

فرموله سازی مسئله

حالت اولیه: آرایه ای شامل چینش اولیه دانش آموزان

اعمال: رفتن به بالا، رفتن به چپ، رفتن به راست، رفتن به پایین \leq که بستگی به محل قرارگرفتن فعلی جعفر دارد.

مدل انتقال:


TargetRow => سطر فعلی جعفر

TargetCol => ستون فعلی جعفر

Cols => طول حیاط مدرسه

Rows => عرض حیاط مدرسه


UP

if targetRow > 0:
 (targetRow, targetCol)  (targetRow - 1, targetCol)


RIGHT

if targetCol < cols - 1:
 (targetRow, targetCol)  (targetRow, targetCol + 1)

DOWN

if targetRow < rows - 1:
 (targetRow, targetCol)  (targetRow + 1, targetCol)

LEFT

if targetCol > 0:
 (targetRow, targetCol)  (targetRow, targetCol - 1)

فضای حالت: چینش‌های مختلف دانش‌آموزان در حیاط

$(m \times n)!$

آزمون هدف: چک کردن کلاس‌های دانش‌آموزان هر سطر و اینکه این کلاس‌ها در یک سطر همه با هم یکسان باشند و اینکه به شکل نزولی سورت شده باشد و #
در سمت چپ یکی از سطرها باشد.

تابع هزینه مسیر: تعداد اعمال اجرا شده

نحوه پیاده سازی الگوریتم IDS

محقق اولیه شروع الگوریتم به عنوان یک مسئله می شود و در الگوریتم IDS، با شروع از محقق اولیه ابتدا الگوریتم DFS با محدودیت محقق اولیه می شود. یعنی جستجوی اول عمق انجام می شود و در هر مرحله پس از انجام الگوریتم بران پس آن، یک قدم عقب می کشد تا به حالتی برسد که با محقق اولیه راه شده قرار دارند به نوبت از مقدار می شود که نویسنده می تواند در هر مرحله، تابع $Goal$ را فراخواند، می شود که از بین هدف در آن انجام شود و ببینیم که به هدف رسیده یا نه و اگر تا محقق رسید و به هدف نرسیده بود، پس راه دیگر پیدا کرده و حالت آن محقق را یک واحد افزایش می دهیم و با محدودیت عمق محدود به حالتی واحد افزایش یافته امان به کار می گیریم و این افزایش محقق یک یک تا بزرگترین جواب در رسیدن به یک واحد هدف امان می یابد.

و برای محدود کردن راه حل یک تست که بدان هر نور راه حل محدود می شود، استفاده کردیم و هم چنین چون به $expand$ کردن یک نور که نور هدف نباشد پس همه آن را تولید می کنیم و او نیز پس از محدود کردن هم تا پس از آن محقق در رسیدن به محقق به پسین بعدی می رسیم پس اول محقق بر صبح راه شد و جستجو DFS است در هر مرحله.

$branching\ factor$ در این مسئله ۴ است و محقق راه حل d (استاندارد است چون من روی مثال هایی که با اینها مثال حرکت زدیم روی حالت هدف ایستاده اند) الگوریتم را تست کردیم پس همیشه جواب می دهد که حل است و هم چنین چون هزینه برای امان پس بچینه هم هست.

از این نظر به نظر می رسد که به علت اینکه الگوریتم به تولید چندباره و مجبور حالات می پردازد، هزینه برای تست تولید نهادهای $expand$ شده و تولید شده زیاد می کنند.

$$N(IDS) = (d)b + (d-1)b^2 + \dots + (1)b^d = O(b^d)$$

از این نظر به نظر می رسد که به علت اینکه الگوریتم به تولید چندباره و مجبور حالات می پردازد، هزینه برای تست تولید نهادهای $expand$ شده و تولید شده زیاد می کنند.

از این نظر به نظر می رسد که به علت اینکه الگوریتم به تولید چندباره و مجبور حالات می پردازد، هزینه برای تست تولید نهادهای $expand$ شده و تولید شده زیاد می کنند.

مثلاً جستجو اول عمق $O(bd)$ است.

نحوه پیاده سازی الگوریتم Bidirectional BFS

در جستجوی دو طرفه اول سطح، چون باید از هر دو جهت به سمت هدف حرکت کنیم، پس برای پیدا کردن هم حالت هدف نهایی که با رفتن از هر یک از این دو جهت به هدف منتهی شود، باید از هر دو جهت به سمت هدف حرکت کنیم. به همین دلیل، در این الگوریتم، دو جهت به سمت هدف حرکت می‌کنیم. اول به سمت جلو (forward) و اول به سمت عقب (backward). در هر مرحله، دو فرانتیئر (Frontier) داریم: یکی برای جلو (forward) و یکی برای عقب (backward). این دو فرانتیئر را به صورت دو صف (queue) قرار می‌دهیم: $queue F$ و $queue B$. هر چه به جلو حرکت می‌کنیم، به $queue F$ اضافه می‌کنیم و هر چه به عقب حرکت می‌کنیم، به $queue B$ اضافه می‌کنیم. وقتی یکی از این دو صف خالی شود، یعنی به هدف رسیده ایم. این روش، زمانی که مسافت بین دو نقطه کوتاه باشد، بسیار سریع‌تر از روش معمولی است.

چون داده‌ها حالت هشدار BFS کامل است، پس الگوریتم کامل است و چون تابع هزینه معتدله را نمی‌بینیم، پس به عنوان هر عمل است، پس هزینه هم هست. چون تقریباً به هر دو طرف می‌رویم، پس به هر دو طرف می‌رویم و به هر دو طرف می‌رویم. به همین دلیل، در این الگوریتم، دو جهت به سمت هدف حرکت می‌کنیم. اول به سمت جلو (forward) و اول به سمت عقب (backward). در هر مرحله، دو فرانتیئر (Frontier) داریم: یکی برای جلو (forward) و یکی برای عقب (backward). این دو فرانتیئر را به صورت دو صف (queue) قرار می‌دهیم: $queue F$ و $queue B$. هر چه به جلو حرکت می‌کنیم، به $queue F$ اضافه می‌کنیم و هر چه به عقب حرکت می‌کنیم، به $queue B$ اضافه می‌کنیم. وقتی یکی از این دو صف خالی شود، یعنی به هدف رسیده ایم. این روش، زمانی که مسافت بین دو نقطه کوتاه باشد، بسیار سریع‌تر از روش معمولی است.

نحوه پیاده سازی الگوریتم A^* و علت قابل قبول بودن هیوریستیک آن

علت قابل قبول بودن هیوریستیک

\Leftarrow در الگوریتم A^* ، هیوریستیک را با این شرط انتخاب کردیم.
 اگر سندها جهت تبدیل نیمه به هر دانشی باشد جایش را در آنجا نگه داریم خود جایگزین سندها "جفر"،
 مثل هیوریستیک فاصله منحنی بران شده $n - \text{path}$ می شود. چون این هیوریستیک هزینه تبدیل را
 است پس قابل قبول است زیرا هر قدر در صراط فاصله منحنی را جای خود در هر cost و لکس و لکس یکم در بران
 یک cost و لکس خاص خاص این فاصله ها بران هم اولیای هم جمع کردیم و این کار را به نحوی cost و لکس ها انجام
 داریم و مجموع $m+n$ را به عنوان هیوریستیک آن برداریم، پس چون این راه حل بران مسئله تبدیل را بهینه است پس
 به همین شرط چون هزینه هر نور را به طریقی خوش بینانه تخمین می زنیم و در این هزینه واقعی است پس بران می شود
 اصل هیوریستیک قابل قبول است.

برای انجام راه حل این الگوریتم برای هر نور که در صف قرار دارد، ابتدا مقدار تابع انتخاب را برابر با هزینه از
 شروع تا آن نور + هیوریستیک آن است را حساب کردیم و در واقع صف اولویت بران صف نور هاست
 فصل Frontier ای در بران نور که محبوب ترین نور است که مجموع cost و هزینه min آن
 است را انتخاب کردیم و آن را در هر صف به نام open و به نام closed شروع cost و لکس بودن را بران
 آن حد کردیم و اگر cost و لکس بود الگوریتم تمام می شود و در این کار را بهینه می کنند.
 چون $\text{cost} < C^*$ می باشد پس cost و لکس کامل است و
 در بران به نام closed ها را به حقیقت نگه داریم تا هیوریستیک را حساب کنیم و محبوب را انتخاب کنند،
 از این طریق چیکاری قضایا محقق می شود و برای هر نور
 بهینه است و تمایز و هم تمایز را نسبت به سایر روش ها به هم می دهد.

مقایسه این سه الگوریتم در سه مثال

EXAMPLE 1

Input : [['17c', '11c', '8c', '4c'], ['4b', '8d', '3b', '3b'], ['10a', '6d', '6a', '5d'], ['5b', '#', '12a', '1a']]

A*	Bidirectional BFS	IDS
depth: 10 Right Up Left Left Down Right Up Up Left Down Num of expanded nodes: 16 Num of produced nodes: 51	found in forward depth: 10 Right Up Left Left Down Right Up Up Left Down Num of expanded nodes: 437 Num of produced nodes: 992	Enter initial depth: 0 depth: 10 Right Up Left Left Down Right Up Up Left Down Num of expanded nodes: 18348 Num of produced nodes: 18360

EXAMPLE 2

Input : [['17c', '8c', '3b', '4c'], ['8d', '11c', '6d', '3b'], ['#', '5b', '6a', '5d'], ['4b', '12a', '10a', '1a']]

A*	Bidirectional BFS	IDS
depth: 14 Down Right Right Up Up Up Left Down Down Left Up Right Down Left Num of expanded nodes: 42 Num of produced nodes: 133	found in forward depth: 14 Down Right Right Up Up Up Left Down Down Left Up Right Down Left Num of expanded nodes: 2259 Num of produced nodes: 4947	Enter initial depth:0 depth: 14 Down Right Right Up Up Up Left Down Down Left Up Right Down Left Num of expanded nodes: 426420 Num of produced nodes: 426436

EXAMPLE 3

Input : [['17c', '11c', '4c', '3b'], ['4b', '3b', '8c', '#'], ['5b', '8d', '6d', '5d'], ['12a', '10a', '6a', '1a']]

A*	Bidirectional BFS	IDS
depth: 6 Up Left Down Left Left Down Num of expanded nodes: 7 Num of produced nodes: 20	found in forward depth: 6 Up Left Down Left Left Down Num of expanded nodes: 53 Num of produced nodes: 152	Enter initial depth:0 depth: 6 Up Left Down Left Left Down Num of expanded nodes: 529 Num of produced nodes: 537

EXAMPLE 4

Input : [['8c', '3b', '6d', '4c'], ['17c', '4b', '3b', '5d'], ['5b', '11c', '8d', '1a'], ['12a', '10a', '6a', '#']]

A*	Bidirectional BFS	IDS
depth: 16 Up Up Left Up Left Left Down Right Down Right Up Up Left Down Left Down Num of expanded nodes: 141 Num of produced nodes: 456	found in backward depth: 16 Up Up Left Up Left Left Down Right Down Right Up Up Left Down Left Down Num of expanded nodes: 4178 Num of produced nodes: 8969	Enter initial depth:0 depth: 16 Up Up Left Up Left Left Down Right Down Right Up Up Left Down Left Down Num of expanded nodes: 789961 Num of produced nodes: 789979

$A^* < \text{Bidirectional BFS} < \text{IDS}$

تعداد گره‌های تولید شده :

$A^* < \text{Bidirectional BFS} < \text{IDS}$

تعداد گره‌های بسط داده شده :

$A^* = \text{Bidirectional BFS} = \text{IDS}$

عمق جواب :