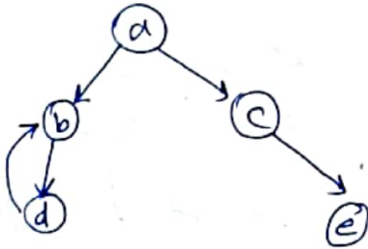


① آبله اگر در دسته باشد و tree search، آنگاه درخت جستجوی نامتناهی خواهد شد، مثلاً dfs که اولویت expand کردن با فرزند چپ باشد، مثل مثال زیر، درخت جستجوی نامتناهی خواهد داشت:



ب) بله درست است، فرض کنیم نامحدود واقعی  $h^*(n)$  باشد، آنگاه داریم:

$$\forall n: f(n) \leq h^*(n) \\ g(n) \leq h^*(n)$$

$$\Rightarrow \forall n \quad \begin{aligned} 0.1 f(n) &\leq 0.1 h^*(n) \\ 0.2 g(n) &\leq 0.2 h^*(n) \end{aligned} \rightarrow \forall n \quad 0.1 f(n) + 0.2 g(n) \leq 0.3 h^*(n)$$

که چون  $h(n)$  همواره نامتناهی است  $\Leftarrow h(n) \leq h^*(n) \Leftarrow$  است

$$\forall n \quad 0.1 f(n) + 0.2 g(n) \leq h^*(n) \rightarrow \text{هر } f \text{ و } g \text{ هم } admissible \text{ است.}$$

ج) غلط است، روش انتخاب بهترین  $k$  فرزند باعث افزایش احتمال افتادن مینیم می برای دهد و برای همین تعداد خوبی حالت رفودم با استفاده از روش  $k$  فرزند تصادفی به حالت ها اضافه می کنیم تا با احتمال کمتری مینیم می یا ماکزیم می بیفتیم.

۴) به قضای حالتی را در نظر بگیرد که ارتفاع درخت  $h$  و branching factor برابر با  $b$  باشد و هدف ما رسیدن به یک optimal goal باشد و سعی شود تو دهنای ارتفاع  $h$ ، optimal goal  
 بایزند، در این حالت dfs، در  $O(h)$ ، optimal goal، رایجی که دی

ID3، در ارد:  $1 + b + \dots + b^{h-1} + 1 = O(b^h)$

$$\hookrightarrow \frac{b^h - 1}{b - 1} + 1 \in O(b^h) \text{ یا } O(b^{h-1})$$

(۱) (۲)

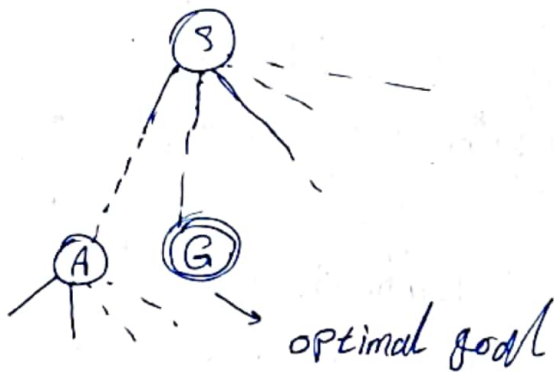
بیشتر از dfs است.

~~به~~  $h \in O(b^h)$ :

$$\log b^h = h \times \log b, \quad b \geq 2 \rightarrow \log b^h \geq h$$

$h \in O(b^h)$  چون  $h$  حتی از لگاریتم  $b^h$  هم کمتر است.

۵) غلط است، در dfs هر بار branching factor، متاهی باشد زیرا در غیر این صورت درخت زیر را در نظر بگیرد



ممکن است در مسیر dfs وارد A شویم

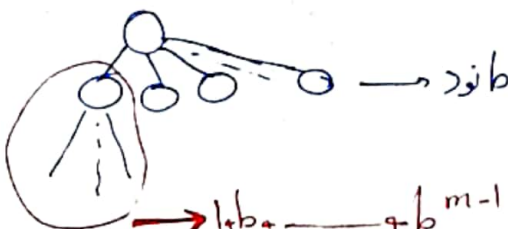
و هیچگاه از آن خارج نمی شویم چون تعداد بچه های

این نود بی شمار است و تا آخر مشغول

گشتن در بین بچه های <sup>و نودهای</sup> A و هیچگاه به

G نمی رسیم و یعنی به optimal goal نمی رسیم.

۶) به درختی با  $\text{branching factor} = b$  و  $\text{max height} = m$  در نظر بگیرد



~~اولویت~~ اولویت هارا به ترتیب عمق نسبت

بدهیم یعنی کسب عمق بیشتر دارد

اولویت بیشتری برای انتخاب شدن

به این ترتیب که  $f(n) = m - \text{depth}(n)$  باین صورت نودی که ~~اولین~~ ارتفاع بیشتری دارد باز می شود و این همان DFS است.

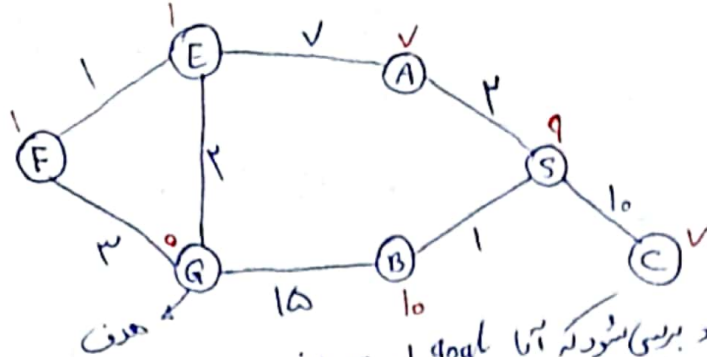
(n) fully observable: است چون در هر لحظه به نقشه ی بازی دسترسی داریم و State ما قابل مشاهده است.

single agent: است چون همواره بین agent در حال تبادل با محیط است، نود فعلی ~~همه~~ (اگر یکی که جدید می آید) در حال تبادل با محیط است. ~~همه~~

deterministic است چون هر action ای با خروجی، تناظر یک به یک دارند، یعنی هر عملیاتی که انجام دهیم ~~همیشه~~ شکل آخر را هر جایی قرار دهیم خروجی ~~همیشه~~ (State) به صورت یکتا مشخص می شود.

sequential است چون هر action ای که انجام می دهیم روی حالت های آینده و کارهای ممکن بعد تأثیر می گذارد.

discrete است زیرا اوقای بازی و حرکات گسسته است.



بافرض اینکه هر فرد موقعی که قرار است expand شود بررسی شود که آیا goal است یا خیر داریم  
 DFS، بسته به اینکه ترتیب در dfs چگونه باشد، رئوس دیده شده به ترتیب می توانند به صورت های زیر باشند  
 موقع expand کردن G همواره می خورد که goal است و expand نمی خورد

S, B, G

S, C, B, G

S, A, E, G

S, A, E, F, G

ترتیب رئوس در مسیر

S, C, A, E, G

S, C, A, E, F, G

greedy: با توجه به اینکه هر بار نودی که کمترین h را دارد انتخاب می شود، ابتدا از S شروع می کنیم و A  
 می تواند دیده شوند که باز به الگوریتم بستگی دارد پس E که کمترین h را دارد انتخاب می شود و پس G  
 انتخاب می شود یعنی به در صورت زیر رئوس می تواند باشند

S, C, A, E, G

S, A, E, G

ترتیب رئوس در مسیر

UCS: هر بار نودی که کمترین فاصله را از مجموعه داشته باشد، به مجموعه اضافه می شود  
 ابتدا B اضافه می شود، پس A اضافه می شود، پس E اضافه می شود. پس F و C هم اضافه می شوند، بسته به الگوریتم  
 چون فاصله آن برابر است به ترتیبی که الگوریتم مشخص می کند اضافه می شوند  
 در نهایت G اضافه می شود و موقع expand کردن به goal است

S, B, A, E, F, C, G

S, B, A, E, C, F, G



$$f(A) = 247 = 9$$

$$f(B) = 1910 = 11$$

$$f(C) = 1047217$$

$$f(E) = 991210$$

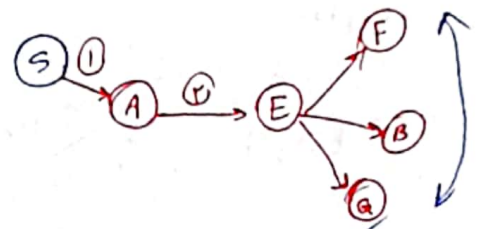
$$f(F) = 1091211$$

$$f(G) = 1190211$$

$A^*$ : ابتدا دارد A می شود و کمترین  $f$  در بین بقیه نودها دارد.

سپس وارد E می شود، حال B و F و G،  $f$  یکسانی دارند و بسته به ترتیب انتخاب برای نودهای با  $f$  مساوی وارد یکسختون می شود.

اگر دارد G شود، ~~در مرحله بعدی~~ به goal رسیده ایم ولی اگر وارد F و B در مرحله بعدی وارد G می شود، پس مسیری که به صورت زیر می تواند باشد:



S, A, E, G

S, A, E, B, G

S, A, E, F, G

S, A, E, B, F, G

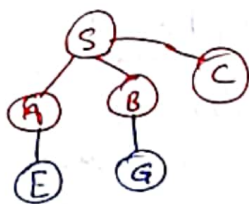
S, A, E, F, B, G

این مسیری که به ترتیب های مختلفی مشاهده شوند خوب است.

G مشاهده شود قبل از بقیه دیگر nodes و متوقف می شود.

BFS: لایه لایه جلو می رود. ابتدا نودهای A, B, C مشاهده و وارد Fringe می شوند.

پس نودهای E و G مشاهده و وارد Fringe می شوند و در نهایت مشاهده می شود که goal, G است.



S, A, B, C, E, G

مسیر!

$$f(x_1) = 2(1) + 2(1) + 1 - 2 - 2 + 9 + 2 \times 4 + 2 \times 5 = 47 \quad (3)$$

$$f(x_2) = 2(1) + 2(8) + 4 - 7 - 9 + 2 \times 2 + 2 \times 0 = 19 \quad (4)$$

$$f(x_3) = 2(1) + 2(9) + 2 - 2 - 2 + 9 + 2 \times 4 + 2 \times 1 = 41 \quad (5)$$

$$f(x_4) = 2 \times 2 + 2 \times 7 + 9 - 1 - 0 + 9 + 2 \times 7 + 2 \times 7 = 75 \quad (1)$$

دستیابی به بیشترین fitness

$$x_3 = 19234548$$

$$x_4 = 27914548$$

$$O_1 = 19230977$$

$$O_2 = 27914548$$

$$x_1 = 21133995$$

$$x_2 = 19234548$$

$$O_3 = 19132948$$

$$O_4 = 21234595$$


$$O_1 \xrightarrow{72} 19230997 \rightarrow f_1 = 2(1) + 2(9) + 2 - 2 - 0 + 9 + 2(9) + 2(7) = 49 \quad (6)$$









$$O_2 \xrightarrow{47} 27914548 \rightarrow f_2 = 2(2) + 2(7) + 9 - 1 - 4 + 5 + 2(2) + 2(8) = 57$$

$$O_3 \xrightarrow{15} 39132948 \rightarrow f_3 = 2(2) + 2(9) + 1 - 2 - 2 + 9 + 2(2) + 2(8) = 71$$

$$O_4 \xrightarrow{75} 21234515 \rightarrow f_4 = 2(2) + 2(1) + 2 - 2 - 4 + 5 + 2(1) + 2(5) = 25$$

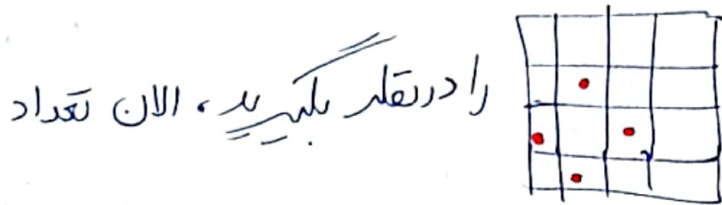
(د) به طور متوسط به fitness از جمعیت اولیه بیشتر شد است.

(۵) می دانیم که حالت  وقتی است که A و B و C و F و G و H ماکزیمم باشند یعنی  $99900999$  حالت بهینه است و D و E مینیمم باشند یعنی ۵ باشند.

با این حالت نمی توانیم بر سر نیز در هر عملیات cross over جای  یک بازه از یکی از این ها با همان بازه در یک جای جای خود و چون  اولین عنصر در هر یک از این ها  اولین عنصر در هر یک از این ها نیست اگر مثلاً اینگونه  cross over کنیم بهینه یک عنصر از  باید یک عنصر دیگر از  جای خود، یعنی اینکه اینگونه آن عنصر در  جای خود، قابل ساختن است ولی از طرفی  با عملیات cross over

رقم جدیدی ساخته نمی شود، تعداد هر عدد یکسان می ماند و ما بین  $9_1 - 9_2$ ، فقط یک کاراکتر منفرد داریم و در جواب بعضی دو کاراکتر صفر پس نمی توانیم بدل mutation به جواب بعضی برسیم.

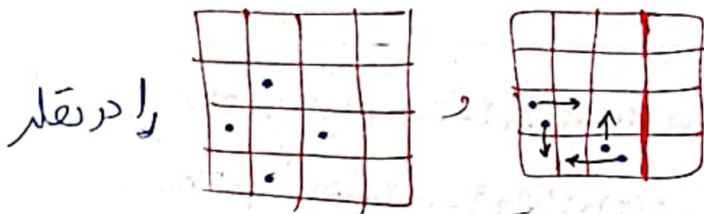
۴- اولی: admissible نیست: حالت



را در تکرار بگیرد، الان تعداد

جفت هایی که یکجا نیستند، ۲ است ولی

ما باید هر حرکت می توانیم را به خاطر وسطی ببریم و یکجا قرار گیرند. که یعنی  $h < h^*$ . بعضی دو حالت

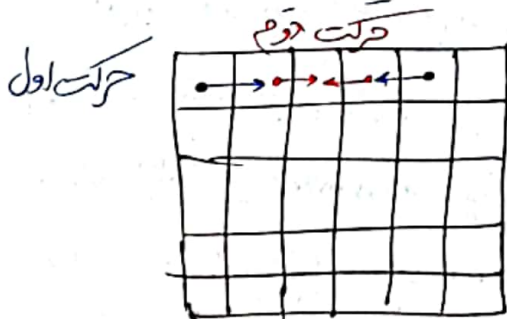


را در تکرار

بگیرد، فاصله این دو رأس ۱ است زیرا باید حرکت می صفرها همزمان حرکت می کنند مطابق فلسفها و شکل اولی به شکل دومی تبدیل می شود ولی اگر  $h$  های آنها را حساب کنیم، داریم:  $h_1 = 4$  و  $h_2 = 6$  ←  $h_2 - h_1 < \text{طول یال بین این دو حالت}$

← monotonic نیست.

دومی admissible است، زیرا اگر فاصله نزدیک ترین نودها کمتر از ۴ باشد، احتمال رسیدن به جواب با کمتر از ۲ حرکت به جواب بعضی برسیم هست مثلاً احتمال رسیدن به جواب با کمتر از ۲ حرکت،  $h$  هر در این حالت صفر است که از  $h$  واقعی کمتر است همچنین اگر فاصله نزدیک ترین نودها حداقل ۴ باشد آنگاه با کمتر از ۲ حرکت نمی توان به جواب رسید.

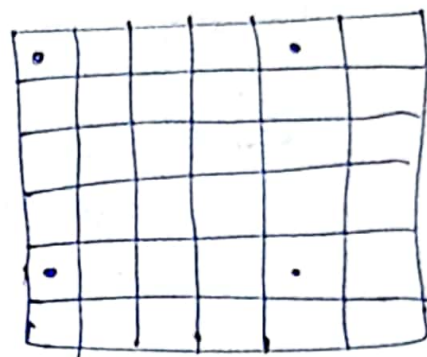
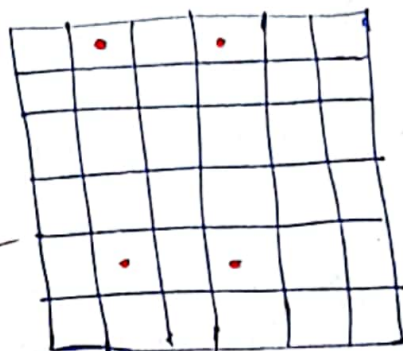


حرکت اول

خرد همی دوتا حداقل به دو حرکت نیاز دارند و بقیه که فاصله ای نزدیکتر مساوی اینجا دارند، نیز بیشتر مساوی اینجا حرکت نیاز دارند ←  $h \geq 2$  واقعی به ازای کمترین فاصله ۴



admissible بود ولی monotonic نیست دو حالت زیر را در نظر بگیرید،



این دو حالت باب علیا

حرکت قابل تبدیل به میزاند

یعنی (۵) وزنِ یالِ بیغسانِ یک است

$$h = 0$$
$$h = r$$

ولی  $\Delta h$  آنها برابر است.

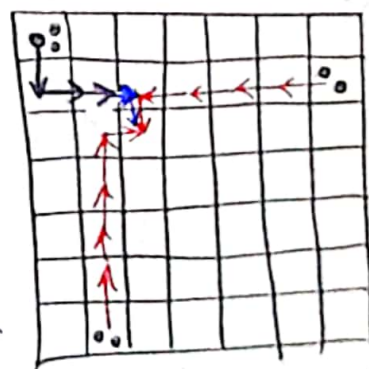
$\hookrightarrow ab \leq h \otimes \rightarrow$  ~~non monotone~~  
 $\downarrow$   $\hookrightarrow h$   $\hookrightarrow$  monotone

سوی قابل قبول نیست ← یکنواخت نیست جعلی است و قابل قبولی  
قابل قبول بودن است : این حالت را در نظر بگیرید :

$$\frac{1}{r}(\text{mcl}(x(\Delta^n))) + \frac{1}{r}(\text{mcl}(x(\Delta^y))) = q \quad \leftarrow$$

در صورتی که طی پنج مرحله می توانست تمام مهری ها را از

دربِ خانہ قرار دھیمیہ  $\leftarrow h > h^* \leftarrow$  اینِ تابع  
قابل قبول ویکٹو اینسیت.





چهارمی: این تابع  $admissible$  است، مطابق بخش قبلی،

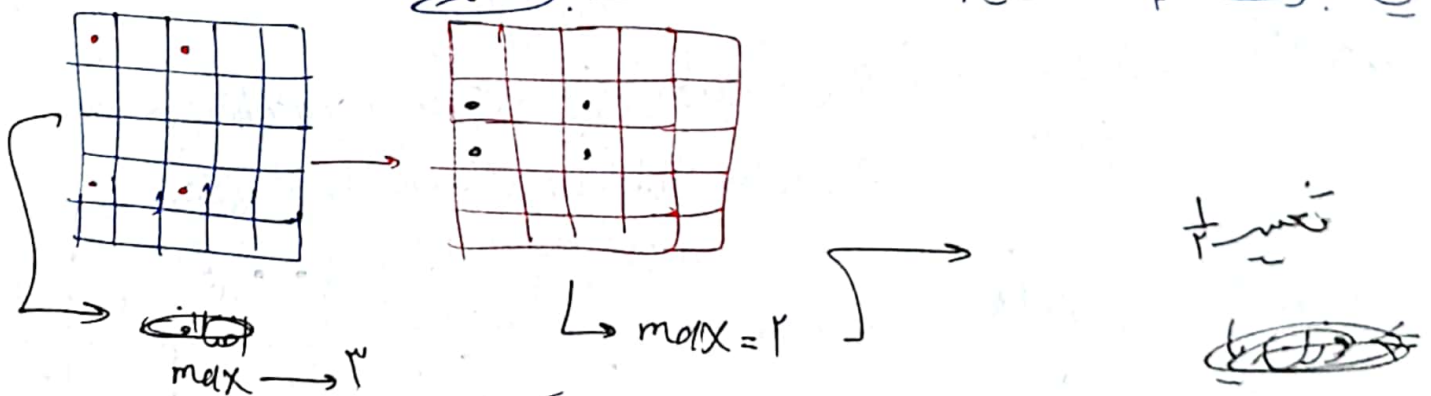
$$\frac{1}{4} \max(n_i - n_j) + \frac{1}{4} \max(y_i - y_j)$$

$admissible$  بود، یعنی کمتر مساوی فاصله واقعی بود، ماکزیمم این دو مورد هر از جهته است  
کوچکتر است یعنی کمتر مساوی از فاصله واقعی است ←

$\max(\frac{1}{4} \max(n_i - n_j), \frac{1}{4} \max(y_i - y_j))$  هم  $admissible$  است.

در مورد  $monotonic$  بودن، می‌دانیم که طی هر حرکت  $\max(\frac{1}{4} \max(n_i - n_j), \frac{1}{4} \max(y_i - y_j))$  نهایتاً بدون تغییر می‌کند زیرا دو تا خود با بیشینه اختلاف نهایت به سمت هر حرکت کند (در جهت  $n$  یا  $y$ )

و اختلاف ۲ مورد تغییر می‌کند، یک تغییر  $\frac{1}{4}$  هم داریم یعنی ماکزیمم ۱ واحد تغییر داریم در ضمن ممکن است ~~تغییر ۱ واحد داشته باشد~~ این عبارت  $\frac{1}{4}$  گاهی باید مثلاً ~~تغییر ۱ واحد داشته باشد~~



همین می‌تواند تغییر منفی واحد باشد ← در یک همواره وزن یا حال ها که  
تغییرات  $\max(\frac{1}{4} \max, \frac{1}{4} \max)$  است.

حر و احد زمانی  
 (ب) اینکه مهره ها بتوانند حرکت کنند آزادی بیشتری به ما می دهد و باعث می شود که  
 یا نمانند

واقعی ما کمتر بود چون بدون تغییر هزینه در هر نوبت می توانیم آن مهره را هر علاوه  
 بر سایرین حرکت دهیم یا نهیم. ولی اگر تعدادی از مهره ها سنگین باشند،

یعنی بعد از هر حرکت نیاز به یک نوبت استراحت دارند این باعث می شود که  $h$  واقعی نسبت به  
 حالتی که  $h$  می توانستند حرکت کنند بزرگتر مساوی شود. ضمن اینکه در حالت اول هر یک مهره می توانست  
 حرکت نکند که محال استراحت در حالت دوم است یعنی ~~در حالت دوم~~ حرکت با وجود  
 مهره سنگین زیر مجموعه ای از حرکت مهره های عادی است پس در حالت دوم

$h$  واقعی نمی تواند کمتر از حالت اول شود چون دقیقاً همان عملیات ها را می توانیم در اولی  
 هم انجام دهیم و به جای استراحت در حالت دوم هم می توانیم <sup>آن مهره را</sup> سر جای خود ~~ماند~~ نگه داریم  
 در حالت دوم ~~در حالت دوم~~ ~~در حالت دوم~~ ~~در حالت دوم~~  $h$  واقعی بزرگتر مساوی  $h$  واقعی حالت اول  
 است  $\leftarrow h_1 \leq h_2$   $\leftarrow$  طبق تعریف  $admissible$  بودن داریم

$\forall n \quad h_a(n) \leq h_1^*(n)$   
 $h_1^*(n) \leq h_2^*(n) \rightarrow h_a(n) \leq h_2^*(n)$   $\rightarrow$   $h_a$   $admissible$  است.

در مورد  $monotonic$  بودن: در ملی یک عملیات در بعضی بدون مهره سنگین می دانیم که  $h_a$   
 یکتا  $monotonic$  است، از طرفی این یعنی اینکه بین هر دو  $State$  ~~ای که~~ ~~یال داره~~  
 باشیم،  $\left[ \text{طول یال} \leq \text{تفاوت } h_a \text{ در دو } State \right]$   $\leftarrow$  طبق توضیحات بالا گفتیم که

مجموع حرکات مهره ها با وجود مهره سنگین زیر مجموعه حرکات مهره ها <sup>بدون</sup> وجود  
 مهره سنگین است ~~چون~~ تمام حالت هایی که با یک حرکت از  $State$  فعلی به آن ها  
 می رویم ~~همسایه~~ های حالت فعلی حساب می شوند و بین  $State$  فعلی و آنها یال است.

و  $h_a$  یکنوا است پس مادل تمامی این  $h_a$  ها بزرگتر مساوی  $\frac{1}{2}$  اختلاف  $h_a$  ها است. حالا اگر به تعدادی از این  $h_a$  ها  $h$  بنویسیم (بخاطر وجود مصرع‌ها سنگین) باز هم مادل تمام  $h_a$  ها  $\leq$  اختلاف  $h_a$  ها است  $\Leftarrow$

$h_a$  در این حالت هم یکنوا یا monotonic است.





۵- هر state، یک جدول  $n \times n$  است که بعضی خانه‌ها صفر و بعضی یک اند  
 هر action معادل انتخاب یک زیر جدول است که به وسیله دو نقطه‌ای آن زیر جدول مشخص می‌شود  
 حالت اولیه جدول داده شده است و  
 حالت نهایی یک جدول تمام یک است.  
 یا بالا سمت راست و پایین سمت چپ  
 یا بالا سمت چپ و پایین سمت راست  
 (فرقی ندارد به هر حال چهار تایی  $(n_1, n_2, n_3, n_4)$  می‌شود)

ضریب انتخاب هر مسئله تمام حالت‌هایی که از نات کردن یک زیر جدول به دست می‌آید یعنی برابر تعداد زیر جدول‌ها می‌باشد، معضات جدول برای  $n$  ها این است که باید دو تا  $n$  و دو تا  $n$  انتخاب کنیم  
 تعداد actions ها  
 ضریب انتخاب  $= \binom{n+1}{2} \times \binom{n+1}{2} = \binom{n+1}{2}^2$


ب) فضای مسئله هر برابر است با تعداد جدول‌های ممکن  
 که هر خانه 2 حالت دارد و سایر جدول  $n^2$  است ← فضای مسئله  
 $O(2^{n^2})$  است

۹- تابع آگنسی مان را برابر با  $\max_{\text{max}}$  دسته‌های سیاه در هر سطر و  $\max_{\text{min}}$  دسته‌های سیاه در هر ستون می‌نویسیم

(دسته‌های سیاه  $\max$  و دسته‌های سیاه  $\max$  ستون‌ها)

که دسته شامل تمام مستطیل‌هایی است که در یک ضلع مشترک اند. مانند  یا   
 به این صورت می‌دانیم که در هر سطر و هر ستون، سطرهایی در میان سیاه سفید اند.

مثلاً  یا  یا 

و در هر سطر که یک زیر جدول داریم، حداکثر یک دسته سیاه در هر سطر و هر ستون  
 که می‌شود و به این ترتیب پس حداکثر به اندازه‌ی  $n$  می‌توانیم دسته‌های سیاه  
 سطرها و ستون‌ها را تعداد دسته‌های سیاه ستون‌ها حرکت لازم داریم که یعنی  $h(n) \leq h^*(n)$   
 که یعنی  $h$  قابل قبول است، از طرفی گفتیم که برای اینکه جدول باید داشته باشیم  
 $h(n) \leq h^*(n)$ ، یا ما اینجا برابر است یعنی حالت‌هایی که بایک بار مات کردن یک  
 زیر جدول می‌توانیم آن به سیم و  می‌دانیم که در هر سطر و هر ستون یک واحد از  
 تابع  $h$  که بیش‌تر از حد لازم می‌شود چرا که زیر جدول‌هایی که انتخاب می‌کنیم تعداد دسته‌های  
 سیاه در هر سطر و هر ستونشان نهایتاً یکی بیشتر از دسته‌های سفید در آن سطر یا ستون است  
 و با مات کردن نهایتاً یک واحد کم می‌شود  $h(n) - h^*(n)$  نهایتاً می‌تواند یک واحد تغییر داشته باشد  
 که  $h(n) \leq h^*(n)$  است پس این تا یکجا درست است.

در واقع هر زیر جدولی که بگیریم، در هر سطح آن تعداد دسته‌های سیاه و سفید  
 یک واحد تفاوت دارند.

1 + سفید = سیاه  $\rightarrow$  (سیاه) (سفید) (سیاه)

سفید = سیاه  $\rightarrow$  (سیاه) (سفید) (سیاه) (سفید)

سفید = 1 + سیاه  $\rightarrow$  (سفید) (سیاه) (سفید)

درست‌ترین حالت همین است  $\leftarrow h$  در هر مرحله نهایتاً یک واحد تغییر می‌کند

~~$h(a) - h(b) \leq \alpha b$~~   $\leftarrow$  آماره  $a$  بزرگتر از  $b$  داریم  
 $h(a) - h(b) \leq \alpha b$

۱-۹.  $h^*(n) = 15$  و  $h^*(c) = 12$  و  $h^*(b) = 14$  ← قابل قبول نیست.

اول B - باز می شود

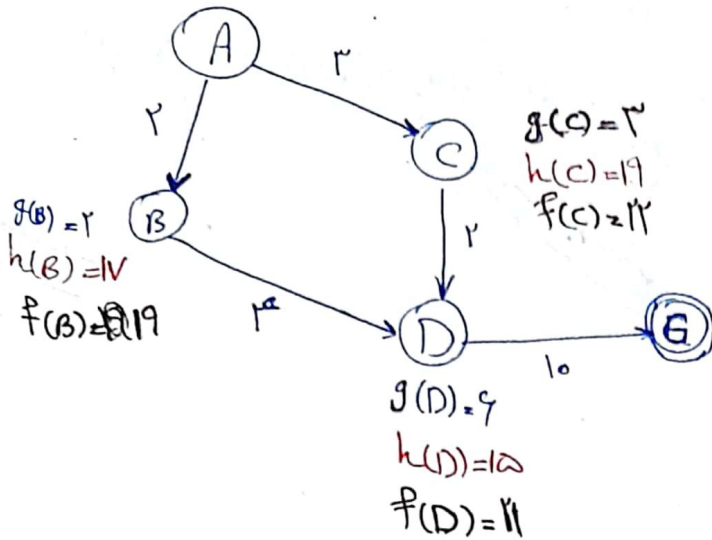
حالا  $g(D)$  برابر با ۶ می شود.

در مرحله بعدی هر D انتخاب می شود، در مرحله بعدش

به G (goal) می رسیم در صورتی که

مسیر بهینه ACDG بود ولی الان

ABDG انتخاب شده است.



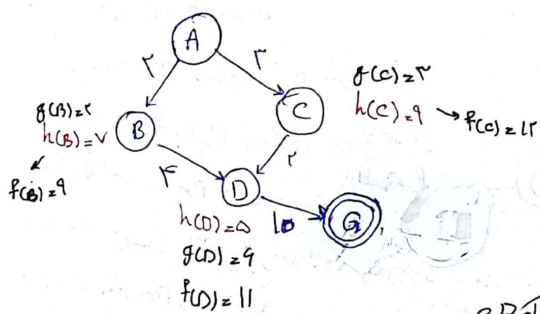
مسیر بهینه: ACDG (الف)

مسیر بهینه: ABDG (ب)



۲. شکل زیر با  $h$  های داده شده را در تکرار بکتر دیده:

$$h^*(B) = 14, h^*(C) = 12, h^*(D) = 10$$



با توجه به  $h$  های داده شده،  $h$  قابل قبول است.

ابتدا B بازی شود چون  $f$  کمتری دارد.

پس بین C و D، D بازی شود چون

$f$  کمتری دارد. در مرحله بعدی هم موقع expand کردن

~~مسیر به A\* پیدا می کند، مسیر بهینه نیست و این مسیر عبارت است از~~

A B D G

مسیری که A\* پیدا می کند، مسیر بهینه نیست و این مسیر عبارت است از

با طول ۱۶ در صورتی که مسیر بهینه A C D G با طول ۱۵ بود و اینجا جایگزین می توان بود  $h$

اتفاق افتاده است چون  $f$  باید به گونه ای بود که  $f(D) < f(C)$  باشد در صورتی که  $f(D)$

کوچکتر از  $f(C)$  شد.

$f(C)$

کمتر از

$f(G)$

است ولی با باز کردن آن چون D ویزیت شده بود الگوریتم A\* می بیند که از طریق C نمی تواند به

optimal goal

برسد و در نتیجه G را expand می کند که همان

optimal goal

هست و بدین ترتیب مسیر بهینه را پیدا نمی کند