به نام خدا

دانشگاه تهران

پردیس دانشکده‌های فنی

دانشکده برق و کامپیوتر

**هوش مصنوعی پاییز 98**

**پروژه یک**

**پکمن بدون روح**

**نام و نام خانوادگی**

**علیرضا زارع نژاد اشکذری**

**شماره دانشجویی**

**810196474**

# شرح مختصر پروژه

هدف از این پروژه آشنایی با روش‌های جست‌و‌جو BFS, DFS, IDS , \*A می‌باشد.

پنج فایل test.txt به ما داده شده است که هر کدام نقشه محیط جست‌و‌جو را نشان می دهد.

درهر یک از شکل ها هر یک از ٪ نشانه ی دیوار و هر کدام از شماره های ۱و۲و۳ نشانه ی غذایی خاص است که عامل های هوشمند P, Q آن‌ها را می خورند. غذای ۱ برای p , و غذای 2 برای q و ۳ مختص هر دو می باشد. هدف نهایی خوردن تمام غذا ها می باشد. همچنین در هر حرکت فقط یکی از عامل می تواندد حرکت کنند و نمی‌توانند از روی هم رد شوند.

# نحوه ی مدل کردن به فضای جست و جو

ابتدا شروع به خواندن از روی فایل test‌ می‌کنیم. در اینجا یک کلاس Node ‌تعریف می‌کنیم که اطلاعات خاصی را در خود نگه می دارد.

مثلاً تعداد غذاها, مختصات عامل های هوشمند و مختصات غذاها و هم چنین یک آرایه دو بعدی به نام map برای چاپ کردن استیت تعریف کردیم.

واضح است که استیت اولیه همان ورودی اولیه است که شامل تعدادی غذا در مختصاتی خاص و همچنین دو عامل هوشمند است.

در اینجا استیت نهایی را استیتی در نظر می‌گیریم که تعداد غذاهایش صفر باشد. Action‌ ها را نیز برای رفتن از یک استیت به استیت دیگر به صورت زیر تعریف می‌کنیم که در هر حرکت عامل های هوشمند p , q به ترتیب می توانند به راست و بالا و چپ و پایین بروند. پس ماکسیمم branch factor برابر هشت می باشد.

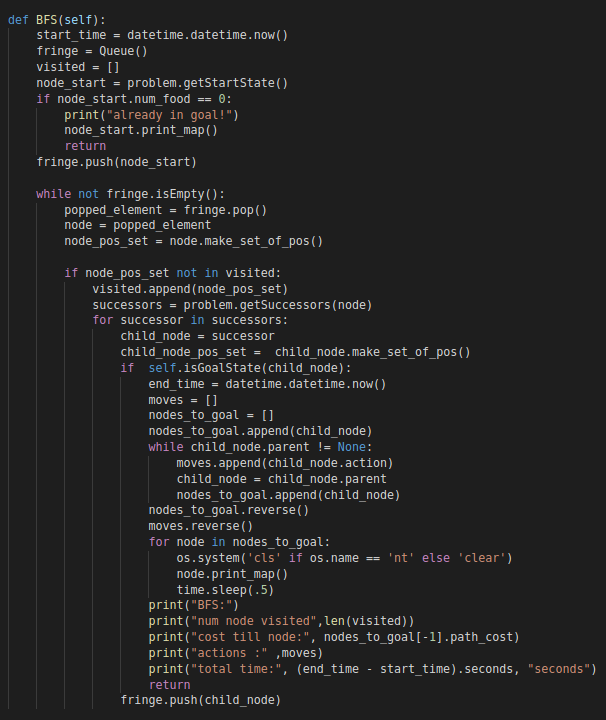
در اینجا با توجه به نوع حرکت انجام شده باید یک سری تغییرات اعمال شود که مثلاً اگه به خانه‌ای که می‌رویم غذا در آن باشد باید آن غذا خورده شود و مختصات آن remove‌ و تعداد غذاها یکی کم شود یا آنکه مختصات عامل های هوشمند آپدیت شود.

در اینجا ابتدا الگوریتم ها توضیح داده می‌شوند و خروجی آن‌ها مقایسه می‌شوند که عمق جواب و تعداد node‌های ویزیت شده و زمان را بررسی می کنیم.

# الگوریتم ‌BFS

در این الگوریتم که به جست و جوی سطح معروف است ابتدا node شروع را بسط می دهیم و همه ی بچه های آن را در یک صف قرار می دهیم. هر بار یک node از صف خارج می کنیم و در صورتی که قبلا ویزیت نشده بود آن را گسترش می دهیم و همین روال را تکرار می کنیم. دقت می کنیم که نود ها سطح به سطح بررسی می شوند و در صورتی که یکی از نود ها در هر سطح goal باشد یعنی تعداد غذاهایش صفر باشد الگوریتم خاتمه می یابد. توجه می کنیم که چون branch factor‌ در اینجا حداکثر 8 است پس اگر عمق جواب n باشد حداکثر n^8 نود دیده خواهد شد. پیاده سازی این الگوریتم در زیر آورده شده است.

این الگوریتم complete و optimal و هزینه ی زمانی آن o(b^d) , و همچنین حافظه ی آن o(b^d) ‌می باشد.

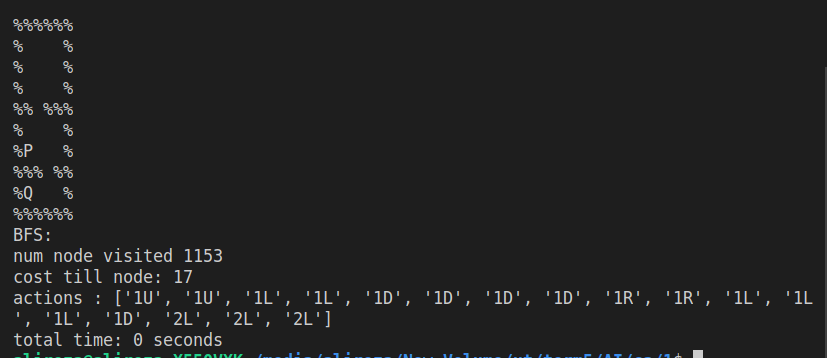


حال خروجی هر کدام از تست ها را برای این الگوریتم بررسی می کنیم.

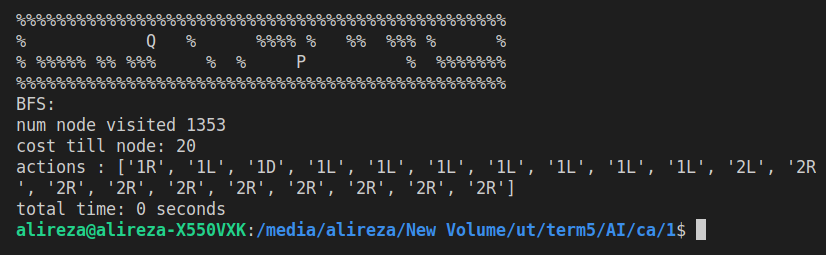


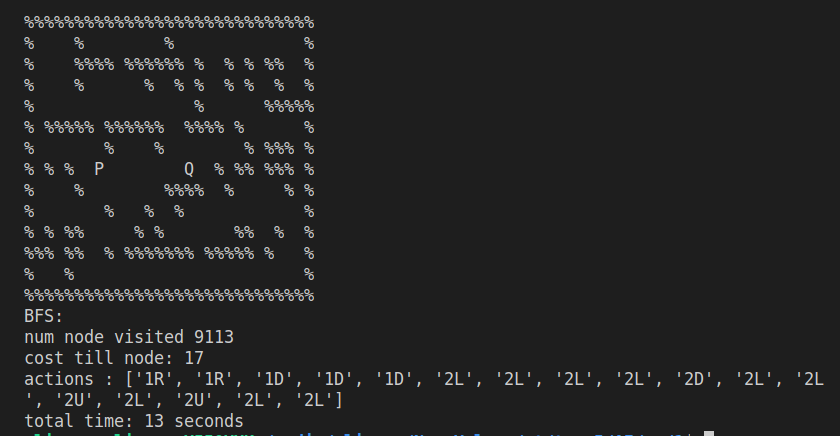
**تست اول‌:**

**تست دوم:**

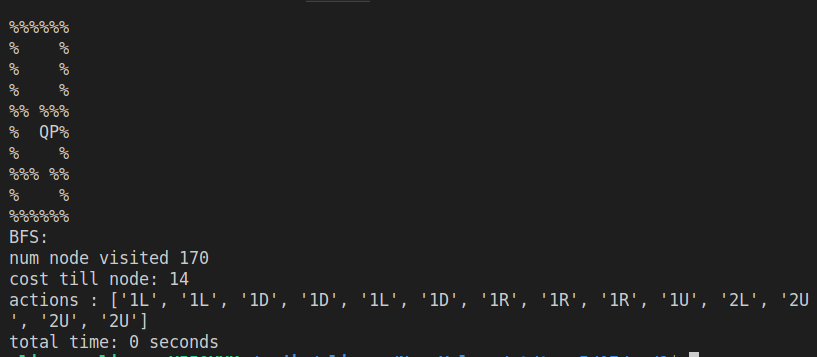


**تست سوم‌ :**

****

**تست چهارم:**

**تست پنجم‌:**



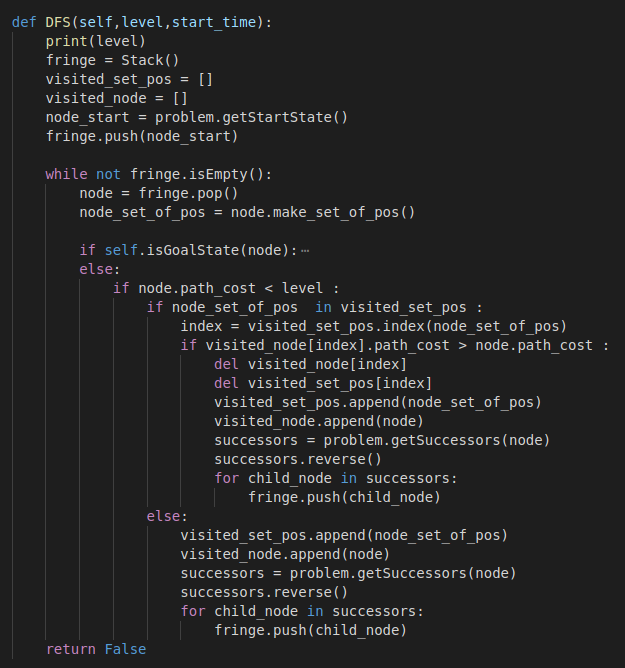
# الگوریتم IDS

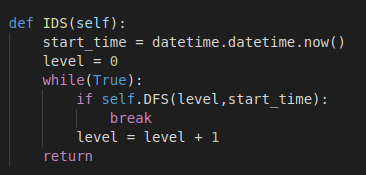
مشابه الگوریتم DFS پیش می رویم یعنی جست جو در عمق بدین ترتیب که از stack به جای صف استفاده می کنیم که هر بار یکی pop‌ می کنیم و در صورتی که قبلا ویزیت نشده بود آن را گسترش می دهیم و به استک push‌ می کنیم. همچنین بر خلاف bfs‌ تشخیص goal‌ بودن هنگام بسط دادن مشخص خواهد شد.

حال به ازای level های مختلف از یک تا جایی که هدف پیدا شود dfs می زنیم. کد الگوریتم در زیر آورده شده است.

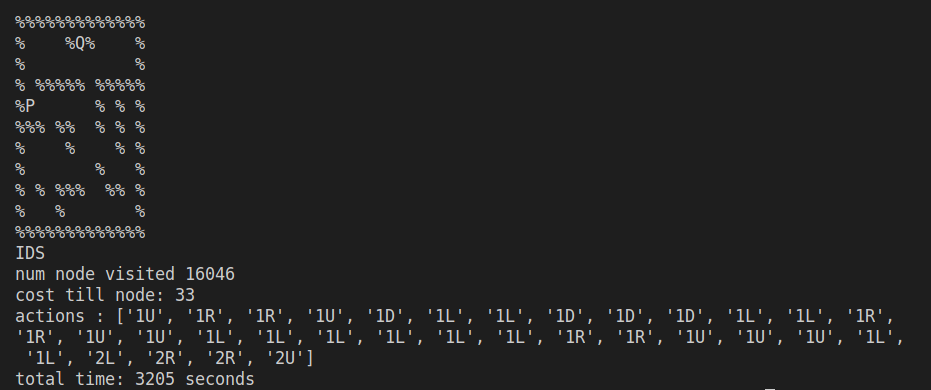
که باید گفت الگوریتم complete‌ و optimal و هزینه زمانی برایر o(b^d) می‌باشد و حافظه o(bd) می باشد.

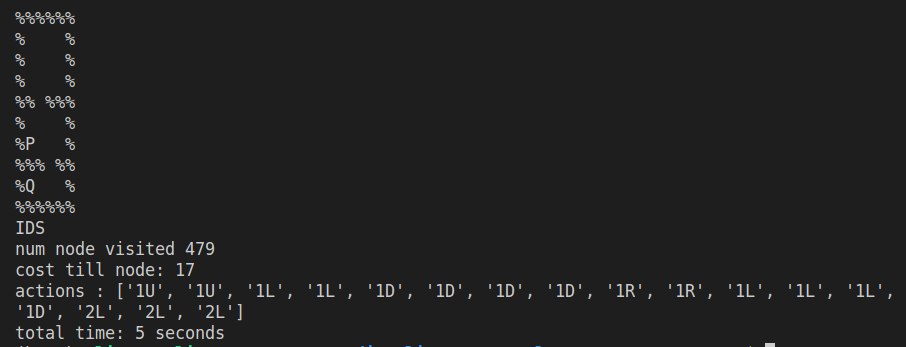
دقت می‌کنیم که IDS بر خلاف DFS باید جواب اپتیمال بدهد. در اینجا بررسی کردیم که اگر یک node قبلاً ویزیت شده باشد و حال اگر path cost آن بیشتر از path cost جاری باشد باید این نود را ویزیت کرده. توجه می‌کنیم زمان این الگوریتم نسبت به سایر الگوریتم ها خیلی بیشتر طول می‌کشد که در خروجی تست ها قابل درک می باشد.



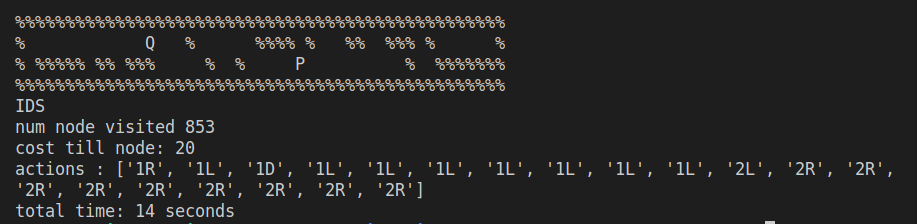


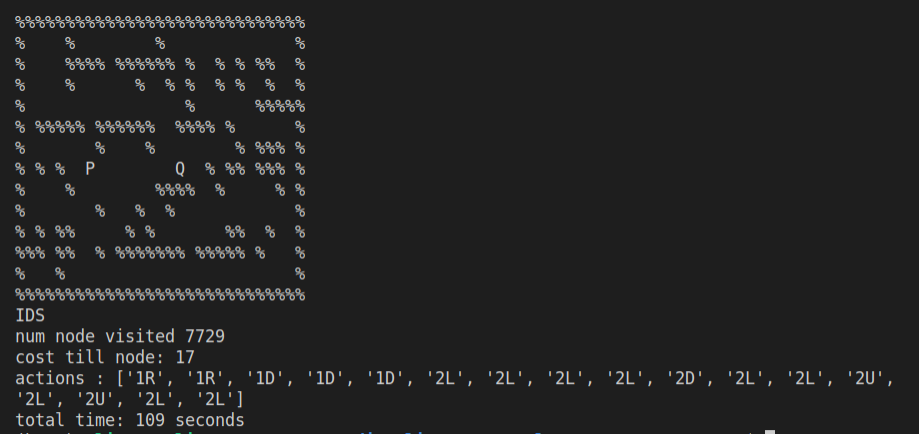
خروجی هر کدام از تست ها به ترتیب آورده شده است.

**تست اول‌:**

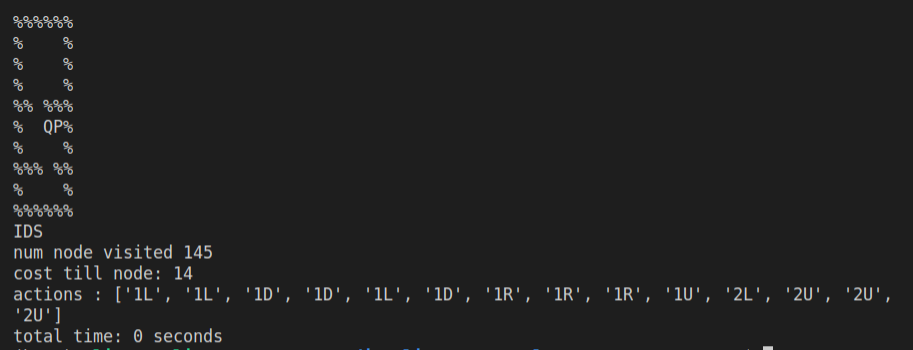
**تست دوم‌:**

**تست سوم‌:**

****

**تست چهارم:**

**تست پنجم‌:**

****

# **الگوریتم A\***

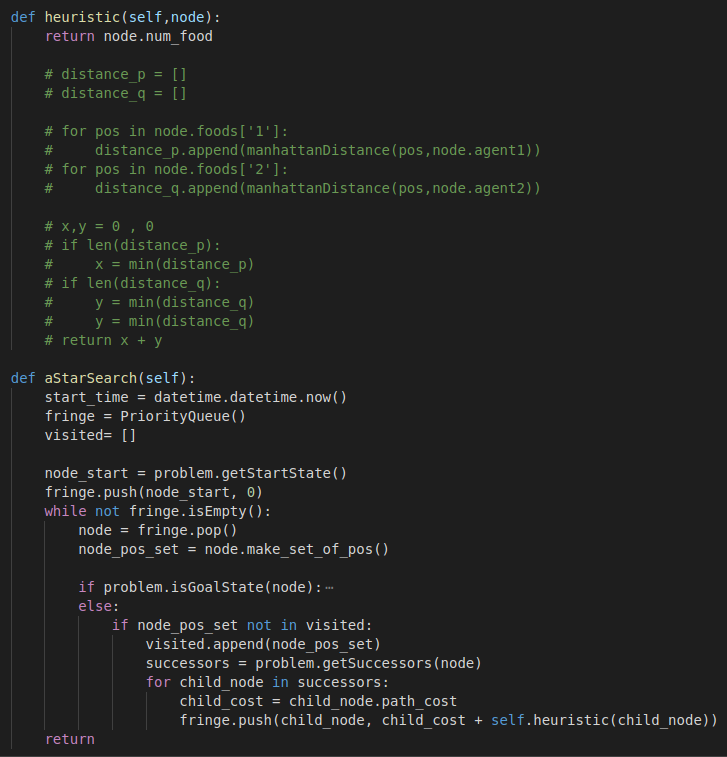
در اینجا دو عامل در انتخاب استیت انتخابی بعدی مهم است که خود را در هزینه تا نود جاری به اضافه ی هزینه از نود بچه تا goal را شامل می شود. هر کدام از نود های بچه که مقدار کمتری داشته باشند انتخاب می شوند. در حقیقت برای پیاده سازی از priority queue استفاده کردیم.

در اینجا دو روش برای hearustic می توانینم انتخاب کنیم. یکی تعداد غذاهای باقی مانده در هر استیت که مشخصا از هزینه واقعی که حداقل در هر حرکت باید یکی جا به جا شویم تا غذا خورده شود کمتر است و این یعنی admissable است.

در روش دوم می توانیم فاصله p را تا نزدیک ترین غذا و فاصله q را تا نزدیک ترین غذا ی خود یعنی ۲ را بدست بیاوریم و مجموع را به عنوان خروجی بدهیم. که باز هم چون باید همه غذا ها خورده شود این مجموع از هزینه واقعی کمتر است. دقت می کنیم که p نمی تواند غذای 2 , و q نمی تواند غذای 1 را بخورد.

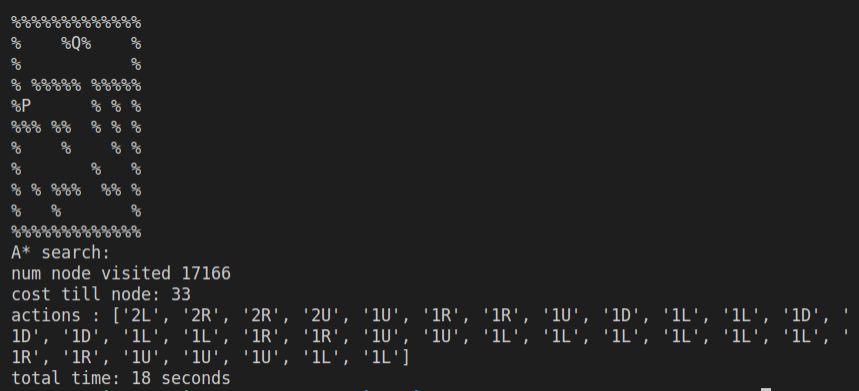
این الگوریتم complete‌ و optimal است و حافظه زمانی و مکانی آن نمایی است.

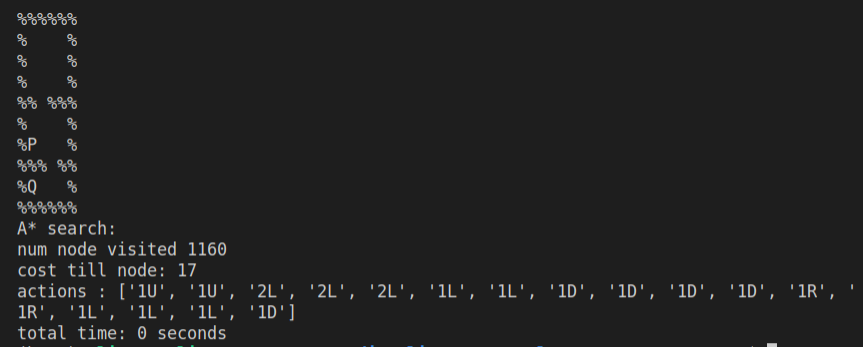
در زیر نحوه پیاده سازی الگویتم آورده شده است. همچنین به وضوح consistency در حالت نخست برقرار است چرا که تعداد غذاها یا ثابت است و یا یکی کم می شود پس هیروستیک در هر نود از هیروستیک بعدی به علاوه یک کمتر است.



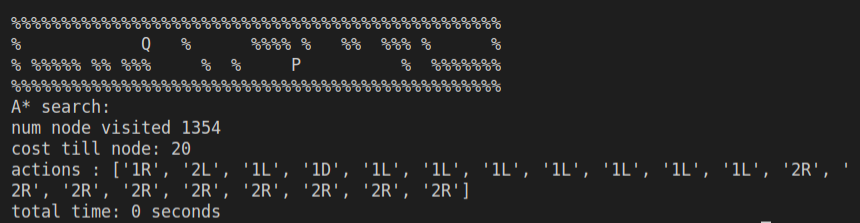
حال خروجی هر کدام از تست ها آورده شده است.

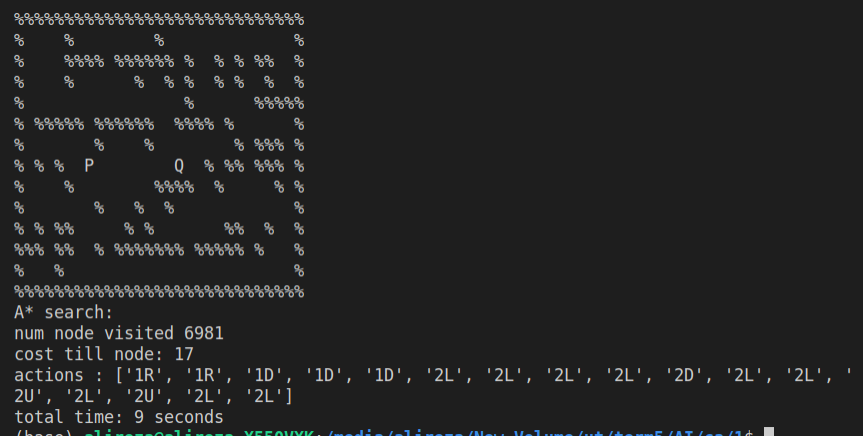
**تست اول‌:**



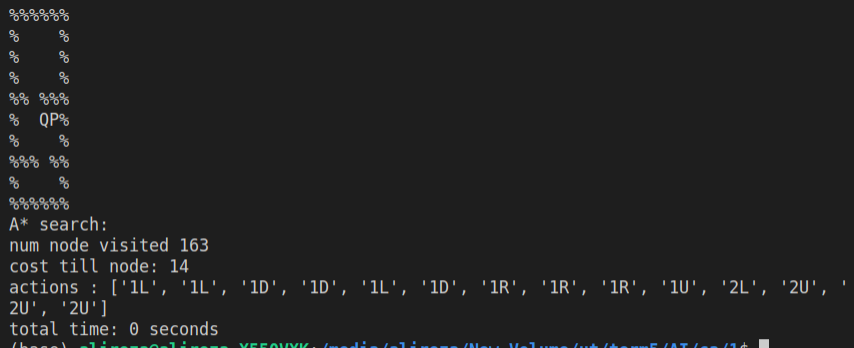
**تست دوم‌:**

**تست سوم:**



**تست چهارم :**

**تست پنجم:**

****

# مقایسه الگوریتم ها با توجه به جدول خواسته شده

**تست اول‌:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| زمان اجرا | تعداد استیت های دیده شده | تعداد استیت های مجزا دیده شده | فاصله از جواب | الگوریتم |
| 22 | 101773 | 17897 | 33 | BFS |
| 3205 | 5803448 | 16046 | 33 | IDS |
| 17 | 97196 | 17166 | 33 | A\* search |

**تست دوم‌:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| زمان اجرا | تعداد استیت های دیده شده | تعداد استیت های مجزا دیده شده | فاصله از جواب | الگوریتم |
| 0 | 6343 | 1153 | 17 | BFS |
| 5 | 558836 | 479 | 17 | IDS |
| 0 | 6306 | 1160 | 17 | A\* search |

**تست سوم‌:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| زمان اجرا | تعداد استیت های دیده شده | تعداد استیت های مجزا دیده شده | فاصله از جواب | الگوریتم |
| 0 | 6657 | 1353 | 20 | BFS |
| 13 | 100679 | 853 | 20 | IDS |
| 0 | 6680 | 1354 | 20 | A\* search |

**تست چهارم‌**‌

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| زمان اجرا | تعداد استیت های دیده شده | تعداد استیت های مجزا دیده شده | فاصله از جواب | الگوریتم |
| 12 | 49543 | 9113 | 17 | BFS |
| 107 | 286345 | 7729 | 17 | IDS |
| 9 | 37268 | 6981 | 17 | A\* search |

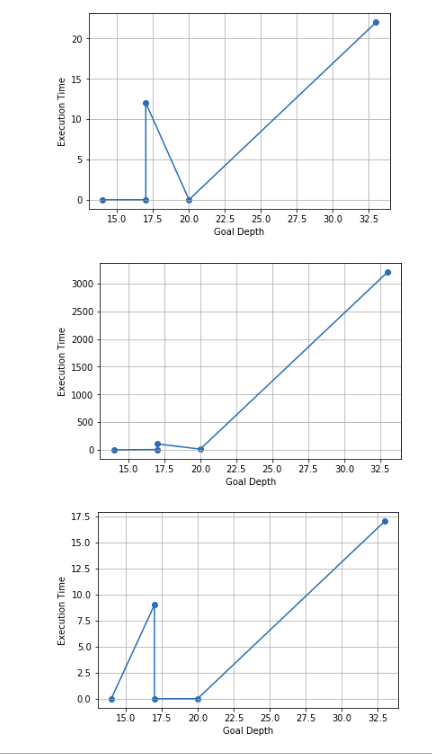
**تست پنجم‌:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| زمان اجرا | تعداد استیت های دیده شده | تعداد استیت های مجزا دیده شده | فاصله از جواب | الگوریتم |
| 0 | 761 | 170 | 14 | BFS |
| 0 | 8417 | 145 | 14 | IDS |
| 0 | 733 | 163 | 14 | A\* search |

با توجه به اینکه هر کدام از الگوریتم ها مسءله را اپتیمال حل می کنند ولی مدت زمان الگوریتم IDS زیاد است و خب در چنین شرایطی اگر branch factor کم باشد bfs می تواند بهتر عمل کند. در مقابل استفاده از سرچ با آگاهی از رفتن به بعضی استیت ها خود داری می کند و در حقیقت تعداد نود های ویزیت شده و زمان کمتری خواهد داشت.

# **نمودار**

در انتها نمودار سه الگوریتم به ترتیب آمده است که محور x آن فاصله از جواب و y نشانه ی هزینه زمانی است.



**پایان**

**پایان**