بسمه تعالی

گزارش پروژه اول هوش مصنوعی - سوال اول - اسلاید پازل

محمدباقر عابدی‌سقّا – ۹۵۳۱۹۰۴

روند مدل سازی مسئله:

ابتدا یک ورودی از کاربر گرفته می‌شود که در هر خط با فاصله ۳ خانه از پازل ورودی هست.

هر حالت در این مسئله به شکل یک رشته ذخیره می‌شود به این صورت که حالت اولیه '123456780' خواهد بود. یعنی سطر اول ۱ ۲ ۳ و سطر دوم ۴ ۵ ۶ و سطر سوم ۷ ۸ ۰ است.

سپس تمام 9! حالت مسئله ساخته شده (به کمک permutation()) و هر کدام در یک دیکشنری همراه با یک index ریخته می‌شود. الگوریتم های جستجوی مسئله با index عمل خواهند کرد نه با رشته حالت ها.

الگوریتم‌ها:

هر الگوریتم جستجو یک لیست از مسیر منتهی یه هدف برمی‌گرداند که گره هدف آخرین المان است.

الگوریتم bfs:

این الگوریتم مطابق سیاست جستجوی سطحی، در هر مرحله یک گره را expand کرده و در صورتی که فرزندان وی visited نباشند آنها را وارد یک صف می‌کند و هر بار تا زمانی که صف خالی نشده، یک گره از انتهای صف خارج کرده و همانند قبل expand می‌کند. در هر بار expand قبل از ورود به صف،goal test روی گره انجام می‌شود.

الگوریتم dfs unlimited:

این الگوریتم مطابق سیاست جستجوی عمقی، در هر مرحله پس از expand کردن حالت اولیه، فرزندان وی را درون یک پشته می‌ریزد و هر بار یک گره از آن بیرون آورده و آن را expand می‌کند. قبل از ورود به پشته تست goal و visited روی گره ها انجام می‌شود. پاسخ این الگوریتم بهینه نخواهد بود.

الگوریتم dfs limited:

این الگوریتم همانند عمقی کار می‌کند با این تفاوت که هر بار قبل از expand کردن یک گره بررسی می‌کند که عمق کاوش شده بیشتر از مقدار limit داده شده به آن نباشد. و در صورتی که بیشتر باشد، ابتدا گره های همسایه در عمق های کاوش شده را جستجو می‌کند. این روش لزوما به جواب نمی‌رسد.

الگوریتم dfs iterative:

این الگوریتم، جستجوی عمقی محدود را به صورت متوالی با عمق های شروع از ۱ آغاز می‌کند. در صورتی که

الگوریتم قادر به یافتن پاسخ نبود، مقدار limit داده شده به آن یک واحد افزایش می‌یابد.

الگوریتم bidirection:
این الگوریتم ابتدا هر دو گره هدف و فعلی را درون یک لیست اضافه می‌کند. در هر مرحله هر دو گره expand شده و گره های فرزند آنها به لیست اضافه می‌شود. در صورتی که این دو مجموعه باهم اشتراک پیدا کنند، مسیر از حالت اول و مسیر از حالت هدف به صورت برعکس با هم پس داده می‌شوند.

الگوریتم *A :
این الگوریتم در هر مرحله پس از expand کردن یک گره، هزینه تمام گره ها از ابتدا تا خودشان را می‌سنجد و و اگر ملاقات نشده باشند، آن را به عنوان گره بعدی در مسیر انتخاب می‌کند.
هزینه در این مسئله با تابع heuritsic فاصله منهتن از جای اصلی یک خانه ی پازل محاسبه خواهد شد.

الگوریتم unform cost search:
این الگوریتم همانند *A عمل خواهد کرد با این تفاوت که هنگام محاسبه هزینه مسیر، فقط هزینه تا این گره را بعلاوه یک می‌کند چون هزینه بین دو حالت در این مسئله واحد است.

توابع کمکی:
تابع get_children در کلاس problem با گرفتن یک حالت یا index آن، حالت های بعدی ممکن آن را پس می‌دهد. باتوجه به ساختار رشته ای هر حالت، فرزندان آن با جابجایی خانه '0' با یکی قبل و بعد خود و سه تا قبل و بعد خود به دست می‌آید.

تابع get_heuristic در کلاس problem فاصله ی منهتن یک گره از جای اصلی خود در حالت پاسخ را پس می‌دهد.

تابع solve در کلاس problem با گرفتن یک تابع جستجو و شرایط خاص الگوریتم های مختلف، آن تابع را با گره اولیه صدا می‌کند. در نهایت پاسخ دریافتی از الگوریتم، توسط تابع printer به صورت یک حلقه چاپ می‌شود.

تابع printer یک حالت یا ایندکس داده شده را به شکل ماتریس چاپ می‌کند.

تابع is_goal ایندکس یک حالت را گرفته و آن را با ایندکس حالت هدف (0) مقایسه می‌کند.

در ادامه پاسخ تمام الگوریتم‌ها برای ورودی داده شده را می‌آوریم:

حالت هدف:

| 3 2 1 |

| 6 5 4 |

| 0 8 7 |

حالت ورودی:

| 3 2 1 |

| 6 5 4 |

| 0 8 7 |

:Bfs answer

| 3 2 1 |
| 6 4 0 |
| 8 5 7 |

| 3 2 1 |
| 6 0 4 |
| 8 5 7 |

| 3 2 1 |
| 6 5 4 |
| 8 0 7 |

| 3 2 1 |
| 6 5 4 |
| 0 8 7 |

Dfs answer (last few steps!)

....

| 3 2 1 |
| 0 4 8 |
| 7 6 5 |

| 3 2 1 |
| 5 4 8 |
| 7 6 0 |

| 3 2 1 |
| 5 4 0 |
| 7 6 8 |

| 3 2 1 |
| 5 0 4 |
| 7 6 8 |

| 3 2 1 |
| 0 5 4 |
| 7 6 8 |

| 3 2 1 |
| 7 5 4 |
| 0 6 8 |

| 3 2 1 |
| 7 5 4 |
| 6 0 8 |

| 3 2 1 |
| 7 5 4 |
| 6 8 0 |

| 3 2 1 |
| 0 5 4 |
| 6 8 7 |

| 3 2 1 |
| 6 5 4 |
| 0 8 7 |

:Ldsf answer

| 3 2 1 |
| 6 4 0 |
| 8 5 7 |

| 3 2 1 |
| 6 0 4 |
| 8 5 7 |

| 3 2 1 |
| 6 5 4 |
| 8 0 7 |

| 3 2 1 |
| 6 5 4 |
| 0 8 7 |

:idfs answer (limit = 3 gets answer)

| 3 2 1 |
| 6 4 0 |
| 8 5 7 |

| 3 2 1 |
| 6 0 4 |
| 8 5 7 |

| 3 2 1 |
| 6 5 4 |
| 8 0 7 |

| 3 2 1 |
| 6 5 4 |
| 0 8 7 |

| 3 2 1 |
| 6 4 0 |
| 8 5 7 |

| 3 2 1 |
| 6 0 4 |
| 8 5 7 |

| 3 2 1 |
| 6 5 4 |
| 8 0 7 |

| 3 2 1 |
| 6 5 4 |
| 0 8 7 |

:Bidirectional answer

| 3 2 1 |
| 6 5 4 |
| 0 8 7 |

| 3 2 1 |
| 6 5 4 |
| 8 0 7 |

| 3 2 1 |
| 6 0 4 |
| 8 5 7 |

| 3 2 1 |
| 6 4 0 |
| 8 5 7 |

;A* answer

| 3 2 1 |
| 6 4 0 |
| 8 5 7 |

| 3 2 1 |
| 6 0 4 |
| 8 5 7 |

| 3 2 1 |
| 6 5 4 |
| 8 0 7 |

| 3 2 1 |
| 6 5 4 |
| 0 8 7 |

:Uniform cont search answer

| 3 2 1 |
| 6 4 0 |
| 8 5 7 |

| 3 2 1 |
| 6 0 4 |
| 8 5 7 |

| 3 2 1 |
| 6 5 4 |
| 8 0 7 |

| 3 2 1 |
| 6 5 4 |
| 0 8 7 |