گزارش پروژه‌ی هوش مصنوعی پریا مهربد ۹۷۳۱۰۶۳

فرموله سازی: فرموله سازی این مسئله مانند مسئله ی پازل ۸ تایی است و محیط آن شناخته شده، کاملا مشاهده پذیر قطعی و گسسته است

حالات: یک حالت محل هر یک از دانش آموزان و جعفر را مشخص میکند

حالت اولیه: هر یک از حالات که از ورودی گرفته میشود

اعمال: جابه جا کردن جعفر (#) با افراد کنارش(چپ ، راست ، جلو، عقب)

آزمون هدف: بررسی اینکه صف ها به حالت مورد نظر رسیده اند یا نه

هزینه مسیر: تعداد جابجایی ها تا رسیدن به هدف

اعمال: جابه جا کردن جعفر در حالت فعلی با دانش آموزان کنارش

پیاده سازی: هر حالت را مثل یک گره در یک گراف در نظر گرفتم که به حالت های پسینش یال وجود دارد. این گره در کد، کلاس Node است و برای ذخیره‌ی حالت صف از یک ساختمان داده به شکل یک لیست دو بعدی از زوج مرتب(tuple)هایی که قد و کلاس هر فرد را نشان می دهد استفاده کردم.مثل روبه‌رو:

```
[('190', 'a'), ('180', 'a'), ('170', 'a')]
[('191', 'b'), ('181', 'b'), ('181', 'b')]
[('176', 'c'), '     #     ', ('168', 'c')]
[('191', 'c'), ('186', 'd'), ('182', 'd')]
```

برای پیاده سازی هر الگوریتم کلاسی به نام Graph در هر فایل وجود دارد که یک یا چند متد برای اجرای الگوریتم و فیلدهای مورد نیاز برای پیاده‌سازی الگوریتم را دارد و تعدادی از متد های آن در هر ۳ فایل مشترک است:

متد is_goal: در واقع آزمون هدف است و بررسی میکند که به حالات نهایی رسیدیم یا نه

متد produce_goals: با استفاده از حالت اولیه یکی از حالات نهایی را درست میکند و جایگشت های حالت نهایی بدست آمده را با استفاده از تابع permutation میسازد.این تابع در فایل مربوط به A* و ids تنها شکل صف هدف را تولید میکند تا در الگوریتم با شکل صف حالات مقایسه شود ولی در الگوریتم جستجوی دوطرفه گره‌های متناظر این صف‌ها را میسازد و parent همگی آنها را یک گره‌ی ساختگی قرار میدهد تا از این گره ساختگی الگوریتم جستجوی اول سطح را اجرا کنیم.

الگوریتم ids: برای پیاده سازی این الگوریتم از تابع ids برای اجرای dfs با عمق محدود استفاده کرده ایم. که عمق اولیه و ماکزیمم عمق از فیلد های کلاس Graph و قابل تعیین اند.همچنین الگوریتم dfs بازگشتی پیاده سازی شده.(جستجوی dfs پیاده سازی شده جستجوی درختی است)

الگوریتم A*: برای پیاده سازی این الگوریتم نیاز به تعیین یک تابع هیوریستیک قابل قبول داریم که هزینه تا مقصد را خوشبینانه حدس بزند.تابع به این صورت عمل میکند که صف هر حالت را با هر یک از حالات هدف مقایسه میکند و مینیمم بین این اعداد را به عنوان مقدار هیوریستیک گره برمیگرداند.

فرض کنیم حالت ابتدایی نسبت به یکی از هدف ها دارای ماکزیمم نابجایی یعنی row x col - 1 تا نابجایی است. واضح است که برای رسیدن به آن هدف حداقل هریک از افراد را باید یک بار با جعفر جابجا کنیم و این تعداد جابجایی مطمئنا کمتر از هزینه ی رسیدن تا آن هدف است، پس مینیمم این جابجایی ها نیز از هزینه ی واقعی کمتر است در نتیجه این هیوریستیک قابل قبول است.

توابع هیوریستیک قابل قبول دیگری نیز وجود دارند مثل اینکه فاصله ی منهتنی هر فرد را تا مکان درستش در هدف اندازه بگیریم و این فاصله را برای هر فرد جمع کنیم ولی پیدا کردن جای متناظر هر فرد در حالات نهایی هزینه بر است.

بعد از آنکه تابع هیوریستیک انتخاب شد برای هزینه‌ی مسیر نیز پارامتری به نام g برای هر گره در نظر می‌گیریم تا هر دفعه از مجموعه ی frontier گره‌ای با کمترین f=g+h یا هزینه برای بسط دادن و اجرای آزمون هدف انتخاب شود.

الگوریتم جستجو دوطرفه: برای اجرای این الگوریتم همانطور که در متد produce_goal توضیح داده شد ابتدا برای هر یک از حالات هدف گره‌ای تولید میکنیم و بعد گره‌ی فرضی را به عنوان parent این گره‌ها قرار میدهیم و با بسط غیر همزمان دو تا الگوریتم bfs با شروع از گره فرضی و شروع از گره اولیه اجرا میکنیم.آزمون هدف هنگام بسط گره انجام میشود و هنگامی که یک گره از یکی از bfsها در مجموعه‌ی frontier دیگری پیدا شود آن گره و گره متناظر آن در frontier دیگری برگردانده میشود تا با استفاده از parent هایشان مسیر رسیدن به هدف را چاپ کنیم.

مقایسه ی عملکرد الگوریتم ها:

	Iterative deepening search	A* search	Bidirectional search
Time complexity	$O(b^d)$	$O(\log h^*(n))$	$O(b^{d/2})$
Space complexity	$O(bd)$	$O(b^d)$	$O(b^{d/2})$

با مقایسه ی پیچیدگی حافظه نتیجه میگیریم الگوریتم ids از لحاظ استفاده از حافظه بهتر است.گرچه برای رسیدن به جواب بیشترین گره را تولید میکند و بسط میدهد ولی در واحد زمان حافظه‌ی کمتری نسبت به دو الگوریتم دیگر استفاده میکند.با توجه به پیچیدگی‌های زمانی بالا میتوان فهمید که الگوریتم A* سریع تر از دو روش دیگر است و الگوریتم ids از لحاظ زمانی بدترین است گرچه باید در نظر داشت در صورت پیاده‌سازی الگوریتم ids به صورت الگوریتم جستجوی گرافی عملکردش بهبود پیدا میکند ولی همچنان A* بهتر است.در زیر مقایسه‌ی الگوریتم‌ها از لحاظ گره‌های بسط داده(explored) و گره‌های تولید شده(produced) در عمق های متفاوت آورده شده.

	Iterative deepening search		A* search		Bidirectional search	
depth/nodes	produced	explored	produced	explored	produced	explored
7	1865	624	25	13	326	129
10	26194	79375	200	106	1023	464
15	32592810	10748795	2006	1091	3827	1998