هوش مصنوعی

گزارش کار پروژه اول

کسری کاشانی

810101490

سوال 1) در ابتدا، نحوه ی مدل کردن هر state را توضیح می دهیم. در این سوال، حالت تمام لامپ ها را (0 یا 1) به شکل یک ماتریس n x n یا یک پازل به عنوان یک state در نظر می گیریم.

- ***initial state***: طبق تعریف قبلی برای هر state، state شروع یا initial state، همان ورودی و حالت اولیه ی سوال است. یعنی حالت اولیه ی لامپ ها را به عنوان initial state در نظر می گیریم.

- ***actions***: هر تغییر وضعیت در یک لامپ و در واقع هر خاموش یا روشن کردن یک لامپ در یک state را به عنوان action هایمان در نظر می گیریم.

- ***transition model***: یعنی اینکه با انجام یک action و تغییر یک لامپ در یک state، به چه state جدیدی می رسیم.

- ***goal state***: همان آخرین state مطلوب است که در آن وضعیت، تمام لامپ ها باید خاموش یا 0 باشند. در واقع یک ماتریس n x n با مقدار 0 است.

- ***path cost***: هزینه ی تبدیل هر state به یک state دیگر، تنها عوض کردن وضعیت یک لامپ است که آن را 1 در نظر می گیریم.

سوال 2) هر الگوریتم استفاده شده برای search را به طور مختصر توضیح می دهیم:

- ***BFS***: در این الگوریتم، جست و جو را به صورت سطحی انجام می دهیم. یعنی ابتدا initial state را بررسی کرده و فرزندان آن را در frontier که یک صف است، در صورتی که قبلا در explored نبوده اند، قرار می دهیم. سپس تمام این فرزندان را به صورت FIFO بررسی می کنیم و با انجام goal test، اگر هر کدام از فرزندان این state ها همان goal state بود، جواب را بر می گردانیم. این الگوریتم، به دلیل اینکه هزینه ها برابر هستند، پاسخ optimal را تولید می کند.

- ***IDS***: در این الگوریتم، جست جو را به کمک یک limit انجام می دهیم. به این صورت که limit را از 0 شروع کرده و در درخت جست و جو، هر بار فقط تا عمق limit را به صورت DFS جست و جو می کنیم و در پایان، در صورت پیدا نشدن goal state، limit را ++ کرده و جست و جو را دوباره از اول و از initial state به همین شکل تکرار می کنیم. در DFS نیز، جست و جو را به صورت عمقی انجام می دهیم. یعنی ابتدا initial state را بررسی کرده و فرزندان آن را در frontier که یک استک است، در صورتی که قبلا در explored نبوده اند، قرار می دهیم. سپس تمام این فرزندان را به صورت LIFO بررسی می کنیم و با انجام goal test، اگر هر کدام از فرزندان این state ها همان goal state بود، جواب را بر می گردانیم. این الگوریتم، به دلیل اینکه هزینه ها برابر هستند، پاسخ optimal را تولید می کند.

- ***A**: در این الگوریتم، برای هر state یک تابع heuristic یا h(n) تعریف می کنیم. همچنین برای هر state تابع evaluation را به شکل f(n) = g(n) + h(n) تعریف می کنیم که تابع g(n) همان هزینه ی رسیدن از نود initial state به نود n است. برای انجام این الگوریتم، ابتدا initial state

را بررسی کرده و فرزندان آن را در frontier که یک min heap است، در صورتی که قبلا در explored نبوده اند، قرار می دهیم. سپس از میان state های داخل frontier، state با کمترین f(n) را انتخاب کرده آن را expand می کنیم. همچنین برای انجام goal test، اگر هر کدام از state ها هنگام اضافه شدن به explored همان goal state بود، جواب را بر می گردانیم. توجه شود که در صورتی که دو state دارای f(n) های برابر باشند، state با state_name کمتر انتخاب می شود به طوری که هر state هنگام اضافه شدن به frontier، state_name اش ++ می شود و لذا هر state یک state_name جدا برای خود دارد و در واقع با این کار، هنگام برخورد با دو نود با f(n) های مساوی، نودی که زود تر وارد frontier شده بود را انتخاب می کنیم. این الگوریتم، در صورتی که تابع h(n) یک تابع consistent باشد، پاسخ optimal را تولید خواهد کرد. توجه شود که heuristic اولی که انتخاب شده، با اینکه در ادامه نشان داده خواهد شد که نه admissible و نه consistent نیست، یک تخمین خوبی است و در تمام تست های سوال پاسخ optimal را تولید می کند و لذا تفاوت این heuristic با heuristic سوم که admissible و consistent است به خوبی حس نمی شود چرا که هر دو پاسخ بهینه را تولید می کنند.

- ***weighted A**** : این الگوریتم، کاملا مشابه الگوریتم A* است با این تفاوت که مقادیر h(n) هر نود برای محاسبه ی f(n)، در یک مقدار ثابت $a > 1$ ضرب می شوند و در این صورت، تعداد نود های کمتری ملاقات می شوند اما ممکن است optimality پاسخ نهایی از بین برود و لذا پاسخ نهایی در این الگوریتم لزوما optimal نخواهد بود. در واقع هرچقدر مقدار a را بیشتر کنیم، سریع تر به goal state نزدیک می شویم در ازای اینکه optimality پاسخ نهایی را بیشتر از دست می دهیم.

با توجه به ویژگی این الگوریتم ها و به طور کلی و حدودی، weighted A* سریع تر از A* سریع تر از IDS برابر با BFS می باشد. همچنین فضای مورد نیاز در

IDS کمتر از BFS کمتر از weighted A* برابر با A* می باشد. همچنین تمام این الگوریتم ها complete هستند. بهینه بودن هر یک از الگوریتم ها نیز بررسی شده بود.

سوال 3) هر سه heuristic استفاده شده را بررسی می کنیم:

- ***heuristic 1***: در این تابع، مقدار h(n) برای هر state را برابر با تعداد لامپ های روشن در آن state در نظر می گیریم. در واقع تعداد کل 1 های موجود در ماتریس هر state را به عنوان h(n) آن state در نظر می گیریم.

$$\begin{matrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{matrix}$$

این تابع admissible نیست. زیرا مثلا برای حالت مقدار تابع h(n) برابر با 5 است در صورتی که هزینه ی واقعی این state برای رسیدن به goal state، خاموش کردن لامپ ردیف دوم ستون دوم است که هزینه ی 1 دارد و لذا رابطه ی $h(n) \leq h^*(n)$ برقرار نیست. همچنین این تابع

$$\begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix}$$

consistent هم نیست. زیرا مثلا در حالت A یعنی که با خاموش

$$\begin{matrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \end{matrix}$$

کردن لامپ ردیف دوم ستون دوم، به حالت B یعنی می رسیم، مقدار h(B) + cost(A to B) برابر با 5 = 4 + 1 و مقدار h(A) برابر با 9 است و لذا رابطه ی h(A) < h(B) + cost(A to B) برقرار نیست.

- ***heuristic 2***: در این تابع، مقدار h(n) برای هر state را برابر با تعداد لامپ های خاموش در آن state در نظر می گیریم. در واقع تعداد کل 0 های موجود در ماتریس هر state را به عنوان h(n) آن state در نظر می گیریم.

$$\begin{matrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{matrix}$$
این تابع admissible نیست. زیرا مثلا برای حالت مقدار تابع h(n) برابر با 4 است در صورتی که هزینه ی واقعی این state برای رسیدن به goal state، خاموش کردن لامپ ردیف دوم ستون دوم است که هزینه ی 1 دارد و لذا رابطه ی $h(n) \leq h^*(n)$ برقرار نیست. همچنین این تابع
$$\begin{matrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{matrix}$$
consistent هم نیست. زیرا مثلا در حالت A یعنی که با روشن
$$\begin{matrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{matrix}$$
کردن لامپ ردیف دوم ستون دوم، به حالت B یعنی می رسیم، مقدار $h(A)$ برابر با 9 و مقدار $cost(A\ to\ B) + h(B)$ برابر با $1 + 4 = 5$ است و لذا رابطه ی $cost(A\ to\ B) + h(B) \geq h(A)$ برقرار نیست.

- ***heuristic 3***: در این تابع، مقدار h(n) برای هر state را برابر با تعداد لامپ های روشن در آن state تقسیم بر 5 در نظر می گیریم. در واقع تعداد کل 1 های موجود در ماتریس هر state تقسیم بر 5 را به عنوان h(n) آن state در نظر می گیریم. این تابع admissible است. چرا که هر لامپ می تواند حداکثر 5 لامپ (با احتساب خودش) را خاموش کند و به عبارتی هر 5 لامپ برای خاموش شدن، به خاموش شدن حداقل 1 لامپ نیاز دارند که کمینه هزینه ی آن برابر 1 است و برای خاموش شدن بیش از 5 لامپ، حداقل هزینه برابر با 2 است و حداکثر h(n) نیز به دلیل تقسیم بر 5 شدن تعداد لامپ های روشن برابر با $10 / 5 = 2$ خواهد بود که برابر با همان حداقل هزینه ی واقعی است. در نتیجه برای تمامی نود ها رابطه ی $h(n) \leq h^*(n)$ برقرار است و لذا این تابع admissible می باشد. همچنین این تابع consistent نیز می باشد چرا که رابطه ی $cost(A\ to\ B) + h(B) \geq h(A)$ برای تمامی نود ها صادق است.

سوال 4) طبق تعاریف و نیز مطابق انتظار، برای هر ورودی، الگوریتم جست و جو با heuristic 3 که consistent است، باید پاسخ های بهینه تری نسبت به heuristic های 1 و 2 که consistent نیستند تولید کند و در واقع هزینه ی رسیدن به goal state که همان طول لیست Solution در جدول است، باید کمتر باشد. اما همان طور که در سوال 2 هم توضیح داده شد، با توجه به اینکه heuristic 1 نیز تخمین خوبی است، هردو تابع heuristic های 1 و 3 در تست های انجام شده پاسخ بهینه را تولید می کنند و لذا با این تست ها نمی توان تفاوت این دو heuristic را مشاهده کرد. همچنین heuristic 1 از heuristic 2 بهتر است و در زمان های خیلی کمتری پاسخ را پیدا می کند به طوری که در برخی از تست ها heuristic 2 به timeout بر میخورد اما heuristic 1 پاسخ را پیدا می کند و در کل یک heuristic بسیار بد است و از تمام الگوریتم های دیگر نیز کند تر کار می کند و با این حال ممکن است پاسخ optimal را هم پیدا نکند. در نهایت، heuristic 3 که consistent است، از بقیه heuristic ها بهتر کار می کند و پاسخ بهینه را در کمترین زمان پیدا می کند.

سوال 5) طبق جدول پاسخ های نهایی موجود در فایل، میانگین زمان اجرای هر کدام از الگوریتم ها را با توجه به تست های انجام شده بدست می آوریم. یعنی در هر الگوریتم زمان اجرای تمام تست ها را جمع کرده و تقسیم بر تعداد تست ها که 9 تاست می کنیم. توجه شود که در برخی از تست ها timeout رخ می دهد و زمان اجرای آن ها را داخل میانگین گیری برابر با حداکثر محدودیت زمانی یعنی 2 دقیقه قرار می دهیم.

BFS: 27.507 sec

IDS: 28.944 sec

A* 1: 14.089 sec

A* 2: 62.723 sec

A* 3: 26.611 sec

weighted A* 1 (a=2): 13.405 sec

weighted A* 1 (a=5): 1.276 sec

weighted A* 2 (a=2): 60.960 sec

weighted A* 2 (a=5): 58.388 sec

weighted A* 3 (a=2): 25.744 sec

weighted A* 3 (a=5): 14.135 sec

همانطور که مشخص است، در تمامی الگوریتم های weighted، با افزایش مقدار a پاسخ در زمان کمتری پیدا می شود و همانطور که گفته شد، تابع heuristic 1 تخمین خوب و heuristic 2 تخمین بسیار بدی است و heuristic 3 نیز با اینکه پاسخ را در زمان بیشتری پیدا می کند، اما اطمینان می دهد که آن پاسخ optimal خواهد بود.

*** *توضیحات مربوط به هر بخش از پروژه و کد، به صورت کامنت در هر تابع و نیز در سوال 2 به طور مختصر و مفید داده شده اند.*