دانشکده‌ی مهندسی کامپیوتر

ساختمان داده‌ها و الگوریتم‌ها

نیم‌سال دوم ۹۷ـ۹۸

گردآورندگان: حسین ابراهیمی، شبنم شیخها

۱۹ فروردین

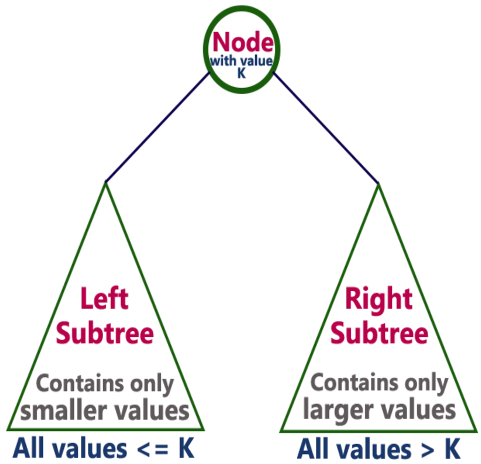
پاسخ کوییز سری دوم

مسئله‌ی ۱. دنباله‌ی جستجو

جست‌وجو به دنباله‌ای از اعداد گفته می‌شود که هنگام جست‌وجوی یک عدد در درخت دودویی جستجو با شروع از ریشه و نوشتن اعداد مربوط به راس‌هایی که در مسیر جست‌وجو می‌بینیم بدست می‌آید. دنباله‌ای از n عدد طبیعی داده شده است، الگوریتمی از $O(n)$ ارائه دهید که ببینیم آیا این دنباله می‌تواند دنباله‌ی جست و جو مربوط به یک درخت دودویی جستجو دلخواه باشد یا خیر.

حل.

برای حل سوال باید به این ویژگی از داده‌ساختار BST توجه کنیم که برای هر گره، تمامی عناصری که در زیردرخت سمت راست آن قرار دارند، دارای مقادیر بزرگتر از ریشه و عناصری که در سمت چپ زیردرخت هستند مقادیری کوچک تر از ریشه دارند.



حال وقتی داریم عنصری را در درخت جست‌وجو می‌کنیم، مسیری از ریشه به یکی از گره‌ها را طی می‌کنیم. در این میان وقتی به هر گره‌ای می‌رسیم یا به زیردرخت سمت راست آن می‌رویم که در این صورت باید تمامی اعدادی که بعد از آن می‌آیند باید از مقدار آن گره بیشتر باشند، یا به زیردرخت سمت چپ آن می‌رویم که به طور مشابه باید اعداد بعد از آن مقدار کمتری از گره موردنظر داشته باشند. حال کافی است برای دنباله داده شده، ویژگی بالا را بررسی کنیم.

دنباله جستجو را آرایه $a_0, a_1, \cdots, a_n$ درنظر بگیرید. a_0 مقدار ریشه درخت می‌باشد. حل سوال در مرتبه زمانی $O(n^2)$ ساده است، کافی است برای هر مقدار a_i را با مقدار a_{i+1} مقایسه کنیم و

سپس برای هر $j \geqslant i + ۲$ باید همان رابطه بین a_i و a_j برقرار باشد یعنی مثلا اگر $a_i < a_{i+۱}$ بود باید $a_i < a_j$ باشد.

حال برای اینکه سوال را در مرتبه زمانی $O(n)$ حل کنیم، بازه‌ای را که هر عنصر بایستی در آن باشد را تعیین کنیم بدین منظور برای هر i، ابتدا $a_{i+۱}$ در بازه‌ای که به آن می‌رسد باید قرار داشته باشد که در غیر این صورت دنباله، دنباله جستجو نیست. سپس a_i را با $a_{i+۱}$ مقایسه می‌کنیم .اگر $a_i < a_{i+۱}$ بود یعنی در زیردرخت سمت راست آن قرار داریم پس حد پایین بازه را $update$ می‌کنیم و اگر $a_{i+۱} < a_i$ باشد، به طور مشابه چون در زیر درخت سمت چپ هستیم، حد بالا را $update$ می‌کنیم. در حالت تساوی آن‌قدر جلو می‌رویم که در یکی از نامساوی‌های بالا صدق کند.

از ابتدای آرایه شروع می‌کنیم، $a_۰$ می‌تواند در بازه $[-\infty, +\infty]$ قرار داشته باشد. سپس $a_۱$ باید در بازه موردنظر قرار داشته باشد که بدیهی است قرار دارد. حال بازه را با توجه به رابطه بین $a_۰$ و $a_۱$، $update$ می‌کنیم به این صورت که اگر $a_۱$ از $a_۰$ بزرگتر بود بازه به $[a_۰, +\infty]$ تغییر می‌کند و اگر کوچکتر بود به این بازه $[-\infty, a_۰]$، سپس $a_۲$ را در این بازه چک می‌کنیم. $update$ کردن و چک کردن را تا انتهای آرایه انجام می‌دهیم و اگر در جایی عنصری در بازه‌ای که به آن می‌رسد، قرار نداشته باشد دنباله جستجو نیست. در غیر این صورت دنباله‌ای جستجو داریم. چون یک بار کل آرایه را پیمایش می‌کنیم، مرتبه زمانی الگوریتم از $O(n)$ است. $\triangleright$

مسئله‌ی ۲. زیر آرایه‌های مطلوب

آرایه‌ای به طول n از اعداد ۳۲ بیتی داریم. به ازای عدد ثابت ۳۲ بیتی k در زمان $O(n)$ تعداد زیر آرایه‌های با XOR حداکثر k را پیدا کنید.

حل.

از ابتدای آرایه شروع می‌کنیم و XOR اعضای آرایه از ابتدا تا اندیس کنونی را در ترای درج می‌کنیم. به عبارتی یک ترای شامل $\mathrm{XOR}(arr[۱ : i])$ داریم که $۱ \leqslant i \leqslant n$. هنگام ساختن ترای، در هر راس متغیری که نشان‌دهنده‌ی تعداد XORها (تا آنجا) در زیردرخت آن است را نیز نگه می‌داریم. (چطور این متغیر را مقداردهی کنیم؟)
در حین ساختن ترای، پس از درج کردن هر عنصر، برای آن بدین صورت عمل می‌کنیم.
به رابطه‌ی زیر دقت کنید:

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

می‌خواهیم به ازای هر $\mathrm{XOR}(arr[۱ : i])$ ببینیم چند زیر آرایه در ترای موجود است که XOR آن‌ها با $\mathrm{XOR}(arr[۱ : i])$ حداکثر k بشود. بیت j عدد $\mathrm{XOR}(arr[۱ : i])$ را a_j که $۱ \leqslant j \leqslant ۳۲$ در نظر بگیرید. فرض کنید k عدد ۰۱۱۰ باشد (برای سادگی اعداد ۴ بیتی در نظر گرفته شده‌اند).
از ریشه‌ی ترای شروع می‌کنیم. باید به سمت رأسی حرکت کنیم که XOR آن با $a_۱$ برابر با صفر شود. زیرا اگر برابر با ۱ شود، یعنی بیشتر از k شده‌است. در حرکت بعدی اگر به سمت رأسی حرکت کنیم که XOR آن با $a_۲$ برابر با صفر شود، همه‌ي زیررشته‌های بعدی XORشان با $\mathrm{XOR}(arr[۱ : i])$ کمتر از k خواهد شد (چرا؟). پس تعداد کلمه‌های آن زیردرخت را با

استفاده از متغیری که قبلا تعریف کردیم به جواب اضافه می‌کنیم. اگر به سمت رأسی حرکت کنیم که XOR آن با $a_۲$ برابر با یک شود، باید مجدد به رأس بعدی نگاه کنیم و ... .
به علت ثابت بودن طول اعداد، درج کردن هر XOR در ترای در زمان ثابت انجام می‌شود.
همچنین با صرف هزینه‌ی ثابت برای هر XOR می‌توان تعداد زیررشته‌هایی که XORشان با آن حداکثر k است را بدست آورد. پس در کل هزینه‌ی $O(n)$ صرف خواهیم کرد.

$\triangleright$

موفق باشید :)