طراحی الگوریتم: پیشرفتهای روشمند

حسین رادمرد

۱۳ اردیبهشت ۱۴۰۳

۱.۰ مسئله ۸۳ رنگ کردن یک گراف دو رنگی

در این مسئله، مانند موارد بعدی، حل مسئله هدف ماست و بهینه بودن جواب برای ما حائز اهمیت نیست. دراینجا، ما الگوریتمی را برای رنگ کردن گراف با دردست داشتن تنها دو رنگ، بررسی میکنیم. نسخه‌ی کلی‌تر(رنگ آمیزی با هر تعداد رنگ) در مسئله‌ی ۵۹، صفحه‌ی ۲۳۶ بررسی میشود. گرچه، نسخه‌ی فعلی بسیار بهینه‌تر است. در ادامه، این بحث بسیار به موضوع ''گرافهای پرکاربرد: گراف‌های دوبخشی'' مرتبط است.

گراف همبند بدون جهت $(G = N, V)$ (ناتهی) به ما داده شده‌ست. ما قصد داریم با رنگ‌های سیاه و سفید گراف را رنگ آمیزی کنیم به گونه‌ای که هیچ دو راس همسایه‌ای دارای رنگ یکسانی نباشند. چنین گرافی را گراف دورنگی مینامیم. الگوریتم حریصانه ای که ما قصد ساخت آن را برای این منظور داریم به پیمایش سطری گراف ها مرتبط است که آن را در ابتدا بررسی کردیم.

چنین گرافی را گراف دورنگی مینامیم. الگوریتم حریصانه ای که ما قصد ساخت آن را برای این منظور داریم به پیمایش سطری گراف ها مرتبط است که آن را در ابتدا بررسی کردیم.

پیمایش سطری گراف: یادآوری

معرفی

اول از همه، اجازه دهید مفاهیم ''فاصله‌ی میان دو راس'' و ''پیمایش سطری'' را برای گراف‌های همبندِ بدون جهت تعریف کنیم.

تعریف ۱۰ (فاصله‌ی میان دو راس): در نظر میگیریم، $G = (N, V)$ یک گراف همبند بدون جهت و s و 's دو راس این گراف باشند. طول کوتاه‌ترین مسیر میان s و 's را فاصله‌ی میان s و 's گویند.

تعریف ۱۱ (جستجوی سطری): فرض کنیم G یک گراف همبند و بدون جهت و s یکی از راس های آن باشد. هر فرایندی که با افزایش فاصله‌ها از راس s با راس های گراف G برخورد میکند به عنوان پیمایش سطری گراف G از s شناخته میشود.

از نمودار (b) در شکل ۸.۷ صفحه‌ی ۳۶۳، میتوانیم نتیجه بگیریم که لیست a, b, c, d, e, f, g, h با پیمایش سطری با شروع از راس a مطابقت دارد. و همین مطلب برای لیست a, c, b, d, e, h, f, g نیز صدق میکند.

تصویر ۸.۷ – یک مثال از گراف. تصویر (a) گرافی را نمایش میدهد که مثالی از حالت مسئله را نشان میدهد. تصویر (b) کوتاه‌ترین مسیر راس a را تا هر راس گراف با خطوط پررنگ نشان میدهد. در تصویر (b)، عددی که در هر راس مشخص است در واقع فاصله‌ی آن راس تا راس a است.

حلقه بدون تغییر

ما علاقه داریم یک الگوریتم بدون تغییر بسازیم؛ یک الگوریتم حریصانه؛ و اینگونه خود را محدود میکنیم. برای جستجوی حلقه بدون تغییر، ادامه ی این ساز و کار به خواننده واگذار میشود. اکنون تصور میکنیم قسمتی از کار انجام شده ست(بخش ۳، صفحه ۹۳ را ببنید). به این ترتیب، برای یک گراف جزئی $G = (N, V)$ (زیرگراف G القا شده با مجموعه رئوس N'، شامل رئوس ابتدایی)، لیستی تشکیل شده از پیمایش سطری G' با شروع از s داریم. عموما* این لیست، CLOSE نامیده میشود. پیشرفت این روند شامل گستردن این لیست با افزودن رئوسی است که در CLOSE نیستند و تا جای ممکن به s نزدیکند.

از آنجایی که هر راسی که در CLOSE حضور نداشته باشد، یک کاندید احتمالی برای انقال به CLOSE است، در غیاب بقیه ی مفروضات، پیشرفت ممکن اما به همان نسبت هزینه بر است. پیشنهاد میکنیم که نسخه اول این ثابت را با اضافه کردن یک ساختمان داده بهبود ببخشید. ساختمان داده OPEN شامل تمام رئوسی ست که در CLOSE حضور نداشته و کاندید این موضوع هستند که همسایه حذاقل یکی از رئوس CLOSE هستند. بیشین*، OPEN به عنوان یک لیست اولویت با مدیریت برروی فاصله ی عناصرش از s بوجود می آید، این موضوع به این دلیل است که عنصری که باید به لیست CLOSE منتقل شود باید نزدیک ترین به s باشد.

بعدها میبینیم که نسخه ساده شده یک لیست اولویت نیز امکان پذیر است. برای ماندگاری این نسخه جدید از ثابت*، بهینه است که سر OPEN را به انتهای لیست CLOSE منتقل کنیم، و – به عنوان همتای تقویت ثابت** – برای معرفی همسایگان ''جدید'' عنصر منتقل شده به ،OPEN عنصرهایی که نه در OPEN نه در CLOSE هستند(این یک انتخاب حریصانه است).

با این حال، با توجه به عنصری e در ،OPEN پرسیدن مستقیم درباره وجود یا عدم وجود یکی از همسایگان آن در OPEN یا CLOSE می تواند پرهزینه باشد. راه حل بهتر شامل تقویت (جدید) با گزاره زیر است: از نظر رنگ آمیزی آینده، یک ''رنگ'' به هر راس گراف اختصاص می یابد، سفید اگر راس در OPEN یا CLOSE باشد، و در غیر این صورت خاکستری (در واقع، در اینجا، دو رنگ نقش مقادیر بولین را بازی می کنند). به شرطی که دسترسی مستقیم به رئوس امکان پذیر باشد، به روز رسانی OPEN آسان تر می شود. در پیشرفت، حفظ این مکمل ناوردا با رنگ آمیزی هر راسی که به OPEN منتقل می شود به رنگ سفید حاصل می شود.

بیایید به استراتژی مدیریت صف OPEN بازگردیم. آیا می توان به جای صف اولویت دار از یک صف ساده FIFO (نگاه کنید به بخش ۸.۱، صفحه ۳۲) استفاده کرد؟ در این صورت، مدیریت OPEN به طور قابل توجهی ساده می شود. برای انجام این کار، زمانی که رأس e از OPEN خارج می شود تا به CLOSE ملحق شود، همسایگان e که نامزد ورود به OPEN هستند باید فاصله ای بیشتر یا مساوی با تمام عناصر موجود در OPEN داشته باشند، که این امر امکان داشتن یک صف مرتب را فراهم می کند. این بدان معناست که اگر e در فاصله k از s باشد، سایر عناصر OPEN در فاصله k یا $(k + ۱)$ از s قرار دارند، زیرا همسایگان ''خاکستری'' e در فاصله $(k + ۱)$ از s قرار دارند. این فرض را به ناوردا اضافه می کنیم. خواننده دعوت می شود بررسی کند که آیا این موضوع با راه‌اندازی حلقه واقعاً برقرار شده است. همچنان باید ثابت کرد که با پیشرفت حفظ می شود. در نهایت، ما ناوردای زیر را پیشنهاد می کنیم که از چهار بند تشکیل شده است.

۱. بسته (CLOSE) یک صف اول–وارد–اول–خارج (FIFO) است که محتوای آن نشان دهنده یک ''پیمایش عمق–اول'' از زیرگراف G است که توسط رئوس موجود در بسته (CLOSE) تشکیل شده است.

۲. باز (OPEN) یک صف اول–وارد–اول–خارج (FIFO) از رئوس همسایه رئوس موجود در بسته (CLOSE) است. اشتراک مجموعه بین باز (OPEN) و بسته (CLOSE) تهی است.

۳. اگر ابتدای صف باز (OPEN) حاوی رئوس با فاصله k از s باشد، سایر عناصر صف باز (OPEN) در فاصله k یا $(k + ۱)$ از s قرار دارند.

۴. در گراف G، رئوس موجود در بسته (CLOSE) یا باز (OPEN) به رنگ سفید رنگ آمیزی می شوند، سایر رئوس خاکستری هستند.

شکل ۹.۷، صفحه ۳۶۶، مراحل مختلف «پیمایش پهنای_اول» گراف شکل ۸.۷، صفحه ۳۶۳ را نشان می‌دهد. در هر گراف شکل، رئوس موجود در بسته (CLOSE) با خطوط خاکستری و رئوس موجود در باز (OPEN) با خطوط دوتایی نمایش داده شده‌اند. فواصل فقط به عنوان یادآوری ذکر شده‌اند، الگوریتم از آنها استفاده نمی‌کند. بیایید به عنوان مثال در مورد مرحله‌ای که منجر به گذار از شکل (e) به شکل (f) می‌شود، توضیح دهیم. در شکل ،(e) بسته (CLOSE) لیست «پیمایش پهنای_اول» زیرگراف القا شده توسط رئوس a، b، c و d را در خود جای داده است. راس ،e سر صف باز ،(OPEN) به انتهای بسته (CLOSE) منتقل خواهد شد. کدام همسایه‌های e قرار است به OPEN ملحق شوند؟ c و b قبلاً در بسته (CLOSE) هستند، بنابراین موردی برای اضافه شدن ندارند. g قبلاً در باز (OPEN) است، تحت تأثیر قرار نمی‌گیرد. تنها راس باقی‌مانده h است که به صف باز (OPEN) ملحق شده و به رنگ سفید رنگ‌آمیزی می‌شود.

ساختارهای داده

از دو نوع ساختار داده در این الگوریتم استفاده می‌شود. مورد اول، صف‌های اول_وارد_اول_خارج ،(FIFO) در صفحه ۳۲ توضیح داده شده‌اند. مورد دوم مربوط به نسخه‌ی «رنگ‌آمیزی‌شده» گراف‌ها است. گراف بدون جهت رنگ‌آمیزی‌شده

در این الگوریتم، نیاز به رنگ‌آمیزی رئوس‌های یک گراف، دسترسی به رنگ آن‌ها و کاوش لیست همسایگان وجود دارد، بنابراین تعاریف زیر ارائه می‌شود (فرض بر این است که مجموعه رنگ‌ها (Colors) تعریف شده است):

- عملگر رنگ‌آمیزی $ColorGr(G, s, col)$: عملیاتی که رأس s گراف G را با رنگ col رنگ‌آمیزی می‌کند.
- تابع رنگ‌رأس $WhichColorGr(G, s)$ نتیجه نوع رنگ‌ها :(Colors) تابعی که رنگ رأس s گراف G را برمی‌گرداند.
- عملگر همسایگان باز $OpenNeighborsGr(G, s)$: عملیاتی که کاوش لیست همسایگان رأس s گراف G را آغاز می‌کند.
- تابع پایان لیست همسایگان $EndListNeighborsGr(G, s)$ نتیجه بولی :(B) تابعی که مقدار درست (true) را برمی‌گرداند در صورتی که کاوش لیست همسایگان رأس s گراف G تمام شده باشد و در غیر اینصورت مقدار نادرست (false) را برمی‌گرداند.
- عملگر خواندن همسایگان $ReadNeighborsGr(G, s, s)$: عملیاتی که هویت رأس ''زیر خواننده'' لیست همسایگان s را در s′ ذخیره کرده و سپس خواننده را یک موقعیت به جلو حرکت می‌دهد.

در این کاربرد، از نظر بیان الگوریتم و کارایی، بهترین بهینه‌سازی نمایش آن با استفاده از لیست همجوری است (برای مشاهده‌ی نمونه‌ای از چنین نمایشی برای گراف‌های جهت‌دار، به شکل (d) در صفحه‌ی ۲۳ مراجعه کنید). بنابراین، گراف به صورت یک ''سه‌تایی'' $G = (N, V, R)$ تعریف می‌شود که در آن R نشان‌دهنده رنگ‌ها است (در حال حاضر ''سفید'' و ''خاکستری'').

الگوریتم

علاوه بر نمودار ،G این الگوریتم از متغیرهای cv راس جاری و فهرست همسایگان برای مرور لیست همسایگان استفاده می کند.

```
constants
n   N1 and n = . . . and N = 1 .. n and Colors = grey, white and
V   N × N and V = . . .
variables
R   N   Colors and G = (N, V, R) and
s   N an d cv   N and neighb   N and CLOSE   FIFO(N) and OPEN   FIFO(N)
begin
```

شکل ۹.۷ – مراحل مختلف کاوش اولویت‌ـعرضی در گراف اسکیما (الف) در شکل ۸.۷ (صفحه ۳۶۳). رئوس با دایره توپر، رئوس CLOSE هستند، آنهایی که با دو دایره مشخص شده اند، رئوس OPEN هستند. مقدار صحیح که هر راس را همراهی می‌کند، فاصله شناخته شده از راس a است. دو صف OPEN و CLOSE به ترتیب در شمال شرقی و جنوب گراف ها نشان داده شده اند.

```
/ coloring all the vertices in grey: /
for w   N do
ColorGr(G, w, grey)
end for;
InitFifo(CLOSE); InitFifo(OPEN);
s   . . . ; / choice of the initial vertex: /
ColorGr(G, s, white);
AddFifo(OPEN, s);
while not IsEmptyFifo(OPEN) do
cv   HeadFifo(OPEN); RemoveFifo(OPEN);
AddFifo(CLOSE, cv);
OpenNeighborsGr(G, cv) ;
while not EndListNeighborsGr(G, cv) do
ReadNeighborsGr(G, cv, neighb) ;
if WhichColorGr(G, neighb) = grey then
ColorGr(G, neighb, white) ;
AddFifo(OPEN, neighb)
end if
```

19. end while
20. end while;
21. write(CLOSE)
22. end

سوال ۱: پیچیدگی مجانبی این الگوریتم از نظر شرایط ارزیابی شده چیست؟
سوال ۲: بر اساس این الگوریتم، اصل الگوریتم رنگ آمیزی حریصانه را توضیح دهید.

الگوریتم رنگ آمیزی دو رنگ گراف

اکنون مجهز هستیم تا به مشکل اصلی در هسته این مسئله بپردازیم: رنگ آمیزی یک گراف با سیاه و سفید. ما الگوریتم بالا را به گونه ای تطبیق خواهیم داد که به صورت متناوب با سیاه و سفید، بر اساس عمق در رابطه با راس شروع، رنگ آمیزی شود، تا زمانی که رئوس ها تمام شوند یا یک عدم امکان کشف شود.

سوال ۳: الگوریتم رنگ آمیزی را طراحی کنید.

سوال ۴: بر اساس این الگوریتم، اصل الگوریتم رنگ آمیزی حریصانه را توضیح دهید.
شکل ۸.۷، صفحه ۳۶۳

سوال ۵: کد الگوریتم را بنویسید و پیچیدگی آن را مشخص کنید.

سوال ۶: مسئله ۵۹، صفحه ۲۳۶، به مسئله عمومی‌تر رنگ‌آمیزی با m رنگ ($m2$) می‌پردازد. امکان تعمیم الگوریتم ارائه شده در پاسخ به سوال ۵، برای m > ۲ را مورد بحث قرار دهید.

تذکر یک ویژگی جالب از گراف‌های دو–رنگ‌پذیر این است که: یک گراف دو–رنگ‌پذیر است اگر و تنها اگر هیچ دوره‌ی فردطولی نداشته باشد. با این حال، این ویژگی سازنده نیست: مشاهده‌ی آن روی یک گراف منجر به رنگ‌آمیزی نمی‌شود!
راه‌حل در صفحه ۳۹۷ است.

۲.۰ مسئله ۸۴: از ترتیب جزئی به ترتیب کلی

دو نسخه از مرتب‌سازی توپولوژیکی مورد بررسی قرار می‌گیرد. نسخه اول ساده لوحانه است اما خیلی کارآمد نیست و طراحی آن آسان است. نسخه دوم نیازمند تقویت یک ناورد [خاصیت تغییرناپذیر] است؛ با استفاده از اشاره‌گرها پیاده‌سازی می‌شود و هر دو مسئله‌ی بهینه‌سازی و دستکاری ساختارهای پویا را به خوبی نشان می‌دهد. الگوریتمی که در اینجا ساخته شده است نزدیک به الگوریتم ماریمونت برای سطح‌بندی یک گراف بدون مدار است.

برای حل این مسئله، ابتدا باید مسئله ۲، صفحه ۳۳ را مطالعه کرد. فرض کنید $(E,)$ یک زوج مرتب باشد به گونه‌ای که E یک مجموعه متناهی با n عضو و ￼ یک رابطه ترتیب جزئی روی E باشد. ما می‌خواهیم روی E یک رابطه ترتیب کلی ￼ سازگار با ￼ بنا کنیم، به این معنی که برای هر زوج (a, b) از $(ab)(ab) : E$.
به هر عضو از $(E,)$ که هیچ پیش‌تر نداشته باشد، عضو مینیمال گفته می‌شود.

مثال در زمینه‌ی دوره‌ی علوم کامپیوتر، ۱c ￼ ۲c نشان‌دهنده‌ی این واقعیت است که دوره‌ی ۱c باید قبل از دوره‌ی ۲c گذرانده شود تا پیش‌نیازهای لازم برای درک دوره‌ی دوم فراهم شود. حال دوره‌های زیر را در نظر بگیرید:

مثالی از رابطه به صورت زیر تعریف می‌شود:

$ab, ac, bd, cb, cd, ce, cf, ed, fe$.

این را می‌توان با نمودار زیر نشان داد:

این نوع گراف با دو ویژگی شناخته می‌شود: جهت‌دار بودن و عاری بودن از دور (مدار). به چنین گرافی، «گراف جهت‌دار غیر دوردار» یا « «DAG گفته می‌شود. در چنین گرافی به یک عضو بدون هیچ پیش‌تر (عضو مینیمال مرتب‌سازی جزئی)، «نقطه‌ی ورود» گفته می‌شود.

هدف این مسئله، ساخت یک الگوریتم حریصانه است که یک ترتیب کلی سازگار با ترتیب جزئی اولیه را ارائه دهد. برای مثال بالا، یک راه‌حل شامل پیشنهاد ترتیب کلی $acbfed$ است.

سوال ۱: اثبات کنید که یک گراف جهت‌دار غیر دوردار (DAG) غیر تهی، که هر یک از نقاط ورود آن حذف شده باشد، همچنان یک گراف جهت‌دار غیر دوردار (DAG) باقی می‌ماند.

سوال ۲: فرض کنیم هر گرافی با $G = (N, V)$ نمایش داده شود و n برابر $card(N)$ باشد. اثبات کنید که برخی DAG ها (گراف جهت‌دار غیر دوردار) به گونه‌ای هستند که $card(V)(n۲)$ است.

اکنون، پیش از اجرای آن روی مثال بالا، به ترسیم کلی حلقه‌ی ‘‘حریصانه’’ الگوریتم می‌پردازیم. روش ساخت استفاده شده مبتنی بر تکنیکِ «اجرای پیشرو» است. سوالات بعدی در مورد الگوریتم، بهینه‌سازی و پیچیدگی آن خواهد بود.

اولین تلاش برای ساخت

فرض کنید $G = (E, V)$ یک DAG ورودی با n راس باشد
ناورد [خاصیت تغییرناپذیر]: فرض کنید S صف خروجی باشد که شامل مجموعه $ES(ESE)$ از رأس‌های مرتب‌سازی‌شده بر اساس یک ترتیب کلی سازگار با ⍰ می‌باشد، به گونه‌ای که هر راس v از E که در ES نیست v) ⍰ E–ES))) طبق ترتیب جزئی بزرگ‌تر از هر راس ES است.
شرط توقف: تمام رأس‌ها در صف S قرار دارند، یعنی $n = |S|$. در واقع، اشتراک ناورد [خاصیت تغییرناپذیر] و شرط قبلی ایجاب می‌کند که S فهرستی مرتب‌سازی‌شده بر اساس ترتیبی سازگار با ترتیب جزئی باشد.
روند کار الحاق یکی از رأس‌های (EES) به صف S است. این کار باعث می‌شود که این راس در زیرگراف القا شده $(Induced(G, EES))$ قرار گیرد و ناورد [خاصیت تغییرناپذیر] در این راستا تقویت خواهد شد.
به منظور تعیین راسی که باید به بهترین نحو در صف قرار گیرد، ضروری است یک ساختار داده‌ای معرفی کنیم که قادر به بهره‌برداری از زیرگراف القا شده $(Induced(G, EES))$ باشد.

طرح جایگزین برای ساخت

به عنوان یک متغیر در نظر گرفته می‌شود. ناورد [خاصیت تغییرناپذیر]: گزاره G" یک DAG است" به نسخه قبلی ناورد اضافه می‌شود.
شرط توقف: بدون تغییر باقی می‌ماند.
به دنبال یکی از نقاط ورود گراف G هستیم تا این راس از (EES) را به صف S منتقل کنیم. به راحتی قابل بررسی است که S نسخه اول ناورد را برآورده می کند و اینکه G، زیرگراف القا شده جدید، در واقع یک DAG است (به دلیل ویژگی‌ای که در پاسخ به سوال ۱ برقرار شده است).
قابل ذکر است که G نقش صف ورودی الگوریتم‌های حریصانه را ایفا می کند و اشتراک ناورد و شرط توقف، در واقع، به معنای دستیابی به هدف مطلوب است.
آغازین سازی: ناورد [خاصیت تغییرناپذیر] از صف خالی S و گراف G که همان گراف اولیه است، برقرار می‌شود.
تابع پایان مناسب $n|S|$ است، زیرا در هر مرحله از پیشرفت، یک عنصر از G به S منتقل می‌شود.
قابل ذکر است که این الگوریتم بر اساس ساخت، خروجی صحیحی را برمی‌گرداند. بدیهی است که این روش، «اجرای پیشرو» را انجام می‌دهد.
سوال ۳: الگوریتم بالا را بر روی مثال ابتدایی اعمال کنید.
سوال ۴: فرض کنید که هر دو تابع $dG(s)$ که درجه ورودی راس s را در گراف G برمی‌گرداند و القا (Induced) (برای مفاهیم مربوط به گراف‌ها به بخش ۵.۱، صفحه ۲۲ مراجعه کنید) در دسترس هستند، کد این الگوریتم را بنویسید. با فرض اینکه نمایش گراف‌ها با فهرستی از جانشین‌ها باشد (به شکل ۳.۱، صفحه ۲۳ مراجعه کنید)، ثابت کنید که پیچیدگی این الگوریتم از نظر تعداد رئوس بازدید شده در $O(n^{۲})$ است.
سوال ۵: در نسخه بدست آمده در پاسخ به سوال ۴، از دیدگاه پیچیدگی، عامل جریمه کننده جستجو برای حداقل بین تمام رئوس‌هایی است که هنوز باید در نظر گرفته شوند. بر اساس تقویت (تقویت) ناورد فوق، راه حل کارآمدتری برای گراف‌های غیرمتراکم (چند قوس با توجه به مجذور تعداد رئوس) پیشنهاد دهید. در مورد کارایی این راه حل چه نتیجه‌ای می‌توان گرفت؟
راه‌حل در صفحه ۴۰۰ است.