به نام خدا

پروژه اول درس هوش مصنوعی

جست و جو

امیرعلی رایگان

۸۱۰۱۹۷۶۲۳

در این پروژه با استفاده از پیاده سازی های مختلف برای جست و جو بین حالت های ممکن فضا میخواهیم به تفاوت پیچیدگی زمانی و پیچیدگی حافظه الگوریتم های مختلف جست و جو پی ببریم.

قبل از شرح الگوریتم ها ابتدا به نحوه ی پیاده سازی بازی و مدل فضا و حالت های آن می پردازیم.

ما برای هر کدام از حالت های جهان یک شئ (object) در نظر میگیریم. به طوری که مختصات اعضای بدن مار و مختصات دانه های درون صفحه را همزمان نگه می دارد. همچنین یک آرایه ی دیگر هم در کلاس state داریم که مسیر تا به این جای این حالت از جهان را نگه می دارد.

```
class state:
    def __init__(self, snakepos, frt, path):
        self.snakepos = snakepos
        self.fruit = frt
        self.path = path
```

کلاس state

پس در این کلاس یک لیست یک بعدی برای مسیر طی شده تا کنون یک لیست دو بعدی برای مختصات مار و یک لیست سه بعدی برای مختصات و امتیاز هر دانه وجود دارد.

بعد از خواندن ورودی ها از فایل با توجه به محتوای فایل یک حالت ابتدایی درست میکنیم و آن را در لیست اصلی برنامه که mainQueue نام دارد اضافه میکنیم. از این به بعد در اجرای هر کدام از الگوریتم ها در کم و اضافه کردن به این لیست راه های مختلفی را پیش میگیریم. همچنین یک set با نام seen داریم که برای loop checking از آن استفاده میکنیم. ویژگی چشم گیری که ما را به استفاده از این data structure مجاب کرد این بود که جست و جوی یک عضو در آن با پیچیدگی زمانی O(1) انجام می شود. هر بار که میخواهیم به یک حالت جدید در جهان برویم ابتدا چک میکنیم که آیا این حالت را در seen داریم یا نه. اگر نداشتیم حالت جدید را به mainQueue و seen اضافه میکنیم. یک نکته ی دیگر هم اینکه برای ذخیره کردن حالت ها در seen کلاس state را در آن اضافه نمیکنیم بلکه با تابعی به نام sett یک کد منحصر به فرد برای هر حالت جهان تولید میکنیم و آن کد را که یک رشته ی چند حرفی است به seen اضافه میکنیم.

```
    def sett(self):
        a = ''
        for i in self.snakepos:
            a += str(i[0])
            a += str(i[1])
        a += '+'
        for i in self.fruit:
            a += str(i[0])
            a += str(i[1])
            a += str(i[2])
```

تابع sett

مورد دیگری که در هر سه الگوریتم یکسان است نحوه ی محاسبه ی تعداد کل حالت ها و همچنین مدت زمان محاسبات است. برای محاسبه ی مدت جست و جو زمان را قبل و بعد از حلقه ی اصلی محاسبه میکنیم و در انتها اختلافش را نمایش می دهیم. برای تعداد کل حالت ها هم هر بار که یک حالت را از لیست اصلی بیرون میکشیم یکی به متغیر مربوطه اضافه میکنیم. همچنین در ابتدای هر حلقه هر بار که یک حالت جهان را از لیست بیرون میکشیم چک میکنیم که آیا این حالت حالت هدف است یا خیر اگر نبود به کارمان ادامه می دهیم. اولین خانه ای که مجموع امتیازات میوه های آن 0 باشد خانه ی هدف ماست.

حال به توضیح هر کدام از الگوریتم ها میرسیم.

الگوریتم BFS :

در این روش جست و جوی نا آگاهانه در هر بار پیمایش حلقه اصلی برنامه محتوای خانه ی اول لیست اصلی بیرون کشیده می شود و تمام فرزندان آن حالت (تمام حالت هایی که میتوان از آن حالت به آنها رفت) به *آخر این لیست* اضافه می شود. اینگونه جست و جوی ما سطحی انجام می شود یعنی ابتدا هر سطح را به صورت کامل و سپس فرزندان آن را برای رسیدن به هدف جست و جو میکنیم. اشکال بزرگ این روش اشغال کردن مقدار زیادی از حافظه می باشد. به عنوان مثال اگر هدف در فاصله ی 4 خانه باشد باید تمام خانه هایی که با 3 حرکت می توان طی کرد را جست سپس به خانه های 4 مرحله ای رسید. اینکه تمام خانه های 3 مرحله ای را میگردیم موجب زیاد شدن حافظه ی اشغالی می شود.

حال نگاهی به خروجی های این الگوریتم می اندازیم.

تست اول :

```
C:\Users\ASUS\anaconda3\python.exe C:/Users/ASUS/PycharmProjects/ca1/BFS.py
Result of BFS algorithm for TEST1
Time duration: 0.2 s
Number of visited States = 6109
Totall number of different States = 4489
Succesfull path distance = 12
Succesfull path is = ['L', 'D', 'L', 'D', 'R', 'U', 'L', 'U', 'U', 'L', 'U', 'L']

Process finished with exit code 0
```

```
C:\Users\ASUS\anaconda3\python.exe C:/Users/ASUS/PycharmProjects/ca1/BFS.py
Result of BFS algorithm for TEST1
Time duration: 0.21 s
Number of visited States = 6109
Totall number of different States = 4489
Succesfull path distance = 12
Succesfull path is = ['L', 'D', 'L', 'D', 'R', 'U', 'L', 'U', 'U', 'L', 'U', 'L']

Process finished with exit code 0
```

تست دوم :

```
C:\Users\ASUS\anaconda3\python.exe C:/Users/ASUS/PycharmProjects/ca1/BFS.py
Result of BFS algorithm for TEST2
Time duration: 2.96 s
Number of visited States = 62469
Totall number of different States = 48544
Succesfull path distance = 15
Succesfull path is = ['R', 'U', 'L', 'L', 'D', 'L', 'U', 'U', 'U', 'U', 'L', 'L', 'L', 'L', 'U']

Process finished with exit code 0
```

```
C:\Users\ASUS\anaconda3\python.exe C:/Users/ASUS/PycharmProjects/ca1/BFS.py
Result of BFS algorithm for TEST2
Time duration: 3.36 s
Number of visited States = 62469
Totall number of different States = 48544
Succesfull path distance = 15
Succesfull path is = ['R', 'U', 'L', 'L', 'D', 'L', 'U', 'U', 'U', 'U', 'L', 'L', 'L', 'L', 'U']

Process finished with exit code 0
```

تست سوم :

```
C:\Users\ASUS\anaconda3\python.exe C:/Users/ASUS/PycharmProjects/ca1/BFS.py
Result of BFS algorithm for TEST3
Time duration: 13.08 s
Number of visited States = 274454
Totall number of different States = 217521
Succesfull path distance = 25
Succesfull path is = ['R', 'U', 'R', 'D', 'D', 'D', 'R', 'R', 'D', 'R', 'R', 'R', 'D', 'D', 'R', 'R', 'U', 'L', 'L', 'D', 'L', 'L', 'L', 'U', 'U']

Process finished with exit code 0
```

```
C:\Users\ASUS\anaconda3\python.exe C:/Users/ASUS/PycharmProjects/ca1/BFS.py
Result of BFS algorithm for TEST3
Time duration: 14.98 s
Number of visited States = 274454
Totall number of different States = 217521
Succesfull path distance = 25
Succesfull path is = ['R', 'U', 'R', 'D', 'D', 'D', 'R', 'R', 'D', 'R', 'R', 'R', 'D', 'D', 'R', 'R', 'U', 'L', 'L', 'D', 'L', 'L', 'L', 'U', 'U']

Process finished with exit code 0
```

الگوریتم IDS :

این الگوریتم هم یک الگوریتم جست و جوی نا آگاهانه است به این معنا که برای حالت های جدید جهان ارزش متفاوت قائل نمیشود و صرفا ترتیب آنها برایش مهم است. در این الگوریتم به طوری مزایای BFS و DFS ترکیب شده است و اشکالات آنها حذف شده.(فضای زیاد BFS و نامتناهی بودن یک شاخه در DFS) به این صورت عمل میکند که هر بار تا یک عمق مشخصی عملیات DFS به صورت کامل انجام می شود و اگر تا آن عمق هدف پیدا نشد مرتبه ی بعد یکی به عمق جست و جو اضافه میشود. برای پیاده سازی این روش یک متغیر عمق (depth) به کلاس state اضافه شد تا فاصله ی هر حالت از حالت ابتدایی نگه داری شود. زمان سپری شده در این روش به کوتاهی BFS نیست. (چون هر بار باید از اول شروع به پیمایش کند)

حال نگاهی به خروجی های این روش بیندازیم:

تست اول:

```
C:\Users\ASUS\anaconda3\python.exe C:/Users/ASUS/PycharmProjects/ca1/IDS.py
Result of IDS algorithm for TEST1
Time duration: 0.42 s
Number of visited States = 1412
Totall number of different States = 10266
Succesfull path distance = 17
Succesfull path is = ['U', 'U', 'R', 'U', 'U', 'L', 'L', 'L', 'U', 'U', 'U', 'L', 'U', 'U', 'R', 'R']

Process finished with exit code 0
```

```
Result of IDS algorithm for TEST1
Time duration: 0.45 s
Number of visited States = 1412
Totall number of different States = 10266
Succesfull path distance = 17
Succesfull path is = ['U', 'U', 'R', 'U', 'U', 'L', 'L', 'L', 'U', 'U', 'U', 'L', 'U', 'U', 'R', 'R']

Process finished with exit code 0
```

تست دوم:

```
C:\Users\ASUS\anaconda3\python.exe C:/Users/ASUS/PycharmProjects/ca1/IDS.py
Result of IDS algorithm for TEST2
Time duration: 6.71 s
Number of visited States = 16779
Totall number of different States = 142222
Succesfull path distance = 25
Succesfull path is = ['R', 'U', 'U', 'U', 'U', 'U', 'U', 'U', 'U', 'U', 'L', 'U', 'L', 'U', 'R', 'U', 'U', 'U', 'L', 'L', 'U', 'L', 'L', 'L', 'L']

Process finished with exit code 0
```

```
C:\Users\ASUS\anaconda3\python.exe C:/Users/ASUS/PycharmProjects/ca1/IDS.py
Result of IDS algorithm for TEST2
Time duration: 7.31 s
Number of visited States = 16779
Totall number of different States = 142222
Succesfull path distance = 25
Succesfull path is = ['R', 'U', 'U', 'U', 'U', 'U', 'U', 'U', 'U', 'U', 'L', 'U', 'L', 'U', 'R', 'U', 'U', 'U', 'L', 'L', 'U', 'L', 'L', 'L', 'L']

Process finished with exit code 0
```

تست سوم:

```
C:\Users\ASUS\anaconda3\python.exe C:/Users/ASUS/PycharmProjects/ca1/IDS.py
Result of IDS algorithm for TEST3
Time duration: 18.92 s
Number of visited States = 52374
Totall number of different States = 478110
Succesfull path distance = 57
Succesfull path is = [ U ,U ,U ,U ,U ,U ,U ,R ,R ,U ,U ,U ,U ,U ,U ,U ,L ,D ,
D ,D ,D ,D ,D ,D ,D ,D ,D ,D ,L ,U ,L ,D ,L ,L ,U ,U ,L ,U ,L ,D ,L ,U ,U ,R ,D ,D ,L ,D ,D ,D ,R ,U ,R ,R ]

Process finished with exit code 0
```

```
C:\Users\ASUS\anaconda3\python.exe C:/Users/ASUS/PycharmProjects/ca1/IDS.py
Result of IDS algorithm for TEST3
Time duration: 22.96 s
Number of visited States = 52374
Totall number of different States = 478110
Succesfull path distance = 57
Succesfull path is = [ U ,U ,U ,U ,U ,U ,U ,R ,R ,U ,U ,U ,U ,U ,U ,U ,L ,D ,
D ,D ,D ,D ,D ,D ,D ,D ,D ,D ,L ,U ,L ,D ,L ,L ,U ,U ,L ,U ,L ,D ,L ,U ,U ,R ,D ,D ,L ,D ,D ,D ,R ,U ,R ,R ]

Process finished with exit code 0
```

الگوریتم A* :

این الگوریتم یک الگوریتم آگاهانه است. به این معنی که ترکیبی از انتخاب حریصانه و uniform cost می باشد. اینگونه کار می کند که یک تابع heuristic داریم که فاصله ی هر حالت را تا حالت هدف تخمین می زند. این تابع باید admissible باشد بدین معنی که در تمام حالت ها حدس کوچکتر یا مساوی واقعیت باشد. با این ترتیب ما همه ی حالت ها را نمی بینیم و فقط حالت هایی که ما را به هدف نزدیک تر میکند را جست و جو میکنیم. هر بار که حالت های جدیدی پیش می آید با توجه به مجموع h(x) و g(x) آن را در جای درست لیست اصلی قرار می دهیم تا در زمان درست به آن حالت دسترسی یابیم. در نتیجه به طور قابل توجهی سریع تر عمل میکنیم و همچنین فضای حافظه ی خیلی کمتری هم استفاده میکنیم.

اولین تابع heuristic که اینجا استفاده کردیم مجموع فواصل سر مار تا امتیاز ها بود. به این صورت که برای هر حالت جهان مجموع فواصل سر مار تا همه ی دانه ها را به صورت وزن دار محاسبه می کنیم یعنی اگر امتیاز دانه ای 2 بود فاصله تا آن دانه را دو برابر می کنیم. گفتنی است که این تابع حتما admissible هست چون ممکن است مار فاصله ی دورتری را بپیماید تا به هر دانه برسد.

نتایج استفاده از این تابع را میتوان به صورت زیر مشاهده کرد:

تست اول:

```
C:\Users\ASUS\anaconda3\python.exe C:/Users/ASUS/PycharmProjects/ca1/A-star.py
Result of A* algorithm for TEST1
Time duration: 0.0 s
Number of visited States = 117
Totall number of different States = 45
Succesfull path distance = 14
Succesfull path is = ['D', 'L', 'D', 'D', 'L', 'D', 'L', 'U', 'L', 'U', 'U', 'R', 'R', 'R']

Process finished with exit code 0
```

```
C:\Users\ASUS\anaconda3\python.exe C:/Users/ASUS/PycharmProjects/ca1/A-star.py
Result of A* algorithm for TEST1
Time duration: 0.0 s
Number of visited States = 117
Totall number of different States = 45
Succesfull path distance = 14
Succesfull path is = ['D', 'L', 'D', 'D', 'L', 'D', 'L', 'U', 'L', 'U', 'U', 'R', 'R', 'R']

Process finished with exit code 0
```

تست دوم :

```
C:\Users\ASUS\anaconda3\python.exe C:/Users/ASUS/PycharmProjects/ca1/A-star.py
Result of A* algorithm for TEST2
Time duration: 0.06 s
Number of visited States = 740
Totall number of different States = 428
Succesfull path distance = 25
Succesfull path is = ['U', 'L', 'D', 'L', 'D', 'D', 'D', 'D', 'D', 'D', 'L', 'U', 'L', 'L', 'L', 'U', 'U', 'U', 'U', 'U', 'L', 'L', 'L', 'L', 'L']

Process finished with exit code 0
```

```
C:\Users\ASUS\anaconda3\python.exe C:/Users/ASUS/PycharmProjects/ca1/A-star.py
Result of A* algorithm for TEST2
Time duration: 0.07 s
Number of visited States = 740
Totall number of different States = 428
Succesfull path distance = 25
Succesfull path is = ['U', 'L', 'D', 'L', 'D', 'D', 'D', 'D', 'D', 'D', 'L', 'U', 'L', 'L', 'L', 'U', 'U', 'U', 'U', 'U', 'L', 'L', 'L', 'L', 'L']

Process finished with exit code 0
```

تست سوم :

```
C:\Users\ASUS\anaconda3\python.exe C:/Users/ASUS/PycharmProjects/ca1/A-star.py
Result of A* algorithm for TEST3
Time duration: 0.31 s
Number of visited States = 1894
Totall number of different States = 953
Succesfull path distance = 36
Succesfull path is = [ D ,D ,R ,D ,R ,R ,R ,D ,R ,R ,R ,D ,L ,U ,R ,D ,L ,U ,L ,U ,
L ,L ,U ,L ,L ,D ,D ,L ,L ,D ,R ,D ,D ,D ,D ,R ]

Process finished with exit code 0
```

```
C:\Users\ASUS\anaconda3\python.exe C:/Users/ASUS/PycharmProjects/ca1/A-star.py
Result of A* algorithm for TEST3
Time duration: 0.42 s
Number of visited States = 1894
Totall number of different States = 953
Succesfull path distance = 36
Succesfull path is = [ D ,D ,R ,D ,R ,R ,R ,D ,R ,R ,R ,D ,L ,U ,R ,D ,L ,U ,L ,U ,
L ,L ,U ,L ,L ,D ,D ,L ,L ,D ,R ,D ,D ,D ,D ,R ]

Process finished with exit code 0
```

تابع heuristic بعدی که در اینجا استفاده کردیم تعداد دانه ها بود. در هر حالت جهان تعداد دانه های موجود در آن لحظه را می شماریم و به عنوان حدس بر می گردانیم. این تابع علاوه بر اینکه admissible می باشد جواب ایده آل یا optimal را به ما می دهد یعنی consistent نیز هست. گفتنی است با اینکه در زمان بیشتری به جواب می رسیم اما جواب بهینه را به ما میدهد.

حال ببینیم نتایج آن چگونه است:

تست اول :

```
C:\Users\ASUS\anaconda3\python.exe C:/Users/ASUS/PycharmProjects/ca1/A-star.py
Result of A* algorithm for TEST1
Time duration: 1.84 s
Number of visited States = 3477
Totall number of different States = 1952
Succesfull path distance = 12
Succesfull path is = [ D ,L ,U ,U ,L ,U ,L ,L ,L ,U ,L ,U ]

Process finished with exit code 0
```

```
C:\Users\ASUS\anaconda3\python.exe C:/Users/ASUS/PycharmProjects/ca1/A-star.py
Result of A* algorithm for TEST1
Time duration: 2.62 s
Number of visited States = 3477
Totall number of different States = 1952
Succesfull path distance = 12
Succesfull path is = [ D ,L ,U ,U ,L ,U ,L ,L ,L ,U ,L ,U ]

Process finished with exit code 0
```

تست دوم :

```
C:\Users\ASUS\anaconda3\python.exe C:/Users/ASUS/PycharmProjects/ca1/A-star.py
Result of A* algorithm for TEST2
Time duration: 143.68 s
Number of visited States = 34706
Totall number of different States = 23505
Succesfull path distance = 15
Succesfull path is = [ R ,L ,L ,U ,R ,U ,U ,U ,L ,L ,U ,L ,L ,L ,L ]

Process finished with exit code 0
```

```
C:\Users\ASUS\anaconda3\python.exe C:/Users/ASUS/PycharmProjects/ca1/A-star.py
Result of A* algorithm for TEST2
Time duration: 141.97 s
Number of visited States = 34706
Totall number of different States = 23505
Succesfull path distance = 15
Succesfull path is = [ R ,L ,L ,U ,R ,U ,U ,U ,L ,L ,U ,L ,L ,L ,L ]

Process finished with exit code 0
```

تست سوم:

پیچیدگی زمانی بسیار بسیار زیاد!

استفاده از weighted A* :

در درس دیدیم که اگر تابع h را در یک الفا ضرب کنیم میبینیم که خیلی سریعتر به جواب می رسیم. این بدین خاطر است که حالت ها سریعتر تفکیک می شوند و عملا حالت های کمتری را میبینیم تا به جواب برسیم. در این روش با وجود اینکه خیلی سریعتر به هدف می رسیم اما جواب بهینه به ما نمیدهد. برای این منظور در هر تست کیس ابتدا یک بار با آلفا برابر 2 و بار دیگر با آلفا برابر 5 برنامه را تکرار میکنیم. با توجه به نتایجی که زیر آمده میبینیم که در هر مرحله تفاوت زمانی چشمگیری رقم میخورد. هر چه آلفا بیشتر تعداد استیت های دیده شده کمتر اما دقت جواب پایین تر.

تست اول:

الفا = 2

```
C:\Users\ASUS\anaconda3\python.exe C:/Users/ASUS/PycharmProjects/ca1/A-star.py
Result of Weighted A* algorithm for TEST1, alpha = 2
Time duration: 0.55 s
Number of visited States = 1123
Totall number of different States = 452
Succesfull path distance = 12
Succesfull path is = [ D ,L ,R ,R ,D ,D ,R ,D ,R ,D ,R ,D ]

Process finished with exit code 0
```

الفا = 5

```
C:\Users\ASUS\anaconda3\python.exe C:/Users/ASUS/PycharmProjects/ca1/A-star.py
Result of Weighted A* algorithm for TEST1, alpha = 5
Time duration: 0.01 s
Number of visited States = 187
Totall number of different States = 71
Succesfull path distance = 12
Succesfull path is = [ D ,L ,U ,U ,L ,L ,U ,L ,L ,U ,L ,U ]

Process finished with exit code 0
```

تست دوم:

الفا = 2

```
C:\Users\ASUS\anaconda3\python.exe C:/Users/ASUS/PycharmProjects/ca1/A-star.py
Result of Weighted A* algorithm for TEST2, alpha = 2
Time duration: 102.42 s
Number of visited States = 17722
Totall number of different States = 10608
Succesfull path distance = 15
Succesfull path is = [ U ,R ,D ,L ,L ,U ,U ,U ,L ,U ,U ,L ,L ,L ,L ]

Process finished with exit code 0
```

الفا = 5

```
C:\Users\ASUS\anaconda3\python.exe C:/Users/ASUS/PycharmProjects/ca1/A-star.py
Result of Weighted A* algorithm for TEST2, alpha = 5
Time duration: 1.61 s
Number of visited States = 2370
Totall number of different States = 1078
Succesfull path distance = 17
Succesfull path is = [ U ,L ,D ,R ,R ,U ,L ,L ,L ,U ,U ,U ,U ,L ,L ,L ,L ]

Process finished with exit code 0
```

تست سوم:

الفا = 2

```
Result of Weighted A* algorithm for TEST3, alpha = 2
Time duration: 499.64 s
Number of visited States = 56919
Totall number of different States = 35267
Succesfull path distance = 25
Succesfull path is = [ U ,R ,D ,D ,R ,D ,D ,R ,R ,R ,D ,R ,D ,R ,R ,U ,L ,L ,D ,
L ,L ,U ,L ,U ]
```

الفا = 5

```
C:\Users\ASUS\anaconda3\python.exe C:/Users/ASUS/PycharmProjects/ca1/A-star.py
Result of Weighted A* algorithm for TEST3, alpha = 5
Time duration: 0.16 s
Number of visited States = 1141
Totall number of different States = 436
Succesfull path distance = 26
Succesfull path is = [ U ,R ,D ,D ,R ,D ,D ,R ,R ,R ,U ,L ,D ,D ,R ,D ,R ,R ,U ,R ,
R ,D ,L ,D ,L ,U ]

Process finished with exit code 0
```

حال اگر بخواهیم تمام داده ها را در جدول بیاوریم داریم:

تست اول

	فاصله جواب	تعداد استیت دیده شده	تعداد استیت مجزا	زمان اجرا
BFS	12	6109	4489	**0.2 s**
IDS	17	10266	1412	**0.44 s**
A* - h1	14	117	45	**0.0 s >**
A* - h2	12	3477	1952	**1.84 s**
Weighted A* (2)	12	1123	452	**0.55 s**
Weighted A* (5)	12	187	71	**0.01 s**

تست دوم

	فاصله جواب	تعداد استیت دیده شده	تعداد استیت مجزا	زمان اجرا
BFS	15	62469	48544	**3.15 s**
IDS	25	142222	16179	**7.01 s**
A* - h1	25	740	428	**0.65 s**
A* - h2	15	34706	23505	**142.5 s**
Weighted A* (2)	15	17722	10608	**102 s**
Weighted A* (5)	17	2370	1078	**1.61 s**

تست سوم

	فاصله جواب	تعداد استیت دیده شده	تعداد استیت مجزا	زمان اجرا
BFS	25	274454	217521	**14.40 s**
IDS	57	478110	52374	**20.08 s**
A* - h1	36	1894	953	**0.37 s**
A* - h2	--	---	---	--
Weighted A* (2)	25	56919	35267	**499 s**
Weighted A* (5)	26	1141	436	**0.16 s**