سوال: آیا ترتیب کاوش همان ترتیبی بود که انتظار داشتید؟ آیا پکمن در راه رسیدن به هدف، به همه مربعهای کاوش شده میرود؟

بله ترتیب درست بود. با توجه به اینکه برای dfs از ساختمان داده ی priorityQueue استفاده شده که از یک minheap استفاده میکند(در نقش maxheap. در ادامه توضیح دادم) ولی در این نوع سرچ ما به ساختمان داده ی استک یا maxheap نیاز داریم و تابع search که کد آنرا نوشتم یک تابع کلی برای هر چهار تا سرچ میباشد. در نتیجه، برای جستجوی dfs اولویت هر نود را در -1 ضرب کردم و صف اولویت در نقش maxheap میباشد. و با توجه به اینکه در صف اولویت افزودن successor به صورت معکوس انجام میگیرد، راه حل طولی برابر با 246 را دارد.

خیر. مسیر پکمن از همه ی مربع های کاوش شده گذر نمیکند. چرا که طبق جستجو، هنگامی که از یک نود به بن بست میرسد، یعنی ممکن است اطراف آن دیوار باشد یا نودهایی باشند که قبلا کاوش شده باشند، آن نود قابل expand نیست و در راه حل و مسیر برگردانده شده وجود ندارد.

سوال: آیا این راه حل کمترین هزینه را دارد؟ اگر نه فکر کنید که جستجوی اول عمق چه کاری را اشتباه انجام میدهد.

خیر. جواب بهینه(کمترین هزینه) را ندارد. چرا که با توجه به نوع سرچ dfs که جستجوی عمق میباشد، همیشه به دنبال راه حل در عمق است و طبق ساختمان داده ی maxheap یا پشته، همیشه نودی برای expand و یا گسترش انتخاب میشود که دارای عمق بیشتری میباشد. و در اینجا که طول مسیر 246 میباشد مشاهده میکنیم که یکی از طولانی ترین مسیر ها برای رسیدن به هدف انتخاب شده است.

سوال: الگوریتم های جستجویی که تا به این مرحله پیاده سازی کرده اید را روی openMaze اجرا کنید و توضیح دهید چه اتفاقی می افتد.

الگوریتم جستجوی dfs:

جستجوی عمق میباشد و با توجه به ویژگی آن که در سوال قبل گفته شد، به دنبال راه حل در عمق میباشد؛ یعنی طولانی ترین مسیرها را برای رسیدن به هدف امتحان میکند. در اینجا نیز یکی از طولانی ترین مسیر ها برای رسیدن به هدف را برمیگرداند.(مسیر با طول 390).

الگوریتم جستجوی bfs:

جستجوی سطح میباشد و به دنبال راه حل و هدف در سطح میباشد. در اینجا با توجه به اینکه هزینه رفتن از نود به نود دیگری یک میباشد، جواب بهینه را پیدا میکند.(مسیر با طول 54)

الگوریتم جستجوی ucs:

با توجه به اینکه هزینه رفتن از یک نود به نود دیگر برابر با یک میباشد، این جستجو همانند bfs عمل میکند و جواب آن همانند bfs میباشد.

جستجوی bfs زیر مجموعه ی جستجوی ucs میباشد که در آن هزینه رفتن از نود به نود دیگر برابر با 1 میباشد.

الگوریتم جستجوی Astar:

مسیر بهینه را پیدا میکند و با استفاده از هیوریستیک Manhattan distance تخمینی از هزینه ی رسیدن به هدف را دارد و تمام نود ها را کاوش[1] نمیکند و نود های کمتری گسترش[2] می یابند.

در این جستجو تعداد نود های گسترش یافته کمتر از سه الگوریتم جستجوی قبلی میباشد و دلیل آن نیز تابع هیوریستیک برای تخمین هزینه برای رسیدن به هدف است.

سوال قسمت شش: cornerHeuristic

سوال:هیوریستیک خود را توضیح دهید و سازگاری آن را استدلال کنید.

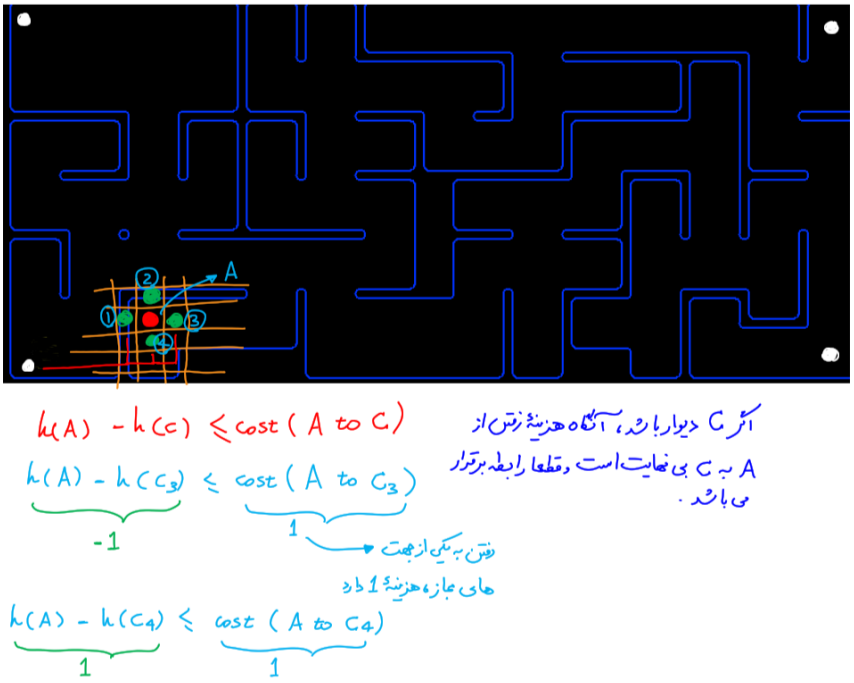
این هیوریستیک بدینگونه میباشد که در ابتدا نقطه ی شروع را به عنوان current node قرار میدهد و به وسیله ی یک for loop روی کرنر ها پیمایش میکند و اگر کرنر قبلا کاوش نشده بود، فاصله ی منهتنی کرنر تا current node را در نظر میگیرد و این عمل بر روی همه ی کرنر های کاوش نشده انجام میگیرد و کوتاه ترین فاصله ی current node تا یک کرنر را بدست می آورد و آن را به مقدار هیوریستیک نهایی که قرار است بازگردانده شود می افزاید. سپس مقدار current node را با نزدیک ترین کرنری که فاصله ی منهتنی آنرا بدست آوردیم برابر قرار میدهد و دوباره فاصله از current node تا نزدیک ترین کرنر کاوش نشده بدست می آورد و مقدار current node را آپدیت میکند و همچنین مقدار هیوریستیک را نیز آپدیت میکند. سپس این مقدار را برمیگرداند.

به طور خلاصه این هیوریستیک فاصله ی استیت فعلی PAC-MAN تا نزدیک ترین گوشه ی کاوش نشده را بدست می آورد و سپس از آن گوشه ای که گفتیم دوباره فاصله را تا نزدیکترین گوشه ی کاوش نشده بدست می آورد و این عمل را تکرار میکند تا گوشه ی کاوش نشده، نداشته باشیم. در نهایت مجموع این فاصله ها را به عنوان هیوریستیک برمیگردانیم.

این هیوریستیک قابل قبول[3] میباشد چرا که آرمانی ترین نوع حرکت در نظر گرفته شده، یعنی فرض شده که هیچ دیواری نداریم و تنها فاصله ی منهتنی در نظر گرفته شده است.

---

[1] Explore
[2] Expand
[3] Admissible



با استفاده از شکل بالا، سازگاری هیوریستیک را توضیح میدهم. اگر نود های سبز رنگ(C) فرزندان نود قرمز رنگ(A) باشند، آنگاه اختلاف مقدار هیوریستک نود قرمز با هر یک از فرزندان حداکثر برابر با 1 میباشد. فرض میکنیم گوشه ی پایین سمت چپ(هر گوشه ای را میتوان مثال زد)، نزدیکترین گوشه ی کاوش نشده باشد، آنگاه تفاوت مقدار هیوریستیک نود قرمز و سبز رنگ ها تنها در رسیدن به گوشه ی پایین سمت چپ میباشد، و هنگامی که هر یک از نودهای سبز یا قرمز رنگ به آن گوشه برسند، باقیمانده مقدار هیوریستیک آنها با همدیگر برابر میشود(طبق الگوریتم برای محاسبه ی هیورسیتک).

برای مثال نود فرزند شماره 4 و نود قرمز رنگ پدر اختلاف مقدار هیوریستیکشان تنها برابر با 1 میباشد. چرا که طبق فاصله ی منهتنی برای رسیدن به نزدیکترین گوشه(گوشه پایین سمت چپ) اختلاف فاصله ی منهتنی آنها، تنها برابر با یک میباشد و در ادامه مسیری که محاسبه میشود برای هر دو یکسان است و همانطور که گفته شد مقدار هیوریستیک پدر فقط یکی از نود قرمز رنگ شماره ی 4 بزرگتر است.

همچنین هزینه ی رفتن از یک نود به نود فرزند در صورتی که مانع وجود نداشته باشد، برابر با یک میباشد.

پس رابطه ی گفته شده در تصویر بالا برقرار میباشد.

سوال قسمت هفت: هیوریستیک برای خوردن همه ی نقطه ها

سوال:هیوریستیک خود را توضیح دهید و سازگاری آن را استدلال کنید.

اگر که نود فرزند فاصله ی منهتنی بیشتری با دورترین غذا را داشت که به این معنی است که مقدار هیوریستیک آن بیشتر از پدرش میباشد و در نتیجه رابطه برقرار است.

این هیوریستیک بدینگونه میباشد که بر روی تمام غذاها و استیت غعلی pac-man جستجوی bfs را انجام میدهد و سپس بزرگترین فاصله بین استیت فعلی پکمن و غذا را برمیگرداند. در پایین به اثبات کانسیستنسی آن میپردازیم.

مساله را به دو حالت تقسیم میکنیم.

اگر هدف یا همان دورترین غذای مدنظر یکی باشد، آنگاه دو حالت وجود دارد.

حالت اول این است که مسیر parent و child یکی باشد که در این حالت اختلاف طول مسیر یا همان اختلاف هیوریستیکشان برابر با یک میباشد.

حالت دوم اینکه مسیر هر دوی آنها متفاوت باشد، در این حالت اگر اختلاف طول دو مسیر بزرگتر مساوی با یک بود، انگاه به تناقض میخوریم، چرا که در این حالت، یک کدام از parent یا child مسیر طولانی تر را انتخاب میکند و همان حالت اول که گفته شد پیش می آید و اختلاف طول مسیر برابر با یک میباشد.

اگر هدف یا دورترین غذا متفاوت باشند، آنگاه تنها حالتی وجود دارد که طول هر دو مسیر برای رسیدن به هر دو غذا برابر باشند، چرا که در صروتی که اختلاف دو مسیر بزرگتر مساوی یک باشد، آنگاه یک کدام از parent و یا child مسیر طولانی تر را انتخاب میکند و دوباره مسیر هر دو یکسان میشود و اختلاف آنها یک میشود.

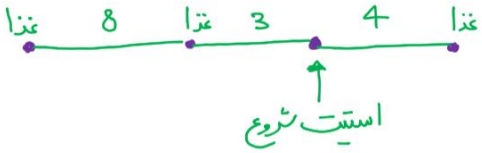
در صورتی که اختلاف طول مسیر parent و child کوچکتر مساوی قدرمطلق یک باشند آنگاه هیوریستیک ما سازگار است.

هزینه ی رفتن از یک نود به یک نود دیگر برابر با یک میباشد و رابطه برقرار است.

در واقع طبق چیزهایی که گفته شد، اختلاف مقادیر هیوریستیک کوچکتر از قدرمطلق یک میباشد و این چیزی که گفته شد کوچکتر مساوی هزینه ی واقعی هست که یک میباشد.

سوال قسمت هشت:

سوال ClosestDotSearchAgent :شما، همیشه کوتاه ترین مسیر ممکن در ماز را پیدا نخواهد کرد. مطمئن شوید که دلیل آن را درک کرده اید و سعی کنید یک مثال کوچک بیاورید که در آن رفتن مکرر به نزدیکترین نقطه منجر به یافتن کوتاهترین مسیر برای خوردن تمام نقاط نمی شود.



این مثال موضوع گفته شده را نشان میدهد. با توجه به راهکار greedy برای پیدا کردن نزدیک ترین غذا، در شکل بالا از استیت شروع به سمت چپ که نزدیکترین غذا(کمترین هزینه) میباشد میرود. سپس از آن استیت به نزدیک ترین غذا که در سمت راست میباشد میرود و سپس به سمت چپ میرود و آخرین غذا رو میخورد.

مسیر بهینه با مسیری که گفته شد فرق دارد. در مسیر بهینه ابتدا به سمت راست میرود و غذا را میخورد و سپس به سمت چپ میرود و دو تا غذای سمت چپ را میخورد که با توجه راهکار greedy، مسیر گفته شده اینگونه نمیباشد.