می‌شود) و از طرفی هزینه‌ی جابه‌جایی بین دو خانه یعنی $c(n, a, n')$ هم یک است، در نتیجه رابطه‌ی اسلاید ۲۳ لکچر Informed Search به وضوح برقرار است.

- در نتیجه، مسیر بهینه توسط این تابع پیدا می‌شود. (البته توضیحات فوق و مشابه آن در کلاس نیز گفته شده و صرفا به جهت یادآوری آورده شده‌اند)

- الگوریتم A* با تابع هیوریستیک واقعی
  - هیوریستیک این تابع به کمک فراخوانی A* قبلی، به ازای هر نقطه، فاصله‌ی واقعی‌اش تا هدف را می‌یابد و به وضوح قابل قبول و سازگار است، زیرا برابر هزینه‌ی واقعی است و تمامی شرایط گفته شده بدیهتا برای آن برقرار است
- الگوریتم A* با تابع هیوریستیک عمودی تا هدف
- الگوریتم A* با تابع هیوریستیک افقی تا هدف
  - به وضوح دو هیوریستیک فوق (بر اساس مطالب لکچرها) نیز سبب می‌شوند تا مسیر بهینه پیدا شود
- الگوریتم A* با تابع هیوریستیک با خروجی تصادفی (که یک هیوریستیک بدون منطق خاص است)
- الگوریتم A* بدون تابع هیوریستیک (به عبارت دیگر، جست‌وجوی هزینه یکنواخت)
- الگوریتم DFS
  - در خصوص بهینه بودن سه مورد فوق در ادامه و در بخش تحلیل خروجی‌ها صحبت شده است.

خروجی اجرای گرافیکی و متنی الگوریتم به ازای ۶ ورودی نمونه داده شده، در فایل output.txt در کنار فایل پایتون موجود است.

نکته: یک عدد seed ثابت به پایتون داده شده تا برنامه هنگام هر بار اجرا، اعداد رندوم یکسانی تولید کند تا قابل دیباگ باشد؛ طبیعتا اگر این کد روی production قرار است deploy شود، می‌توان این خط را حذف کرد.

تحلیل خروجی‌ها

- فایل SampleRoom.xml (فایل اصلی قرار داده شده در CWی درس)
- ابتدا A* با Manhattan Heuristic فراخوانی شده که پس از ۲۳ بار چک نودها (شماره‌ها با حساب نود اولیه نوشته شده‌اند)، مسیری به طول ۱۲ از پایین–راست پیدا کرده است که مسیر بهینه‌ی رسیدن ربات به باتری

در این grid نیز همین است و ربات برای یافتن این مسیر ۸۷ مربع را به شکل فیزیکی پیموده است. هیوریستیک واقعی نیز مشخصا همین پاسخ را داده است. ای‌استار با هیوریستیک عمودی و افقی نیز توانسته مسیر بهینه را پیدا کند، اما نیاز به چک کردن گام‌های بیش‌تری (۲ گام بیش‌تر نسبت به منهتنی – در کل، ۲۵ گام) داشته است (زیرا از to the point بودن هیوریستیک و میزان آگاهانه بودنش کاسته شده است) و هیوریستیک‌های عمودی و افقی به ترتیب ۹۱ و ۱۰۵ مربع را فیزیکی پیموده‌اند تا مسیر بهینه را بیابند. هیوریستیک رندوم لزوما to the point نیست اما در این مثال توانسته از نظر تعداد گام برای یافتن هدف، هیوریستیک همواره فاصله‌ی افقی یا همواره عمودی – به دلیل randomness ذاتی‌اش – بهتر عمل کند و در ۲۴ گام (و ۴۱ حرکت فیزیکی)، هدف را یافته است، اما به دلیل آن که این هیوریستیک admissible نیست (زیرا ممکن است حتی اگر فاصله‌ی ربات تا مقصد یک باشد، عددی بیش‌تر از یک را خروجی دهد و به مثال نقض، ثابت می‌شود که این هیوریستیک، کران پایینی از فاصله‌ی واقعی ربات تا باتری نیست)، مسیر بهینه خروجی داده نشده است و طول مسیر پیدا شده توسط این هیوریستیک، ۱۴ است و نه ۱۲. در نسخه‌ی بدون هیوریستیک نیز با ۲۶ چک و ۱۱۵ گام فیزیکی، مسیر به طول ۱۲ (بهینه) پیدا شده است، زیرا می‌دانیم (از اسلاید ۴۳ لکچر Solving Problems by searching) که جست‌وجوی هزینه یکنواخت، بهینه است. در انتها، الگوریتم DFS به وضوح بهینه نیست (مطابق اسلاید ۵۸ لکچر Solving Problems by searching) که در این مثال نیز مشاهده می‌شود که در ۲۶ چک، مسیر به طول ۱۴ – و نه ۱۲ – پیدا شده است؛ ولی به دلیل ماهیت آن که سعی می‌کند تعدادی مسیر را تا انتها (عمق) رفته و سپس مسیرهای دیگر را تست کند، در این مثال، با تعداد گام فیزیکی کم‌تری (۳۷ گام) به نسبت A*، به جواب البته نه لزوما بهینه (و در این مثال، نابهینه) می‌رسد.

- فایل SampleRoom2.xml - ورودی‌ای که در آن جای ربات و باتری کمی عوض شده است
- ابتدا A* با Manhattan Heuristic فراخوانی شده که پس از ۹ بار چک نودها (شماره‌ها با حساب نود اولیه نوشته شده‌اند) و ۱۰ گام فیزیکی، مسیری به طول ۷ از سمت راست پیدا کرده است که مسیر بهینه‌ی رسیدن ربات به باتری در این grid نیز همین است. هیوریستیک واقعی نیز مشخصا همین پاسخ را داده است. ای‌استار با هیوریستیک عمودی و افقی نیز توانسته مسیر بهینه را پیدا کند، اما نیاز به چک کردن گام‌های بیش‌تری (به ترتیب ۷ و ۲ گام بیش‌تر نسبت به منهتنی – در کل، به ترتیب ۱۶ و ۱۱ گام) و نیز به ترتیب ۶۰ و ۳۲ گام فیزیکی داشته است (زیرا از to the point بودن هیوریستیک و میزان آگاهانه بودنش کاسته شده است). این بار هیوریستیک رندوم نیز توانسته به تصادف مسیر بهینه را بیابد (به طول ۷) و در ۱۲ گام و ۱۶ گام فیزیکی به این یافته رسیده است. در نسخه‌ی بدون هیوریستیک نیز با ۱۷ چک (که تعداد چک‌ها بیش‌تر بوده،

چون هیوریستیک مشخصی نداشتیم که به ما در جست‌وجوی سریع‌تر و آگاهانه کمک کند) و تعداد قابل توجه ۸۰ گام فیزیکی، مسیر به طول ۷ (بهینه) پیدا شده است، زیرا می‌دانیم که جست‌وجوی هزینه یکنواخت، بهینه است. در انتها، الگوریتم DFS نیز این بار توانست مسیر بهینه را بیابد (در حالی که تضمین نمی‌شد که بتواند) و در ۸ گام کلی و البته حتی از نظر فیزیکی هم ۸ گام، توانسته این مسیر را بیابد.

- فایل SampleRoom3.xml – ورودی‌ای که مشابه ورودی دوم بوده، ولی جای یک مانع (ستون اول از راست سطر سوم) در آن متفاوت است، به گونه‌ای که این مانع، مسیر از راست را مسدود کرده است
  - ابتدا A* با Manhattan Heuristic فراخوانی شده که پس از ۲۶ بار چک نودها (شماره‌ها با حساب نود اولیه نوشته شده‌اند) و ۴۴ گام فیزیکی، مسیری به طول ۱۵ از سمت چپ پیدا کرده است که مسیر بهینه‌ی رسیدن ربات به باتری در این grid نیز همین است. هیوریستیک واقعی نیز مشخصا همین پاسخ را داده است. ای‌استار با هیوریستیک عمودی و افقی نیز توانسته‌اند مسیر بهینه را با ۲۶ گام پیدا کنند، ولی تعداد گام‌های فیزیکی پیموده شده توسط آن‌ها بیش‌تر و به ترتیب برابر ۸۲ و ۵۸ است. این بار هیوریستیک رندوم نیز توانسته به تصادف مسیر بهینه را بیابد (به طول ۱۵) و در ۲۵ چک و ۴۲ گام فیزیکی، به این یافته رسیده است. در نسخه‌ی بدون هیوریستیک نیز با ۲۶ چک و ۷۲ گام فیزیکی، مسیر به طول ۱۵ (بهینه) پیدا شده است، زیرا می‌دانیم که جست‌وجوی هزینه یکنواخت، بهینه است. در انتها، الگوریتم DFS نیز این بار توانست مسیر بهینه را بیابد (در حالی که تضمین نمی‌شد که بتواند) و در ۲۴ چک و ۳۰ گام فیزیکی، توانسته این مسیر را بیابد.
- فایل SampleRoom4.xml – ورودی‌ای که به جای ابعاد ۶ در ۶ قبلی‌ها، ابعاد ۷ در ۷ دارد تا اثر تغییر ابعاد قابل بررسی باشد و البته موانع در آن به شکلی چیده شده‌اند که مسیری از ربات تا باتری وجود ندارد و تمام مسیرها به بن‌بست می‌خورند
  - تمام روش‌ها در این ورودی، پاسخ صفر به عنوان طول مسیر بر می‌گردانند که در این کد به این معناست که مسیری از ربات به باتری وجود ندارد. در DFS و نیز A* با هیوریستیک منهتن، واقعی، افقی و عمودی و نیز بدون هیوریستیک، با ۷ چک و ۱۰ گام فیزیکی و در A* با هیوریستیک رندوم با ۷ چک و ۱۱ گام فیزیکی می‌توان به این نتیجه رسید.
- فایل SampleRoom5.xml – ورودی‌ای که مشابه ورودی چهارم بوده، ولی جای یک مانع (ستون اول از چپ سطر سوم) در آن متفاوت است، به گونه‌ای که این مانع، مسیر از چپ را که مسدود بود، آزاد کرده است
  - ابتدا A* با Manhattan Heuristic فراخوانی شده که پس از ۲۱ بار چک نودها (شماره‌ها با حساب نود اولیه نوشته شده‌اند) و ۳۲ گام فیزیکی، مسیری به طول ۱۵ از سمت چپ پیدا کرده است که مسیر بهینه‌ی رسیدن ربات به باتری در این grid نیز همین است. هیوریستیک واقعی نیز مشخصا همین پاسخ را داده است.

ای‌استار با هیوریستیک عمودی و افقی نیز توانسته‌اند مسیر بهینه را با به ترتیب ۲۲ چک / ۴۰ گام فیزیکی، و ۲۴ چک / ۴۰ گام فیزیکی (تعداد گام‌ها بیش‌تر از منهتن شد زیرا همان‌طور که پیش‌تر گفته شد از to the point بودن هیوریستیک‌ها نسبت به منهتن کاسته شده است) پیدا کنند. این بار هیوریستیک رندوم نیز توانسته به تصادف مسیر بهینه را بیابد (به طول ۱۵) ولی در ۳۳ چک و ۶۶ گام فیزیکی به این یافته رسیده است. در نسخه‌ی بدون هیوریستیک نیز با ۲۴ چک و ۴۶ گام فیزیکی، مسیر به طول ۱۵ (بهینه) پیدا شده است، زیرا می‌دانیم که جست‌وجوی هزینه یکنواخت، بهینه است. در انتها، الگوریتم DFS به وضوح بهینه نیست که در این مثال نیز مشاهده می‌شود که در ۳۶ چک و ۴۸ گام فیزیکی، مسیر به طول ۱۷ – و نه ۱۵ – پیدا شده است.

- فایل SampleRoom6.xml - ورودی‌ای که با ابعاد غیرمربعی (۵ سطر و ۳ ستون) طراحی شده تا صحت عملکرد کد در چنین ورودی‌هایی را نیز بسنجد
- ابتدا A* با Manhattan Heuristic فراخوانی شده که پس از ۷ بار چک نودها (شماره‌ها با حساب نود اولیه نوشته شده‌اند) و ۶ گام فیزیکی، مسیری به طول ۷ از سمت چپ پیدا کرده است که مسیر بهینه‌ی رسیدن ربات به باتری در این grid نیز همین است. هیوریستیک واقعی نیز مشخصا همین پاسخ را داده است. ای‌استار با هیوریستیک عمودی و افقی نیز توانسته‌اند مسیر بهینه را با به ترتیب ۸ چک / ۱۰ گام فیزیکی و ۱۱ چک / ۲۰ گام فیزیکی (تعداد گام‌ها بیش‌تر از منهتن شد زیرا همان‌طور که پیش‌تر گفته شد از to the point بودن هیوریستیک‌ها نسبت به منهتن کاسته شده است) پیدا کنند. این بار هیوریستیک رندوم نیز توانسته به تصادف مسیر بهینه را بیابد (به طول ۷) ولی در ۱۱ چک و ۱۴ گام فیزیکی به این یافته رسیده است. در نسخه‌ی بدون هیوریستیک نیز با ۱۲ چک و ۲۲ گام فیزیکی، مسیر به طول ۷ (بهینه) پیدا شده است، زیرا می‌دانیم که جست‌وجوی هزینه یکنواخت، بهینه است. در انتها، الگوریتم DFS نیز این بار توانست مسیر بهینه را بیابد (در حالی که تضمین نمی‌شد که بتواند) ولی در ۱۳ چک و ۱۴ گام فیزیکی توانسته این مسیر را بیابد.

در کل می‌توان نتیجه گرفت که اگر دنبال بهینه بودن جواب حاصل نباشیم، به دلیل ماهیت سریع‌تر رسیدن به جواب و عمدتا تعداد گام‌های فیزیکی کم‌تر، A* با هیوریستیک واقعی / منهتن و نیز DFS از میان الگوریتم‌های امتحان شده در فوق، بهتر عمل می‌کنند، اما اگر به دنبال جواب بهینه باشیم، گزینه‌ی DFS حذف شده و A* با هیوریستیک واقعی / منهتن مناسب‌ترین‌اند؛ هم‌چنین، چون محاسبه‌ی هیوریستیک واقعی به اطلاع از موانع قبل از حرکت نیاز داشته و در عمل ناممکن است، هیوریستیک منهتن میان گزینه‌های امتحان شده در فوق، در کل وضعیت بهتری داشته است، زیرا هم بهینه بوده و هم به نسبت در تعداد گام‌های فیزیکی کم‌تری به جواب می‌رسد.