به نام خدا

# دانشگاه تهران، دانشکده مهندسی برق و کامپیوتر
# تحلیل و طراحی الگوریتم‌ها

تمرین کتبی اول
موعد تحویل: دوشنبه ۹ اسفند ۱۴۰۰، ساعت ۲۳:۵۹
طراح: حسام اسداله‌زاده، asadzadeh.hesam@ut.ac.ir

---

۱. (۱۴ نمره) برای مسائل زیر، با استفاده از روش تقسیم و حل، الگوریتم‌هایی با زمان اجرای $O(\log n)$ ارائه دهید.

(آ) آرایه‌ی مرتب‌شدهٔ $A$ (به صورت صعودی و به طول $n$) از اعداد صحیح متمایز را در نظر بگیرید. الگوریتمی بنویسید که اندیس $i$ را بیابد به طوری که $A[i] = i$ باشد.

(ب) فرض کنید $A$ یک آرایه‌ی یک‌بعدی با اندازه‌ی $n$ از اعداد طبیعی متمایز باشد. $A[i]$ را ماکسیمال گوییم اگر از هر دو خانهٔ کناری‌اش (در صورت وجود) کوچک‌تر نباشد. الگوریتمی بنویسید که یک عنصر ماکسیمال را برگرداند. (ممکن است بیش از یک عنصر ماکسیمال وجود داشته باشد که ارائه فقط یکی از آن‌ها به عنوان جواب کافی خواهد بود).

- پاسخ آ: با توجه به اینکه آرایه مرتب شده است از روشی مشابه binary search استفاده می‌کنیم. ابتدا عضو میانی آرایه را انتخاب کرده و اختلافِ مقدار و اندیس آن را محاسبه می‌کنیم ($A[i] - i$). اگر این مقدار مثبت بود، نیمهٔ چپ آرایه و در غیراین صورت، نیمهٔ راست آرایهِ را انتخاب می‌کنیم. (دقت کنید که آرایه صعودی و اعداد متمایز هستند پس آرایه به صورت صعودی اکید مرتب شده. در نتیجه اگر عضو میانی پاسخ سوال نباشد، مقدار $A[i] - i$ حداقل برابر ۱ خواهد بود. پس مقدار هیچ‌یک از عناصرِ بعد از عنصر میانی نمی‌تواند برابر اندیسش باشد.) این کار را به صورت بازگشتی تا رسیدن به جواب ادامه می‌دهیم. واضح است که اگر هیچ عضوی از آرایه شرط سوال را برآورده نکند، جواب وجود نخواهد داشت.
- پاسخ ب: باز هم از روشی مشابه binary search استفاده می‌کنیم. ابتدا باید بررسی کنیم که عنصر میانی، عنصر ماکسیمال است یا خیر. اگر عنصر میانی ماکسیمال نبود، عنصر سمت راستِ عنصر میانی را بررسی می‌کنیم. اگر عنصر سمت راست بزرگ‌تر از عنصر میانی بود، پس حتما یک عنصر ماکسیمال در زیرآرایهِ راست وجود دارد. و اگر عنصر سمت چپی بزرگ‌تر بود، عنصر ماکسیمال در زیرآرایه چپ خواهد بود. این کار را آن‌قدر تکرار می‌کنیم تا به پاسخ برسیم.
- واضح است که زمان اجرای هر دو پاسخ بالا مانند binary search از مرتبه $O(\log n)$ خواهد بود.

۲. (۱۶ نمره) آرایه‌ی مرتب‌شده $A$ (به صورت صعودی و به طول $n$) را در نظر بگیرید. به زیر دنبالهٔ متوالی $\{A[i], A[i+1], ..., A[i+j]\}$ یک دنباله با سطح جدایی $k$ می‌نامیم اگر و فقط اگر اختلاف هر دو عضو متوالی این دنباله حداکثر برابر $k$ باشد.

(آ) با استفاده از روش تقسیم و حل، الگوریتمی با زمان اجرای $O(n \log n)$ ارائه دهید که بزرگ‌ترین زیردنباله با سطح جدایی $k$ را در آرایه بیابد.

(ب) سعی کنید الگوریتمی با زمان اجرای $O(n)$ برای حل مسئله ارائه دهید.

- پاسخ آ: آرایه را به دو قسمت تقسیم می‌کنیم. حال ۳ حالت داریم: حالت اول و دوم آن است که کل جواب در نیمهٔ راست یا نیمهٔ چپ آرایه باشد و حالت سوم آن است که جواب بین دو نیمه مشترک باشد. برای یافتن این جواب، از نیمه شروع میکنیم و به سمت چپ می‌رویم و تا جایی که شرط برقرار بماند یک واحد به طول زیردنباله اضافه می‌کنیم. همچنین از نیمه شروع کرده و به سمت راست می‌رویم و پاسخ این بخش را نیز پیدا می‌کنیم. حال اگر نقاط مرزی دو قسمت، فاصله کمتر از $k$ داشته باشند، این دو بازه، با هم یکی می‌شوند و طول پاسخ دو قسمت با هم جمع می‌شود. وگرنه که جواب، ماکسیمم جواب یافت‌شده توسط دو قسمت خواهد بود. واضح است که این الگوریتم مانند الگوریتم merge sort رفتار می‌کند و زمان اجرای آن $O(n \log n)$ خواهد بود.

- پاسخ ب: با یک بار پیمایش آرایه می‌توانیم به پاسخ برسیم. به این صورت که دو متغیر $max_{seq}$ و $counter$ تعریف می‌کنیم و هردو را برابر ۱ قرار می‌دهیم. حال تا زمانی که شرط سوال برقرار بود، یک اندیس به جلو حرکت می‌کنیم و متغیر $counter$ را یک واحد افزایش می‌دهیم. اگر به اندیسی رسیدیم که شرط مسئله نقض شد، مقدار متغیر $max_{seq}$ برابر $max(counter, max_{seq})$ شده و مقدار $counter$ را مجددا برابر ۱ قرار می‌دهیم. در نهایت مقدار $max_{seq}$ طول بزرگ‌ترین زیردنباله با سطح جدایی $k$ خواهد بود.

۳. (۱۵ نمره) بُرنا زمان تولد و مرگ تعداد زیادی از مشاهیر تاریخ از قرون وسطی تا پایان جنگ جهانی دوم را در دفتر خود نوشته. او می‌خواهد بداند کدام دو نفر در لیست او بیشترین همپوشانی را در طول عمر خود داشته‌اند. برای این کار او آرایه‌ای از سه‌تایی‌های مرتب تشکیل داده که هر عضو آن به شکل مقابل است: $(birth, death, name)$. الگوریتمی با زمان اجرای $O(n \log n)$ ارائه دهید که این دو نفر را برای برنا بیابد. (در واقع باید بیشینه $min\{A[i].death, A[j].death\} - max\{A[i].birth, A[j].birth\}$ را به ازای $i \neq j$ بیابید). مثال:

$$n = 4, A = [(1889, 1945, AdolfHitler), (1451, 1506, ChristopherColumbus), (1878, 1953, JosephStalin),$$

$$(1483, 1546, MartinLuther)] \rightarrow answer = (AdolfHitler, JosephStalin)$$

- پاسخ: ابتدا آرایه را برحسب زمان تولد مشاهیر مرتب می‌کنیم ($O(n \log n)$). حال مسئله را به صورت بازگشتی برای نیمهٔ راست و نیمهٔ چپ آرایه حل می‌کنیم. برای این کار، ابتدا در نیمهٔ چپ آرایه اندیس $i$ را طوری می‌یابیم که $A[i].death$ بیشینه باشد. ($1 \leq i \leq \frac{n}{2}$). حال در نیمهٔ راست آرایه، اندیس $j$ را طوری می‌یابیم که:

$$min\{A[i].death, A[j].death\} - max\{A[i].birth, A[j].birth\}$$

بیشینه باشد ($\frac{n}{2} \leq j \leq n$). دقت کنید که آرایه برحسب زمان تولد به صورت صعودی مرتب شده. پس با پیدا کردن بیشینه زمان مرگ در سمت چپ و بیشترین همپوشانی متناظر آن در سمت راست، بیشترین همپوشانی بین این دو نیمه به دست خواهد آمد. حال ۳ حالت وجود دارد. حالت اول و دوم آن است که هر دو نفر در نیمهٔ راست یا چپ زیرآرایه باشند (یعنی پاسخ‌هایی که از طریق زیرمسائل چپ و راست به دست آمده ($2T(\frac{n}{2})$))و حالت سوم این‌که یکی از آن‌ها در نیمهٔ چپ و دیگری در نیمهٔ راست آرایه قرار داشته باشد ($O(n)$). پس بین این ۳ مقدار ماکسیمم گرفته و اندیس‌های متناظر آن را به عنوان پاسخ زیرمسئله گزارش می‌کنیم.

به این ترتیب پاسخ کل مسئله در $O(n \log n)$ بدست می‌آید.

$$T(n) = 2 \times T(\frac{n}{2}) + O(n) \xrightarrow{\text{Master Theorem}} T(n) \in O(n \log n)$$

۴. (۱۵ نمره) روی میزی $n$ جعبه وجود دارد که داخل هر یک از آنها مقداری پول موجود است. جعبه‌ها از $0$ تا $n - 1$ شماره‌گذاری شده‌اند و می‌دانیم که داخل جعبهٔ $i$اُم، $M[i]$ دلار قرار داده شده است. جعبه‌ها بین $m$ نفر تقسیم خواهند شد. هرکدام از این $m$ نفر تعدادی از جعبه‌ها را به صورت متوالی دریافت می‌کنند (به طور مثال جعبه‌های ۴ و ۵ و ۶ را می‌توان به یک نفر اختصاص داد ولی حق دادن جعبه‌های ۴ و ۶ به یک نفر وجود ندارد). هر طوری که جعبه‌ها را بین افراد تقسیم کنیم، یک نفر هست که مجموع پولی که دارد از بقیه بیشتر خواهد بود. این مقدار را بین همهٔ افرازهای ممکن، $MaxMoney$ می‌نامیم. می‌خواهیم کمترین $MaxMoney$ را به ازای تمام روش‌هایی که می‌توان جعبه‌ها را بین افراد تقسیم کرد به دست آوریم. فرض کنید مقدار کل پول موجود حداکثر برابر $k$ دلار است. از آنجایی که تعداد جعبه‌ها بسیار زیاد است، الگوریتمی ارائه دهید که مسئلهٔ بالا را در $O(n \log k)$ حل کند.

- پاسخ: برای حل این مسئله، از binary search روی مقدار پول همه جعبه‌ها استفاده می‌کنیم. در ابتدا مقدار کمینه برای مقدار $MaxMoney$ را صفر و مقدار بیشینه را برابر جمع کل پول موجود در جعبه‌ها ($k$) در نظر می‌گیریم. حال بررسی می‌کنیم که آیا می‌توان میانگین این دو ($mid$) را به افراد داد یا خیر. اگر امکان‌پذیر بود، روی بازهٔ صفر تا میانگین، این کار را تکرار می‌کنیم و در غیراین‌صورت این کار را برای بازهٔ میانگین تا بیشینه تکرار می‌کنیم.

برای بررسی این که آیا می‌توان مقدار مشخصی را به افراد داد یا خیر، به این صورت عمل می‌کنیم که روی آرایه $M$ پیمایش کرده و به ترتیب مقدار پول را به افراد اختصاص می‌دهیم. به این صورت که تا زمانی که مقدار کل پول اختصاص داده شده به فرد از مقدار $mid$ بیشتر شود روی آرایه $M$ جلو می‌رویم و وقتی مقدار پول اختصاص داده شده بیشتر شد، سراغ نفر بعدی می‌رویم. در هنگام انجام این کار اگر نیاز شد که تعداد افراد بیشتر از $m$ نفر باشد، به این نتیجه می‌رسیم که این مقدار پول قابل اختصاص دادن

به $m$ نفر نیست. درغیراین‌صورت، می‌توان این مقدار پول را به افراد اختصاص داد. در نهایت مینیمم مقدار $MaxMoney$ به دست آمده را گزارش می‌کنیم. زمان اجرای این الگوریتم:

$$T(k) = T(\frac{k}{2}) + O(n) \xrightarrow{\text{Master Theorem}} T(k) \in O(n \log k)$$

۵. (۲۰ نمره) درخت دودویی، درختی‌ست که هریک از رئوس آن دقیقا **دو** یا **صفر** فرزند دارند. درخت دودویی کامل با ارتفاع $k$، درخت دودوییِ‌ای است که تا ارتفاع $k-1$، همه رئوس آن دقیقا دو فرزند دارند و راس‌های ارتفاع $k$اُم هیچ فرزندی ندارند. توجه کنید که $k$ بزرگ‌ترین عدد ممکنی‌ست که $2^k - 1$ کوچک‌تر از $n$ باشد ($n \geq 2^k - 1 \geq 1$). حال رئوس درخت دودویی را از $1$ تا $2^k - 1$ شماره‌گذاری می‌کنیم طوری که فرزندان راس $i$اُم دو راس $2i$ و $2i-1$ باشند.

حال $n$ نقطه روی صفحه داریم به طوری که هیچ ۳ نقطه‌ای روی یک خط نمی‌باشند. الگوریتمی با زمان اجرای $O(n \log^2 n)$ ارائه دهید که تعدادی از این نقاط را طوری به هم وصل کند که یک درخت دودویی کامل با ارتفاع $k$ تشکیل شود. (یال‌های درخت نباید یکدیگر را قطع کنند)

- پاسخ: ابتدا باید ریشهٔ درخت را انتخاب کنیم. ریشه باید نقطه‌ای باشد که یک سمت آن هیچ نقطهٔ دیگری وجود نداشته باشد؛ به طور مثال، پایین‌ترین یا راست‌ترین نقطه باشد ($O(n)$). پس از انتخاب کردن ریشه، باید سایر نقاط را برای تشکیل دادن زیردرخت چپ و راست به دو قسمت تقسیم کنیم. برای این کار، نقاط را برحسب زاویهٔ خط واصل نقاط و ریشه و خط افق مرتب می‌کنیم ($O(n \log n)$) و سپس نقاط مرتب شده را به دو قسمت تقسیم می‌کنیم. (نکته: اگر پایین‌ترین نقطه را به عنوان ریشه انتخاب کنیم، زوایا از ۰ تا ۱۸۰ درجه خواهند بود و در این بازه، $\cos(x)$ نزولی‌ست. پس کافی‌ست نقاط را برحسب کسینوس زوایا مرتب کرده و آن را وارونه کنیم.) حال این کار را به ازای زیردرخت چپ و راست ادامه می‌دهیم تا درخت دودویی تشکیل شود.

$$T(n) = 2 \times T(\frac{n}{2}) + O(n \log n) + O(n) = O(n \log^2 n)$$

- نکته ۱: دقت کنید که در هر زیرمسئله باید زوایای سایر نقاط نسبت به ریشه‌ی آن زیردرخت محاسبه و این مقادیر مرتب شوند.
- نکته ۲: انتخاب هر معیار دیگری غیر از زاویه نسبت به ریشه (مانند مرتب کردن نسبت به محور $x$ یا $y$) می‌تواند باعث شود که یال‌های درخت یکدیگر را قطع کنند.

۶. (۲۰ نمره) ضرب ماتریس‌ها و بردارها یکی از عملیات مهم و رایج در الگوریتم‌های هوش مصنوعی و یادگیری عمیق به شمار می‌روند و انجام این محاسبات در زمان بهینه، بسیار حائز اهمیت می‌باشند. ماتریس‌های $M_0, M_1, M_2, \ldots$ به صورت زیر تعریف می‌شوند:

- $M_0$ ماتریس $[1]$ است.
- برای هر $k > 0$، ماتریس $M_k$ یک ماتریس $2^k \times 2^k$ است که به صورت زیر تعریف می‌شود:

$$M_k = \begin{bmatrix} M_{k-1} & M_{k-1} \\ M_{k-1} & -M_{k-1} \end{bmatrix}$$

**بردار ستونی**، ماتریسی با ابعاد $n \times 1$ است و شامل $n$ درایه در ستونی واحد است. ضرب ماتریس در بردار ستونی به شکل زیر تعریف می‌شود:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \rightarrow \mathbf{Ax} = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \ldots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \end{bmatrix}.$$

بردار ستونی $v$ با تعداد اعضای $n = 2^k$ را در نظر بگیرید. ثابت کنید که حاصل ضرب $M_k v$ را می‌توان در زمان $O(n \log n)$ محاسبه کرد. فرض کنید محاسبات جمع و ضرب در زمان $O(1)$ انجام می‌شوند. (پاسخ نهایی یک بردار ستونی با اندازه $2^k \times 1$ خواهد بود)

- پاسخ: بردار ستونی $v$ را به دو قسمت $v_1$ و $v_2$ تقسیم می‌کنیم:

$$v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

حال داریم:

$$M_k v = \begin{bmatrix}(M_k v)_1\\(M_k v)_2\end{bmatrix} = \begin{bmatrix}M_{k-1}v_1 + M_{k-1}v_2\\M_{k-1}v_1 - M_{k-1}v_2\end{bmatrix} = \begin{bmatrix}M_{k-1}(v_1+v_2)\\M_{k-1}(v_1-v_2)\end{bmatrix}$$

پس می‌توانیم $M_k v$ را با محاسبه $v_1 + v_2$ و $v_1 - v_2$ و محاسبه $M_{k-1}(v_1+v_2)$ و $M_{k-1}(v_1-v_2)$ به صورت بازگشتی بیابیم. واضح است که پاسخ زیرمسئله در مرحله $i$اُم، یک بردار ستونی با اندازه $2^i \times 1$ خواهد بود و با روی هم گذاشتن پاسخ دو زیرمسئله به صورت ستونی، برداری با اندازه $2^{i+1} \times 1$ تشکیل خواهد شد. به این ترتیب در نهایت پاسخ مسئله یعنی برداری ستونی با ابعاد $2^k \times 1$ به دست خواهد آمد. جمع و تفریق نیمه‌های بالا و پایین بردار $v$ از مرتبه $O(n)$ می‌باشند. پس داریم:

$$T(n) = 2 \times T(\frac{n}{2}) + O(n) \xrightarrow{\text{Master Theorem}} T(n) \in O(n \log n)$$