ساختمان داده‌ها و الگوریتم‌ها

نیم‌سال دوم ۹۷ــ۹۸

گردآورندگان: حسین ابراهیمی، شبنم شیخها

دانشکده‌ی مهندسی کامپیوتر

پاسخ تمرین سوم — درخت دودویی جستجو، ترای

درخت دودویی جستجو

مسئله‌ی ۱. سنگین‌ترین مسیر

در یک درخت دودویی وزن یک مسیر از راس a به b را مجموع اعداد رئوس این مسیر در نظر بگیرید. الگوریتمی خطی ارائه دهید که وزن سنگین‌ترین مسیر یک درخت را بیابد.

حل.

مسیرهایی که می‌تواند از هر گره عبور کند، به ۴ دسته زیر تقسیم می‌شوند:

الف) مسیری که به گره ختم می‌شود و به فرزندان گره نمی‌رود
ب) مسیری که از خود گره و فرزند راست آن می‌گذرد.
ج) مسیری که از خود گره و فرزند چپ آن عبور می‌کند.
د) مسیری که از گره و هر دو فرزند آن می‌گذرد.

برگ‌ها فقط می‌توانند حالت اول را داشته باشند و این حالت پایه برای ما می‌شود. حال کافی است از ریشه شروع کرده و به صورت بازگشتی مقادیر بالا را برای فرزندان آن حساب کنیم تا به برگ‌ها برسیم و پس از حساب شدن مقادیر بالا، برای هر گره ماکسیمم بگیریم.

▷

مسئله‌ی ۲. پیمایش درخت

الف) الگوریتمی از مرتبه زمانی خطی طراحی کنید که لیست مرتب شده از عناصر د.د.ج را خروجی دهد.

ب) با داشتن کدام یک از پیمایش‌های پیش‌ترتیب، میان‌ترتیب و پس‌ترتیب از یک د.د.ج می‌توان آن را به طور یکتا ساخت؟ اثبات کنید یا مثال نقض بیاورید.

ج) الگوریتمی ارائه دهید که با گرفتن پیمایش پیش‌ترتیب یک درخت جستجوی دودویی تمام برگ‌های درخت را (بدون ساختن درخت) پیدا کند

حل.

الف) بر روی درخت پیمایش میان‌ترتیب اجرا کنید که مرتبه زمانی آن خطی است.

ب)

۱. پیش‌ترتیب : بله. اولین عدد در پیمایش، مقدار ریشه درخت است، حال بر روی اعداد جلو رفته و اولین عددی که از ریشه بزرگتر می‌باشد را پیدا می‌کنیم. اعدادی که بین اولین عدد و عدد پیدا شده هستند برای زیر درخت سمت چپ ریشه و اعداد بعد از آن برای زیر درخت سمت راست ریشه هستند. حال به صورت بازگشتی همین کار را برای این دو مجموعه انجام می‌دهیم و در نهایت به ریشه می‌چسبانیم.

۲. میان‌ترتیب : خیر. خروجی این پیمایش لیست مرتب شده از اعداد داخل د.د.ج است پس برای هر دو د.د.جای که عناصر برابری در ساختار خود دارند، پیمایش میان‌ترتیب آن‌ها یکسان است.

۳. پس‌ترتیب : بله. مانند پیش‌ترتیب عمل می‌کنیم.

ج)

برای حل سوال از داده‌ساختار stack استفاده می‌کنیم به این صورت که لیست پیش‌ترتیب درخت را با دو پوینتر i و j پیمایش می‌کنیم که مقدار ابتدایی آن‌ها ۰ و ۱ است. حال اگر a[j] < a[i] بود بدین معناست که در زیردرخت سمت چپ ریشه حرکت می‌کنیم و هنوز به برگی نرسیده‌ایم برای همین، a[i] را در stack ، push می‌کنیم و مقدار هر دو پوینتر را ۱ واحد اضافه می‌کنیم.

حال اگر به حالتی برخوردیم که a[j] > a[i] بود و a[j] از مقدار بالای استک بزرگتر بود بدین معناست که a[i] برگ درخت می‌باشد و تا جایی که a[j] از مقدار بالای استک بزرگتر باشد از استک pop می‌کنیم. این روند را تا انتهای لیست انجام می‌دهیم که در این صورت اردر زمانی از O(n) خواهد بود.

▷

مسئله‌ی ۳. نیاکان مشترک

الف) الگوریتمی طراحی کنید که در د.د.ج عمیق‌ترین جد مشترک دو گره (LCA) را پیدا کند. دقت کنید عمیق‌ترین جد مشترک دو گره در واقع از سمت دو گره نزدیکترین جد مشترک است.

ب) با استفاده از قسمت قبل الگوریتمی طراحی کنید با داشتن د.د.ج و دو عنصر داده شده، بزرگترین عنصر در مسیر بین دو عنصر داده شده را پیدا کند.

ج) کم ترین فاصله بین دو راس متمایز x و y در ددج را بدست آورید. بهترین مرتبه زمانی برای حل این سوال چیست؟ راه حل خود را با تحلیل ارائه دهید

حل.

الف) از یکی از گره‌ها شروع کرده و پدر گره را بدست می‌آوریم و مقدار $visited$ آن را $true$ می‌کنیم. این عمل را روی اجداد این گره تکرار می‌کنیم تا به ریشه برسیم. از گره دیگر شروع کرده و اجداد این گره را پیدا می‌کنیم، حال اولین گره‌ای که در این اجداد مقدار $visited$اش $true$ بود،

عمیق‌ترین جد مشترک است.

ب) با استفاده از الگوریتم قسمت قبل LCA دو گره را بدست می‌اوریم. مسیر یکتا بین گره اول، LCA و گره دوم را در نظر بگیرید. از LCA شروع کرده و در هر مرحله فرزند راست گره فعلی را انتخاب می‌کنیم حال اولین گره‌ای که فرزند راستش در مسیر نباشد جواب مورد نظر است.

ج) از ریشه درخت شروع می‌کنیم، اگر مقدار x و y بزرگتر از مقدار ریشه بود، به فرزند راست ریشه می‌رویم، اگر هر دو کوچک‌تر باشند به فرزند چپ ریشه و اگر یکی از دو مقدار کوچکتر و دیگری بزرگ‌تر باشد در این صورت این گره، LCA برای دو گره x و y می‌باشد. (روش دیگری برای یافتن عمیق‌ترین جد مشترک) حال با جستجو از این گره، فاصله‌ی LCA تا x و y را بدست می‌آوریم. مجموع این فاصله‌ها،کمترین فاصله بین x و y است.

$\triangleright$

مسئله‌ی ۴. تشخیص د.د.ج

ای داده شده است، از مرتبه زمانی خطی تشخیص دهید که این درخت، یک درخت جستجوی دودویی است یا خیر.(رویه‌ی بازگشتی‌ای برای سوال ارائه دهید و مرتبه زمانی آن را تحلیل کنید.)

حل.

در ابتدا تمام مقادیر در درخت می‌توانند در بازه $[-\infty, +\infty]$ باشند حال از ریشه درخت شروع می‌کنیم، مقدار آن را r فرض کنید، حال تمام عناصر سمت راست ریشه باید در بازه‌ی $[r, +\infty]$ و همچنین عناصر سمت چپ ریشه در بازه $[-\infty, r]$ باشند، حال به صورت بازگشتی برای فرزند چپ و راست ریشه مقادیر بازه‌ای که به آن‌ها می‌رسد $update$ می‌کنیم تا به برگ‌ها برسیم. حال اگر عنصری وجود داشت که در بازه‌ای به آن گره می‌رسد قرار نداشته باشد درخت موردنظر د.د.ج نیست.

$\triangleright$

مسئله‌ی ۵. گذار در درخت

الف) یک د.د.ج و یک عدد داده شده‌است. روشی پیشنهاد کنید که نزدیکترین عنصر د.د.ج به عدد را پیدا کند.

ب) ابتدا روشی با استفاده از پیمایش د.د.ج پیشنهاد کنید که برای هر kداده شده، kمین عنصر درخت را در زمان خطی برگرداند. سپس با پیشنهاد کردن تغییری در ساختار د.د.ج کاری کنید که برای هر n عنصر داده شده بعد از قرار گیری عناصر در د.د.ج بتوانید در $O(h)$ که h ارتفاع درخت است برای هر kداده شده kمین عنصر درخت را برگردانید.

ج) با کمک ایده ی قسمت ب نشان دهید می‌توان در $O(h)$ جواب داد که چند عنصر بینa و b وجود دارد.

د) نشان دهید با شروع از هر راس در درخت جستجوی دودویی، می‌توان در $O(h+k)$، k عنصر بعدی آن را یافت که در آن، منظور از h ارتفاع درخت است.

حل.
الف) از ریشه درخت شروع می‌کنیم حال اگر عدد با ریشه برابر بود، خود عدد را خروجی می‌دهیم در غیر این صورت اگر عدد از مقدار ریشه بزرگتر بود پس باید نزدیک‌ترین عدد یا فرزند راست ریشه است یا در زیردرخت سمت راست ریشه هست اگر کوچکتر بود برای فرزند چپ ریشه چنین وضعیتی داریم، حال نزدیک‌ترین عدد را به عدد داده شده در زیر درخت موردنظر به صورت استقرایی پیدا می‌کنیم و سپس با ریشه مقایسه کرده و هر کدام نزدیک‌تر بود، خروجی می‌دهیم.

ب) برای پیدا کردن عنصر kام درخت، کافی است در زمان مرتبه خطی روی درخت پیمایش میان‌ترتیب کنیم و به سادگی عنصر kام در زمان خطی بدست می‌آید. برای حل قسمت دوم کافی است وقتی می‌خواهیم درخت را بسازیم، برای هر گره تعداد عناصر زیر درخت سمت چپش را ذخیره می‌کنیم. حال برای پیدا کردن عنصر kام از ریشه شروع می‌کنیم در صورتی که تعداد عناصر زیر درخت چپ ریشه که مثلا با N نشان می‌دهیم، $k - ۱$ باشد، ریشه عنصر مطلوب است. در غیر این صورت اگر N از k کوچک‌تر باشد عنصر مطلوب در زیر درخت سمت راستش وجود دارد. حالا باید در زیر درخت سمت راست عنصر $k - N - ۱$م را جستوجو کنیم که اینکار را به صورت استقرایی انجام می‌دهیم. در حالتی که N از k بزرگتر باشد هم به طور تقریباً مشابه است.

ج) در این قسمت علاوه بر تعداد عناصر زیر درخت سمت چپ هر گره، تعداد عناصر زیر درخت سمت راست آن را نیز نگه می‌داریم. حال برای حل سوال کافی است عمیق‌ترین جد مشترک a و b را بدست می‌آوریم حال می‌توان تعداد عناصر بین a تا b را به شکل زیر بدست می‌آوریم:

در مسیر b به lca هر جا فرزند چپ بوده باشیم، باید تعداد زیردرخت راست پدر بعلاوه خود پدر کم شود.

در مسیر a به lca هرجا فرزند راست بوده باشیم، باید تعداد زیردرخت چپ پدر بعلاوه خود پدر کم شود.

د) فرض کنید گره ابتدایی با sو گره kام بعد از آن را t بنامیم. مسیر یکتایی که بین این دو راس در درخت وجود دارد را p بنامید. همچنین پایین‌ترین جد مشترک آنها را با l نشان دهید. طول p حداکثر برابر $۲h$ است. برای یافتن جواب علاوه بر رئوس p، تنها زیردرخت‌هایی از رئوس مسیر به(جز s و t و l)، و به صورت کامل دیده می‌شوند (بدین معنی که در آنها راس اضافه‌ای که داخل جواب نباشد دیده نمی شود.) بنابراین هر راس از این زیردرخت‌ها، یکی از k راس مدنظر است که دقیقا هم یک بار دیده شده‌اند. پس هزینه‌ای به اندازه $۲h + k$ که از $O(h + k)$ است، کرده‌ایم.

$\triangleright$

مسئله‌ی ۶. وسط عناصر

الف) الگوریتمی طراحی کنید که در $O(logn)$ میانه‌ی دو آرایه‌ی مرتب را که هر کدام دقیقا n عضو دارند را بیابد.

ب) الگوریتمی طراحی کنید که با انجام پیش‌پردازشی از مرتبه زمانی $O(n)$ روی دو د.د.ج، بتواند میانه‌ی کل عناصر دو د.د.ج را در $O(h)$ پیدا کند که h، ارتفاع درخت بلندتر است و n مجموع عناصر دو درخت است.

دو د.د.ج پس از پیش پردازش باید قابلیتهای یک د.د.ج عادی را داشته باشد.

حل.

الف) حل سوال در $O(n)$ ساده است کافی است دو آرایه را ادغام کرده و عنصر وسط را خروجی دهیم. حال برای حل سوال در $logn$ باید میانه‌ی دو آرایه را با هم مقایسه کنیم، میانه آرایه اول را $m_۱$ و میانه آرایه دوم را $m_۲$ می‌نامیم، اگر $m_۱ = m_۲$ باشد، برابر با میانه دو آرایه هستند و الگوریتم تمام است. فرض کنید اگر $m_۱ > m_۲$ باشد، واضح است که میانه کل در نیمه دوم آرایه اول و نیمه اول آرایه دوم نیست. در نتیجه سایز مسئله به n کاهش می‌یابد و با استقرا بعد از $log(۲n)$ مرحله که از $O(logn)$ هست به جواب می‌رسیم.(اثبات کنید که میانه آرایه دوم با سایز n برابر است با مجموعه دو آرایه اولیه)

ب)

پیش‌پیمایش را به این صورت انجام می‌دهیم که آرایه میان‌ترتیب از هر دو درخت را بدست می‌آوریم که در زمان خطی قابل انجام است.

حال مسئله تبدیل می‌شود به پیدا کردن میانه دو آرایه مرتب که طول متفاوتی دارند و مجموع عناصر دو آرایه برابر با n است که این مسئله با استفاده از ایده‌ای مشابه با قسمت قبل در اردر زمانی O(logn) که کوچکتر از O(h) است، بدست می‌آید.

▷

مسئله‌ی ۷. درخت‌های آینه‌ای

الگوریتمی خطی ارائه دهید که با گرفتن دو درخت دودویی، مشخص کند که آیا قرینه‌ی آیینه‌ای هم هستند یا نه.

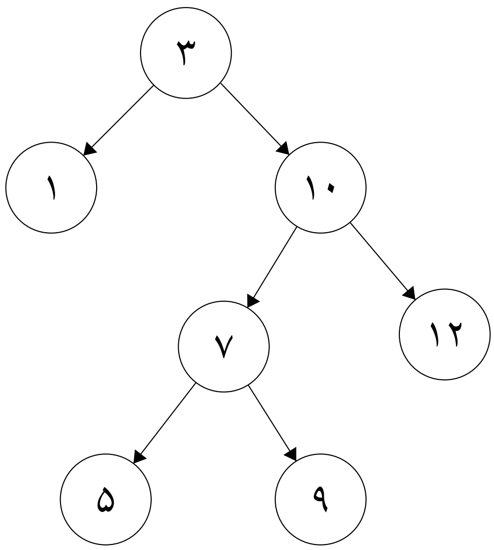
حل. از ریشه‌ی هر دو درخت شروع می‌کنیم ابتدا باید مقدار دو ریشه باید باهم برابر باشد سپس فرزند راست ریشه اول باید برابر با فرزند چپ ریشه دوم و همچنین فرزند چپ ریشه اول با فرزند راست ریشه دوم برابر باشد. حال این رویه را به صورت بازگشتی برای [فرزند راست ریشه اول و فرزند چپ ریشه دوم] و [فرزند چپ ریشه اول و فرزند راست ریشه دوم] اجرا می‌کنیم تا به برگ‌ها برسیم. ▷

مسئله‌ی ۸. مجموعه‌های جستجو

فرض کنید جستجو برای کلید k در یک BST به یک برگ ختم شود. سه مجموعه زیر را در نظر بگیرید: A، کلیدهای سمت چپ مسیر جستجو، B، کلیدهای روی مسیر و C، کلیدهای سمت راست مسیر. آیا به ازای هر سه کلید $c \in C$، $b \in B$ و $a \in A$ باید داشته باشیم $a \leqslant b \leqslant c$ ؟ در صورتی که جواب شما مثبت است اثبات کنید و در صورتی که منفی باشد، مثال نقض بیاورید.

حل.

خیر، برقرار نیست. مثال نقض:



مسیر جستجو را $<۳,۱۰,۷,۹>$ در نظر بگیرید. حال عنصر ۵ در مجموعه A و عنصر ۳ در مجموعه B قرار دارد. این دو در رابطه بیان شده صدق نمی‌کنند. $\triangleright$

ترای

مسئله‌ی ۹. ترتیب الفبایی

الگوریتمی از مرتبه‌ی زمانی $O(n)$ برای مرتب سازی n کلمه بر اساس ترتیب الفبایی ارائه دهید.

حل.

ابتدا کلمات را در ترای درج می‌کنیم. سپس پیمایش پیش‌ترتیب انجام می‌دهیم. چرا این پیمایش کلمات را سورت می‌کند؟ $\triangleright$

مسئله‌ی ۱۰. XOR بهینه

فرض کنید آرایه‌ای از اعداد ۳۲ بیتی داریم.

الف) به ازای عدد a، بیشترین XOR آن با اعضای آرایه را پیدا کند.

ب) در زمان $O(n)$ جفت عدد با کمترین XOR را پیدا کنید.

ج) در زمان $O(n)$ زیرآرایه با بزرگترین XOR را پیدا کنید.

حل.

الف) مدلی از ترای را در نظر بگیرید که یال‌های آن به جای حروف الفبا، نمایانگر بیت صفر و یک باشند. اعداد آرایه را در این ترای درج کنید. حال بیت‌های عدد a را در نظر بگیرید. عددی که بیشینه XOR با a دارد، تا حد امکان بیت‌هایش مخالف بیت‌های a است. با توجه به این نکته، چگونه در ترای دنبال این عنصر می‌گردید؟

ب) راه حل نسبتا غیر بهینه این است که ابتدا همه‌ی اعداد را در ترای درج کنیم. سپس عملیاتی که در مورد قبل در رابطه با a انجام دادیم را به ترتیب روی همه‌ی اعداد آرایه انجام دهیم و بیشینه مقدار را پیدا کنیم. راه حل بهینه تر چیست؟

ج) از رابطه‌ی زیر استفاده می‌کنیم:

$$\mathrm{xor}(arr[i:j]) = \mathrm{xor}(arr[۱:i-۱]) \oplus \mathrm{xor}(arr[۱:j])$$

یک ترای متشکل از XOR زیر آرایه‌های به شکل $arr[۱:i]$ می‌سازیم و این مقادیر را در جایی ذخیره می‌کنیم. سپس مقادیر را یکی یکی بررسی می‌کنیم و مطابق با مورد الف بیشینه XOR آن با اعضای ترای را بدست می‌آوریم. راه حل بهینه‌تر که هم‌چنین حافظه‌ي کمتری نیز بخواهد چیست؟

$\triangleright$

مسئله‌ی ۱۱. طولانی‌ترین پیشوند مشترک

الگوریتمی ارائه دهید که طولانی‌ترین پیشوند مشترک مجموعه‌ای از کلمات با حداکثر اندازه‌ي ثابت را در زمان خطی بدست آورد.

حل.
کلمات را در ترای درج می‌کنیم. از ریشه‌ی ترای شروع می‌کنیم. تا اولین راسی که بیش از یک فرزند داشته باشد ادامه می‌دهیم. حروف پیمایش شد، حروف طولانی‌ترین پیشوند مشترک هستند.
$\triangleright$

مسئله‌ی ۱۲. داستان‌نویسی

داستانی به طول n که از کلمات حداکثر k حرفی (k عدد ثابت) تشکیل شده است داریم.

الف) الگوریتمی ارائه دهید که تعداد کلمات متفاوت آن را در زمان $O(kn)$ بدست آورد.

ب) الگوریتمی ارائه دهید که با انجام پیش‌پردازشی از $O(kn)$ تعداد دفعات تکرار کلمه‌ي w را در در زمان $O(k)$ بدست آورد.

ج) الگوریتمی ارائه دهید که کلمه‌ي با بیشترین تکرار را پیدا کند.

حل.

الف) کلمات را در ترای درج می‌کنیم. سپس ترای را پیمایش می‌کنیم و تعداد راس‌هایی که انتهای کلمات را نشان می‌دهند تعداد کلمات متفاوت هستند.

ب) هنگامی که عنصری را در ترای درج می‌کنیم به جای اینکه به متغیری نشان‌دهنده‌ی انتهای کلمه داشته باشیم، متغیری برابر با تعداد کلمات در نظر میگیریم. هرگاه کلمه‌ای را درج کردیم، این متغیر را یکی افزایش می‌دهیم.

ج) مشابه با مورد قبل کلمات را در ترای درج می‌کنیم و سپس ترای را پیمایش می‌کنیم و ماکسیمم تعداد را به دست می‌آوریم.

▷

مسئله‌ی ۱۳. ترای پیشرفته

متنی شامل رشته‌هایی از حروف داریم (به عبارتی، یک داستان :))). با ایجاد تغییراتی در ساختار ترای هنگام وارد کردن رشته‌ها، قابلیت‌های زیر را به ترای اضافه کنید.

الف) ویژگی auto-complete. به عبارتی به ازای رشته‌ی ناقص w اگر این رشته پیشوند رشته‌های از قبل ذخیره شده‌ای باشد، آن ها را چاپ کند.

ب) برگرداندن تعداد رشته‌هایی که رشته‌ی w پیشوند آن است.

حل.

الف) کلمات را در ترای درج می‌کنیم. سپس به ازای رشته‌ی w داده شده، ابتدا آن را در ترای پیدا می‌کنیم، سپس از راس انتهایی آن پیمایش پیش‌ترتیب می‌زنیم.

ب) برای هر راس ترای یک متغیر در نظر می‌گیریم. هنگام درج کردن هر کلمه در ترای هرگاه از راسی رد شدیم، متغیر تعریف شده را یکی افزایش می‌دهیم. کلمه‌ی w را سرچ می‌کنیم، مقدار این متغیر در راس آخر آن جواب مسئله است.

▷

موفق باشید :)