حسن کاظمی طهرانی

9629041

گزارش پروژه مبانی هوش مصنوعی

فرموله سازی مسئله:

فرموله سازی این مسئله به این صورت می باشد که :

حالت اولیه : وضعیت اولیه قرارگیری کارت ها در زمین ها حالت اولیه این مسئله می باشد.

State : حالت های مختلف قرار گیری کارت ها در زمین ها وضعیت های مختلف مسئله را نشان می دهد.

حالت هدف: حالت هدف این مسئله زمانی است که همه زمین های بازی به صورت نزولی مرتب شده باشند و یک رنگ باشند و یا خالی باشند.

Action : در این مسئله action به برداشتن بالاترین کارت یک زمین و قرار دادن در زمین خالی و یا بر روی کارت با شماره بیشتر گفته میشود.

هزینه مسیر: تعداد اقدام هایی که صورت گرفته است تا از حالت اولیه به وضعیت فعلی برسیم هزینه مسیر این مسئله می باشد.(هزینه هرجابجایی کارت برابر 0.2 در نظر گرفته شده است)

(برای پیاده سازی الگوریتم ها از شبه کد اسلاید استفاده شده است و ایده پیاده سازی از کتاب می باشد.)

الگوریتم bfs :

برای پیاده سازی این الگوریتم فضای مسئله در قالب یک کلاس به نام playground تعریف شده که هر node مارا تشکیل می دهد علاوه بر آن نیز این کلاس می تواند حالت هدف را تشخیص دهد.همچنین پدر این node و اکشن ای که انجام شده برای رسیدن به این node را ذخیره میکند.

این الگوریتم به این صورت عمل میکند ابتدا یک وضعیت اولیه به عنوان ورودی میگیرد.بررسی میکند که هدف می باشد یانه. اگر هدف نبود آن را به مجموعه پیشران frontier اضافه میکند.سپس به صورت FIFO از این لیست node ها را استخراج می کند و به ازای اقدامات مجاز بچه هارا تولید می کند که در صورت هدف بودن این بچه ها ، بچه هدف را به عنوان پاسخ خروجی میدهد در صورت هدف نبودن بچه ها به مجموعه پیشران اضافه می شوند.همچنین اگر قبلا وضعیت بچه ها در مجموعه پیشران موجود بود اقدامی انجام نمیشود.همچنین اگر همه node های داخل مجموع پیشران تمام شوند و پاسخ پیدا نشود "شکست" را به عنوان خروجی می دهد.

با این نوع پیاده سازی node ها سطح به سطح بررسی می شوند و جستجو اول سطح پیاده سازی می شود.همچنین این الگوریتم تعداد گره تولید شده و بسط شده را به عنوان خروجی می دهد.

گره های تولید شده برابر است با مجموع طول frontier و explored

گره های بسط داده شده برابر است با طول explored

الگوریتم ids :

برای پیاده سازی این الگوریتم نیز مانند bfs از کلاس playground استفاده شده است.

این الگوریتم به این صورت عمل می کند که از محدودیت تعیین شده توسط کاربر تا بینهایت الگوریتم dls را اجرا می کند . در صورتی که به جواب برسد جواب را باز میگرداند در صورت شکست نیز کار متوقف شده و اجرا پایان می پذیرد و شکست را به عنوان خروجی می دهد.

الگوریتم dls نیز به این صورت عمل میکند که تا عمق محدود شده به صورت اول عمق گره های فرزند را به اجزای اکشن های موجود تولید می کند و هدف بودن آن ها را نیز بررسی میکند.در صورتی که به عمق محدود شده برسد و نه شکست بخورد و نه به پاسخ برسد cutoff می شود و اعلام میکند که ممکن است با عمق بیشتر بتواند به پاسخ برسد.

به اجزای هر بچه ای که تولید می شود یکی به گره های تولید شده اضافه می شود همچنین به اجزای تولید بچه ای از گره والد ، گره والد به مجموعه بسط داده شده ها اضافه می شود.

الگوریتم *A :

برای این الگوریتم به کلاس playground یک متد اضافه شده است که هزینه مسیر به علاوه هیوریستیک حالت موجود را محاسبه و باز میگرداند.

این الگوریتم از لحاظ ساختار کد مشابه bfs می باشد با این تغییرات که هنگام انتخاب node از مجموعه پیشرانfrontier گره ای را انتخاب میکند که هیوریستیک به علاوه هزینه مسیر کمتری دارد.همچنین هنگامی که فرزند تولید می کند و وضعیت فرزند داخل مجموعه frontier موجود بود اگر هزینه فرزند جدید از گره ی داخل مجموعه کمتر بود با آن گره جابه جا می شود. مشابه bfs :

گره های تولید شده برابر است با مجموع طول frontier و explored

گره های بسط داده شده برابر است با طول explored

هیوریستیک استفاده شده در این الگوریتم به این صورت است که:

در هر زمین بازی اگر همه کارت ها همرنگ نباشند یک واحد به هیوریستیک اضافه می شود و اگر نزولی نباشد یک واحد دیگر به هیوریستیک اضافه می شوند.

این هیوریستیک قابل قبول است زیرا هزینه ای که تخمین می زند از حالت واقعی کمتر است.علت این است که این برای بدست اوردن این هیوریستیک از تکنیک بانک های اطلاعتی الگو استفاده شده است.

به این صورت که مسئله به دو زیر مسئله (نزولی بودن و همرنگ بودن) شکسته شده است.هزینه حل مسئله نزولی بودن و هزینه حل مسئله همرنگ بودن یک حد پایین برای حل مسئله اصلی که همرنگ بودن و نزولی بودن به صورت همزمان می باشد. در نتیجه این هیوریستیک قابل قبول می باشد.(هزینه حل زیر مسئله از مسئله بزگتر قطعا و به وضوح کمتر است.)

مقایسه الگوریتم ها:

ابتدا اجرای دو وضعیت اولیه توسط الگوریتم های فوق آورده شده است:

مسئله اول:

bfs:

```
PS C:\university\AI\project> & C:/Users/HP/AppData/Local/Programs/Python/Python39/python.exe c:/university/AI/project/bfs.py
5 2 4
5g 4g 3g 2r 1g
5r 4r 3r 2g 1r
#
#
final state is :

g-5 g-4 g-3 g-2
r-5 r-4 r-3
g-1
r-2 r-1

actions:

0 ---> 2
0 ---> 3
1 ---> 3
1 ---> 0

soloution depth:4
nodes created:38
nodes expanded:17
PS C:\university\AI\project>
```

Ids:

```
Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\university\AI\project> & C:/Users/HP/AppData/Local/Programs/Python/Python39/python.exe c:/university/AI/project/ids.py
5 2 4
5g 4g 3g 2r 1g
5r 4r 3r 2g 1r
#
#
initial limit:1
final state is :

g-5 g-4 g-3 g-2
r-5 r-4 r-3
g-1
r-2 r-1

actions:

0 ---> 2
0 ---> 3
1 ---> 3
1 ---> 0

soloution depth:4
nodes created:153
nodes expanded:36
PS C:\university\AI\project>
```

A*:

```
PS C:\university\AI\project> & C:/Users/HP/AppData/Local/Programs/Python/Python39/python.exe c:/university/AI/project/Astart.py
5 2 4
5g 4g 3g 2r 1g
5r 4r 3r 2g 1r
#
#
final state is :

g-5 g-4 g-3 g-2
r-5 r-4 r-3
g-1
r-2 r-1

actions:

0 ---> 2
0 ---> 3
1 ---> 3
1 ---> 0

soloution depth:4
nodes created:36
nodes expanded:16
PS C:\university\AI\project>
```

مسئله دوم:

Bfs:

```
PS C:\university\AI\project> & C:/Users/HP/AppData/Local/Programs/Python/Python39/python.exe c:/university/AI/project/bfs.py
5 2 4
5g 4g 3r 2g 1g
5r 4r 3g 2r 1r
#
#
final state is :

g-5 g-4 g-3
r-5 r-4
r-3 r-2 r-1
g-2 g-1

actions:

0 ---> 2
0 ---> 3
2 ---> 3
0 ---> 2
1 ---> 0
1 ---> 2
0 ---> 2
1 ---> 0

soloution depth:8
nodes created:339
nodes expanded:219
PS C:\university\AI\project>
```

Ids:

```
Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\university\AI\project> & C:/Users/HP/AppData/Local/Programs/Python/Python39/python.exe c:/university/AI/project/ids.py
5 2 4
5g 4g 3r 2g 1g
5r 4r 3g 2r 1r
#
#
initial limit:0
final state is :

g-5 g-4 g-3
r-5 r-4
r-3 r-2 r-1
g-2 g-1

actions:

0 ---> 2
0 ---> 3
2 ---> 3
0 ---> 2
1 ---> 0
1 ---> 2
0 ---> 2
1 ---> 0

soloution depth:8
nodes created:85588
nodes expanded:17937
PS C:\university\AI\project>
```

A*:

```
PS C:\university\AI\project> & C:/Users/HP/AppData/Local/Programs/Python/Python39/python.exe c:/university/AI/project/Astart.py
5 2 4
5g 4g 3r 2g 1g
5r 4r 3g 2r 1r
#
#
final state is :

g-5 g-4 g-3
r-5 r-4
r-3 r-2 r-1
g-2 g-1

actions:

0 ---> 2
0 ---> 3
2 ---> 3
0 ---> 2
1 ---> 0
1 ---> 2
0 ---> 2
1 ---> 0

soloution depth:8
nodes created:309
nodes expanded:192
PS C:\university\AI\project>
```

همانطور که در بالا مشخص هست برای مسائل مختلف تعداد گره تولید شده در ids از همه بیشتر و در A* از همه کمتر است.

علت این امر این است که در A* به علت هیوریستیک قابل قبول جستجو هوشمندانه تر و آگاهانه تر شده و سریع تر به هدف میل می کند.

و در ids به علت اجرای الگوریتم در عمق های غیر ضروری و cutoff شدن گره های تولیدی بسیار بیشتر و کند تر است.

همچنین مشاهده می شود که هر 3 به جواب بهینه میرسند علت این امر این است که bfs سطر به سطر بررسی می کند در نتیجه همواره به کوتاه ترین مسیر دست می یابد.A* نیز با توجه به هیوریستیک قابل قبول و همچنین بسط دادن گره ها با کمترین هزینه به مسیر بهینه می رسد.همچنین ids با توجه به این که از محدودیت سطر 0 شروع می کند با اینکه به صورت عمقی جستجو می کند اما از سطر محدود می شود و سطر آن یکی یکی افزایش می یابد در نتیجه به جواب بهینه میرسد.

از این پروژه نتیجه میگیریم که bfs پیاده سازی آسانی دارد ، با توجه به مسئله بهینه است اما نسبت به جستجو آگاهانه سرعت کمتری دارد.

A* سرعت هم گرایی بیشتری نسبت به بقیه دارد و بهینه است اما پیدا کردن هیوریستیک قابل قبول پیاده سازی را مقداری دشوار میکند.

Ids از هر دوخاصیت جستجو اول عمق و اول سطح استفاده میکند و فضای ذخیره سازی کمی لازم دارد اما در عین حال سرعت پایین تری دارد و بهینه است.