پاسخ تمرین سری دوم درس طراحی الگوریتم‌ها

مدرس: علی شریفی زارچی
گروه طرح سؤال: آقایان مهدی صفرنژاد بروجنی، علیرضا رضایی، اشکان نوروزی فرد، مهران خدابنده و خانم ها مریم علی اکبرپور و نرگس نوروزی

۲۹ مهر ۱۳۹۰

۱

سؤال

آرایه n عضوی $x_1, x_2, \ldots, x_n$ از اعداد صحیح داده شده است. با استفاده از حداکثر $O(n)$ مقایسه، عضوی را (در صورت وجود) پیدا کنید که بیش تر از $\frac{n}{2}$ بار تکرار شده باشد.

پاسخ

راه حل $O(n \log n)$ با روش تقسیم و حل:

رشته را به دو قسمت مساوی تقسیم می کنیم و الگوریتم را روی هر قسمت انجام می‌دهیم. بر اساس جواب های بدست آمده از دو فراخوانی ۴ حالت پیش می آید که با بررسی آنها در زمان $2n$ جواب مسئله برای آرایه ی کلی را بدست می آوریم. با حل $T(n) = 2T(n/2) + 2n$ به زمان $T(n) = O(n \log n)$ می رسیم.

راه حل اول $O(n)$:

اگر همچین عددی وجود داشته باشد پس میانه هم هست. فرض می کنیم به کمک الگوریتمی خطی بتوانیم میانه را پیدا کنیم. بعد از پیدا کردن میانه همه اعداد را با آن مقایسه می کنیم و بررسی می کنیم که آیا این عدد بیشتر از $n/۲$ بار تکرار شده یا نه. الگوریتم های مختلفی برای پیدا کردن میانه در $O(n)$ وجود دارد. به طور مثال الگوریتمی تصادفی مشابه $Quicksort$ برای این کار وجود دارد و الگورتیمی قطعی و کمی پیچیده تر که در کتاب موجود است.

راه حل دوم $O(n)$:

اعداد را به $\lfloor n/۲ \rfloor$ جفت افراز می کنیم (غیر از حداکثر یک عدد) و اعداد هر جفت را با هم مقایسه می کنیم. جفت های نامساوی را دور می ریزیم. می توان نشان داد این کار جواب مسئله را در صورت وجود تغییر نمی دهد. از جفت های مساوی نیز یکی را دور می ریزیم و یکی را نگه می داریم. عدد تکی (در صورت وجود) نگاه داده می شود. می توان نشان داد این بار نیز در صورت وجود جواب بدون تغییر باقی می ماند. و اعداد حداکثر $\lceil n/۲ \rceil$ شده اند. این کار را می توان ادامه داد و بنابر این می توان مجموعه را به یکی کاهش داد. و برای تعداد مقایسه داریم $T(n) = T(\lceil n/۲ \rceil) + \lfloor n/۲ \rfloor$. که $T(n) = O(n)$. در آخر باید عدد به دست آمده را با کل آرایه اولیه مقایسه کنیم که آیا تعداد تکرار آن بیشتر از نصف هست یا خیر.

۲

سؤال

n عدد طبیعی داریم و می خواهیم بزرگترین و کوچکترین آن‌ها را پیدا کنیم. با استفاده از روش تقسیم و حل الگوریتمی ارائه دهید که این کار را با تعداد مقایسه کمی انجام دهد. تعداد مقایسه ها را به دست آورید.

پاسخ

با فرض کردن عدد n به عنوان یک توان دو مسئله حل می‌شود و برای عدد های غیر از توان دو با استفاده از حدود توان دو $order$ مربوط بدست می‌آید. برای حل ابتدا آرایه را به دو قسمت تقسیم می کنیم و برای هر قسمت بزرگترین و کو چکترین عنصر ها را بدست می آوریم و با مقایسه بزرگترین عدد ها با هم و کوچکترین ها با هم بزرگترین و کوچکترین عدد کل آرایه را بدست می آوریم.
تعداد مقایسه ها از رابطه $T(n) = T(\lfloor n/۲ \rfloor) + T(\lceil c/۲ \rceil) + ۲$ به دست می آید که برابر $\lceil ۳n/۲ \rceil - ۲$ می شود.

۳

سؤال

k آرایه با اندازه n از اعداد مرتب شده به صورت صعودی داریم. می خواهیم این آرایه ها را طوری با هم ترکیب کنیم که یک آرایه کلی از اعداد مرتب شده به صورت صعودی داشته باشیم که دارای همه این nk

عدد باشد.
۱. راه حل پیشنهادی اول این است که ابتدا آرایه اول و دوم را ترکیب کنیم و سپس آرایه ترکیب شده را با آرایه سوم و سپس حاصل را با آرایه چهارم و الی آخر ترکیب کنیم. زمان اجرای لازم برای انجام این الگوریتم را بدست آورید.
۲. الگوریتمی با زمان اجرای کمتر و با استفاده از روش تقسیم و حل ارائه کنید و زمان اجرا را بدست آورید.

پاسخ

مرتبه‌ی الگوریتم راه‌حل اول: اگر مشابه ادغام $Merge\ sort$ عمل کنیم، برای ادغام دو رشته‌ی مرتب به طول $L_۱$ و $L_۲$ از $O(L_۱ + L_۲)$ زمان نیاز داریم. پس زمان لازم برای هر مرحله به صورت زیر است:

$$\begin{gathered} n + n = ۲n \\ ۲n + n = ۳n \\ \vdots \\ (k-۱)n + n = kn \end{gathered}$$

پس در مجموع این الگوریتم از مرتبه‌ی $n(\frac{k(k+۱)}{۲} - ۱)$ یا همان $O(k^۲ n)$ است.
الگوریتم پیشنهادی: از روش تقسیم و حل استفاده می‌کنیم. برای راحتی کار فرض کنید که عدد k توانی از ۲ باشد. آرایه‌ها را به دو دسته $\frac{k}{۲}$ تقسیم می‌کنیم. مسئله را برای هر دو دسته حل می‌کنیم و دو آرایه مرتب شده مربوط به این دو دسته را با روش ادغام در $Merge\ sort$ ادغام می‌کنیم. با کمی دقت در می‌یابیم که زمان اجرای این الگوریتم از مرتبه‌ی $O(nk\log k)$ است.

۴

سؤال

رئوس یک چند ضلعی محدب به ترتیب پادساعت گرد داده شده است. می دانیم هیچ دو راس آن مختصات x یا y برابر با یکدیگر ندارند. رئوس چند ضلعی با شروع از سمت چپ ترین راس (با کمترین x) داده شده است. راس با بیشترین y (بالاترین راس) و راس با بیشترین x (راست ترین راس) را در $O(\log n)$ پیدا کنید.

پاسخ

یک آرایه را $unimodal$ می نامیم هرگاه از ابتدای آرایه تا جایی اعداد به صورت صعودی باشند و از جایی به بعد نزولی. حال ادعا می کنیم این نقطه تغییر جهت در چنین آرایه ای را می توان در $O(\log n)$ بدست آورد. ابتدا عنصر میانی را نگاه می کنیم اگر از عدد قبل بزرگ تر و از عدد بعد کوچک تر باشد در قسمت صعودی آرایه قرار دارد و می توانیم نقطه تغییر را در نیمه دوم آرایه که آن هم $unimodal$ هست جستجو کنیم. همین طور اگر از عدد قبل کوچک تر و از عدد بعد بزرگ تر بود در قسمت نزولی قرار دارد و جواب در نیمه اول آرایه هست. و اگر از عنصر بعد بزرگ تر و از عنصر قبل هم بزرگ تر باشد خود، عنصر مورد

نظر است.
به همین صورت مانند جست و جوی دودویی ادامه می دهیم تا این عنصر را پیدا کنیم. حال توجه می کنیم که این نقاط بر حسب مولفه x، $unimodal$ هستند و نقطه با بیشترین x هم همان نقطه تغییر جهت است. پس در $O(\log n)$ می شود آن را پیدا کرد. سپس اگر از این نقطه تا انتهای نقطه ها به آرایه نگاه کنیم بر حسب مولفه y، $unimodal$ است و عنصر با بیشترین y نقطه تغییر جهت است و آن را می توان در $O(\log n)$ پیدا کرد.

۵

سؤال

در یک آرایه معمولی عضو میانه عضو $\lfloor \frac{n}{۲} \rfloor$ آرایه درحالت مرتب شده است. اما اگر هرکدام از اعضای آرایه یعنی x_i ها یک وزن w_i هم داشته باشند به طوری که $\sum_{i=۱}^{n} w_i = ۱$ میانه وزن دار چنین آرایه ای را اینگونه تعریف می کنیم :
آخرین اندیس i در حالت مرتب شده به طوری که داشته باشیم $\sum_{j=۱}^{i} w_i \leq \frac{۱}{۲}$
حال فرض کنید الگوریتمی داریم که میانه یک آرایه بدون وزن را در $O(n)$ محاسبه می کند. با استفاده از این الگوریتم میانه وزن دار آرایه داده شده را در $O(n)$ بدست اورید.

پاسخ

ابتدا عنصر میانه را پیدا می کنیم حال روی آرایه حرکت می کنیم و وزن عناصری را که از عنصر میانه کوچک تر هستند جمع می کنیم و این مقدار را k می نامیم. اگر k از $\frac{۱}{۲}$ بزرگتر باشد میانه وزن دار در نیمه اول آرایه قرار دارد یعنی کافیست عناصر کوچک تر از میانه را در یک آرایه بریزیم و اخرین عنصری در این آرایه که جمع وزن ها تا انجا از $\frac{۱}{۲}$ کوچکتر است را پیدا کنیم. و اگر k از $\frac{۱}{۲}$ کوچکتر باشد این عنصر در نیمه دوم آرایه قرار دارد و باید در نیمه دوم به دنبال اخرین عنصری بگردیم که جمع وزن از میانه تا انجا حداکثر $\frac{۱}{۲} - k$ باشد. به همین صورت می توان بازگشتی عمل کرد و در هر مرحله میانه آرایه فعلی را پیدا کرد و این اعمال را انجام داد. به این صورت برای تحلیل زمانی رابطه زیر را داریم: $T(n) = T(n/۲) + O(n)$ و بنابر این رابطه $T(n) = O(n)$ می باشد.

۶

سؤال

در کشور بربرستان قانونی در بانکهای آن وجود دارد که می توان n بربر (واحد پول بربرستان) را به بانک داد و به جای آن سه مقدار $\lfloor \frac{n}{۲} \rfloor$ و $\lfloor \frac{n}{۳} \rfloor$ و $\lfloor \frac{n}{۴} \rfloor$ را از بانک گرفت اما چون در این کشور فقط مقادیر طبیعی به عنوان پول وجود دارد ممکن است مقداری از پول شما بسوزد. یعنی اگر به بانک ۵ بربر بدهید، ۲ و ۱ و۱ بربر از بانک می توانید بگیرید که به صرفه نیست. به ازای مقدار n بربر نشان دهید حداکثر چه مقدار پول می توان بدست آورد.

پاسخ

مسئله را به صورت استقرایی حل می‌کنیم.
پایه‌ی استقرا: جواب مسئله برای اعداد ۱ تا ۳ به سادگی قابل محاسبه است.
فرض استقرا: فرض کنید برای اعداد کمتر از n بیشترین مقدار پول که با تبدیل n بربر به دست می‌آوریم را داریم. (اگر تبدیل مقدار n سودی نداشت این مقدار را برابر با خود n در نظر می‌گیریم.)
گام استقرا: حال کافی است که مجموع مقدار این پول را برای هر سه مقدار $\lfloor \frac{n}{2} \rfloor$ و $\lfloor \frac{n}{3} \rfloor$ و $\lfloor \frac{n}{4} \rfloor$ محاسبه کنیم و با مقدار n مقایسه کنیم. اگر این مقدار بیشتر از n که تبدیل بیشترین مقدار حاصل از تبدیل n را برابر با همین مجموع در نظر می‌گیریم. در غیر این صورت خود n برابر با بیشترین مقدار پول خواهد بود.
توجه کنید که این الگوریتم را می‌توان به نحوی $dynamic$ نیز دانست و به سادگی از $O(n)$ قابل پیاده‌سازی است. اگر از روش ممویز استفاده کنید، قابلیت پیاده سازی از مرتبه‌ی $\log n$ را نیز دارد. از آنجایی که حل مسئله برای $n/4$ زیر مجموعه‌ی مسئله‌ی $n/2$ است. می‌توانیم مقادیر محاسبه شده را در آرایه‌ای نگهداری کنیم و نیازی به محاسبه‌ی سود برای $n/4$ نداریم و مسئله تنها به دو زیر مسئله‌ی $n/3$ و $n/2$ تبدیل می‌شود.

۷

سؤال

یک رشته متقارن است اگر خودش با برعکس خودش برابر باشد، مانند «مادام». الگوریتمی از زمان $O(n^2)$ ارائه دهید که حداقل تعداد حرفی که لازم است به رشته ورودی (با طول n) اضافه کنیم تا رشته متقارن شود را به دست آورد.

پاسخ

مسئله را به روش پویا حل می‌کنیم. آرایه پویا زیر را تعریف می‌کنیم:
$d[i][j]$ را برابر با حداقل تعداد حروف اضافه شده برای متقارن کردن رشته‌ی شامل عناصر i تا j در نظر می‌گیریم.
مقدار دهی اولیه: واضح است که $d[i][i] = 0$
نحوه پر کردن: اگر عنصر اول و آخر رشته با هم برابر نباشد، باید به اول یا آخر یک عضو اضافه کنیم تا اول و آخر شبیه هم شوند و هزینه‌ی متقارن کردن رشته‌ی بین این دو عنصر را نیز اضافه کنیم. یعنی:

$$d[i][j] = min(d[i][j-1]+1, d[i+1][j]+1)$$

حال اگر عنصر i با j برابر باشد، یعنی ابتدا و انتهای رشته متقارن است. پس کافیست تنها هزینه‌ی رشته‌ی بین این دو حرف را در نظر بگیریم.

$$d[i][j] = min(d[i][j-1]+1, d[i+1][j]+1, d[i+1][j-1])$$

برای پر کردن این جدول در مرحله kام، خانه های $d[i][i+k]$ را پر می کنیم. جواب مسئله برابر با $d[1][n]$ می‌شود و مرتبه‌ی الگوریتم برابر با $O(n^2)$ است.

۸

سؤال

فرض کنید می خواهید به سفری بروید که در طول مسیر n هتل وجود دارند. این هتل ها در فواصل $a_۱ < a_۲ < ... < a_n$ از مبدا قرار دارند. در طول سفر تنها می‌توان در این هتل ها اقامت داشت و شما می توانید انتخاب کنید که در طول سفر در کدام هتل ها اقامت داشته باشید. هتلی که در فاصله a_n از مبدا قرار دارد مقصد شما می باشد. شما میتوانید در روز ۲۰۰ مایل مسافرت کنید ولی این امکان به دلیل محدودیت هتل ها همیشه امکان پذیر نمی باشد. جریمه برای میزان سفر در هرروز برابر $(۲۰۰ - x)^۲$ می باشد که x طول سفر در آن روز به مایل می باشد. هدف شما بدست آوردن ترتیبی از استراحت در هتل ها است که کمترین میزان جریمه کل سفرها را داشته باشید. الگوریتمی کارا ارائه دهید که ترتیب هتل های محل توقف را برای این بهترین سفر ارائه دهد.

پاسخ

این مسئله را به روش پویا حل می‌کنیم. فرض کنید $d[i]$ نشان‌دهنده کمترین میزان هزینه برای رسیدن به هتل iام باشد. برای راحتی کار فرض می‌کنیم هتلی در مختصات $a_. = ۰$ (که با هزینه‌ی صفر به آنجا رسیده‌ایم) وجود دارد. اگر فرد هتل i را انتخاب کند شب قبل را در هتلی مانند هتل jام سپری کرده است که هم $a_i - a_j < ۲۰۰$ و هم میزان $d[j] + (a_i - a_j)^۲$ کمینه باشد. پس میزان $d[i]$ به سادگی بر اساس $i - ۱$ مقدار قبلی آن قابل محاسبه است و مرتبه‌ی این محاسبه $O(i)$ است. واضح است که جواب مسئله برابر با $d[n]$ است و مرتبه‌ی پیچیدگی برابر با $O(۱ + ۲ + \cdots + n)$ یا همان $O(n^۲)$ است.