به نام خدا

جواب تمرین تئوری اول طراحی الگوریتم

تقسیم و حل

سینا کچویی

سوال ۱.

سوال یک. منچسترسیتی

n تیم در لیگ جزیره شرکت می‌کنند. پس از پایان رقابت های لیگ ، در مجموع مسابقات هر تیم تعدادی گل به ثمر رسانده و تعدادی گل نیز دریافت کرده است. تیم‌هایی را «برتر» می‌نامیم که هیچ تیمی در لیگ وجود نداشته باشد که از آن‌ها بیشتر گل زده و کمتر گل خورده باشد. با این فرض که تعداد گل های خورده و گل های زده هیچ دو تیمی برابر نیست، الگوریتمی از مرتبه زمانی$O(nlogn)$ ارائه دهید که تمامی تیم‌های برتر لیگ جزیره را پیدا کند.

راه حل. تیم ها را نصف کرده و هر تکه به ترتیب از کمترین تا بیشترین گل خورده مرتب می کنیم (هیچ دو تیمی گل خورده مساوی ندارند و این مرتب سازی یکتاست) . حال هر تکه که به صورت بازگشتی مرتب شده تحویل گرفتیم، را درس همدیگر ترکیب می کنیم و مانند merge sort عمل می کنیم فقط با این تفاوت که بیشترین گل زده شده تیم های درون مجموعه را نگهداری می کنیم بدین صورت که در هر گام تیم با کمترین گل خورده بین دو تکه را به مجموعه نهایی اضافه کرده و اگر بیشترین گل زده شده در مجموعه قبلی از تیم کنونی کمتر باشد، تیم فعلی برتر است و تعداد آنها افزایش می‌یابد (چرا که از میان تیم هایی که گل کمتری از تیم فعلی زده اند، هیچ تیمی گل بیشتر نزده !) در غیر اینصورت برتر نیست (چرا؟) با ادامه این روند می‌توانید تعداد تیم های برتر را بدست آورید و تیم ها مرتب شده باشند که میتوانیم در خروجی آن را بدهیم .

سوال ۲.

سوال دوم. اورست

ماتریس دو بعدی A از اعداد طبیعی با m سطر و n ستون در اختیار داریم . «قله»ی ماتریس خانه ای مانند i, j است که از چهار خانه‌ی مجاورش نا کوچکتر باشد. به عبارتی دیگر قله در شرایط زیر خواهیم داشت:

$$A[i,j] \geq A[i+1,j] A[i,j] \geq A[i,j+1] A[i,j] \geq A[i-1,j] A[i,j] \geq A[i,j-1]$$

الگوریتمی با پیچیدگی زمانی $O(nlogm)$ ارائه دهید که یکی از قله های ماتریس را پیدا کند. دقت کنید هر ماتریس حداقل یک قله دارد (کافی است عدد بیشینه در ماتریس را در نظر بگیرید). برای حل مشکل وجود نداشتن درایه‌های همسایه ها، فرض کنید که حاشیه ماتریس با مقادیر $-\infty$ پر شده است.

راه حل. این مسئله مانند مسئله یافتن قله در یک آرایه‌ی یک بعدی است با این تفاوت که باید در هر مرحله، عنصر ماکزیمم آن سطری که در حال بررسی هستیم را بیابیم و آن عنصر را با عناصر همسایه‌اش مقایسه کنیم.

سطر میانی را در نظر بگیرید و عنصر ماکزیمم آن را پیدا کنید.
ایندکس سطر میانی "mid" باشد ، مقدار ماکزیمم در سطر میانی "max" و عنصر ماکزیمم در "$mat[mid][max\_index]$" باشد.

1. اگر $max >= A[mid+1][max\_index]$ & $max >= A[mid-1][max\_index]$ ، پس max یک قله است و جواب را پیدا کردیم.
2. اگر $max < mat[mid-1][max\_index]$ ، پس یعنی قسمت بالای سطر میانی ماتریس حتما دارای یک قله است. زیرا ماکزیمم سطر میانی، از عنصر مجاور سطر بالایی خود بزرگتر است، پس در نتیجه ماکزیموم سطر بالایی حتما از عنصر زیری خود بزرگتر خواهد بود. الگوریتم را مجدد به صورت بازگشتی برای این نیمه‌ی بالا انجام می‌دهیم.
3. اگر $max < mat[mid+1][max\_index]$ ، پس یعنی قسمت پایین سطر میانی ماتریس حتما دارای یک قله است. زیرا ماکزیمم سطر میانی، از عنصر مجاور سطر پایینی خود بزرگتر است، پس در نتیجه ماکزیموم سطر پایینی حتما از عنصر بالایی خود بزرگتر خواهد بود.پس الگوریتم را مجدد به صورت بازگشتی برای این نیمه‌ی پایین انجام می‌دهیم.

چون در هر مرحله، به صورت باینری سرچ، سطر میانی را می‌یابیم، هزینه‌اش $O(logm)$ ، و چون برای هر سطر باید عنصر ماکزیموم را با هزینه‌ی $O(n)$ بیابیم، پس هزینه‌ی کل برابر $O(nlogm)$ خواهد بود.

سوال ۳.

سوال سوم. کمینه‌ی تابع

دنباله‌ی $a_1, a_2, ..., a_n$ از اعداد صحیح در نظر بگیرید. تابع f را برای هر i, j $(1 \leq i < j \leq n)$ به صورت زیر تعریف می‌کنیم:

$$f(i,j) = (i-j)^2 + (a_{i+1} + a_{i+2} + ...a_j)^2$$

الگوریتمی با زمان اجرای $O(nlogn)$ ارائه دهید که کمینه مقدار f را محاسبه کند .

راه حل. آرایه x را به صورت $x_i = i$ تعریف و آرایه y را به صورت $y_i = \sum_{j=1}^{i} a_j$ تعریف می‌کنیم. حال شکل مساله عوض شد و در حقیقت ما به دنبال کمترین فاصله بین نقاط با داشتن طول

و عرض آن‌ها هستیم. این همان مسئله کلاسیکی است که در کلاس استاد به شرح آن پرداخته اند و هزینه‌ی آن برابر
$O(nlogn)$ می‌باشد.
در اصل با این راه حل، قسمت اول معادله یعنی $(i-j)^2$ برابر $(x_i-x_j)^2$ که توان دوعه اختلاف طول‌ها است، و $a_{i+1}+a_{i+2}+...+a_j$ برابر $(y_i-y_j)^2$ که توان دوعه اختلاف عرض‌ها است خواهد بود که می‌خواهیم جمع این ۲ مقدار که برابر فاصله‌ی بین ۲ نقطه i, j می‌باشد کمینه شود.

سوال ۴.

سوال چهارم. رنگ آمیزی درست حسابی

فرض کنید n نقطه روی محور اعداد حقیقی داده شده است. می خواهیم این n نقطه را طوری رنگ‌آمیزی کنیم که به ازای هر بازه $[a, b]$ روی محور اعداد حقیقی ، از بین نقاطی که در این بازه قرار گرفته‌اند ، حداقل یک نقطه وجود داشته باشد که رنگ آن با رنگ بقیه نقاط داخل بازه متفاوت باشد. نشان دهید با $[logn]$ رنگ قابل انجام است و الگوریتمی برای این امر ارائه دهید.

راه حل. کافی است نقطه وسط در این محور به رنگی دلخواه و متفاوت با سایر رنگ ها، رنگ آمیزی کنیم در ادامه نقاط هر تکه را جداگانه با سایر رنگ ها، رنگ کنیم چون هر تکه حداکثر $[n/2]$ نقطه دارد پس حداکثر $logn - 1 = [\log n/2]$ رنگ مورد استفاده است که شرایط برقرار گردد چون یک رنگ اولیه استفاده کردیم در نهایت در کل از $logn$ رنگ استفاده شد.
حال هر بازه درون این محور یا شامل نقطه وسط می‌باشد که در این صورت رنگ نقطه وسط با سایرین متفاوت است، در صورتی که بازه انتخاب شده از نقطه وسط گذر نکند در این صورت این بازه درون یکی از دو تکه سمت راست یا چپ قرار می گیرد. پس چون دو تکه به نحوی رنگ گشته اند که شرایط مساله را داشته باشند پس درون بازه مد نظر نقطه ای موجود است که رنگ آن با سایرین یکسان نباشد.

سوال ۵.

سوال پنجم. شلم

n تا ورق بازی داریم که روی هر کدام یک عدد نوشته شده است. در هر گام می‌توانیم دو ورق انتخاب کرده و صرفا عدد روی آن‌ها را مقایسه کنیم و تنها متوجه آن می‌شویم که دو کارت برابر اند یا خیر. در نهایت هدف فهمیدن آن است که آیا بیشتر از نیمی از ورق ها عدد یکسانی دارند یا خیر.

راه حل. فرض کنیم که اعداد نوشته شده روي هر کارت یک آرایه اي از اعداد به صورت $< a_1, \ldots, a_n >$ را تشکیل می دهند. در واقع ما به دنبال عددي از بین این اعداد هستیم که بیش از

$n/2$ بار تکرار شده است. می دانیم که در این آرایه نمی توانیم دو عدد مختلف داشته باشیم که هردو بیش از $n/2$ بار تکرار شده باشند.

(آ) الگوریتمی بر مبنای تقسیم و حل ارائه دهید که با انجام $O(nlogn)$ این کار را انجام دهد.

راه حل. شبیه sort merge عمل می کنیم و در هر مرحله آرایه ورودي را به دو قسمت می شکنیم و تابع را روي ورودي جدید صدا می‌زنیم تا زمانی که به یک آرایه با یک عنصر برسیم.در یک آرایه با یک عنصر، همان عنصر جواب می باشد و به عنوان جواب برگردانده می شود. در هنگام *merge* کردن دو زیر آرایه نیز چهار حالت پیش می آید که به صورت زیر عمل می کنیم:

حالت اول: هیچ کدام از دو زیر آرایه عنصري ندارند که بیش از $n/2$ بار تکرار شده باشد و آرایه حاصل از *merge* شدن این دو زیر آرایه نیز چنین عنصري نخواهد داشت.

حالت دوم: زیر آرایه سمت چپ چنین عنصري دارد ولی زیر آرایه سمت راست ندارد. در این حالت روي تمامی عناصر هر دو زیر آرایه پیمایش انجام میدهیم و تعداد تکرار عنصري را که به عنوان عنصر پر تکرار در زیر آرایه سمت چپ بود را به دست می آوریم. اگر تعداد تکرار این عنصر بزرگتر از نصف طول آرایه حاصل از *merge* شدن بود، مقدار آن عنصر به عنوان عنصر پر تکرار برگردانده می شود. در غیر این صورت عنصر پر تکرار نداریم.

حالت سوم: زیر آرایه سمت راست چنین عنصري دارد ولی زیر آرایه سمت چپ ندارد که شبیه حالت دوم عمل می کنیم.

حالت چهارم: هر دو زیر آرایه عنصر پر تکرار دارند که در این صورت تعداد تکرار هر دو عنصر را در آرایه حاصل از *merge* شدن دو زیر آرایه به دست می آوریم و اگر یکی از آن ها تعداد تکرارش بیشتر از نصف طول آرایه حاصل از *merge* شدن بود، به عنوان عنصر پرتکرار برگردانده می‌شود(می‌دانیم که یک آرایه حداکثر یک عنصر پر تکرار دارد) و در غیر این صورت عنصر پر تکرار نداریم.

توجه کنید که منظور از عنصر پر تکرار، عنصري است که بیشتر از $n/2$ بار در یک آرایه به طول n تکرار شده است. زمان اجرای الگوریتم از رابطه بازگشتی $T(n) = 2T(n/2) + O(n)$ پیروي می کند که با استفاده از قضیه اصلی در می یابیم که زمان اجرای الگوریتم $O(n\log n)$ می‌باشد.

(ب) الگوریتمی ارائه دهید که این کار را با $O(n)$ انجام دهد.

راه حل. فرض کنیم که دو متغیر $index - majority$ و $count$ داریم که به ترتیب با

مقدار هاي 0 و 1 مقدار دهی شده اند. روي عناصر آرايه و با شروع از عنصر دوم a_2 شروع به پيمايش می کنيم و به هر عنصر با انديس i که می‌رسيم مقدار آن عنصر را با مقدار $index-majority$ مقايسه می کنيم. اگر با هم برابر بودند مقدار $count$ را يک واحد زياد می کنيم و به سراغ عنصر بعدي می رويم و در غير اينصورت مقدار $count$ را يک واحد کم می کنيم و اگر مقدار $count$ صفر شد، $index-majority$ را برابر با i و مقدار $count$ را برابر با 1 می کنيم. بعد از اتمام اين پيمايش، تعداد تکرار $index-majority$ را در آرايه اعدادي که داريم، می شماريم. اگر اين مقدار بزرگتر از $n/2$ بود به عنوان عنصر پرتکرار شناخته می شود و در غير اين صورت عنصر پرتکرار نداريم.

سوال ۶.

سوال ششم. زیرآرایه‌ی بیشینه

آرایه‌ای از اعداد داده شده است. الگوریتمی با مرتبه‌ی زمانی $O(n)$ ارایه دهید تا زیرآرایه‌ای از اعداد پشت سر هم را پیدا کند که مجموع اعداد آن بیشینه شود. شبه کد الگوریتم خود را بنویسید.

راه حل. در هر مرحله آرايه را به دو قسمت چپ (L) و راست (R) تقسيم می کنيم. هر قسمت در خروجی خود می‌بايست ۴ مقدار $maxSuffix$, $maxPrefix$, $maxSum$, $totalSum$ را محاسبه کند و برگرداند. حال می‌دانيم که $totalSum$ برابر $L.totalSum + R.totalSum$ می‌باشد. حال برای مقدار $maxPrefix$ اگر نخواهد که از وسط آرايه عبور کند، مقدارش برابر $L.maxPrefix$ می‌باشد و در صورتی که اين محدوديت نباشد برابر $L.totalSum + R.maxPrefix$ می‌باشد. در نهايت $maxSum$ را محاسبه می کنيم. اين مقدار برابر ماکزيموم $L.maxSuffix + R.totalSum$ و $R.maxSuffix$ می‌باشد.

مرتبه‌ی زمانی: $T(n) = 2T(n/2) + d$ که برابر $O(n)$ می‌باشد.

```
function FMS-COMPARE(A,low,high)
        if low = high then
                return (A[low], A[low], A[low], A[low])
        else
                mid ← (low + high) / 2
                Left ← FMS-COMPARE(A, low, mid)
                Right ← FMS-COMPARE(A, mid + 1, high)
                return COMPARE(A,Left,Right)
        end if
end function

function COMPARE(A,L,R)
        totalSum ← L.totalSum + R.totalSum
        maxPrefix ← MAX(L.maxPrefix,L.totalSum + R.maxPrefix)
        maxSuffix ← MAX(R.maxSuffix, R.totalSum + L.maxSuffix)
        maxSum ← MAX(L.maxSum, R.maxSum,
        L.maxSuffix + R.maxPrefix)
        return (totalSum,maxSum,maxPrefix, maxSuffix)
end function
```

سوال ۷.

سوال هفتم. میانه‌ی وزن‌دار

در یک آرایه‌ی معمولی عضو میانه عضو $[n/2]$ است. اما اگر هر کدام از اعضای آرایه یعنی x_i ها یک وزن w_i هم داشته باشند که $\sum_{i=(1,n)} w_i = 1$ میانه‌ی وزن دار چنین آرایه‌ای را اینگونه تعریف می‌کنیم:

آخرین اندیس i در حالت تعریف شده به طوری که داشته باشیم:

$$\sum_{i=(1,n)} w_i \leq 1/2$$

حال فرض کنید الگوریتمی داریم که میانه یک آرایه بدون وزن را در $O(n)$ محاسبه کند. با استفاده از این الگوریتم میانه وزن‌دار آرایه داده شده را در $O(n)$ بدست آورید.

راه حل. ابتدا عنصر میانه را پیدا کنیم. حال روی آرایه حرکت می‌کنیم و وزن عناصری را که از میانه کوچک‌تر هستند جمع می‌کنیم و این مقدار را k می‌نامیم. اگر k از ۱/۲ بزرگتر بود میانه وزن‌دار در نیمه‌ی اول آرایه قرار دارد یعنی کافیست عناصر کوچکتر از میانه را در یک آرایه دیگر بریزیم و آخرین عنصری در این آرایه که جمع وزن‌ها تا آنجا از ۱/۲ کوچکتر است را پیدا کنیم. و اگر از ۱/۲ کوچکتر باشد این عنصر در نیمه‌ی دوم آرایه قرار دارد و باید در نیمه‌ی دوم به دنبال آخرین عنصری بگردیم که جمع وزن از میانه تا آنجا حداکثر $k - 1/2$ باشد. به همین صورت می‌توان بازگشتی عمل کرد و در هر مرحله میانه آرایه فعلی را پیدا کرد و این اعمال را انجام داد.
تحلیل زمان اجرا: $T(n) = T(n/2) + O(n)$ که برابر $O(n)$ می‌باشد.