ساختمان داده‌ها و الگوریتم‌ها

نیم‌سال دوم ۹۷ـ۹۸

دانشکده‌ی مهندسی کامپیوتر

گردآورندگان: امیرمجتبی صبور ، علیرضا موسوی

پاسخنامه تمرین عملی چهارم

مسئله‌ی ۱. موش آزمایشگاهی و توپ‌هایش

بدیهیست که برای حل سوال باید بتوانیم در هر مرحله عضو بیشینه و کمینه را پیدا کنیم. برای این کار در زمان $O(n)$ یک هرم بیشینه و یک هرم کمینه می‌سازیم. سپس در هر مرحله یک عضو از هرم بیشینه و یک عضو از هرم کمینه را به عنوان x_i و x_j می‌گیریم و سپس قرار می‌دهیم $x'_i = \lfloor \frac{x_i+x_j}{۲} \rfloor$ و $x'_j = \lceil \frac{x_i+x_j}{۲} \rceil$ و سپس مجددا x'_i و x'_j را داخل هر دو هرم درج می‌کنیم. دقت کنید که با این کار هردفعه تعداد اعضای هر دو هرم یکی زیاد می‌شود (عضو بیشینه در هرم کمینه و عضو کمینه در هرم بیشینه باقی می‌مانند) اما از آنجایی که این اعضا هیچگاه خارج نمی‌شوند پس نیازی به حذف کردن آن‌ها نیست. این الگوریتم در زمان $O(n + klg(n + k))$ اجرا می‌شود.

مسئله‌ی ۲. متین و کیانوش

برای این سوال ابتدا باید گراف نهایی را کمی بررسی کنیم. گراف نهایی گراف کاملی است که هر یال ۲ رنگ دارد و در هر مثلث تعداد یال های قرمز زوج است. در این گراف ۱ راس را در نظر بگیرید و تمامی همسایه هایی که با یال قرمز به آنها وصل شده است را در یک سمت و تمامی آنهایی که با یال آبی به آنها وصل شده اند را در سمت دیگر قرار دهید. با استفاده از فرض زوج بودن یالهای قرمز در هر مثلث به این نتیجه میشه رسید که تمامی یال های بین این دو بخش قرمز، و تمامی یالهای درون این دو بخش سبز خواهند بود. یعنی گراف ما تشکیل شده است از ۲ بخش که تمامی یال هایی که دو سر آن در دو بخش مختلف اند قرمز و بقیه سبز هستند.

حالا برای اینکه گراف داده شده مان را بتوانیم به چنین بخشی تبدیل کنیم، میشه دید که اگر DSU ای n تایی برای رئوس در نظر بگیریم و هر دو راسی که با یال سبز به هم وصل اند را $Union$ بگیریم، آنگاه هر $Component$ نهایی ای DSU باید به طور کامل در یکی از این دو بخش قرار بگیرد.

پس راه حل میشود اینکه ابتدا یک DSU میسازیم و یالهای سبز را $Union$ میگیریم برای دو سرش. سپس یالهای قرمز را در نظر میگیریم. اگر یال قرمزی باشد که هر دو سرش داخل یکی از این $Component$ ها قرار بگیرد نمیشود این گراف را به گراف نهایی تبدیل کرد. وگرنه اگر ما هر کدام ازین $Component$ ها را یک راس در گراف جدیدمان در نظر بگیریم و یالهای قرمز بین این رئوس قرار خواهند گرفت. گراف اولیه مان را میتونیم به گراف نهایی تبدیل کنیم اگر و تنها اگر این گراف جدید ۲ رنگ پذیر باشد.

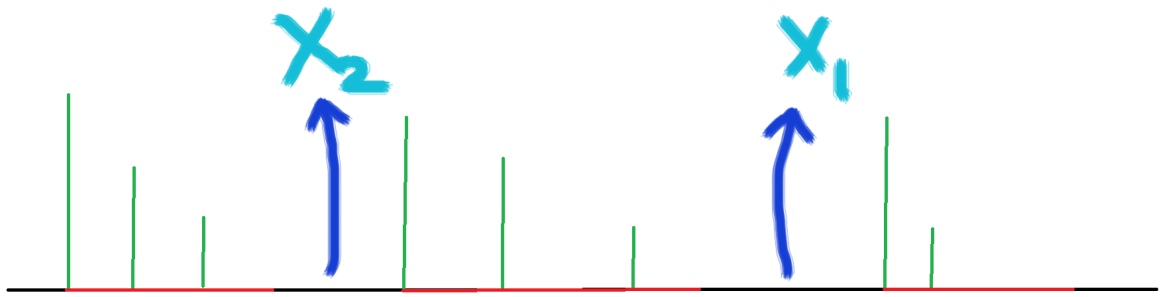
پس کافیست در $O(n)$ دو رنگ پذیری گراف جدید را چک کنیم. سپس یک DSU برای گراف جدید تشکیل دهیم و دو سر یالهای قرمز را ایندفعه $Union$ بگیریم. سپس جواب نهایی برابر میشود با $Num\ of\ Components-۱$ ۲.

مسئله‌ی ۳. برج‌های لگویی

در این مسئله ابتدا حالت ساده تر از مسئله کلی را بررسی میکنیم. حالتی را بررسی میکنیم که اگر دومینوی اول را بزنیم، قطعا دومینوی دوم نمی افتد. در این حالت ما باید بیاییم راست ترین نقطه ای را که با انداختن دومینوی i ام دومینو های افتاده در آن ختم میشوند را در نظر بگیریم. دومینوی بعدی آنرا در نظر بگیریم. فاصله ی بین این راست ترین نقطه تا شروع دومینوی بعدی را به یه $Counter$ ای اضافه کنیم و سپس همینکار را با دومینوی بعدی تکرار کنیم و آنقدر برویم تا دومینوی j ام بیفتد. نتیجه ی $Counter$ جوابی است که ما باید در خروجی چاپ بکنیم.

از طرفی وقتی ما میخواهیم جواب مربوط به فیلم برداری از دومینوی i ام تا j ام را حساب کنیم دومینو های قبل i و بعد j هیچ تاثیری نخواهند داشت. پس یک راه حساب کردن جواب این است که ۲ تا $Stack$ و یدونه DSU در نظر بگیریم و سپس از j امین برج به عقب بیاییم. در $Stack$ اول ما تمامی دومینو هایی را میخواهیم نگه داریم که به ترتیب صعودی باشند از بالای استک به پایین آن و با افتادن هر عضو استک، بعدی ها نیفتند.

در این صورت اگر فرضا تا دومینوی k ام اینکار را انجام داده باشیم شکل چنین چیزی میشود:



و هدف ما این است که $Partial\ Sum$ های X_i هارو توی $Stack$ دوم نگه داریم. به طوریکه ته استک در این شکل مثلا $X_۱$ میباشد. بالایی اش برابر با $X_۱ + X_۲$ است الی آخر. خب برای اینکه دومینوی جدید بعدی را وارد این مراحل بکنیم، کافیست برایش مقدار $Right = Start + Length$ رو تعریف کنیم. سپس از سر استک اولی Pop انجام میدهیم اگر $Start$ آن کمتر مساوی مقدار $Right$ باشد. اگر Pop رو انجام دادیم، $Right$ رو باید برابر با $Max(Right, Right(Stack۱.Top()))$ قرار بدیم. و چون هر یک از اعضای $Stack$ اولمون نماینده ی یکی از دومینو های شروع هر یک از اون بازه های قرمز است، سر استک دوم رو هم Pop میکنیم. اینکار را آنقدر انجام میدهیم که $Start$ دومینوی سر استک اول بیش از $Right$ بشود. در آنصورت فاصله ی $Right$ با $Start$ سر استک برابر با X_i جدید میشود. پس به $Stack$ اول این برج را $Push$ میکنیم و به استک دوم،

مجموع سر آن بعلاوه ی این X_i را $Push$ میکنیم. اینطوری تمامی فرض های روی استک هامون برقرار خواهند شد. وقتی به دومینوی i ام رسیدیم، کافیست عدد سر $Stack$ دوم را در خروجی چاپ بکنیم.

حال باید این روش را برای تعداد بیش از ۱ زوج (i,j) گسترش بدیم. برای اینکار تمامی این زوج ها را به ترتیب نزولی بر حسب عضو اول $Sort$ بکنید. سپس به ترتیب از اول درخواست ها را بررسی میکنیم. در ابتدا یک DSU و ۲ تا $Stack$ میسازیم. هر وقت به درخواست (i,j) رسیدیم، عملیات بالا را از دومینویی که توقف کردیم (که در ابتدا راست ترین است) انجام میدهیم تا به راس i ام برسیم. حالا اینجا تفاوتی که داریم با حالت قبل این است که j امین دومینو لزوما راست ترین دومینویی نیست که ما ازش شروع کردیم و در نتیجه سر استک دوم جواب نخواهد بود. باید ببینیم در این طول، دومینوی j ام در کدام یک از آن $Component$ ها قرار دارد. بعد باید ببینیم این $Component$ چندمین از راست است. مثلا فرضا k امین از راست میشود. سپس جواب برابر با سر استک دوم منهای k امین عضو از ته استک دوم. خب حالا برای اینکه این $Component$ ها را هندل کنیم از DSU استفاده میکنیم. در عملیاتمان، هر وقت از $Stack$ اول Pop میکردیم، دو $Component$ مربوط به دومینوی حال حاضر در عملیات و دومینویی که از $Stack$ اول Pop کردیم را در DSU مان $Union$ میگیریم . حال اگر اینکار را به روش درختی پیاده سازی کنیم، برای هر $Node$ یک $Index$ تعریف میکنیم . و هر وقت که دومینو را خواستیم $Push$ کنیم به استک اول، از آنجایی که اعضای استک اول نماینده های اول $Component$ های مختلف اند، $Index$ که همان شماره ی اینکه چندمین $Component$ از راست است را کافیست بذاریم $Stack۱.size()$. در این صورت هر وقت بخواهیم ببینیم j امین دومینو توی کدوم $Component$ است کافیه یبار توی DSU مون $Find$ اش کنیم. و سپس $Index$ اش همان چیزی خواهد شد که ما بدنبالش هستیم.

و اینطوری میتوانیم تمامی جواب های فیلم برداری ها را در $O(nlog(n)+qlog(n))$ پیدا کنیم.