به نام او

پاسخ آزمون عملی نخست شاززز

۱۲ بهمن ۱۳۹۵

بنیامین و منبع MRX ..... BenjaminAndMRXsSource

اگر به ازای هر یک از پسورد ها یک راس در نظر بگیریم به صورتی که اگر در کامپیوتر اول باشد سفید و در غیر اینصورت سیاه باشد (n راس) و از راس i به راس j یال باشد اگر و تنها اگر برای به دست آوردن پسورد j، پسورد i نیاز باشد.

می‌دانیم این گراف DAG (گراف جهت دار بدون دور جهت دار) است چون هیچ پسوردی پیش نیاز خود نمی‌شود.

ما در هر حرکت می‌توانیم یکی از راس های ریشه (راس با درجه ورودی صفر) را حذف کنیم چون به پسورد آن دسترسی داریم اما اگر تغییر رنگ دهیم یک حرکت حساب می‌شود.

به صورت حریصانه در هر مرحله اگر راسی همرنگ راس که پیش از این حذف شده در میان راس های ریشه وجود داشت، آن را حذف می‌کنیم در غیر این‌صورت یک راس ریشه با رنگ جدید حذف می‌کنیم. هر دو حالت را نیز به ازای رنگ راس اولیه امتحان می‌کنیم.

اثبات درستی : فرض می‌کنیم روش بهتری وجود دارد اولین مرحله ای را فرض می‌کنیم که الگوریتم بهتر در حالی که ریشه سیاه دارد و رنگ راس حذف شده قبلی سیاه باشد ریشه سفید حذف کند (بدون کم شدن از کلیت مسئله)؛ در این صورت اگر در همین مرحله ریشه سیاه را حذف کنیم بعد هر راسی که در حالت های قبل می‌توانست حذف شود نیز همچنان می‌تواند حذف شود همچنین هیچ حرکتی اضافه نمی‌شود. پس تعداد حرکات بیشتر نمی‌شود.

پیاده سازی و تحلیل پیچیدگی : دو صف نگه می‌داریم که یکی تمام ریشه های سفید است و یکی تمام ریشه های سیاه بعد هر بار که یک راس حذف می‌شود می‌توان فهمید که کدام یک از همسایه‌هایش ریشه می‌شوند و آنها را به صف هایشان اضافه کرد. و زمانی که یک صف از یک رنگ خالی شد مجبوریم سراغ صف دیگر برویم. هر راس حداکثر یک بار حذف می‌شود و به ازای راس حذف شده از درجه ورودی تمام همسایه ها یکی کم می‌کنیم و راس های با درجه ورودی صفر را به صف ها اضافه می‌کنیم. پس به ازای هر یال نیز یک عملیات انجام می‌دهیم پس مجموع عملیات ها از $O(n+m)$ خواهد بود.

بنیامین و آخرین پسورد ..... BenjaminAndFinalPassword

کوتاه ترین پیشوند ناتهی برابر با پسوند یک رشته که برابر با کل رشته نیست را **حاشیه** آن رشته می‌گوییم. برای مثال حاشیه رشته $abab$، ab خواهد بود.

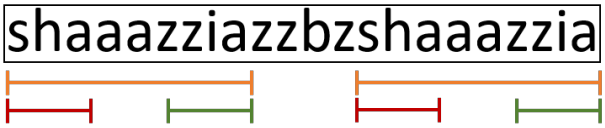
یک رشته **بدون حاشیه** است اگر هیچ پیشوندی برابر هیچ پسوند از آن نباشد.

لم ۱ : طول حاشیه یک رشته نمی‌تواند از $\frac{n}{2}$ بیش‌تر باشد.

برهان خلف: فرض می‌کنیم طول کوتاه ترین پیشوند و پسوند مشترک از $\frac{n}{2}$ باشد. قسمت مشترک نیز یک پیشوند و پسوند مشترک است که کوتاه تر از قبلی است. پس تاوقتی که بیش‌تر از $\frac{n}{2}$ باشد می‌توان آن را کوتاه کرد.

لم ۲: اگر رشته حاشیه رشته s را t بنامیم؛ رشته t بدون حاشیه است.

برهان خلف فرض می‌کنیم که t حاشیه داشته باشد، اگر حاشیه آنر را در نظر بگیریم حاشیه رشته s نیز خواهدبود درحالی که طول آن کمتر است.



مسئله kـامین رشته n حرفی را به ترتیب دیکشنری می‌خواهد. برای این کار حرف ها را از چپ به راست ابتدا a می‌گذاریم، اگر تعداد رشته های بدون حاشیه با پیشوند مشخص شده بیشتر مساوی k باشد برای پیشوند جدید kـامین را میابیم و در غیر اینصورت تعداد این رشته ها را از k کم می‌کنیم بعد این حرف را b می‌گذاریم و k امین رشته با پیشوند جدید را میابیم.

جلو می‌رویم و رشته res را که جواب است را به شکل گفته شده می‌سازیم.

$dp[i][j]$ را تعریف می‌کنیم تعداد رشته های بدون حاشیه به طول j که پیشوند به طول i آن با رشته res برابر است.

برای محاسبه $dp[i][j]$ ابتدا مقدار آن را ۲^{j-i} قرار می‌دهیم سپس حالت های نامطلوب را از جواب کم می‌کنیم.

در حالت های نامطلوب می‌دانیم که رشته حاشیه دارد به ازای هر طول k ($۱ \le k \le \frac{n}{۲}$) تعداد رشته های به طول j که طول حاشیه آن‌ها دقیقا k است را پیدا می‌کنیم و از $dp[i][j]$ کم می‌کنیم. طبق لم ۲ می‌دانیم که هر یک از این حاشیه ها خودشان رشته ای بی حاشیه است. که دو حالت وجود دارد :

- اگر $k \le i$ کل حاشیه در پیشوند مشخص شده است؛ پس تنها کافی است که قسمت مشخص شده بدون حاشیه بوده و باقی حروف نیز می‌تواند $۲^{max(۰,j-i-k)}$ حالت داشته باشند. (اگر پیشوند مشخص شده از $j-k$ نیز بیشتر باشد باید چک کرد که آن قسمت نیز برابر پیشوند باشد)
- اگر $k > i$ همچنان چون پیشوند به طول k حاشیه کل رشته است، باید بدون حاشیه باشد که تعداد حالت های این حاشیه برابر است با $dp[i][k]$ و بقیه حروف نیز $۲^{j-۲\times k}$ حالت دارند.

اکنون پس از اضافه شدن حرف iـام، $dp[i][n]$ تعداد رشته های بدون حاشیه با پیشوند مشخص شده را نشان میدهد که با روش گفته شده می‌توان کل رشته را پیدا کرد.

تحلیل پیچیدگی : هر بار که حرف i ام اضافه می‌شود به ازای j از i تا n $dp[i][j]$ محاسبه می‌شود که هر یک از آنها با $O(j)$ آپدیت می‌شود.

پس در مجموع به ازای اضافه شدن هر حرف از $O(n^۲)$ عملیات انجام داده می‌شود، پس مجموع عملیات ها از $O(n^۳)$ خواهد بود.

سفر بنیامین و ماری ..... BenjaminsAndMarysTravel

مدل سازی مسئله به گراف : به ازای هر شهر یک راس و به ازای هر نوع پرواز یک یال جهت‌دار از شهر مبدا به شهر مقصد پرواز در نظر می‌گیریم که وزن آن ممکن است تغییر کند.

جواب مسئله دارای دو حالت کلی است :

- حالت غیر ممکن : ابتدا با الگوریتم Dijkstra وجود حداقل یک روش برای سفر از راس ۱ به راس n در زمان h را بررسی می‌کنیم (بدین ترتیب که وزن یال v به u زمانی تعیین می‌شود که در الگوریتم Dijkstra روی راس v باشیم ، یعنی می‌دانیم کمترین زمانی که می توان به راس v رسید چیست و می‌خواهیم بهترین زمان برای رسیدن به بقیه راس ها را آپدیت کنیم).

  شرط لازم و کافی برای ممکن بودن این است که بعد از اجرای الگوریتم Dijkstra بهترین زمان رسیدن به راس n کمتر مساوی h باشد.

- در صورتی که فهمیدیم حداقل یک روش برای رسیدن از راس ۱ به راس n در زمان h وجود دارد باید بیشترین میزان لذت از سفر را محاسبه کنیم.

  فرض کنید پاسخ مسئله در این حالت برابر ans باشد یعنی سفری وجود دارد که کمینه زمان ماندن در کافه ها بیشتر مساوی ans باشد؛ پس اگر بتوانیم برای یک x دلخواه با زمان اجرای خوب بررسی کنیم آیا سفری وجود دارد که میزان لذت آن بیشتر مساوی x باشد یا نه ، آنگاه می توان کل مسئله را با BinarySearch حل کرد (چون اگر تابع چک باینری سرچ به ازای یک مقدار x مقدار $true$ برگرداند به ازای همه x های کوچکتر هم مقدار $true$ برخواهد گرداند.)

  حالا فرض کنید می‌خواهیم وجود سفری با حداقل میزان لذت x را چک کنیم ؛ کافی است دوباره همان الگوریتم Dijkstra در بخش اول را اجرا کنیم با این تفاوت که به ازای هر راس میانی مثل v (غیر از راس ۱ و راس n) وقتی قرار است از روی v بهترین زمان بقیه را آپدیت کنیم ابتدا بهترین زمان خود v را به علاوه ی x کنیم. و در نهایت باید بررسی کنیم که بهترین زمان راس n کمتر مساوی h شده باشد.

تحلیل پیچیدگی : با توجه به این که وزن یال‌ها را در هر وضعیت می‌توان در $O(۱)$ به دست آورد کل مسئله در $O(Dijkstra \times BinarySearch)$ یا $O(m \times log(n) \times log(h))$ حل می‌شود.

غافل از خویش مشو
یک سر موی
عمر آویخته از
یک سر موست

موفق باشید