ساختمان داده‌ها (۲۲۸۲۲)

مدرس: حسین بومری

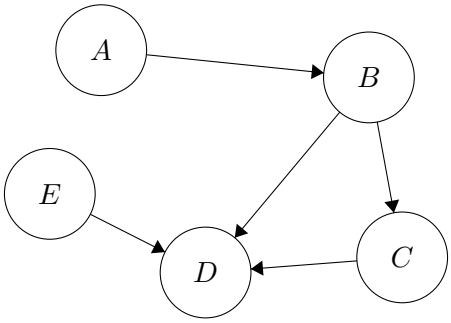
[زمستان 99]

جلسه ۲۷: گراف۱ و توپولوجیکال سورت[1]

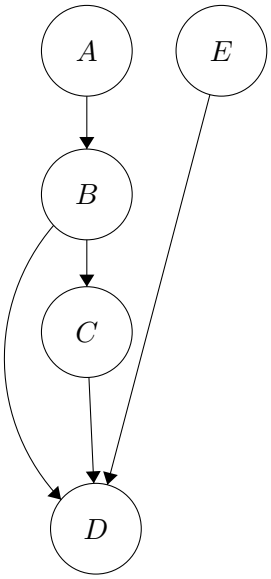
نگارنده: محمد مهدی زارع

در جلسات گذشته درباره مولفه های همبندی در گراف بدون جهت حرف زدیم که مرتبه زمانی آن $O(v+e)$ بود و همچنین درباره ترتیب جزیی هم حرف زده بودیم .

گراف زیر را در نظر بگیرید



که ترتیب جزیی آن به صورت زیر است



و ویژگی ترتیب جزیی این است که یالی به سمت بالا نداریم و الگوریتم ساختن آن این بود که ابتدا راس های که یال ورودی ندارند را در سطح اول بنویسیم سپس این راس ها را حذف کنیم و باز راس هایی که یال ورودی ندارند را حذف کنیم و در آخر یال ها را رسم کنیم.

[1] topological sort

۱ مرتب‌سازی توپولوژیکی

هدف ازمرتب‌سازی توپولوژیکی این است که برای یک گراف بدون دور ترتیبی از عناصر ارائه بدهد که همه یال ها روبه جلو باشند (به یک طرف باشند) برای مثال برای گراف بالا ترتیب

$$A, E, B, C, D$$

را به ما میدهد. در الگوریتم بالا که با استفاده از آن توانستیم مرتب‌سازی توپولوژیکی انجام دهیم مرتبه زمانی هان به صورت زیر است و در آن از داده ساختار هیپ استفاده شده است و یکی از جاهایی هست که عملیات $decreas \quad key$ به درد ما میخورد :

پیمایش گراف و شناسایی یال ها : m

ساختن هیپ کمکی : n

بیرون آوردن مینیمم از هیپ : $nlogn$

عملیات $decreas \quad key$ به ازای هر یال : $mlogn$

$$T : O((m+n)logn)$$

میتواینم الگوریتم دیگری هم برای مرتب‌سازی توپولوژیکی ارائه دهیم .

میخواهیم از dfs استفاده کنیم. در dfs یال ها سه رنگ دارن که هر کدام میتواند خاکستری باشد که یعنی در حال پیمایش هست یا سیاه باشد که یعنی کل خانواد آن سیاه شدند یا سفید باشد که یعنی تا حالا دیده نشده است . حال بیاید در لحظه سیاه شدن هر راس ،آن را در یک پشته بگذارید .

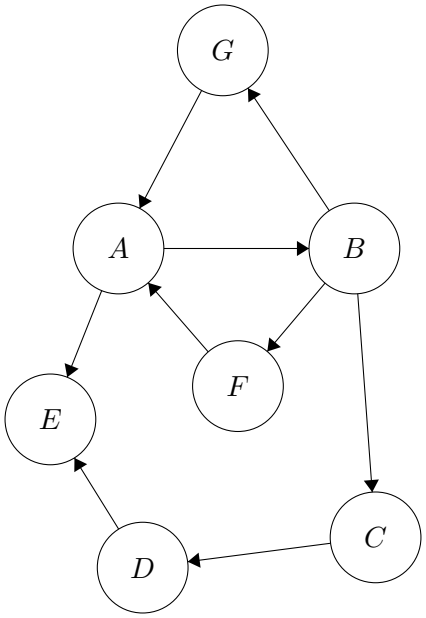
ادعا ۱. ادعا میکنیم باز همه جهت ها به یک سمت هستند

اثبات. برهان خلف : فرض کنید یک یال با جهت برعکس وجود دارد $x \to y$. دو حالت وجود دارد . اول اینکه y جز پدر هایx باشد یعنی از y به x یک مسیر است .پس یک دور وجود می آید و با فرض بدون دور بودن گراف در خلاف است . حالت دوم اینکه y جز پدر هایx نباشد پس زمانی که به x رسیدیم میتوانستیم yرا ببینیم پسy را باید زودتر می دیدیم که باز با ترتیب ارائه شده در تناقض است و فرض خلف باطل است و ادعا درست است. □

مرتبه زمانی این الگوریتم همان مرتبه زمانی dfs است $O(m+n)$

۲ مولفه های قویا همبند

حال میخواهیم از مرتب‌سازی توپولوژیکی استفاده کنیم و مولفه های قویا همبند را بدست آوریم. گراف زیر را در نظر بگیرید



مولفه های قویا همبند به صورت زیر هستند :

$1.c \qquad 2.d \qquad 3.e \qquad 4.abcg$

تعریف ۱. مولفه های قویا همبند :مولفه هایی که با شروع حرکت از یکی به توان به دیگری رسید .

۱.۲ الگوریتم اول

برای هر دو راس امتحان میکنیم آیا قویا همبند هستند یا نه .

این الگوریتم را میتوان با $n^2(m+n)$ حساب کرد که $m+n$ همان الگوریتم dfs است .

۲.۲ الگوریتم دوم

ابتدا به ازای هر راس وهمه راس هایی که به انها مسیر دارد را حساب میکنیم $O(n(n+m))$

برای هر دو راس چک میکنیم ایا در ماتریس های مجاورت هم هستند یا نه $O(n^2)$ پس این الگوریتم هم مرتبه زمانی $O(n(n+m))$ است.

۳.۲ الگوریتم سوم

ابتدا الگوریتم مرتب‌سازی توپولوژیکی را روی گراف خود اجرا کنیم $O(n+m)$ وسپس ترتیب خروجی راس ها را بدست اوریم حال گراف را برعکس کنید (یال ها را برعکس کنید) حال از سمت اخر لیست دوباره الگوریتم dfs را اجرا میکنیم در این گام اگر از یک راس شروع کنیم ، هر رای جدیدی که ببینیم با ان راس یک مولفه قویا همبند هستند . برای مثال گراف بالا را در نظر بگیرید

ترتیب خروجی اجرای مرتب‌سازی توپولوژیکی $\qquad e|g|d|c|f|b|a|$

اجرای dfs روی عکس گراف و لیست خروجی از گام اول :

$$|e|g(1)|d|c|f(1)|b(1)|a(1)|$$

$$|e|g(1)|d|c(2)|f(1)|b(1)|a(1)|$$

$$|e|g(1)|d(3)|c(2)|f(1)|b(1)|a(1)|$$

$$|e(4)|g(1)|d(3)|c(2)|f(1)|b(1)|a(1)|$$

پس مرتبه زمانی این الگوریم $O(m+n)$ است .

۳ درخت های پوشا

تعریف ۲. به درختی پوشا میگوییم که دور نداشته باشد و همه راس ها به هم مسیر دارند.

میخواهیم الگوریتیمی برای پیدا کردن درخت پوشا روی گراف ارایه دهیم .

با پیمایش هر دو الگوریتم dfs ، bfs میتوان درخت پوشا یدست آورد و یال هایی که روی آنها حرکت میکنیم درخت پوشا هستند. وفرق آنها این است که در الگوریتم dfs درختی که بدست می آوریم عمق کمینه دارد ولی در bfs ممکن است بیشتر باشد .و مرتبه زمانی این الگوریتم ها $O(m+n)$ است .

حال فرض کنید یال ها وزن‌دار هستند .

۱.۳ درخت پوشا کمینه

تعریف ۳. به درختی که مجموع وزن یال های آن کمینه باشد درخت پوشا کمینه[۲] میگویند

برای پیدا کردن این نوع درخت ها دو الگوریتم کروسکال [۳]و پریم[۴] وجود دارد که اگر از فیبوناچی هیپ استفاده کنیم مرتبه زمانی خیلی بهینه تر میشود.

[2] MST
[3] kruskal
[4] prim