به نام خدا

مقایسه الگوریتم های اول سطح(BFS) و اول عمق (DFS) در مسأله 8-وزیر

استاد: آقای دکتر فیضی درخشی

تهیه کننده گزارش: امیر محسن یوسفی واقف

فهرست مطالب

مقدمه

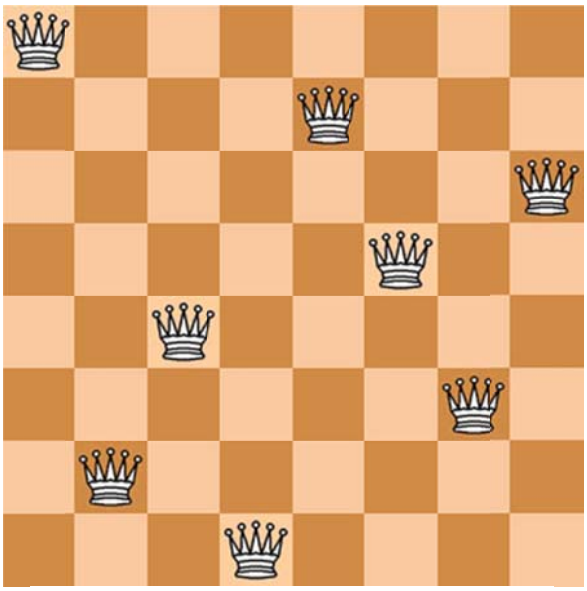
صورت مسئله : هشت وزير را در هشت خانه شطرنج (8*8) طوري قرار دهيد كه هيچكدام يكديگر را تهديد نكنند. وزير در خانه هاي شطرنج به صورت عرضي،طولي و قطري مي تواند حركت كند. اين مسئله قابل تعميم به مسئله N وزير در يك شطرنج N*N است.

تاريخچه: اين مسئله در سالي 1848 توسط شطرنج بازي به نام Max Bezzel عنوان شد و رياضي دانان بسياري ازجمله Gauss و Georg Cantor بر روي اين مسئله كار كرده و در نهايت آنرا به N وزير تعميم دادند. اولين راه حل توسط Franz Nauck در سال 1850 ارائه شد كه به همان مسئله N وزير تعميم داده شد. پس از آن Gunther راه حلي با استفاده از دترمينان ارائه داد كه J.W.L. Glaisher آنرا كامل نمود.

در سال 1979 ، Edsger Dijkstra با استفاده از الگوريتم عقب گرد اول عمق اين مسئله را حل كرد.

حال ما در اين پروژه می خواهيم اين مسأله را با روشی غير هوشمند كه عبارتست از توليد و تست تمامی حالات ممكن با دو الگوريتم رياضی و غيرهوشمند جستجوی اول عمق و جستجوی اول سطح عمق انجام دهيم و نتيجه حاصله از آنها را به معرض مقايسه بگذاريم.(شكل1)

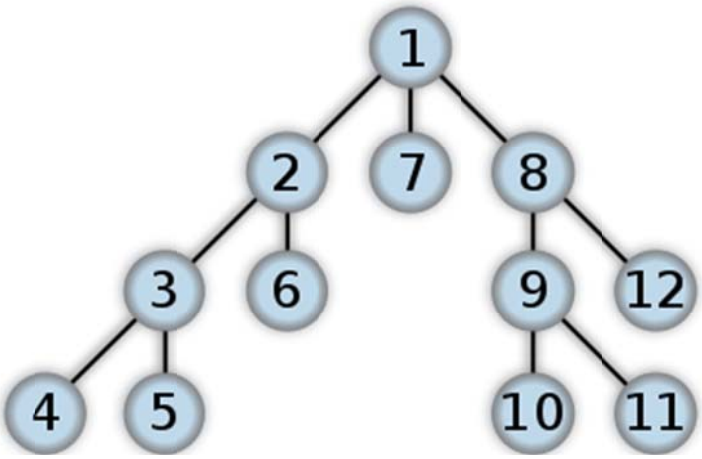
راه حل: مسئله 12 راه حل يكتا دارد كه با در نظر گيري تقارن و چرخش به 92 حالت قابل تبديل است. بايد تكنيكهايي جهت كاهش حالات ،روش Brute Force يا امتحان تك تك جواب ها انجام شود. تعداد همه حالاتي كه مي تواند در روش Brute Force چك شود برابر 16,777,216 يا هشت به توان هشت است!



جستجوی اول عمق (DFS)

در نظریهٔ گراف، جستجوی عمق اول (به انگلیسی: Depth-first Search، به‌اختصار DFS) یک الگوریتم پیمایش گراف است که برای پیمایش یا جستجوی یک درخت یا یک گراف به کار می‌رود.

استراتژی جستجوی عمق اول برای پیمایش گراف، همانطور که از نامش پیداست "جستجوی عمیق‌تر در گراف تا زمانی که امکان دارد" است. (شکل2)



چگونه کار می‌کند؟

الگوریتم از ریشه شروع می‌کند (در گراف‌ها و یا درخت‌های بدون ریشه راس دلخواهی به عنوان ریشه انتخاب می‌شود) و در هر مرحله همسایه‌های رأس جاری را از طریق یال‌های خروجی رأس جاری به ترتیب بررسی کرده و به محض روبه‌رو شدن با همسایه‌ای که قبلاً دیده نشده باشد، به صورت بازگشتی برای آن رأس به عنوان رأس جاری اجرا می‌شود. در صورتی که همهٔ همسایه‌ها قبلاً دیده شده باشند، الگوریتم عقب‌گرد می‌کند و اجرای الگوریتم برای رأسی که از آن به رأس جاری رسیده‌ایم، ادامه می‌یابد. به عبارتی الگوریتم تا آنجا که ممکن است، به عمق بیشتر و بیشتر می‌رود و در مواجهه با بن بست عقب‌گرد می‌کند. این فرایند تامادامیکه همهٔ رأس‌های قابل دستیابی از ریشه دیده شوند ادامه می‌یابد.

همچنین در مسائلی که حالات مختلف متناظر با رئوس یک گراف‌اند و حل مسئله مستلزم یافتن رأس هدف با خصوصیات مشخصی است، جستجوی عمق اول به صورت غیرخلاق عمل می‌کند. بدین‌ترتیب که هر دفعه الگوریتم به اولین همسایهٔ یک رأس در گراف جستجو و در نتیجه هر دفعه به عمق بیشتر و بیشتر در گراف می‌رود تا به رأسی برسد که همهٔ همسایگانش دیده شده‌اند که در حالت اخیر، الگوریتم به اولین رأسی بر می‌گردد که

همسایهٔ داشته باشد که هنوز دیده نشده باشد. این روند تا جایی ادامه می‌یابد که رأس هدف پیدا شود و یا احتمالاً همهٔ گراف پیمایش شود. البته پیاده‌سازی هوشمندانهٔ الگوریتم با انتخاب ترتیب مناسب برای بررسی همسایه‌های دیده نشدهٔ رأس جاری به صورتی که ابتدا الگوریتم به بررسی همسایه‌ای بپردازد که به صورت موضعی و با انتخابی حریصانه به رأس هدف نزدیک‌تر است، امکان‌پذیر خواهد بود که معمولاً در کاهش زمان اجرا مؤثر است.

از نقطه نظر عملی، برای اجرای الگوریتم، از یک پشته (stack) استفاده می‌شود. بدین ترتیب که هر بار با ورود به یک رأس دیده نشده، آن رأس را در پشته قرار می‌دهیم و هنگام عقب‌گرد رأس را از پشته حذف می‌کنیم. بنابراین در تمام طول الگوریتم اولین عنصر پشته رأس در حال بررسی است. جزئیات پیاده‌سازی در ادامه خواهد آمد.

وقتی در گراف‌های بزرگی جستجو می‌کنیم که امکان ذخیرهٔ کامل آنها به علت محدودیت حافظه وجود ندارد، در صورتی که طول مسیر پیمایش شده توسط الگوریتم که از ریشه شروع شده، خیلی بزرگ شود، الگوریتم با مشکل مواجه خواهد شد. در واقع این راه‌حل ساده که "رئوسی را که تا به حال دیده‌ایم ذخیره کنیم" همیشه کار نمی‌کند. چراکه ممکن است حافظهٔ کافی برای این کار نداشته باشیم. البته این مشکل با محدود کردن عمق جستجو در هر بار اجرای الگوریتم حل می‌شود که در نهایت به الگوریتم تعمیق تکراری (Iterative Deepening) خواهد انجامید.

الگوریتم جستجو

پیمایش با انتخاب رأس r به عنوان ریشه آغاز می‌شود. r به عنوان یک رأس دیده شده برچسب می‌خورد. رأس دلخواه r1 از همسایگان r انتخاب شده و الگوریتم به صورت بازگشتی از r1 به عنوان ریشه ادامه می‌یابد.از این پس در هر مرحله وقتی در رأسی مانند v قرار گرفتیم که همهٔ همسایگانش دیده شده‌اند،اجرای الگوریتم را برای آن رأس خاتمه می‌دهیم. حال اگر بعد از اجرای الگوریتم با ریشهٔ r1 همهٔ همسایگان r برچسب خورده باشند، الگوریتم پایان می‌یابد. در غیر این صورت رأس دلخواه r2 از همسایگان r را که هنوز برچسب نخورده انتخاب می‌کنیم و جستجو را به صورت بازگشتی از r2 به عنوان ریشه ادامه می‌دهیم. این روند تامادامیکه همهٔ همسایگان r برچسب نخورده‌اند ادامه می‌یابد.

شبه کد این الگوریتم جستجو به صورت زیر می باشد:

```
1-  Un=[s]; Ex=[];
2-  N = Head(Un);
3-  Un = Un – [n]
```

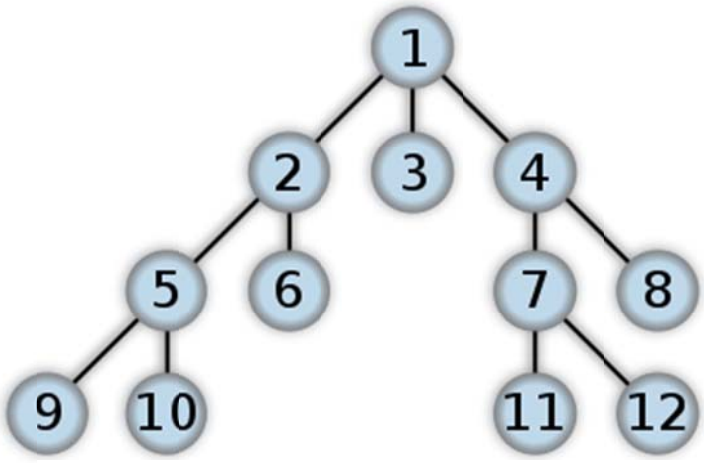
4- If N=G then exit('Success')
5- If N don't be in Ex then
    a. Ex = Ex + [n]
    b. Expand n using operators and produce its children (x1,…,xk)
    c. Add x1,….,xk to the front of Un=[x1,…,xk] + Un
6- Goto 2

پیچیدگی زمان اجرای الگوریتم O(b^m) و مرتبه حافظه مصرفی اجرای الگوریتم O(mb) می باشد که b فاکتور انشعاب و m حداکثر عمق درخت می باشد. لازم به ذکر است که دقت نمائید زمان اجرای الگوریتم نمائی و حافظه مورد نیاز آن به صورت خطی می باشد.

جستجوی اول سطح (BFS)

در نظریهٔ گراف، جستجوی اول سطح (به انگلیسی: Breadth-first Search، به‌اختصار: BFS) یکی از الگوریتم‌های پیمایش گراف است.

استراتژی جستجوی سطح اول برای پیمایش گراف، همانطور که از نامش پیداست «جستجوی سطح به سطح گراف» است.(شکل3)



چگونه کار می‌کند؟

الگوریتم از ریشه شروع می‌کند (در گراف‌ها و یا درخت‌های بدون ریشه رأس دلخواهی به عنوان ریشه انتخاب می‌شود) و آن را در سطح یک قرار می‌دهد. سپس در هر مرحله همهٔ همسایه‌های رئوس آخرین سطح دیده شده را که تا به حال دیده نشده‌اند بازدید می‌کند و آنها را در سطح بعدی می‌گذارد. این فرایند زمانی متوقف می‌شود که همهٔ همسایه‌های رئوس آخرین سطح قبلاً دیده شده باشند. همچنین در مسائلی که حالات مختلف متناظر با رئوس یک گراف‌اند و حل مسئله مستلزم یافتن رأس هدف با خصوصیات مشخصی است که در عین حال در بین همهٔ رئوس هدف با آن خصوصیات به ریشه نزدیک‌ترین باشد، جستجوی سطح اول به صورت غیرخلاق عمل می‌کند. بدین ترتیب که الگوریتم هر دفعه همهٔ همسایه‌های یک رأس را بازدید کرده و سپس به سراغ رأس بعدی می‌رود و بنابراین گراف سطح به سطح پیمایش خواهد شد. این روند تا جایی ادامه می‌یابد که رأس هدف پیدا شود و یا احتمالاً همهٔ گراف پیمایش شود. براساس آنچه گفته شد پیاده‌سازی هوشمندانهٔ الگوریتم آنقدر مؤثر نخواهد بود.

از نقطه نظر عملی، برای پیاده‌سازی این الگوریتم از صف استفاده می‌شود. بدین ترتیب که در ابتدا ریشه در صف قرار می‌گیرد. سپس هر دفعه عنصر ابتدای صف بیرون کشیده شده، همسایگانش بررسی شده و هر همسایه‌ای که تا به حال دیده نشده باشد به انتهای صف اضافه می‌شود. جزئیات پیاده‌سازی در ادامه خواهد آمد.

الگوریتم جستجو

پیاده‌سازی این الگوریتم مشابه پیاده‌سازی جستجوی عمق اول است با این تفاوت که به جای پشته از صف استفاده می‌شود.

شبه کد این الگوریتم جستجو به صورت زیر می باشد:

7- Un=[s]; Ex=[];
8- N = Head(Un);
9- Un = Un – [n]
10- If N=G then exit('Success')
11- If N don't be in Ex then
    a. Ex = Ex + [n]
    b. Expand n using operators and produce its children (x1,…,xk)
    c. Add x1,….,xk to the end of Un= Un + [x1,…,xk]
12- Goto 2

پیچیدگی زمان اجرای الگوریتم O(b^d) و مرتبه حافظه مصرفی اجرای الگوریتم O(b^d) می باشد که b فاکتور انشعاب و d عمق کم عمق ترین جواب می باشد. لازم به ذکر است که دقت نمائید زمان اجرای الگوریتم نمائی و حافظه مورد نیاز آن نیز به صورت نمایی می باشد.

مقایسه دو الگوریتم جستجو

آمار بدست آمده از اجرای الگوریتم اول عمق(DFS) بر روی مسأله 8 وزیر عبارتست از:

Start time is: 12/4/2010 11:20:06 AM
End time is: 12/4/2010 11:20:10 AM
Elapsed time for execution is: 00:00:03.2651868
Number of made node: 1485577
Number of read node: 1485549
Max number of nodes in Un: 57
Answer node is: 15863724

آمار بدست آمده از اجرای الگوریتم اول سطح(BFS) بر روی مسأله 8 وزیر عبارتست از:

Start time is: 12/4/2010 11:29:20 AM
End time is: 12/4/2010 11:29:43 AM
Elapsed time for execution is: 00:00:22.5902920
Number of made node: 19173961
Number of read node: 3696597
Max number of nodes in Un: 16777216
Answer node is: 15863724

با توجه به زمان اجرای الگوریتم در هر دو روش مشاهده می کنیم که زمان الگوریتم اول سطح تقریباً 7 برابر زمان الگوریتم اول عمق می باشد که علت آن مواجه با تمامی جواب های الگوریتم در آخرین عمق و در نیمه چپ درخت می باشد که الگوریتم اول سطح مجبور می باشد برای یافتن جواب تمامی سطوح را تا لایه آخر مورد تست قرار دهد ولی الگوریتم اول عمق می بایستی هر بار تا عمق درخت برود و به همین ترتیب به جستجو ادامه دهد که با توجه به وجود برخی جواب ها در نیمه چپ این الگوریتم سریعتر به جواب می رسد.

همچنین با دقت در بیشترین تعداد عناصر در Un که ما در اینجا به عنوان حافظه مصرفی آن را مورد توجه قرار می دهیم مشاهده می کنیم که مرتبه حافظه مصرفی الگوریتم اول عمق بسیار کمتر از الگوریتم اول سطح می باشد و تفاوت مرتبه خطی و نمایی دقیقاً در اینجا قابل مشاهده می باشد.

با توجه به نود جواب درمی یابیم که هر دو جواب یکی است زیرا تمامی جواب ها در عمق آخر درخت کامل می باشد و هردو الگوریتم عمق آخر را از چپ به راست پیمایش می کنند، بنابراین هر دو به اولین جواب می رسند که 1،5،8،6،3،7،2،4 می باشد (منظور آن این است که اولین وزیر در سطر 1 ستون 1 می باشد، دومین وزیر در سطر2 ستون5 می باشد، سومین وزیر در سطر3 ستون8 می باشد و.....) و چون هر دو الگوریتم اولین چواب را مد نظر قرار داده و از برنامه خارج می شوند جواب هردو الگوریتم فقط همین جواب و یکتا می باشند.

مراجع

1- دانشنامه آزاد ویکیپدیا