**Artificial intelligence**

**تمرین عملی اول**

**شاهین شاهنواز**

**810800013**

**عنوان پروژه:** تمرین شماره‌ی

**هدف:** پیاده‌سازی الگوریتم‌های و و به کمک حداقل تابع

**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 یا خاموش باشد. بنابراین ما یک ماتریس از و 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 به و از به 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:

: من هر حالت از جدول را با و های موجود، یک میدانم.

: جدول اولیه‌ای که به ما در ورودی می‌دهند.

: جدولی که تماما همه خانه‌هایش باشد. در فایل کد یعنی پازلی که تابع آن شده باشد.

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

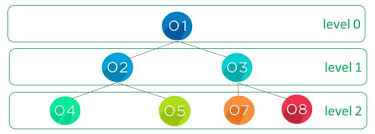
: وقتی یک خانه را انتخاب میکنیم و کلیدش را میزنیم خود آن خانه و خانه‌های مجاوز میخورند و به جدول جدیدی میرسیم.

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 موجود در فولدر جواب کامنت‌هایی گذاشته شده که از لحاظ کدی مراحل حل پروژه ذکر شده باشد.

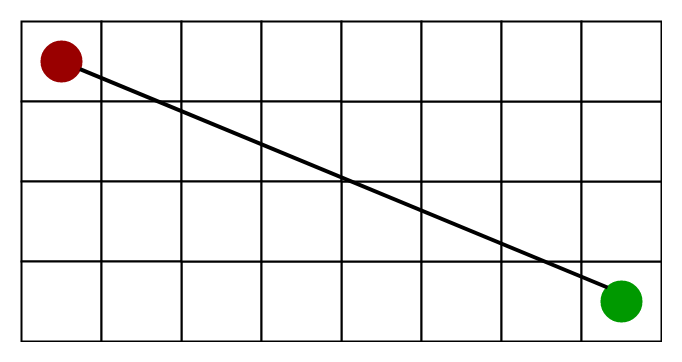
فایل کد کامل شده‌ی نت بوکی است که در اختیار گذاشته شده. در ابتدا ایمپورت کردن کتابخانه ها، تعریف پازل و توابع مرتبط با آن و ساخت پازل های رندوم انجام شده.

حال میرسیم به بحث **الگوریتم :** در این الگوریتم هدف این است که به ازای هر (اینجا ها به گراف تشبیه شده اند) اولین باری که به آن میرسیم را به عنوان کوتاه ترین مسیر از به آن نگاه کنیم. چرا که اگر الان که آن را زودتر دیده ایم جواب نباشد در ادامه تعداد های دیده شده بیشتر یا مساوی با همین مقدار کنونی است. اردر الگوریتم از لحاظ زمانی برای گراف راسی و یالی خواهد بود. که متناسب با تعداد رئوس یا ها هست عددی حول خواهد بود و که متناسب با تعداد یالهاست که تعداد عملیات ممکن است چیزی حدود 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 ای که ما برای زمان گذاشته ایم.)



**الگوریتم :** اینجا هدف چیزی مانند پیاده سازی ترکیبی از و با هم است. اجرای 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 بنابراین میتوان انتظار داشت در جدول مقایسه کارآمدی الگوریتمها، نسبت به از لحاظ زمانی ضعیف تر عمل کند.

**الگوریتم :** در اینجا یک حدس اولیه از میزان فاصله ای که در هر استیت تا استیت نهایی وجود دارد میزنیم که همان توابع ما این وظیفه را به گردن میگیرند. سپس در هر حالت به آن ای که کوچیک ترین ممکن را دارد میرویم که ممکن است و تاحالا آن را ویزیت نکرده ایم. دقت کنید که طول کوتاه ترین مسیر از استیت اولیه به دلخواه ما را ذخیره میکند و برای هر ، 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 است که میگوید به ازای هر دو گره ای در گراف اگر شرط برقرارباشد ( تابع یعنی طول مسیری که از راس به میرود) الگوریتم 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 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 1 دارند را خروجی دهد. این از شرط پیروی نمیکند چرا که ممکن است با یک حرکت 5 خانه (خانه انتخابی و 4 خانه مجاور آن) همگی از 1 به صفر تغییر حالت دهند و این صرفا با خرج هزینه 1 عملیات انجام شده باشد. اما برای شروع بد نیست.
* در ادامه یک تابعی داریم که اتفاقا از شرط پیروی میکند. اینگونه که به ازای هر پازل، تعداد خانه های روشن آن تقسیم بر را خروجی میدهد. دلیل پیروی این تابع از شرط این است که به ازای هر دو به نام های فرض کنید تعداد خانه های روشن آنها به ترتیب باشد. حال

یعنی میزان تعداد عملیاتی که انجام میدهیم برای اینکه به برسد حداقل برابر با سقف (تفاضل تعداد خانه های روشن آنها) تقسیم بر خواهد بود. چون با هر عملیات حداکثر خانه تغییر حالت میدهند. بنابراین با تعداد کمتر به نود نمیتوانیم برسیم. حال که میدانیم به ازای هر راسی تابع ابتکاری آن عددی ثابت است پس . پس این تابع جواب بهینه را به ما میدهد.

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

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

