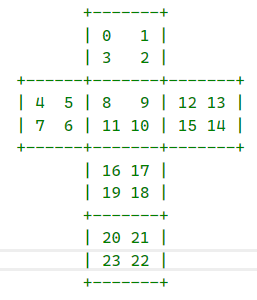
گزارش پروژه‌ میانترم درس مبانی هوش مصنوعی

محمدرضا غفرانی ۹۶۳۱۰۵۳

# سوال ۱

## قسمت اول

در این قسمت مکعب روبیک به صورت یک آرایه ۳۶ تایی مدل سازی شده‌است. ترتیب قرار گرفتن اعضا کنار هم طبق شکل زیر است:



پیاده‌سازی این قسمت مطابق با شبه کدی که در اسلاید‌های درس در رابطه با IDS بود انجام شده‌است. بدین صورت که یک تابع به نام depth\_limited\_search تعریف شده است. این تابع سه ورودی می‌گیرد، روبیکی که باید روی آن کار کند، حداکثر عمقی که باید برود، و جوابی که تاکنون برای مسئله یافته است. این تابع در واقع روی روبیک مورد نظر dfs می‌زند با این تفاوت که از ابتدا عمق را بررسی می‌کند که به عمق مورد نظر رسیده است یا خیر و همچنین ولی می‌خواهد کار را ادامه داده و خود را دوباره صدا بزند یک واحد از limit کم می‌کند. در پیاده‌سازی این بخش به صورت درختی پیش رفته ایم زیرا که حالت تکراری را با یک واحد کاستن از limit صدا می‌زند و در نتیجه احتمال افتادن در یک حلقه بینهایت وجود ندارد.

## قسمت دوم

در این قسمت یک کلاس Node تعریف شده است، که مقادیر زیر را ذخیره می‌کند:

* Cube: مکعب روبیک مربوط آن گره که همانند قسمت قبلی مقادیر را نگهداری می‌کند.
* Parent: والدی که با چرخاندن آن Node فعلی به وجود آمده است
* Phase: وجهی از Node والد که با چرخاندن آن در جهت direction نود فعلی به وجود آمده است.
* Direction: جهتی که اگر آن Phase از Node والد را درآن جهت بچرخانیم Node به وجود آید.

عمده کار در قسمت bidirectional search رخ می‌دهد. این تابع که براساس شبه کدی که در اسلاید های درس وجود دارد پیاده سازی شده است، بدین شکل عمل می‌کند که با تمامی نود های هدف را به یک نود اولیه وصل کرده که با گسترش دادن آن تمامی حالت های هدف تولید می‌شود. منظور از qf همان Qf و منظور از qb همان Qb در اسلاید های درس است. همچنین ef و eb به ترتیب explored set مربوط به قسمت پیشرونده و عقبگرد است. شیوه کار نیز به صورت BFS غیر همزمان است که در آن ابتدا تمامی گره های مربوط به یک حالت ساخته شده و اگر در explored set مربوطه نباشند به qf و با qb بر اساس این که مربوط به قسمت پیشرونده و یا عقبگرد هستند اضافه می‌شوند. سپس یک Node از qf برداشته می‌شود و چک می‌شود که در eb وجود دارد یا خیر. اگر وجود داشت قسمت مربوط به یافتن مسیر فعال می‌شود و در غیر این صورت همین عملیات برای qb و ef تکرار می‌شود تا به جواب برسیم.

## قسمت سوم

در این قسمت نیز کلاس Node همانند قسمت قبلی است. در این قسمت الگوریتم A\* مطابق با شبه‌کد موجود در اسلایدهای درس پیاده‌سازی شده است. نکته قابل توجه در اینجا آن است که هزینه هر گام برابر ۸ در نظر گرفته شده است، تا هیوریستیک داده شده قابل قبول باشد. در این روش نیز یک لیست الویت‌دار ساخته می‌شود که در اینجا ما یک لیست ساده در نظر گرفته‌ایم ولی هنگام برداشتن از آن Node را انتخاب می‌کنیم که کمترین f را داشته باشد. به همین شیوه ادامه می‌دهیم تا به جواب برسیم و با دیگر Node در لیست ما وجود نداشته باشد و ما هنوز به هدف نرسیده باشیم که در این صورت می‌گوییم مسئله جواب نداشته است.

* در قسمت اول تعداد گره های تولید شده در تابع rotate یک واحد زیاد می‌شود چون هنگامی که این تابع فراخوانی می‌شود یک مکعب جدید با چرخاندن مکعب قبلی تولید می‌شود. در قسمت دوم و سوم این عبارت را در constructor مربوط به کلاس Node گذاشته‌ایم گرچه می‌شد در قسمت rotate نیز قرار داد.
* در قسمت اول تعداد گره‌های بسط داده شده هنگامی شمرده می‌شود که تابع depth\_limited\_search فراخوانی شده باشد چون همانطور که گفته شد این تابع یک مکعب را گرفته و شروع به پردازش آن می‌کند، بنابراین باید در این قسمت باشد. در قسمت دوم بعد از هنگامی که یک Node از qb و یا qf برداشته شد یک واحد زیاد می‌شود چون در اینجا عملیات پردازش روی آن Node آغاز می‌شود. در قسمت سوم هنگامی که یک Node از مجموعه frontier برداشته شد به تعداد گره های بسط داده شده یک واحد اضافه می‌شود چرا که در اینجا نیز عملیات پردازش Node آغاز می‌شود.
* عمق جواب در قسمت اول در برابر limitدر تابع depth\_limited\_search\_decorator است. و هنگامی که مقدار آن زیاد می‌شود مقدارش زیاد می‌شود. در قسمت دوم و سوم چون با استفاده از parent به مسیری که منجر به حل روبیک می‌شود می‌رسیم بنابراین در آن تابع (‌ یعنی تابع find\_solution) در هر حلقه یک بار به مقدار عمق جواب اضافه می‌کنیم.
* ماکزیمم تعداد گره ها

شیوه کار این الگوریتم ها روی مثال های زیر به شرح زیر است: