



مدرس: دکتر شهرام خزایی

آنالیز الگوریتم‌ها

تمرین سری دو

شماره دانشجویی: ۴۰۱۱۰۰۰۷۱

نام و نام خانوادگی: کثرال باغستانی

فرض ۱: الگوریتمی داریم که در زمان خطی مولفه‌های قویا همبند را در گراف جهت دار پیدا میکند.
فرض ۲: الگوریتمی داریم که در زمان خطی sort topological را برای یک dag پیدا میکند. در ادامه این الگوریتم‌ها را ارائه می‌دهم.

پرسش ۱

پرسش ۲

پرسش ۳

ابتدا از روی لیست f_i ها گرافی جهت دار که راس‌های آن کارمندان و یال‌های آن به صورت جهت دار از هر کارمند به کارمندی است که باید به آن خبر رسانی کند تشکیل می‌دهیم. سپس با الگوریتم 1 مولفه‌های قویا همبند را $O(|V| + |E|)$ پیدا می‌کنیم که در اینجا تعداد یال‌ها دقیقاً با تعداد راس‌ها برابر است (زیرا هر کارمندی دقیقاً به یک کارمند یال جهت دار دارد). پس $O(n)$.
ادعا:

هر شخصی خبری که میشوند به همه افراد منتقل میشود اگر و تنها اگر تمام افراد در یک مولفه قویا همبند قرار گیرند.
اثبات:

اگر تمام افراد در یک مولفه قویا همبند قرار گیرند به وضوح از هر راس (کارمند) به راس‌های دیگر مسیری جهت دار هست پس خبر به خوبی پخش میشود.

حال میخواهیم نشان دهیم اگر تمام افراد در یک مولفه قویا همبند نباشند فردی وجود دارد که اگر خبر به او برسد همه مطلع نمیشوند. می‌دانیم هر dag حداقل یک sink دارد پس اگر خبر به یکی از افراد در مولفه‌ای که sink است برسد خبر فقط در همان مولفه میماند و افراد در مولفه‌های دیگر از آن خبر مطلع نمیشوند پس ادعا ثابت میشود.
ادعا:

اگر در sort topological هر راسی به راس بعد از خود یال جهت دار داشته باشد و راس آخر را به راس اول به یالی جهت دار وصل کنیم dag ما با کمترین تعداد ممکن دور دار میشود و همه مولفه‌ها در یک مولفه قرار میگیرند.
اثبات:

اگر در sort topological بخواهیم بتوانیم از راس i به راس $i+1$ برویم حتماً باید یال جهت دار از i به $i+1$ وجود داشته باشد زیرا از هیچ یالی که

پرسش ۴

اشتباه ممکنه چنتا سینک داشته باشیم. طبق الگوریتم ۱ ابتدا مولفه های قویا همبند را برای شهر ها و مسیر هایی که از قبل بین آنهاست پیدا میکنیم. سپس آنها را sort topological میکنیم ممکن است در sort topological ما مولفه های همبندی ای وجود داشته باشند که به هیچ وجه به هم راه ندارند و اصلا به هم وابسته نیستند این مولفه ها را در دسته های جداگانه ای قرار میدهیم سپس برای هر u_i sink آن را در بخشی که هست پیدا میکنیم و برای هر v_i source آن را. حالا اگر u_i و v_i به هم مسیر جهت دار نداشته باشند یال جهت داری از سینک u_i به سورس v_i وصل میکنیم. این کار باعث میشود مسیر های جهت داری بین u_j هایی که از u_i به آنها راه هست به v_k هایی که در مسیر رسیدن به v_i قرار دارند ایجاد شود. این کار ممکن است کار مارا برای ساخت مسیر کم کند. حالا کافیت درستی الگوریتم را اثبات و زمان آن را تحلیل کنیم.

ادعا:

این الگوریتم به ازای هر i v_i را به v_j مسیر دار میکند.

اثبات:

در هر مرحله با اجرای DFS مشخص میشود که آیا مسیری بین این دو راس هست یا خیر و اگر نباشد به سادگی با وصل کردن سینک u_i به سورس v_i این مسیر ساخته میشود.

ادعا:

این الگوریتم کمترین تعداد مسیر مورد نیاز را به ما میدهد.

اثبات:

برای اثبات این ادعا باید دو چیز را نشان دهیم یکی اینکه اگر در مرحله i ام بین دو شهر u_i و v_i مسیری جهت دار نباشد حداقل یک مسیر جدید باید ایجاد شود برای اتصال این دو به هم و مورد بعد این است که نشان دهیم با هر جایگشتی از u_i ها تعداد مسیر های به دست آمده یکسان است. مورد اول بدیهی است پس کافیت نشان دهیم که با هر ترتیبی u_i ها را انتخاب و با الگوریتم داده شده مسیری از آن به v_i ایجاد کنیم تفاوتی در تعداد مسیر ها ایجاد نمیشود. برای اینکار کافیت نشان دهیم در یک جایگشت داده شده با عوض کردن جای دو عنصر دلخواه u_i و u_j تفاوتی ایجاد نمیشود. چون میدانیم از هر جاگشتی با عوض کردن دو تا دوتا عناصر میتوان به هر جایگشت دلخواه رسید.

عناصر را به سه دسته قبل از u_i و بین u_i و u_j و بعد از u_j تقسیم میکنیم.

جایگشت اول به صورت $u_1, \dots, u_i, \dots, u_j, \dots, u_m$ و جایگشت دوم به صورت $u_1, \dots, u_j, \dots, u_i, \dots, u_m$ است. میتوان جایگشت ها را به صورت زیر نگاه کرد $u_i, \dots, u_j, \dots, u_m$ و $u_j, \dots, u_i, \dots, u_m$ زیرا قبل از آن دقیقا تغییرات یکسانی را در گراف ایجاد کرده چهار حالت را بررسی میکنم:

۱: u_i و u_j به ترتیب در جایگشت های اول و دوم مسیری اضافه نکنند: در این حالت وقتی u_i در جایگشت اول مسیر ایجاد نکرده یعنی تا قبل از آن مرحله به v_i متناظر خود وصل شده پس در جایگشت دوم نیز لزومی به اتصال ندارد و همچنین برای u_j . پس این دو عنصر از جایگشت بی اثر میشوند و هر دو جایگشت یکی است.

۲: u_i در جایگشت اول مسیر ایجاد کند و u_j در جایگشت دوم مسیر ایجاد نکند: در این حالت در جایگشت اول مسیر ایجاد شده توسط u_i باعث بی اثر شدن عناصری در جایگشت شده که سینک آنها با سینک u_i و سورس آنها با سورس v_i یکی بوده است. در نتیجه در جایگشت دوم اگر یکی از این عناصر زودتر از u_i ظاهر شود u_i و همه آن عناصر را بی اثر کرده و تعداد مسیر ها ثابت میماند. برای u_j چون در جایگشت دوم بی اثر است در جایگشت اول نیز (مانند استدلال قسمت قبل) بی اثر است. در نتیجه دو حالت دیگر که به صورت زیر هستند نیز با همین استدلال ثابت میشوند.

۳: u_j در جایگشت اول مسیر ایجاد کند و u_i در جایگشت دوم مسیر ایجاد نکند.

۴: u_i و u_j به ترتیب در جایگشت های اول و دوم مسیری اضافه کنند.
پس ادعا ثابت میشود.

تحلیل زمانی:

در هر مرحله برای پیدا کردن مولفه های قویا همبند و sort topological به $O(n)$ زمان نیاز داریم (که n مجموع راس و یال های گراف است).
و اما در نهایت برای m راس باید این کار را انجام دهیم پس $O(nm)$ زمان مورد نیاز برای اجرای این الگوریتم میباشد.

پرسش ۵

پرسش ۶

برای اثبات این الگوریتم کافیت دو چیز را ثابت کنیم که به صورت دو ادعا در زیر ارائه میکنم.

در ابتدا به این نکته توجه میکنیم که در درخت هر گاه مسیری بین دو رأس داریم آن مسیر تنها مسیر بین آن دو رأس است. (زیرا در غیر این صورت دور تشکیل میشود) در نتیجه مسیری که بین دو رأس i و j در درخت BFS وجود دارد ابتدا از i به ریشه میرویم و اگر j در این مسیر نبود از j به سمت ریشه میرویم و اولین جایی که به مسیر از i به ریشه برخورد کردیم میتوانیم به سمت i برویم. پس هر مسیری یک بخش بالا رونده به سمت ریشه و یک بخش پایین رونده به سمت رأس مورد نظر دارد. و نکته دوم قابل توجه این است که قطر درخت هرگز نمیتواند بین i و j باشد که یکی در مسیر دیگری به ریشه است. زیرا فاصله آن رأس تا ریشه بیشتر از فاصله آن تا رأس دیگر میشود.

ادعا ۱:

اگر قطر گراف طول D داشته باشد هر رأس را به عنوان ریشه درخت BFS انتخاب کنیم حتما یک سر قطر در سطح آخر می افتد.

اثبات:

رأس v را به عنوان ریشه درخت BFS در نظر میگیریم. سر اول قطر از آن فاصله H و سر دوم قطر از آن فاصله $D - H$ دارد. در این صورت ارتفاع درخت BFS برابر با $\max(H, D - H)$ میشود. زیرا اگر بیشتر شود رأسی به فاصله بیشتر از این دو، از ریشه درخت وجود دارد که این یعنی آن رأس به عنوان سر دیگر قطر شناخته میشود.

ادعا ۲:

اگر در درخت BFS یک راس از سطح آخر انتخاب کنیم و BFS را روی آن اجرا کنیم مستقل از اینکه آن کدام رأس است عدد یکسانی به عنوان قطر به ما میدهد.

اثبات:

فرض کنیم a_1, a_2, \dots, a_n رأس های سطح آخر ما هستند. اگر a_i راس i را به عنوان سر دیگر قطر خروجی دهد اگر a_j را نیز در نظر بگیریم. یا جد مشترک آن با i همان جد مشترک a_i با i است که در این صورت همان فاصله را با i دارد. و یا جد مشترک آن با i متفاوت از جد مشترک a_i و i است که در این صورت اگر در نظر بگیریم جد مشترک a_j و i پایین تر از جد مشترک a_i و i است. اگر i در سطح آخر نباشد آنگاه از جد مشترک i و a_j بجای رفتن به i به a_j میرویم. که این یعنی طول قطر برای a_i بیشتر میشد. پس اگر جد مشترک متفاوت داشتیم نیز فاصله a_i و a_j با هم را به عنوان خروجی برای a_j در نظر میگیریم. (زیرا برابر با فاصله a_i و i بود). حالا کافیت نشان دهیم که هیچ کدام از a_i ها نمیتوانند عدد بزرگ تری از دیگری به قطر نسبت بدهند. این بخش نیز به وضوح معلوم است

چون نشان دادیم هر عددی که از هر کدام از a_i ها بگیریم به سادگی میتوان به a_j دیگر نیز راسی نسبت داد که همان عدد را بدهد. پس همگی مساوی و ماکسیمم نداریم.

و اثبات کامل میشود.