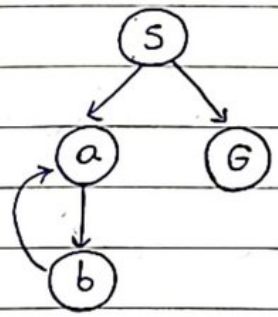
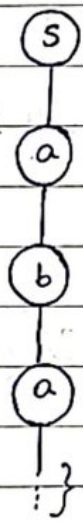
DATE / / SUBJECT :

علیرضا میرکنی ۴۰۱۱۰۶۶۱۷

۱- الف) درست. فضای حست‌وجوی مسائل زیر را در نظر بگیرید که در آن S وضعیت شروع و G وضعیت پایان می‌باشد. اگر از الگوریتم DFS برای انجام Tree Search روی این فضای حست‌وجو استفاده بکنیم، درخت حست‌وجوی حاصل دارای عمق و در نتیجه تعداد رئوس نامتناهی خواهد بود. پس مثالی یافتیم که در آن فضای حست‌وجو متناهی و درخت حست‌وجو نامتناهی می‌باشد.




فضای حست‌وجو:




درخت حست‌وجو:

عمق نامتناهی

* اگر DFS را از رأس S شروع کنیم و در ابتدا وارد رأس a شویم، در ادامه مکرراً بین رئوس a و b جابجا می‌شویم و هیچ‌گاه به رأس G نخواهیم رسید، در نتیجه درخت حست‌وجو دارای عمق و تعداد رئوس نامتناهی می‌باشد.

ب) درست. برای هر رأس (node) دلخواه باشد n در فضای حست‌وجو، اگر $h^*(n)$ برابر فاصله واقعی رأس n از نزدیک‌ترین goal باشد، چون توابع f و g قابل قبول هستند، طبق تعریف داریم:

$$f(n) \leq h^*(n) \quad \wedge \quad g(n) \leq h^*(n) \Rightarrow 0.7\, f(n) \leq 0.7\, h^*(n) \quad \wedge \quad 0.2\, g(n) \leq 0.2\, h^*(n)$$

DATE / / SUBJECT :

پس داریم: (توابع $f$، $g$ و $h^*$ همگی نامنفی فرض شده‌اند)

$$0.7\,f(n) + 0.2\,g(n) \le 0.7\,h^*(n) + 0.2\,h^*(n) = 0.9\,h^*(n) \le h^*(n)$$

در نتیجه $h^* \ge 0.7f + 0.2g$ بوده و طبق تعریف، $0.7f + 0.2g$ یک تابع قابل قبول می‌باشد.

ج) نادرست. اگر از روش انتخاب بهترین $K$ فرزند برای local beam search استفاده کنیم، در بسیاری از مواقع همه $K$ استیت ذخیره شده به یک local hill ختم (همگرا) می‌شوند. برای رفع این مشکل از روش انتخاب $K$ فرزند تصادفی استفاده می‌کنیم تا diversity (تمایز) استیت‌های ذخیره شده افزایش پیدا کند و احتمال گیر کردن در local optima کاهش یابد.

د) درست. فضای حالت زیر را در نظر بگیرید که شامل $n$ رأس (node) با شماره‌های ۱ تا $n$ می‌باشد. رأس ۱ وضعیت شروع و رأس $n$ وضعیت پایان (goal) می‌باشد.


اگر از DFS برای جست‌وجوی این فضای حالت استفاده کنیم، در $O(n)$ به رأس $n$ (goal) می‌رسیم، چرا که هر رأس دقیقاً یک بار expand می‌شود تا در نهایت به رأس $n$ برسیم. این در حالی است که اگر از IDS برای جست‌وجوی این فضای حالت استفاده کنیم، در $O(n^2)$ به رأس $n$ خواهیم رسید، چرا که رأس ۱ $n$ بار، رأس ۲ $n-1$ بار، ... و رأس $n$ ام ۱ بار expand می‌شود و در مجموع $O(n^2)$ عمل expansion داریم. $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$

DATE / / SUBJECT :

ه) نادرست. طبق مطالب تحلیل شده در کلاس و اسلایدهای درسی، می دانیم که پیچیدگی زمانی الگوریتم BFS برابر $O(b^{d+1})$ و پیچیدگی زمانی الگوریتم IDS برابر $O(b^d)$ می باشد، به طوری که b درجه انشعاب و d عمق جواب با کمترین هزینه می باشد. پس در هر دو این الگوریتم ها، اگر درجه انشعاب (b) نامتناهی باشد، ممکن است الگوریتم قادر به پیدا کردن یک جواب (در صورت وجود) نباشد و در نتیجه کامل نخواهد بود. پس برای کامل بودن الگوریتم IDS، متناهی بودن درجه انشعاب لازم است. (اگر تعداد رئوس یک عمق نامتناهی باشد هیچ iteration مربوط به آن عمق عام عی شود به goal عی رسیم)

و) درست. اگر در الگوریتم Best-first search، تابع هزینه (cost function) را برای رأس (node) دلخواه n، به صورت مقابل تعریف کنیم: $f(n) = -depth(n)$ ، در این صورت در هر مرحله رأسی (عمق رأس n) expand می شود که کوچکترین f و در نتیجه بیشترین عمق را داشته باشد که این دقیقا همان الگوریتم DFS می باشد.

ز) محیط فعالیت برای این عامل، fully observable (چرا که کل صفحه بازی را مشاهده می کند نه بخشی از آن را)، single agent (چرا که تنها یک عامل هوشمند در محیط فعالیت وجود دارد)، deterministic (چرا که عامل در هر مرحله به طور قطعی تصمیم می گیرد که چکار کند)، sequential (چرا که تصمیمات گذشته عامل هوشمند بر تصمیمات فعلی تاثیر است)، و discrete است. پس موارد ذکر شده به ترتیب درست، درست، درست، نادرست و درست هستند.

DATE / / SUBJECT :

۲- الف) DFS : در این الگوریتم اولویت expansion برای رأس با عمق بیشتر است. فرض می‌کنیم اگر در یک مرحله چند رأس با عمق یکسان وجود داشتند، اولویت expansion برای رأسی است که هزینه نوشته شده روی یالی که آن رأس را به والدش متصل می‌کند، کمتر است. با این فرض، ترتیب دیدن رئوس به صورت زیر است:

$S \xrightarrow{*} B \rightarrow G$ (به وضعیت پایان رسیدیم و توقف می‌کنیم) (در مرحله * می‌توانستیم به رئوس A و C نیز برویم)

ب) greedy : در این الگوریتم اولویت expansion برای رأسی است که کمترین heuristic را دارد.

ترتیب دیدن رئوس به صورت زیر است:

$S \xrightarrow{*} A \rightarrow E \Rightarrow G$ (به وضعیت پایان رسیدیم و توقف می‌کنیم) (در مرحله * می‌توانستیم به رأس C نیز برویم)

ج) UCS : در این الگوریتم اولویت expansion برای رأسی است که کمترین فاصله (مجموع هزینه یال‌های مسیر) را از رأس شروع دارد. دقت کنید که در این الگوریتم، فاصله رئوس از رأس شروع در هر مرحله update می‌شود، چرا که یال‌هایی که یک سر آنها رأسی است که به fringe اضافه شده، relax می‌شوند. در نتیجه ترتیب دیدن رئوس به صورت زیر خواهد بود: (برای هر رأس n، f(n) که فاصله آن از رأس شروع می‌باشد نوشته شده است)

$S \rightarrow B \rightarrow A \rightarrow E \rightarrow F \rightarrow C \rightarrow G$ (به وضعیت پایان رسیدیم و توقف می‌کنیم)
$f=0 \quad f=1 \quad f=2 \quad f=9 \quad f=10 \quad f=10 \quad f=11$

د) $A^*$ : در این الگوریتم اولویت expantion برای رأسی است که کمترین مقدار $f = h + g$ را دارد، به طوری که g هزینه‌ای است که برای رسیدن از رأس شروع به آن رأس متحمل شده‌ایم. در نتیجه با توجه به h داده شده، ترتیب مشاهده رئوس به صورت زیر خواهد بود: (برای هر رأس n، f(n) زیر آن نوشته شده است)

DATE / / SUBJECT :

$S \to A \to E \xrightarrow{*} G$ (به وضعیت پایان رسیدیم و متوقف می‌شویم) (در حلقه * می‌توانستیم به رئوس F و B نیز برویم)

$f=9 \quad f=9 \quad f=10 \quad f=11$

دقت کنید که هم در الگوریتم $A^*$ و هم در الگوریتم UCS، اگر چند راس با f کمینه وجود داشتند، به صورت تصادفی یکی را انتخاب می‌کنیم و آن را expand می‌کنیم. همچنین دقت کنید که مقدار f برای راس G در هر دو این الگوریتم‌ها برابر هزینه کلی مسیر بهینه از وضعیت شروع به وضعیت پایان می‌باشد. همچنین در الگوریتم $A^*$، داریم:

برای راس A: $g=2, h=7$ برای راس S: $g=0, h=9$

برای راس E: $g=9, h=1$ برای راس G: $g=11, h=0$

ه) BFS: در این الگوریتم اولویت expantion برای راسی است که عمق کمتری دارد. فرض می‌کنیم اگر در یک مرحله چند راس با عمق کمینه وجود داشتند، اولویت expansion برای راسی است که هزینه نوشته شده روی یالی که آن راس را به والدش متصل می‌کند، کمتر است. با این فرض، ترتیب دیدن رئوس به صورت زیر خواهد بود:

$S \to B \to A \to C \to E \to G$ (به وضعیت پایان رسیدیم و متوقف می‌شویم)

مسیر به دست آمده در هر قسمت از راس شروع به راس پایان به صورت زیر می‌باشد:

الف) SBG ب) SAEG ج) SAEG د) SAEG

ه) SBG

DATE / / SUBJECT :

۳- الف)

$fitness(x_1) = 3\times2 + 2\times1 + 8 - 3 - 2 + 9 + 2\times4 + 3\times5 = 47$

$fitness(x_2) = 3\times1 + 2\times9 + 4 - 7 - 6 + 3 + 2\times2 + 3\times0 = 19$

$fitness(x_3) = 3\times8 + 2\times9 + 2 - 3 - 4 + 5 + 2\times4 + 3\times8 = 74$

$fitness(x_4) = 3\times3 + 2\times7 + 9 - 1 - 0 + 9 + 2\times7 + 3\times7 = 75$

ب) طبق اعداد به دست آمده در بخش الف، $x_3$ و $x_4$ بیشترین fitness را بین تمامی نمونه‌ها دارند. پس عملیات crossover را بر روی این دو نمونه انجام می‌دهیم. این کار به دو طریق قابل انجام است:

① $O_1$ : نیمه اول نمونه $x_4$ + نیمه دوم نمونه $x_3$ : 37912548 ⇝ fitness = 42

② $O_2$ : نیمه اول نمونه $x_3$ + نیمه دوم نمونه $x_4$ : 89230977 ⇝ fitness = 85

همچنین $x_3$ و $x_1$، به ترتیب دومین و سومین کروموزوم از لحاظ بیشتر بودن fitness هستند در نتیجه عملیات crossover را به صورت دو نقطه‌ای بر روی این دو کروموزوم انجام می‌دهیم. این کار به دو طریق قابل انجام است:

① $O_3$ : بقیه از کروموزوم $x_1$ + CDEF از کروموزوم $x_3$ : 21234595 ⇝ fitness = 35

② $O_4$ : بقیه از کروموزوم $x_3$ + CDEF از کروموزوم $x_1$ : 89832948 ⇝ fitness = 84

ج) عملیات‌های mutation داده شده را به ترتیب از چپ به راست روی $O_1$ تا $O_4$ انجام می‌دهیم:

$O_1$ روی 72 ⇝ $O_1' = 37912598$ ⇝ fitness = 98

DATE / / SUBJECT :

$O_2$ روی ۴۷ $\rightsquigarrow$ $O'_2 =$ ۸۹۲۰۰ ۹۷۷ $\rightsquigarrow$ fitness = ۸۸

$O_3$ روی ۱۵ $\rightsquigarrow$ $O'_3 =$ ۷۱۲۳۴۵ ۹۵ $\rightsquigarrow$ fitness = ۵۰

$O_4$ روی ۷۵ $\rightsquigarrow$ $O'_4 =$ ۸۹۸۳۲۹۹۸ $\rightsquigarrow$ fitness = ۹۶

مجموع فیتنس جمعیت اولیه برابر ۲۱۵ = ۷۵ + ۷۴ + ۱۹ + ۴۷ می باشد و مجموع فیتنس جمعیت جدید ۹۶ + ۵۰ + ۸۸ + ۶۸ = ۳۰۲

می باشد. پس fitness جمعیت جدید نسبت به جمعیت اولیه بهتر شده است.

۵) بدون استفاده از عملیات mutation، امکان رسیدن به جواب بهینه وجود ندارد.

اثبات: طبق تعریف fitness، واضح است که پاسخ بهینه، کروموزوم ۹۹۹۰۰۹۹۹ می باشد. (چون که در بین fitness دزن مثبت دارند را با بزرگ ترین رقم ممکن (۹) جایگزین کردیم، و چون که در بین fitness دزن منفی دارند را با صفر جایگزین کردیم). دقت کنید که با توجه به دو نوع crossover ای که می توانیم روی جفت رشته ها انجام دهیم، دو رقم سمت چپ کروموزوم های هر جمعیتی که با شروع از این جمعیت اولیه به آنهای می رسیم زیر مجموعه دو رقم سمت چپ کروموزوم های اولیه می باشد. یعنی هر کروموزومی که بدون عملیات mutation با شروع از این جمعیت اولیه ساخته می شود، دو رقم سمت چپ متعلق به مجموعه {۲۱، ۱۹، ۸۹، ۳۷} می باشد، در حالی که در جواب بهینه دو رقم سمت چپ برابر ۹۹ می باشد. پس با شروع از این جمعیت اولیه و بدون انجام عملیات mutation، نمی توان به جواب بهینه رسید. حال با استدلال، نشان می دهیم که گزاره ★ درست می باشد:

DATE / / SUBJECT :

پایه: درستی حکم برای جمعیت اولیه واضح می باشد.

گام: فرض کنید با شروع از جمعیت اولیه، به جمعیت $G_1$ رسیده ایم که حکم برای آن برقرار می باشد. حال جمعیت $G_2$ را با استفاده از جمعیت $G_1$ تولید می کنیم. برای یک کروموزوم مانند $C$، دو رقم سمت چپ $C$ را با $f(C)$ عایش می دهیم. داریم:

$\forall C$ کروموزوم $\in G_2 \; ; \; \exists C_1 ، C_2 \in G_1 \mid C = C_1$ نتیجه اول + $C_2$ نتیجه دوم

$C = C_2$ از CDEF + $C_1$ بقیه از

در هر دو حالت بالا، خواهیم داشت:

حکم برای جمعیت $G_2$ نیز برقرار می باشد. $\Leftarrow f(C) = f(C_1) \in \{21 ، 19 ، 89 ، 37\}$ طبق فرض

پس حکم با استفاده از استقرا ثابت گردید.

DATE / / SUBJECT :

۴- الف) ① این تابع اکتشافی، قابل قبول نیست. (برای $n \geq 3$). دقت کنید که اگر حالتی از قرارگیری مهره ها را در نظر بگیریم که $n-1$ مهره در یک خانه قرار گرفته اند و یک مهره در یکی از خانه های مجاور خالی این خانه قرار گرفته است، به وضوح مقدار $h^*$ (فاصله واقعی از نزدیک ترین goal) برای این حالت برابر ۱ می باشد (چرا که با یک حرکت می توانیم آن مهره ای که باقیه مهره ها در یک خانه نیست را به آن خانه ای که $n-1$ مهره دیگر در آن قرار گرفته اند منتقل کنیم و به goal، یعنی قرارگیری هر مهره ها در یک خانه از جدول برسیم)، اما مقدار تابع اکتشافی برای این حالت برابر $n-1$ می باشد (تعداد جفت مهره هایی که در کنار یکدیگر قرار ندارند). پس برای $n \geq 3$، داریم:

$h = n-1 \geq 3-1 = 2 > 1 = h^* \Rightarrow$ تابع اکتشافی مذکور، قابل قبول نیست.

از آنجایی که قابل قبول بودن شرط لازم برای یکنوا بودن است، پس این تابع اکتشافی یکنوا هم نمی باشد.

② این تابع اکتشافی، قابل قبول است. اگر مقدار این تابع اکتشافی برای حالتی از جدول برابر صفر باشد که قطعا $h \leq h^*$ خواهد بود و شرط قابل قبول بودن برقرار است. (می دانیم مقدار $h^*$ یا فاصله واقعی از نزدیک ترین goal، همواره نامنفی است). اگر مقدار این تابع اکتشافی برای حالتی از جدول برابر ۲ باشد، این بدین معنی است که فاصله نزدیک ترین مهره ها در آن حالت بیشتر یا مساوی ۲ است. دقت کنید که در هر مرحله، هر مهره حداکثر یک خانه از مکان اولیه اش فاصله می گیرد و در نتیجه فاصله بین دو مهره دلخواه در هر مرحله، حداکثر ۲ واحد کاهش می یابد. (حداکثر یک واحد ناشی از حرکت مهره اول و حداکثر یک واحد ناشی از حرکت مهره دوم و

در مجموع حداکثر دو واحد.) پس دگر صرفا همین نزدیک ترین مهره ها را در نظر بگیریم، چون فاصله شان حداقل ۴ است و این فاصله در هر گام حداکثر ۲ تا کم می شود، به حداقل ۲ گام برای قرار دادن این دو مهره در یک خانه از جدول نیاز داریم. در نتیجه $h = 2 \geq h^*$ خواهد بود و در این حالت نیز شرط قابل قبول بودن برقرار است. در نتیجه، تابع اکتشافی داده شده، قابل قبول می باشد.

اما این تابع اکتشافی یکنوا نیست. می دانیم برای یکنوا بودن این تابع اکتشافی، باید برای هر دو حالت مجاور از جدول در گراف حالات مانند $S_1$ و $S_2$، داشته باشیم $h(S_1) - h(S_2) \leq 1$. دقت کنید که از $S_1$ به $S_2$ می رویم و هزینه این جابجایی برابر ۱ می باشد). $S_1$ را حالتی در نظر بگیرید که در آن فاصله نزدیک ترین مهره ها دقیقا برابر ۴ باشد. طبق تعریف تابع اکتشافی، $h(S_1) = 2$ می باشد. حال یکی از این دو مهره (که فاصله آنها برابر ۴ است) را به گونه ای حرکت دهید که فاصله دو مهره به ۳ کاهش یابد (فرض کنید موانع به گونه ای قرار گرفته اند که این کار امکان پذیر است). با انجام این حرکت به حالت مجاور $S_2$ می رسیم که در آن فاصله نزدیک ترین مهره ها برابر ۳ است و در نتیجه $h(S_2) = 0$ خواهد بود (طبق تعریف). پس داریم: $h(S_1) - h(S_2) = 2 > 1$ که با تعریف یکنوایی در تناقض است.

پس این تابع اکتشافی یکنوا نیست. به عنوان مثال، یک جدول ۹×۹ را در نظر بگیرید که در ۴ گوشه، در وسط ۴ ضلع (ستون ها و سطرهای بالا و پایین) و همچنین در خانه وسط آن مهره قرار گرفته است. فاصله نزدیک ترین مهره ها دقیقا برابر ۴ است و می توانیم با یک حرکت این فاصله را به ۳ برسانیم. پس یک مثال نقض هم معرفی کردیم.

DATE / / SUBJECT :

③ این تابع اکتشافی قابل قبول نیست.

| | | | |
|---|---|---|---|
| | $C_1$ | | |
| $C_4$ | $C_7$ | $C_5$ | $C_2$ |
| | $C_6$ | | |
| | $C_3$ | | |

قسمت مقابل از جدول را در نظر بگیرید. حالتی از جدول را در نظر بگیرید

که در آن $\lfloor \frac{n}{4} \rfloor$ تا از مهره‌ها در خانه $C_1$، $\lfloor \frac{n}{4} \rfloor$ تا از مهره‌ها در خانه $C_2$،

$\lfloor \frac{n}{4} \rfloor$ تا از مهره‌ها در خانه $C_3$ و باقی مهره‌ها در خانه $C_4$ قرار گرفته‌اند. مقدار تابع اکتشافی داده شده برای این حالت

برابر $h = \frac{1}{2} \times 3 + \frac{1}{2} \times 3 = 3$ خواهد بود. اما مقدار $h^*$ (فاصله واقعی از نزدیک‌ترین goal) برای

این حالت، برابر ۲ می‌باشد، چرا که در یک مرحله مهره‌های موجود در خانه‌های $C_1$ و $C_4$ را به خانه $C_7$، مهره‌های موجود

در خانه $C_2$ را به خانه $C_5$ و مهره‌های موجود در خانه $C_3$ را به خانه $C_6$ منتقل می‌کنیم. پس در مرحله بعد مهره‌های موجود در خانه

های $C_5$ و $C_6$ را به خانه $C_7$ منتقل می‌کنیم و در مجموع، در دو مرحله به goal می‌رسیم و همه مهره‌ها در خانه $C_7$ قرار می‌گیرند.

پس داریم: $h^* = 2 < h = 3$ و در نتیجه این تابع اکتشافی قابل قبول نیست. از آنجایی که قابل قبول بودن شرط

لازم برای یکنوایی است، پس این تابع اکتشافی یکنوا هم نیست.

④ این تابع اکتشافی هم قابل قبول و هم یکنوا است. از آنجایی که یکنوایی شرط کافی برای قابل قبول بودن است،

کافی است نشان دهیم که این تابع اکتشافی یکنوا است. می‌دانیم تابع $h$ یکنوا است، اگر برای هر دو حالت مجاور

از جدول در گراف حالت مانند $S_1$ و $S_2$ داشته باشیم: $h(S_1) - h(S_2) \leq 1$ (دقت کنید که از $S_1$ به $S_2$ می‌رویم و هزینه

این جابجایی برابر ۱ می‌باشد.) حال دو حالت مجاور از جدول مانند $S_1$ و $S_2$ را در نظر بگیرید.

برای جابجایی از حالت $S_1$ به حالت $S_2$، تعدادی از مهره‌ها را به خانه‌های جدیدشان منتقل می‌کنیم. دقت کنید که در طی این فرایند، برای هر جفت مهره دلخواه، فاصله افقی $(|x_i - x_j|)$ و نیز فاصله قائم $(|y_i - y_j|)$ آنها، هر کدام حداکثر ۲ واحد تغییر می‌کند (حداکثر یک واحد ناشی از حرکت مهره اول و حداکثر یک واحد ناشی از حرکت مهره دوم و در مجموع حداکثر ۲ واحد). در نتیجه در حین جابجایی بین حالات $S_1$ و $S_2$، $\max_{i,j}(|x_i - x_j|)$ و $\max_{i,j}(|y_i - y_j|)$ هر کدام حداکثر ۲ واحد و در نتیجه $\frac{1}{2}\max(\max(|x_i - x_j|), \max(|y_i - y_j|))$ که همان مقدار تابع اکتشافی است، حداکثر یک واحد تغییر می‌کند. پس داریم:

$$|h(S_1) - h(S_2)| \leq 1 \Rightarrow h(S_1) - h(S_2) \leq 1$$

این تابع اکتشافی یکنوا و در نتیجه قابل قبول است.

ب) $h_a$ برای مسئله جدید یکنوا و قابل قبول است. واضح شرایط جدیدی که به مسئله اضافه شده، سبب به کاهش $h^*$ (فاصله واقعی از نزدیکترین goal) برای هیچ حالتی نمی‌شود. به عبارت دیگر، مقدار $h^*$ برای هر حالت دلخواهی از جدول با اضافه شدن این شرایط تغییری نمی‌کند یا افزایش می‌یابد. پس همچنان شرط $h_a \leq h^*$ برای تمامی حالات برقرار می‌باشد (چرا که قبلا در مسئله اصلی برقرار بوده است) و در نتیجه $h_a$ همچنان قابل قبول می‌باشد.

DATE / / SUBJECT :

از طرف دیگر، دقت کنید که هر دو حالت مجاور از جدول مانند $S_1$ و $S_2$ در مسئله جدید، در مسئله اولیه نیز مجاور هستند، چرا که تمامی حرکاتی که در یک مرحله، مجاز به انجام آنها در مسئله جدید هستیم در مسئله اولیه نیز حرکات مجازی هستند. (در مسئله جدید در برخی مراحل، نمی توانیم برخی از مهره ها را حرکت دهیم. درحالی که در مسئله اولیه می توانیم تمامی مهره ها را در تمامی مراحل حرکت دهیم یا به دلخواه حرکت ندهیم پس اگر مهره هایی را که مجاز به حرکت دادن آنها در مسئله جدید نیستیم در مسئله اولیه به انتخاب خودمان حرکت ندهیم، حالت های مجاور متناظر مسئله جدید را در مسئله اولیه خواهیم داشت) چون $h_a$ در مسئله اولیه یکنوا بود، طبق تعریف یکنوایی خواهیم داشت $h(S_1) - h(S_2) \leq 1$ و در نتیجه مجددا طبق تعریف یکنوایی، $h_a$ برای مسئله جدید نیز یکنوا می باشد و چون یکنوایی شرط کافی برای قابل قبول بودن است، قابل قبول نیز می باشد.

DATE / / SUBJECT :

۵- الف) می توان هر حالت از این جدول را با یک ماتریس مربعی $n \times n$ نمایش داد، به طوری که هر درایه از این ماتریس برابر صفر یا یک می باشد. اگر یک درایه برابر صفر باشد، خانه متناظر به آن درایه در جدول سفید رنگ است و اگر یک درایه برابر یک باشد، خانه متناظر به آن درایه در جدول سیاه رنگ خواهد بود. با این مدل سازی، فضای حالت شامل تمامی ماتریس های مربعی $n \times n$ با درایه های دودویی می باشد. حالت اولیه، ماتریسی است که حل مسئله را از آن شروع می کنیم و در ابتدای حل سوال به ما داده می شود. حالت نهایی نیز ماتریس $O_{n \times n}$ (ماتریس مربعی $n \times n$ که تمامی درایه های آن صفر هستند) می باشد که جدول متناظر آن، جدول تمام سفید است. هر کنش را می توان با یک ۴ تایی مرتب از اعداد طبیعی به شکل $(x_1, y_1, x_2, y_2)$ $\begin{matrix} 1 \le x_1, x_2 \le n \\ 1 \le y_1, y_2 \le n \end{matrix}$ نمایش داد که معادل است با معکوس کردن رنگ خانه های موجود در زیرجدولی که خانه بالا چپ آن خانه $(x_1, y_1)$ و خانه پایین راست آن خانه $(x_2, y_2)$ است.

ضریب انشعاب برابر است با حداکثر تعداد حالات می در یک حالت که در اینجا برابر تعداد زیرجدول های جدول می باشد (چرا که به ازای هر زیرجدول، می توانیم در حالت فعلی رنگ خانه های آن زیرجدول را معکوس کنیم و به یک حالت مجاور برسیم). حال تعداد زیرجدول های یک جدول $n \times n$ را می سنجیم. در یک جدول $n \times n$، $n+1$ خط افقی و $n+1$ خط قائم داریم. اگر ۲ خط افقی و ۲ خط قائم انتخاب بکنیم، از تلاقی این ۴ خط یک زیرجدول حاصل می شود. پس تعداد کل زیرجدول ها برابر است با:

$$\text{ضریب انشعاب} = \text{تعداد زیرجدول ها} = \binom{n+1}{2}^2$$

DATE / / SUBJECT :

ب) اندازه فضای مسئله برابر تعداد کل ماتریس های برمی $n \times n$ با درایه های دودویی می باشد. $n^2$ درایه داریم که هرکدام می توانند ۰ یا ۱ باشد، پس طبق اصل ضرب، تعداد این ماتریس ها و درنتیجه اندازه فضای مسئله برابر $2^{n^2}$ می باشد، یا به عبارت دیگر $O(2^{n^2})$ است.

ج) سطری دلخواه از جدول را در نظر بگیرید. این سطر، از چپ به راست، از تعدادی بلوک سفید و تعدادی بلوک سیاه تشکیل شده است. منظور از بلوک، تعدادی خانه متوالی در سطر است که همگی هم رنگ هستند و در دو شرط زیر صدق می کنند: ① یا در سمت راست راست ترین خانه، خانه ای وجود ندارد یا خانه ای که در سمت راست راست ترین خانه قرار گرفته، با آن غیرهم رنگ است.

② یا در سمت چپ چپ ترین خانه، خانه ای وجود ندارد یا خانه ای که در سمت چپ چپ ترین خانه قرار گرفته، با آن غیرهم رنگ است.

تابع heuristic را، تعداد بلوک های سیاه در سطر اول قرار می دهیم. این heuristic، قابل قبول و یکنوا می باشد. می دانیم که یکنوا بودن، شرط کافی برای قابل قبول بودن است. پس کافی است نشان دهیم که این تابع heuristic، یکنوا می باشد. طبق تعریف یکنوایی، باید نشان دهیم برای هر دو حالت مجاور از جدول مانند $S_1$ و $S_2$، داریم: $h(S_1) - h(S_2) \leq 1$ (دقت کنید که از $S_1$ به $S_2$ می رویم و هزینه این جابجایی برابر ۱ می باشد). برای این منظور، نشان می دهیم با معکوس کردن رنگ خانه های هر زیر جدول دلخواه از جدول، تعداد این تابع اکتشافی حداکثر یک واحد تغییر می کند.

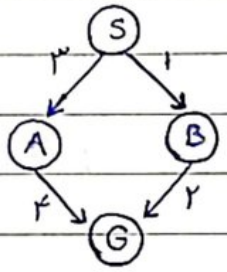
DATE / / SUBJECT :

اثبات: فرض کنید زیرجدولی از جدول را انتخاب و رنگ خانه‌های آن را معکوس می‌کنیم. قسمتی از سطر اول که در این زیرجدول قرار دارد را در نظر بگیرید. دقت کنید که اگر چند بلوک سیاه در این قسمت باشند، بعد از معکوس کردن رنگ خانه‌ها، بلوک‌های سفید بین این بلوک‌های سیاه به بلوک سیاه تبدیل می‌شوند. به عبارت دیگر برای سفید کردن K بلوک سیاه، حداقل K-1 بلوک سفید تبدیل به بلوک سیاه می‌شوند. با استدلالی مشابه، می‌توان نتیجه گرفت که برای سفید کردن K بلوک سیاه، حداکثر K+1 بلوک سفید تبدیل به بلوک سیاه می‌شوند. ★ در نتیجه اگر K تا از بلوک‌های سیاه کم شود، حداقل K-1 و حداکثر K+1 بلوک سیاه جدید ایجاد می‌شود و در نتیجه مقدار تابع اکتشافی حداکثر یک واحد تغییر می‌کند. پس این تابع اکتشافی یکنواخت و قابل قبول می‌باشد. (برای حالت پایه، می‌دانیم h برای حالت goal برابر صفر است، چرا که در حالت goal خانه سیاه نداریم)

★ K-1 بلوک سفید میان بلوک‌های سیاه و ۲ بلوک سفید در طرفین بلوک‌های سیاه

DATE / / SUBJECT :

۶- ① گراف جهت دار زیر را در نظر بگیرید که در آن S رأس شروع و G رأس goal می باشد. مقدار تابع heuristic برای رأس های مختلف این گراف نیز نوشته شده است.




$h(S) = 9$ $h(B) = 8$

$h(A) = 1$ $h(G) = 0$

در این گراف جهت دار، الگوریتم $A^*$ مسیر بهینه را پیدا می کند.

الف) مسیر بهینه، مسیر SBG می باشد که هزینه کلی آن ۳ است.

ب) الگوریتم $A^*$ در هر مرحله رأسی را expand می کند که $f = g + h$ کمتری داشته باشد، به طوری که g هزینه ای است که برای رسیدن از رأس شروع به آن رأس متحمل شده ایم. بنابراین، مسیری که الگوریتم $A^*$ در این مثال پیدا می کند، مسیر زیر خواهد بود: (برای هر رأس n، مقدار f(n) زیر آن نوشته شده است)

S → A → G (به وضعیت پایان رسیدیم و متوقف می شویم) ← الگوریتم $A^*$، مسیر SAG را پیدا می کند

$f=9$ $f=4$ $f=7$

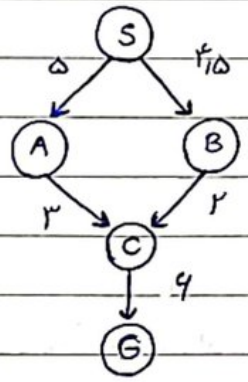
دقت کنید که در اجرای الگوریتم $A^*$، داریم:

برای رأس S: $g=0, h=9$ برای رأس A: $g=3, h=1$

برای رأس G: $g=7, h=0$

DATE / / SUBJECT :

② گراف جهت دار بدون دور زیر را در نظر بگیرید که در آن S رأس شروع و G رأس goal می باشد. مقدار تابع heuristic نیز برای رأس های مختلف این گراف نوشته شده است.




$h(S) = 9$ $h(A) = 3$

$h(B) = 5$ $h(C) = 1$ $h(G) = 0$

دقت کنید که تابع heuristic معرفی شده، قابل قبول می باشد، چرا که داریم:

$9 = h(S) \leq h^*(S) = 12,5$ $3 = h(A) \leq h^*(A) = 9$

$5 = h(B) \leq h^*(B) = 8$ $1 = h(C) \leq h^*(C) = 9$

$0 = h(G) \leq h^*(G) = 0$

الف) مسیر بهینه، مسیر SBCG می باشد که هزینه کلی آن 12,5 است.

ب) با توجه به توضیحات قسمت قبل، مسیری که الگوریتم $A^*$ در این مثال پیدا می کند، مسیر زیر است:

الگوریتم $A^*$، مسیر SACG را پیدا می کند (به وضعیت پایان رسیدم و متوقف می شویم).

$S \xrightarrow{f=9} A \xrightarrow{f=8} C \xrightarrow{f=9} G \xrightarrow{*} f=12$

دقت کنید که در مرحله *، مقدار f برای رأس B برابر 9,5 می باشد که از $f(G)$ کمتر است. اما چون رأس B همسایه ای ندارد که قبلا دیده نشده باشد، این رأس را expand نمی کنیم و رأس G را expand می کنیم.