استفاده از الگوریتم های جستوجوی ناآگاهانه برای حل مساله 8-puzzle

محسن لیاقت ۶۱۰۳۹۸۱۶۳

۱۴ بهمن ۱۴۰۱

فهرست مطالب

۱ تعاریف

states : تمام حالات ممکن قابل دسترس از حالت آغازین ($\frac{9!}{2}$ حالت)

Actions : بالا، پایین، چپ یا راست بردن کاشی ۰

Goal test : همه کاشی ها در محل درست باشند

path cost : ۱-

transition model : در قالب یک تابع با همین عنوان در کلاس ENV_8puzzle در فایل puzzle8.py آمده است.

۲ انتخاب تابع Heuristic

تابع Heuristic مورد نظر من مجموع کوتاه ترین فاصله هر کاشی با محل درست آن است که با نام hf در کلاس ENV_8puzzle در فایل puzzle8.py آمده است. دلیل انتخاب این تابع موارد زیر است :

- به وضوح admissible است زیرا ما مجموع کوتاه ترین فاصله ها را حساب می کنیم و هزینه واقعی از این کمتر نخواهد بود.
- consistent است زیرا باتوجه به اینکه هزینه هر عمل ۱ است خواهیم داشت.

$$\text{cost}_{(n,a,n')} + \text{hf}_{(n')} = 1 + \text{hf}_{(n')} \begin{cases} \text{place its to closer become tile a if} & = 1 + \text{hf}_{(n)} - 1 = \text{hf}_{(n)} \\ \text{O.W} & \geqslant 1 + \text{hf}_{(n)} \geqslant \text{hf}_{(n)} \end{cases}$$

- مقدار آن به هزینه واقعی تقریبا نزدیک است.

۳ مثال

همه مثال های داده شده اجرا شده اند و نتیجه اجرای آنها به همراه حافظه مصرفی آنها در هر الگوریتم در فایل اکسل به پیوست آمده است.(به دلیل محدودیت های سخت افزار لپتاپ خودم صرفا توانستم حداکثر عمق recursion را ۲۵۰۰ قرار دهم.)
به دلیل اینکه تعداد مثال ها زیاد بود از روی فایل مثال ها فایل های دیگری ساختم به نام easyex.text , medex.text , hardex.text که شامل همان مثال ها بود فقط فایل منظم تر شده بود و سپس همه ورودی ها را از آن فایل گرفتم و خروجی ها را در فایل res.csv ذخیره کردم سپس از روی آن فایل یک فایل اکسل ساختم. همه این فایل ها اگر سامانه ارور ندهد پیوست می شود :)
به دلیل اینکه پیچیدگی زمانی IDS نمایی است برای مثال های medium و hard دیگر این الگوریتم را اجرا نکردم (راستش چند بار اجرا کردم ولی هر بار بعد از گذشت چند ساعت هنوز همه مثال ها حل نشده بود.)

۱.۳ تحلیل نتایج

- الگوریتم DFS در حالات بسیاری به دلیل پر شدن استک terminate می کند نه به دلیل اینکه پاسخ را پیدا کرده است.
- الگوریتم BFS نسبت به DFS از حافظه بیشتری استفاده می کند و چون این حافظه به صورت هیپ است و سیستم عامل قابلیت صفحه بندی حافظه را دارد این الگوریتم با موفقیت پاسخ را می یا بد
- الگوریتم UCS چون از min-heap استفاده می کند حافظه و زمان بیشتری را برای یافتن پاسخ نسبت به BFS نیاز دارد.
- اگر به زمان پاسخ گویی IDS توجه شود مشاهده می شود که رشد این زمان نسب به رشد طول پاسخ بهینه نمایی است.
- الگوریتم A^* کمترین زمان و حافظه را نسبت به دیگر الگوریتم ها برای یافتن پاسخ نیاز دارد.