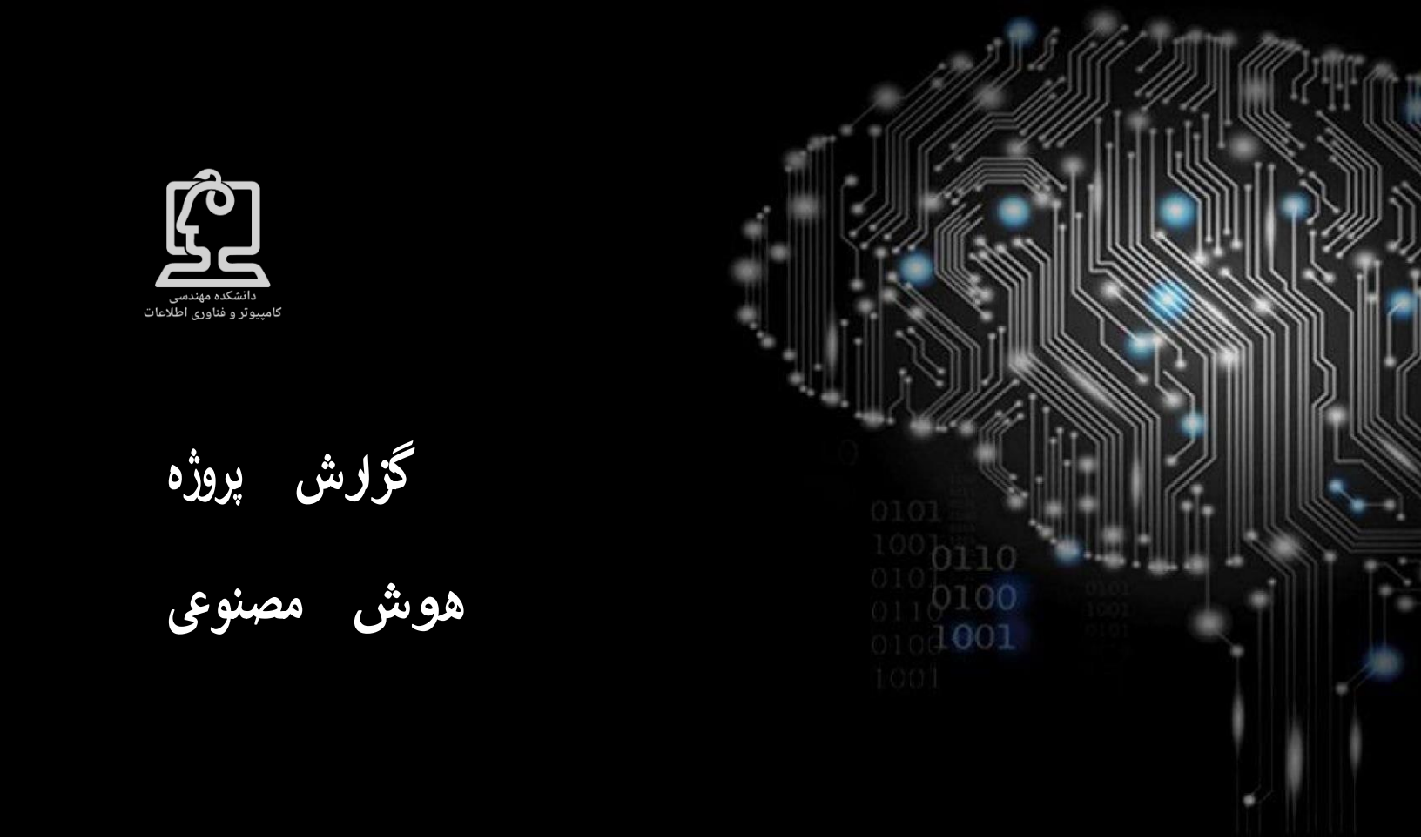

دانشکده مهندسی
کامپیوتر و فناوری اطلاعات

گزارش پروژه

هوش مصنوعی

محمدرضا قادری 9627057

پاییز 99

در حالت

مدلسازی :

برای حل این مسئله ما ورودی ها رو به صورت لیست از کاربر دریافت میکنیم و در یک ماتریس دو بعدی نگه داری میکنیم که هر سطر از این ماتریس یک ناحیه از بازی ما هست.

برای تولید هر نود پسین از حالت فعلی اون نود به تابع Child_Node فرستاده میشود که با توجه به الگوریتم مورد استفاده نوع ساخت فرزندان تغییر خواهد کرد.

با توجه به فرزندان تولید شده در هر action که اتفاق میافتد باید تابع Goal صدا زده میشود که تشخیص بدیم که به هدف رسیدیم یا نه .

(هدف پایانی به طرق مختلفی قابل بیان بود که به دلیل راحتی فقط در هر سطر رنگ ها یکی باشد و به صورت صعودی sort شده باشد کفایت شده (قابل تغییر به حالت های دیکر نیز هست)).

روش های حل این سوال به شرح زیر میباشد:

1. جستجوی اول سطح یا : BFS

در این مسئله از الگوریتم گرافی جستجوی bfs استفاده کردیم که در این حالت یک آرایه

برای نگه داری نودهای که فرزندان آنها تولید نشده نگه میداریم(آماده برای تولید فرزندان)

که frontier نام دارد و یک آرایه برای نگه داری نودهای که انها چک شده(که دوبار اونها رو

بسط ندهیم) که explored نام دارد. در این هدف رو موقع تولید نود چک میکنم تا بهینه تر

باشد. اگر درنهایت به پاسخی مناسب نرسیم حالت اولیه بر گردانده میشود(این زمانی اتفاق

میافتد که frontier خالی شود (که دیگر چیزی برای بسط نداریم((😂

نتایج حاصل از کد برای این الگوریتم :

(متاسفانه به دلیل زیاد بودن حالت ، مجبور شدم یه حالت ساده در نظر بگیرم و به یک

حالت ساده تر برسیم تا به جواب نزدیک تر باشد)

در این حالت من دوتا رنگ b , r رو انتخاب کردم که 4 زمین داشته باشم و نهایت هر رنگ 5 کارت داشته باشم

```
C:\Users\mohammad reza\Desktop>py bfs.py
Enter the col :4
Enter Num of color :2
Enter Num of each Color :5
5b 4b 3b 2r 1b
5r 4r 3r 2b 1r
#
#
Goal is : [['5b', '4b', '3b', '2b'], ['5r', '4r', '3r'], ['1b'], ['2r', '1r']]
Generated Nodes:   38
Expanded Nodes:   17
Solution Depth is :  4
```

2. جستجوی درختی اول عمق با افزایش تدریجی عمق :IDS

در این مساله ، هدف پیاده سازی الگوریتم dfs برای عمق های متفاوت است. یعنی در ابتدای

الگوریتم عمقی رو در نظر میگیریم و برای آن dfs رو پیاده سازی میکنیم. اگر در عمق مورد

نظر در یکی از شاخه ها به جواب رسیدیم الگوریتم عمق را باز میگرداند ، در غیر اینصورت

حداکثر عمق را افزایش میدهد و دوبار الگوریتم رو اجرا میکنیم.این روش باعث میشود تا ما

به جواب بهینه برسیم ،بدین صورت که در dfs اگر جواب در عمق بالاتری (نزدیک به ریشه)

بود، امکان داشت به حالت هدف دیگری برسیم در عمق پایینتر. ولی در این روش اطمینان داریم که جواب پیدا شده بهینه است، در غیر اینصورت در حالت قبل (عمق قبلی) به آن جواب دست میبافتیم. الگوریتم درختی است پس explored نداریم . ممکن است تکراری داشته باشیم.

نتایج حاصل از کد برای این الگوریتم:

اگر limit را در ابتدا صفر بگذاریم میتوان عمق جواب رو بدست آورد و تعداد نود های بسط داده شده

```
C:\Users\mohammad reza\Desktop>py ids.py
Enter the col :4
Enter Num of color :2
Enter Num of each Color :5
5b 4b 3b 2r 1b
5r 4r 3r 2b 1r
#
#
Goal is :[['5b', '4b', '3b', '2b'], ['5r', '4r', '3r'], ['1b'], ['2r', '1r']]
Generated Nodes:  193
Limit is :  4
```

حالا اگر عمق ابتدایی را افزایش دهیم تعداد نود های بسط داده شده کمتر خواهد شد

به ترتیب برای عمق های اولیه 1 و 2 و 3 و 4 و 85 داریم:

```
C:\Users\mohammad reza\Desktop>py ids.py
Enter the col :4
Enter Num of color :2
Enter Num of each Color :5
Enter First Limit :1
5b 4b 3b 2r 1b
5r 4r 3r 2b 1r
#
#
Goal is : [['5b', '4b', '3b', '2b'], ['5r', '4r', '3r'], ['1b'], ['2r', '1r']]
Generated Nodes:   166
Solution Depth:   4
```

```
C:\Users\mohammad reza\Desktop>py ids.py
Enter the col :4
Enter Num of color :2
Enter Num of each Color :5
Enter First Limit :2
5b 4b 3b 2r 1b
5r 4r 3r 2b 1r
#
#
Goal is : [['5b', '4b', '3b', '2b'], ['5r', '4r', '3r'], ['1b'], ['2r', '1r']]
Generated Nodes:   162
Solution Depth:   4
```

```
C:\Users\mohammad reza\Desktop>py ids.py
Enter the col :4
Enter Num of color :2
Enter Num of each Color :5
Enter First Limit :3
5b 4b 3b 2r 1b
5r 4r 3r 2b 1r
#
#
Goal is : [['5b', '4b', '3b', '2b'], ['5r', '4r', '3r'], ['1b'], ['2r', '1r']]
Generated Nodes:   142
Solution Depth:   4
```

```
C:\Users\mohammad reza\Desktop>py ids.py
Enter the col :4
Enter Num of color :2
Enter Num of each Color :5
Enter First Limit :4
5b 4b 3b 2r 1b
5r 4r 3r 2b 1r
#
#
Goal is : [['5b', '4b', '3b', '2b'], ['5r', '4r', '3r'], ['1b'], ['2r', '1r']]
Generated Nodes:   74
Solution Depth:   4
```

```
C:\Users\mohammad reza\Desktop>py ids.py
Enter the col :4
Enter Num of color :2
Enter Num of each Color :5
Enter First Limit :85
5b 4b 3b 2r 1b
5r 4r 3r 2b 1r
#
#
Goal is : [['5b', '4b', '3b', '2b'], ['5r', '4r', '3r'], ['1b'], ['2r', '1r']]
Generated Nodes:   74
Solution Depth:   4
```

بعد از عمقی که جواب رو توش پیدا میکنیم هر قدر هم غمق اولیه رو زیاد کنیم

تاثیری نخواهد داشت.

البته اگر جستجوی IDS ما گرافی بود نتایج از نظر نود های بسط داده شده بهتر میشد.

3.جستجوی A:*

هیوریستیک 1)

در هر ردیف اولین کارتی از سمت چپ که عدد n را داشته باشد در نظر میگیریم و در ادامه اگر باقی کارت های هم رنگ آن را به ترتیب در آن ردیف وجود داشته باشد اوکی ولی اگر موجود نباشد به اندازه کارتهایی که نیست از یک واحد کم میکنیم حداقل یک جابه جایی لازم است تا کارت را به جای خود برسانیم اگر در پایان رنگی را محاسبه نکرده باشیم در n ضرب میکنیم و به مقدار هیوریستیک آن اضافه میکنیم. (این حالت وقتی پیش می آیدکه بطور مثال عدد n برای رنگ b در ردیفی باشد که عدد n رنگ r هم باشد پس باید تک تک رنگ های r رو ببریم یه خط دیگه)

4 2 5

5b 4b 3b 2r 1b

5r 4r 3r 2b 1r

#

#

برای مثال در این مثال ردیف اول رو چک میکنیم اولین کارتی که مقدار n رو داره از سمت چپ کارت 5b هست پس این ردیف متعلق به رنگ b خواهد بود. حالا در این

ردیف 4b هست یا نه که هست حالا دنبال 3b و که اینم هست و 2b که این دیگه نیست پس Cost ما با 1 جمع میشه هزینه ای که باید بدیم تا کارت 2b بیاد سر جاش حداقل و بعد 1b که هست .

حالا برای ردیف بعد هم به همین ترتیب که در نهایت Cost این state میشود 2.

```
C:\Users\mohammad reza\Desktop>py astarall.py
Enter the col :4
Enter Num of color :2
Enter Num of each Color :5
5b 4b 3b 2r 1b
5r 4r 3r 2b 1r
#
#
Goal is :[[['5b', '4b', '3b'], ['5r', '4r', '2r'], ['3r', '1r'], ['2b', '1b']], 4]19
Generated Nodes:  31
Expanded Nodes:  13
```

هیوریستیک 2)

در این هیوریستیک از به کارت های موجود توی هر ردیف نگاه میکنیم ببینیم که ایا توی این ردیفی آیا sort آن صعودی است و اینکه رنگشون یکی هست یا نه اگر بود که هیچی اگر نه یکی به Hit اضافه میکنیم.

```
C:\Users\mohammad reza\Desktop>
C:\Users\mohammad reza\Desktop>py astar.py
Enter the col :4
Enter Num of color :2
Enter Num of each Color :5
5b 4b 3b 2r 1b
5r 4r 3r 2b 1r
#
#
Goal is :[[['5b', '4b', '3b', '2r', '1b'], ['5r', '4r', '3r', '2b'], [], ['1r']], 4]
Generated Nodes:  12
Expanded Nodes:  4
```

مقایسه معیار های مسئله :

تعداد گره های تولید شده در الگوریتم A* کمینه خواهد بود و بسته به نوع هیوریستیک تعریفی ما دارد و بعد الگوریتم BFS و در نهایت الگوریتم IDS هست.

تعداد گره های بسط داده شده همانند گره های تولید شده خواهند یعنی اول A* و بعد BFS و بعد IDS کمترین گره بسط شده را دارند

عمق جواب ها که در الگوریتم های BFS و IDS عمق بهینه باز میگردد ولی برای الگوریتم A* این گونه نخواهد بود و ما به مسئله عمق کاری نداریم و به هزینه میسر توجه میکنیم.