Subject:
Date

۱- ایده حل: درخت کارتزین. درخت کارتزین، درختی باینری است که دارای خاصیت min-heap

است. یعنی اگر این درخت را به صورت میان ترتیب پیمایش کنیم، خود آرایه را به همان ترتیب اول،

خروجی می دهد [مثالش تو پاسخنامه هست]. حالا فرض کنید یه کوئری [۱,۲] داریم، و ۱' و ۲'

به ترتیب رئوس متناظر با عناصر A[1] و A[2] در درخت کارتزین A هستند.

پایین ترین جد مشترک ۱' و ۲' را در نظر بگیرید. این رأس در پیمایش میان ترتیب، حتماً بین ۱' و ۲'

است (تعریف پیمایش) چون درخت کارتزین هم از نوع هرم کمینه است، LCA از تمام فرزندان

و نوادگانش کوچکتر است. ← مینیمم بازه ی داده شده، همان LCA است.

۲- ایده حل: تشکیل گراف مسئله و اجرای الگوریتم دایسترا از رأس متناظر با تن تن

و با هدف رسیدن به مقصد. ← کمترین فاصله تا مقصد بدست می آید.

نتیجه ی منطقی ← اگر فاصله ی یکی از دزدها تا مقصد، کمتر از فاصله تن تن تا مقصد باشد،

تن تن رو می گیره در غیر این صورت، تن تن می تواند فرار کند. برای مدل سازی خودروها هم

که قسمت پیچیده ی سوالمه، می توانیم یک گراف با نصف وزن های گراف اول (چون ماشین در نصف

مدت زمان دزدها را منتقل می کند) ترسیم کنیم و از رئوس دارای ماشین گراف دوم، به گراف اول

یال جهت دار با وزن صفر وصل کنیم. بعد، از مقصد در گراف دوم، یه الگوریتم دایسترا اجرا می زنیم

تا کمترین فاصله ی دزدها پیدا شه.

Subject:

Date

۳- اگر ماتریس مجاورت را داشته باشیم، از قوانین و قضایای جبر خطی میشه ثابت کرد که اگر این ماتریس رو به توان k برسونیم، درایه‌ی (i و j) ام برابر تعداد مسیرهای به طول k از i به j است. پس کافیه ماتریس A^2 را حساب کنیم و به ازای هر درایه‌ی ناصفر از A و A^2 در گراف G^2 یال قرار بدیم.

برای لیست مجاورت: ابتدا ترانهاده‌ی G را تعریف می‌کنیم که نمادش G^T هست. G^T همان G است ولی با جهت‌های یال معکوس.

به ازای هر راس یه لیست نگه میداریم و برای مبدا u، بررسی می‌کنیم که آیا با ۲ تا واسطه به v رسیده یا نه. ترانهاده می‌تونه راحت‌تر این کار رو انجام بده. مثلاً از v به u رو باید تو ترانهاده بررسی کنیم، و این ممکنه کار راحت‌تری باشه.

PAPCO

Subject:
Date

۴- قسمت اول: با DFS مسیر مولفه همبندی را بدست آورد. (اما این فقط تعداد رو میگه و برای بدست آوردن خود مولفه‌ها، از یه الگوریتم به اسم Kosaraju استفاده میشه که با استفاده از G و G^T، مولفه‌های همبندی را حساب می‌کند. بعد از این کار، برای هر راس، یک متغیر scc قرار میدهیم که بیانگر مولفه‌ی قویا همبندی است که یک راس در آن قرار دارد. بعد برای هر یال uv، یک یال از u.scc به v.scc وصل می‌کنیم تا گراف تشکیل شود.

برای قسمت دوم، یک لیست تو در تو قرار میدهیم که A[i] بیانگر لیست رئوس حاضر در مولفه‌ی همبندی iام است. بعد یک DFS میزنیم و یال‌هایی که ۲ مولفه‌ی همبندی را به هم وصل می‌کند را حذف می‌کنیم. حالا برای اینکه یال‌هایی که داریم را طوری در مولفه‌ی همبندی قرار دهیم که کمترین اندازه‌ی E را داشته باشیم، می‌دانیم که کمترین تعداد یال لازم برای ساخت یک مولفه قویا همبند با n راس، n تا یال است که تشکیل دور می‌دهند. برای هر مولفه‌ی همبندی که داریم، برای رئوس آن دور تشکیل می‌دهیم. لیست رئوس را هم که از A داریم.

Subject:
Date

۵ـ یک راه‌حل ساده، الگوریتم بلمن ـ فورد است اما پیچیدگی زمانی Order-4 داره

پس از الگوریتم دیگری بنام John on استفاده می‌کنیم. این الگوریتم، یک بار بلمن ـ فورد

را اجرا می‌کند (با یک سری ملاحظات البته، که با انتهای الگوریتم برود) و |V| بار هم الگوریتم

دایسترا. اول با بلمن فورد حل می‌کنیم آیا دور منفی داره یا نه. بعد برای هر راس V در مجموعه

رئوس G، $h(v)$ که همان فاصله v است که توسط بلمن ـ فورد محاسبه شده را پیدا

می‌کنیم و برای هر یال از یال‌های G، وزن را آپدیت می‌کنیم

$$weight^{*} = weight(u,v) + h(v) - h(u)$$

بعد، به ازای همه رئوس در G، دایسترا می‌زنیم.

PAPCO

Subject:
Date

۶- در اینجا تعداد ماکسیمم گوی‌هایی که جابجا شوند مهم است. بازه‌های متوالی از گوی‌های انجام شده را به عنوان یک مجموعه (set) در نظر می‌گیریم. نماینده‌ی کل مجموعه هم مینیمم عناصر حاضر در مجموعه است (به دلیل شرط دوم سؤال). برای هر گوی جدید، اگر مجموعه‌ای برای a_i وجود نداشته باشد، این مجموعه را می‌سازیم و اگر وجود داشته باشد از قبل، و کمینه‌اش m باشد، مجموعه m-1 را می‌سازیم و این گوی را در خانه‌ی m-1 ام قرار می‌دهیم. در آخر، مجموعه m-1 را با دو مجموعه‌ی کناری‌اش (m-2, m) اجتماع می‌گیریم تا یک بازه‌ی یکپارچه بدست بیاید ← m تا ساخت و m تا اجتماع ← پیچیدگی زمانی سؤال

۷- یک روش ساده، DFS است اما پیچیدگی‌اش زیاد است. می‌توانیم از یک راس دلخواه DFS را اجرا کنیم و در هر مرحله که از یک راس Back track کردیم، آن راس را در stack ذخیره کنیم. هر وقت در پیمایش به راس ابتدایی رسیدیم و دیگر نتوانستیم DFS را ادامه دهیم، آن راس را pop می‌کنیم و از یک راس بازدید نشده‌ی دیگر، دوباره DFS می‌زنیم. در انتها راس موجود در بالای استک، راس مادر است.

اثبات درستی ← برهان خلف. موجود در پاسخنامه

Subject:
Date

۸- از DFS تغییریافته استفاده می‌کنیم. یعنی طوری آن را تغییر می‌دهیم که برای ویزیت کردن رئوس، تعداد دفعاتی تعریف می‌کنیم و تعداد مولفه‌های همبندی گراف را در یک متغیر ذخیره می‌کنیم، که در ابتدا صفر است. ← نتیجه می‌شود که رئوسی که در یک بار صدا زده شدن DFS-Visit توسط DFS، ویزیت می‌شوند، حتماً در یک مولفه‌ی همبندی هستند و برعکس.

یک تابع بازگشتی که کار اصلی DFS را انجام می‌دهد

اثبات در پاسخنامه

اما به طور کلی و شهودی، چون به هم راه دارند، پس در یک مولفه هستند.

۹- برهان خلف: فرض می‌کنیم e در MST نباشد ← پس آن را اضافه می‌کنیم. چون طبق فرض در ابتدا ما یک MST داشته‌ایم، پس حتماً بین u و v یک مسیر از قبل بوده و ما با اضافه کردن این یال، یک دور به دست آورده‌ایم. حالا یال با بیشترین وزن را حذف می‌کنیم تا دور بدون حذف بشه. با این کار، وزن کل ما نسبت به قبل کمتر می‌شود ← به تناقض رسیدیم چون در ابتدا فرض کرده بودیم MST داریم.

۱۵- طبق فرض، میدانیم بیشترین تعداد یال در مسیرها از راس source، برابر m است.

از خاصیت relax کردن یال ها که در درس داشتیم، میدانیم که پس از m بار پیمایش، تمام فواصل درست محاسبه شده اند و relax هم شده اند. پس یک متغیر جدید اضافه می کنیم که بررسی کند پس از هر پیمایش، آیا تغییری رخ داده یا نه. (از نوع Boolean)

شبه کد هم در پاسخنامه هست.