Artificial intelligence

تمرین عملی اول

شاهین شاهنواز

۸۱۰۸۰۰۰۱۳

عنوان پروژه: تمرین شماره‌ی ۱

هدف: پیاده‌سازی الگوریتم‌های bfs و ids و $A*$ به کمک حداقل ۲ تابع $heuristic$

توضیح کلی پروژه: به صورت خلاصه مسئله این چنین است که یک ماتریس داریم که در هر خانه آن یک لامپ قرار دارد. در ابتدا هر لامپ میتواند روشن (۱) یا خاموش (۰) باشد. بنابراین ما یک ماتریس از ۰و۱ ها داریم. حال در هر عملیات اگر ما کلید هر لامپ دلخواهی را بزنیم علاوه بر خودش، لامپ‌های مجاورش نیز (بالا/ پایین/ چپ/ راست) در صورت وجود، تغییر حالت میدهند. (از ۰ به ۱و از ۱ به ۰ تبدیل می‌شوند.) هدف مسئله چنین است که باید کمترین تعداد عملیات ممکن را بیابیم که در نهایت ماتریس اولیه ما را به یک ماتریس که همه‌ی خانه‌هایش صفر است تبدیل کند.

نحوه مدل کردن پروژه: برای اینکه مسئله را به مسائل جست‌وجو مدل کنیم باید حداقل به دنبال متناظر کردن چند المان در مسئله خود بگردیم:

state: من هر حالت از جدول را با ۰و۱ های موجود، یک $state$ میدانم.

initial state: جدول اولیه‌ای که به ما در ورودی می‌دهند.

goal state: جدولی که تماما همه خانه‌هایش ۰ باشد. در فایل کد یعنی پازلی که تابع $is\_solved()$ آن $True$ شده باشد.

action: همین که هر لامپ را انتخاب کنیم برای زدن یا نه. در فایل کد متناظر با تابع get_moves خواهد بود.

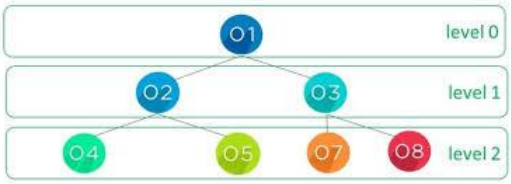
successor function: وقتی یک خانه را انتخاب میکنیم و کلیدش را میزنیم خود آن خانه و خانه‌های مجاوز $toggle$ میخورند و به جدول جدیدی میرسیم.

مسئله را به گرافی متناظر میکنیم. رئوس آن مجموعه استیت هاست و یال های بین آن همان عملیاتی است که باعث میشود دو استیت یا پازل به هم تبدیل شوند.

در پروژه از الگوریتم‌های مختلفی استفاده شده است که به معرفی ویژگی های هر یک میپردازیم. همچنین در فایل $AI\_CA۱\_F۰۳\_student\_notebook.ipynb$ موجود در فولدر جواب کامنت‌هایی گذاشته شده که از لحاظ کدی مراحل حل پروژه ذکر شده باشد.

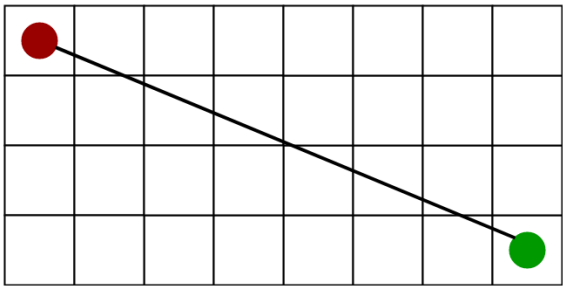
فایل کد کامل شده‌ی نت بوکی است که در اختیار گذاشته شده. در ابتدا ایمپورت کردن کتابخانه ها، تعریف پازل و توابع مرتبط با آن و ساخت پازل های رندوم انجام شده.

حال میرسیم به بحث **الگوریتم BFS**: در این الگوریتم هدف این است که به ازای هر $node$ (اینجا $state$ ها به $node$ گراف تشبیه شده اند) اولین باری که به آن میرسیم را به عنوان کوتاه ترین مسیر از $initial\ state$به آن نگاه کنیم. چرا که اگر الان که آن را زودتر دیده ایم جواب نباشد در ادامه تعداد $node$ های دیده شده بیشتر یا مساوی با همین مقدار کنونی است. اردر الگوریتم از لحاظ زمانی برای گراف n راسی و m یالی $O(m+n)$ خواهد بود. n که متناسب با تعداد رئوس یا $state$ ها هست عددی حول $۲^{(puzzle\ size^۲)}$ خواهد بود و m که متناسب با تعداد یالهاست که تعداد عملیات ممکن است چیزی حدود $n * \frac{puzzle\ size^۲}{۲}$ خواهد بود. بنابراین میتوان پیشبینی کرد که از یک جایی به بعد خیلی این الگوریتم از لحاظ زمانی خوب عمل نمیکند و انتظار داریم برای مثال هایی با پازلی با اندازه بزرگتر الگوریتم خوب کار نکند. (با توجه به $limit$ ای که ما برای زمان گذاشته ایم.)



الگوریتم IDS: اینجا هدف چیزی مانند پیاده سازی ترکیبی از bfs و dfs با هم است. اجرای dfs با عمق محدود. اگر به جواب رسیدیم که حل است و اگر نه عمق را یکی اضافه میکنیم تا به جواب برسیم. از لحاظ زمانی وضعیت بدتری دارد نسبت به bfs بنابراین میتوان انتظار داشت در جدول مقایسه کارآمدی الگوریتمها، نسبت به bfs از لحاظ زمانی ضعیف تر عمل کند.

الگوریتم $A*$: در اینجا یک حدس اولیه از میزان فاصله ای که در هر استیت تا استیت نهایی وجود دارد میزنیم که همان توابع $heuristic$ ما این وظیفه را به گردن میگیرند. سپس در هر حالت به آن $node$ ای که کوچیک ترین $g(node) + h(node)$ ممکن را دارد میرویم که ممکن است و تاحالا آن را ویزیت نکرده ایم. دقت کنید که $g(node)$ طول کوتاه ترین مسیر از استیت اولیه به $node$ دلخواه ما را ذخیره میکند و برای هر $node$، $h(node)$ عددی ثابت است که همان ابتدای مسئله گرفته ایم. این الگوریتم لزوما جواب بهینه را به ما نمیدهد. اما میتوان یک شرط به توابع حدسی خود اضافه کنیم که بهینه بودن جواب را تضمین کند. آن شرط $consistency$ است که میگوید به ازای هر دو گره a, b ای در گراف اگر شرط $c(a,b) + h(b) >= h(a)$ برقرارباشد (تابع $c(a,b)$ یعنی طول مسیری که از راس a به b میرود) الگوریتم $A*$ جواب بهینه را به ما میدهد. در مورد توابع حدسی جلوتر توضیح داده شده. این الگوریتم از لحاظ بهینگی بهتر از دو الگوریتم قبل است اما در مورد لزوما بهینه جواب دادن آن باید شرط $consistency$ هم رعایت شود در غیر اینصورت لزوما جواب درستی به ما نمیدهد.



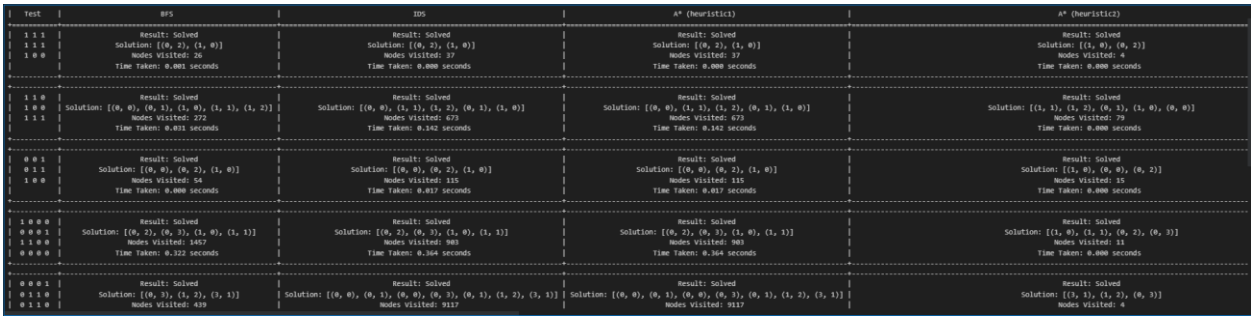
توابع ابتکاری:

- ساده ترین تابعی که به ذهن هر فرد میرسد به ازای هر پازل این است که تعداد خانه‌هایی که مقدار ۱ دارند را خروجی دهد. این از شرط $consistency$ پیروی نمیکند چرا که ممکن است با یک حرکت ۵ خانه (خانه انتخابی و ۴ خانه مجاور آن) همگی از ۱ به صفر تغییر حالت دهند و این صرفا با خرج هزینه ۱ عملیات انجام شده باشد. اما برای شروع بد نیست.۱ $heuristic$

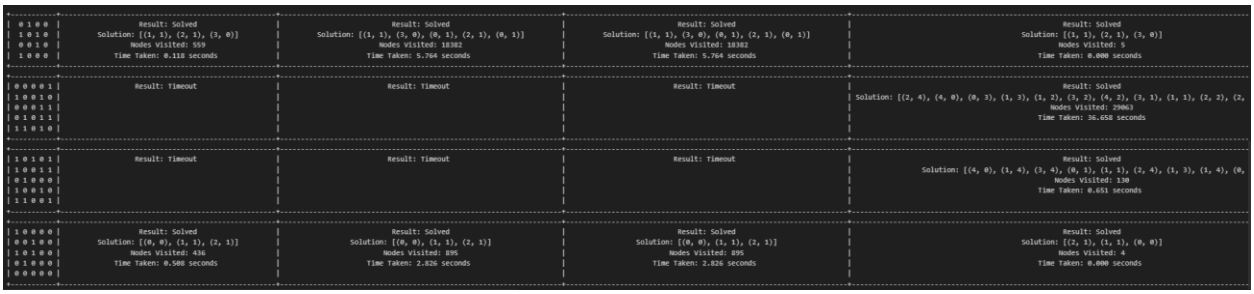
- در ادامه یک تابعی داریم که اتفاقا از شرط $consistency$ پیروی میکند. اینگونه که به ازای هر پازل، تعداد خانه های روشن آن تقسیم بر ۵ را خروجی میدهد. دلیل پیروی این تابع از شرط این است که به ازای هر دو $node$ به نام های a, b فرض کنید تعداد خانه های روشن آنها به ترتیب c, dباشد. حال $cost(a,b) \geq ceil(|c-d)/5)$ یعنی میزان تعداد عملیاتی که انجام میدهیم برای اینکه a به b برسد حداقل برابر با سقف (تفاضل تعداد خانه های روشن آنها) تقسیم بر (۵) خواهد بود. چون با هر عملیات حداکثر ۵ خانه تغییر حالت میدهند. بنابراین با تعداد کمتر به نود b نمیتوانیم برسیم. حال که میدانیم به ازای هر راسی تابع ابتکاری آن عددی ثابت است پس $cost(a,b) + h(b) \geq ceil\left(\frac{|c-d|}{5}\right) + \frac{d}{5} \geq \frac{c}{5}$. پس این تابع جواب بهینه را به ما میدهد. $heuristic2$

در نهایت جدول ها و نتایج نهایی: نتیجه نهایی برای هر حالت در فایل $res.txt$ ذخیره شده.

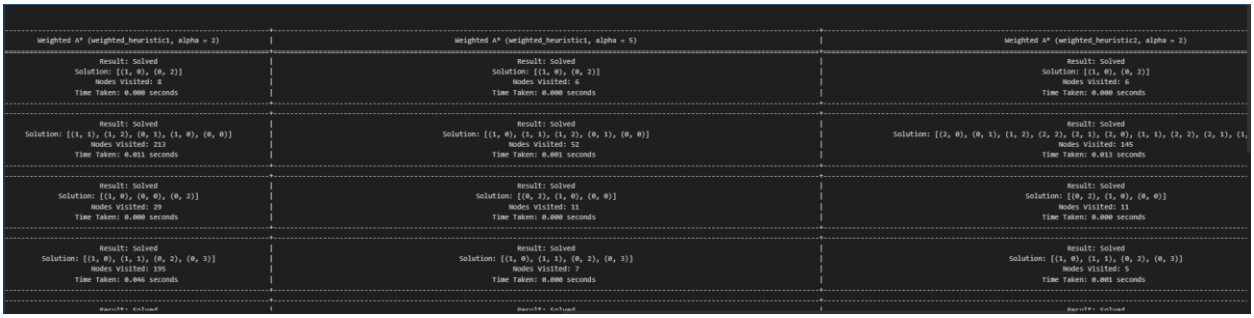
اما چیزی که قابل مشاهده است این است که تا یک جایی که bfs و ids جواب میدهند جواب ها بهینه است. اگر $a*$ با تابع ابتکاری دومی که زدیم تایم لیمیت نخورد جواب بهینه را میدهد و $a*$ با تابع ابتکاری اول صرفا یک جواب حدسی و حدودی به ما میدهد که لزوما بهینه نیست.



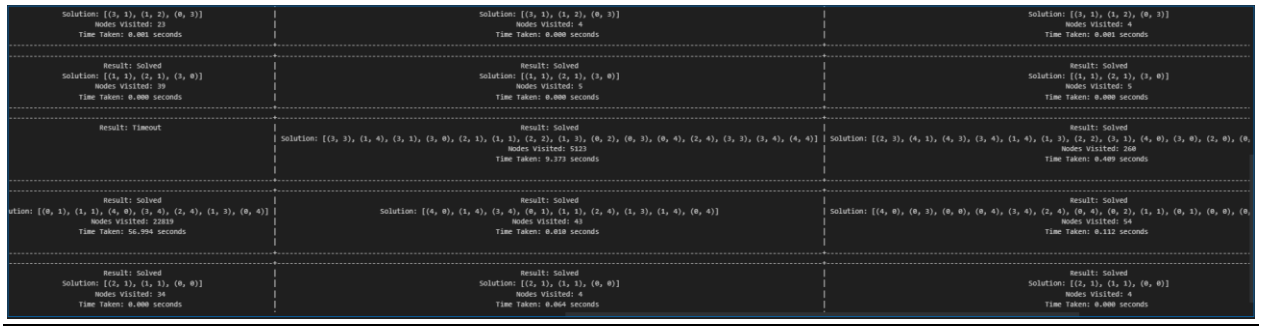
Test	BFS	IDS	A* (heuristic1)	A* (heuristic2)
1 1 1 1 1 1 1 0 0	Result: Solved Solution: [(0, 2), (1, 0)] Nodes Visited: 26 Time Taken: 0.001 seconds	Result: Solved Solution: [(0, 2), (1, 0)] Nodes Visited: 37 Time Taken: 0.000 seconds	Result: Solved Solution: [(0, 2), (1, 0)] Nodes Visited: 37 Time Taken: 0.000 seconds	Result: Solved Solution: [(1, 0), (0, 2)] Nodes Visited: 4 Time Taken: 0.000 seconds
1 1 0 1 0 0 1 1 1	Result: Solved Solution: [(0, 0), (0, 1), (1, 0), (1, 1), (1, 2)] Nodes Visited: 272 Time Taken: 0.031 seconds	Result: Solved Solution: [(0, 0), (1, 1), (1, 2), (0, 1), (1, 0)] Nodes Visited: 673 Time Taken: 0.142 seconds	Result: Solved Solution: [(0, 0), (1, 1), (1, 2), (0, 1), (1, 0)] Nodes Visited: 673 Time Taken: 0.142 seconds	Result: Solved Solution: [(1, 1), (1, 2), (0, 1), (1, 0), (0, 0)] Nodes Visited: 79 Time Taken: 0.000 seconds
0 0 1 0 1 1 1 0 0	Result: Solved Solution: [(0, 0), (0, 2), (1, 0)] Nodes Visited: 54 Time Taken: 0.000 seconds	Result: Solved Solution: [(0, 0), (0, 2), (1, 0)] Nodes Visited: 115 Time Taken: 0.017 seconds	Result: Solved Solution: [(0, 0), (0, 2), (1, 0)] Nodes Visited: 115 Time Taken: 0.017 seconds	Result: Solved Solution: [(1, 0), (0, 0), (0, 2)] Nodes Visited: 15 Time Taken: 0.000 seconds
1 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0	Result: Solved Solution: [(0, 2), (0, 3), (1, 0), (1, 1)] Nodes Visited: 1457 Time Taken: 0.322 seconds	Result: Solved Solution: [(0, 2), (0, 3), (1, 0), (1, 1)] Nodes Visited: 983 Time Taken: 0.364 seconds	Result: Solved Solution: [(0, 2), (0, 3), (1, 0), (1, 1)] Nodes Visited: 983 Time Taken: 0.364 seconds	Result: Solved Solution: [(1, 0), (1, 1), (0, 2), (0, 3)] Nodes Visited: 11 Time Taken: 0.000 seconds
0 0 0 1 0 1 1 0 0 1 1 0	Result: Solved Solution: [(0, 3), (1, 2), (3, 1)] Nodes Visited: 439	Result: Solved Solution: [(0, 0), (0, 1), (0, 0), (0, 3), (0, 1), (1, 2), (3, 1)] Nodes Visited: 9117	Result: Solved Solution: [(0, 0), (0, 1), (0, 0), (0, 3), (0, 1), (1, 2), (3, 1)] Nodes Visited: 9117	Result: Solved Solution: [(3, 1), (1, 2), (0, 3)] Nodes Visited: 4



0 1 0 0 1 0 1 0 0 0 1 0 1 0 0 0	Result: Solved Solution: [(1, 3), (2, 1), (3, 0)] Nodes Visited: 559 Time Taken: 0.118 seconds	Result: Solved Solution: [(1, 1), (3, 0), (0, 1), (2, 1), (0, 1)] Nodes Visited: 18382 Time Taken: 5.764 seconds	Result: Solved Solution: [(1, 1), (3, 0), (0, 1), (2, 1), (0, 1)] Nodes Visited: 18382 Time Taken: 5.764 seconds	Result: Solved Solution: [(1, 3), (2, 1), (3, 0)] Nodes Visited: 5 Time Taken: 0.000 seconds
0 0 0 0 1 1 0 0 1 0 0 0 0 1 1 0 1 0 1 1 1 1 0 1 0	Result: Timeout	Result: Timeout	Result: Timeout	Result: Solved Solution: [(2, 4), (4, 0), (0, 3), (1, 3), (1, 2), (3, 2), (4, 2), (3, 1), (1, 1), (2, 2), (2, Nodes Visited: 29063 Time Taken: 36.658 seconds
1 0 1 0 1 1 0 0 1 1 0 1 0 0 0 1 0 0 1 0 1 1 0 0 1	Result: Timeout	Result: Timeout	Result: Timeout	Result: Solved Solution: [(4, 0), (1, 4), (3, 4), (0, 1), (1, 1), (2, 4), (1, 3), (1, 4), (0, Nodes Visited: 130 Time Taken: 0.651 seconds
1 0 0 0 0 0 0 1 0 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 0	Result: Solved Solution: [(0, 0), (1, 1), (2, 1)] Nodes Visited: 436 Time Taken: 0.508 seconds	Result: Solved Solution: [(0, 0), (1, 1), (2, 1)] Nodes Visited: 895 Time Taken: 2.826 seconds	Result: Solved Solution: [(0, 0), (1, 1), (2, 1)] Nodes Visited: 895 Time Taken: 2.826 seconds	Result: Solved Solution: [(2, 1), (1, 1), (0, 0)] Nodes Visited: 4 Time Taken: 0.000 seconds

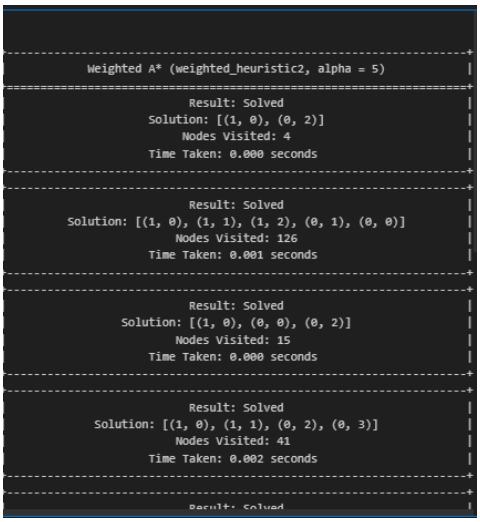


Weighted A* (weighted_heuristic1, alpha = 2)	Weighted A* (weighted_heuristic1, alpha = 5)	Weighted A* (weighted_heuristic2, alpha = 2)
Result: Solved Solution: [(1, 0), (0, 2)] Nodes Visited: 8 Time Taken: 0.000 seconds	Result: Solved Solution: [(1, 0), (0, 2)] Nodes Visited: 6 Time Taken: 0.000 seconds	Result: Solved Solution: [(1, 0), (0, 2)] Nodes Visited: 6 Time Taken: 0.000 seconds
Result: Solved Solution: [(1, 1), (1, 2), (0, 1), (1, 0), (0, 0)] Nodes Visited: 213 Time Taken: 0.011 seconds	Result: Solved Solution: [(1, 0), (1, 1), (1, 2), (0, 1), (0, 0)] Nodes Visited: 52 Time Taken: 0.001 seconds	Result: Solved Solution: [(2, 0), (0, 1), (1, 2), (2, 2), (2, 1), (2, 0), (1, 1), Nodes Visited: 145 Time Taken: 0.013 seconds
Result: Solved Solution: [(1, 0), (0, 0), (0, 2)] Nodes Visited: 29 Time Taken: 0.000 seconds	Result: Solved Solution: [(0, 2), (1, 0), (0, 0)] Nodes Visited: 11 Time Taken: 0.000 seconds	Result: Solved Solution: [(0, 2), (1, 0), (0, 0)] Nodes Visited: 11 Time Taken: 0.000 seconds
Result: Solved Solution: [(1, 0), (1, 1), (0, 2), (0, 3)] Nodes Visited: 195 Time Taken: 0.046 seconds	Result: Solved Solution: [(1, 0), (1, 1), (0, 2), (0, 3)] Nodes Visited: 7 Time Taken: 0.000 seconds	Result: Solved Solution: [(1, 0), (1, 1), (0, 2), (0, 3)] Nodes Visited: 5 Time Taken: 0.001 seconds
Result: Solved	Result: Solved	Result: Solved

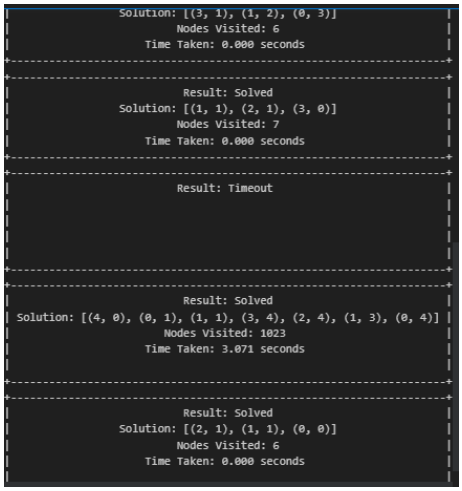


Solution: [(3, 1), (1, 2), (0, 3)] Nodes Visited: 23 Time Taken: 0.001 seconds	Solution: [(3, 1), (1, 2), (0, 3)] Nodes Visited: 4 Time Taken: 0.000 seconds	Solution: [(3, 1), (1, 2), (0, 3)] Nodes Visited: 4 Time Taken: 0.001 seconds
Result: Solved Solution: [(1, 1), (2, 1), (3, 0)] Nodes Visited: 39 Time Taken: 0.000 seconds	Result: Solved Solution: [(1, 1), (2, 1), (3, 0)] Nodes Visited: 5 Time Taken: 0.000 seconds	Result: Solved Solution: [(1, 1), (2, 1), (3, 0)] Nodes Visited: 5 Time Taken: 0.000 seconds
Result: Timeout	Result: Solved Solution: [(3, 3), (1, 4), (3, 1), (3, 0), (2, 1), (1, 1), (2, 2), (1, 3), (0, 2), (0, 3), (0, 4), (2, 4), (3, 3), (3, 4), (4, 4)] Nodes Visited: 5123 Time Taken: 9.373 seconds	Result: Solved Solution: [(2, 3), (4, 1), (4, 3), (3, 4), (1, 4), (1, 3), (2, 2), (3, 1), (4, 0), (3, 0), (2, 0), (0, Nodes Visited: 260 Time Taken: 0.409 seconds
Result: Solved ution: [(0, 1), (1, 1), (4, 0), (3, 4), (2, 4), (1, 3), (0, 4)] Nodes Visited: 22839 Time Taken: 56.994 seconds	Result: Solved Solution: [(4, 0), (1, 4), (3, 4), (0, 1), (1, 1), (2, 4), (1, 3), (1, 4), (0, 4)] Nodes Visited: 43 Time Taken: 0.010 seconds	Result: Solved Solution: [(4, 0), (0, 3), (0, 0), (0, 4), (3, 4), (2, 4), (0, 4), (0, 2), (1, 1), (0, 1), (0, 0), (0, Nodes Visited: 54 Time Taken: 0.112 seconds
Result: Solved Solution: [(2, 1), (1, 1), (0, 0)] Nodes Visited: 34 Time Taken: 0.000 seconds	Result: Solved Solution: [(2, 1), (1, 1), (0, 0)] Nodes Visited: 4 Time Taken: 0.064 seconds	Result: Solved Solution: [(2, 1), (1, 1), (0, 0)] Nodes Visited: 4 Time Taken: 0.000 seconds

---



Weighted A* (weighted_heuristic2, alpha = 5)
Result: Solved Solution: [(1, 0), (0, 2)] Nodes Visited: 4 Time Taken: 0.000 seconds
Result: Solved Solution: [(1, 0), (1, 1), (1, 2), (0, 1), (0, 0)] Nodes Visited: 126 Time Taken: 0.001 seconds
Result: Solved Solution: [(1, 0), (0, 0), (0, 2)] Nodes Visited: 15 Time Taken: 0.000 seconds
Result: Solved Solution: [(1, 0), (1, 1), (0, 2), (0, 3)] Nodes Visited: 41 Time Taken: 0.002 seconds
Result: Solved



Solution: [(3, 1), (1, 2), (0, 3)] Nodes Visited: 6 Time Taken: 0.000 seconds
Result: Solved Solution: [(1, 1), (2, 1), (3, 0)] Nodes Visited: 7 Time Taken: 0.000 seconds
Result: Timeout
Result: Solved Solution: [(4, 0), (0, 1), (1, 1), (3, 4), (2, 4), (1, 3), (0, 4)] Nodes Visited: 1023 Time Taken: 3.071 seconds
Result: Solved Solution: [(2, 1), (1, 1), (0, 0)] Nodes Visited: 6 Time Taken: 0.000 seconds