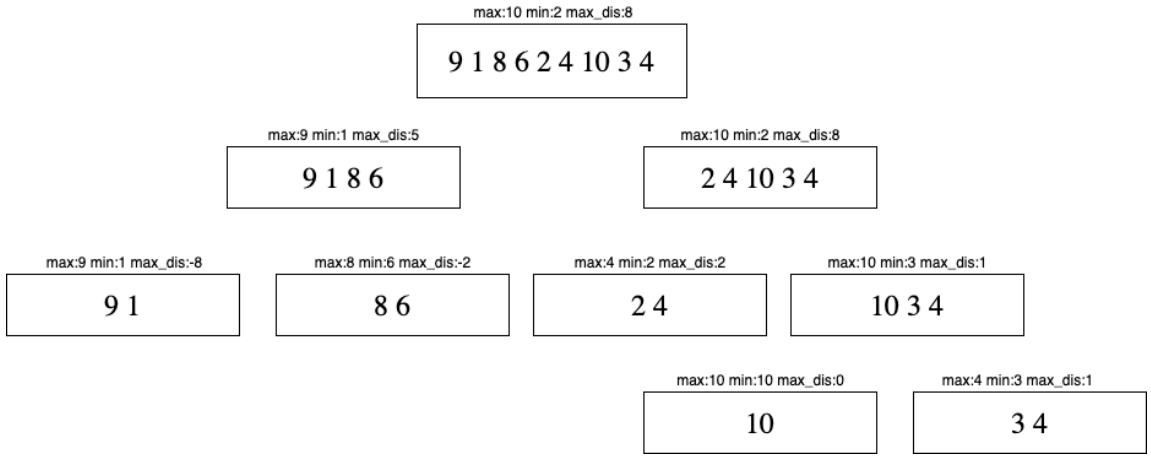
به نام خدا

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

تمرین کتبی دوم با موضوع تقسیم و حل
موعد تحویل: شنبه ۱۶ اسفندماه ۱۳۹۹، ساعت ۲۳:۵۹
طراح: مجید دلیری، محمدهادی حجت، majiddl.2099@gmail.com

---

۱. (10 نمره) روش حل این سوال به این صورت می‌باشد که باید آرایه را به دو نیم تقسیم کرده و درهر تکه دو عنصری که بیشترین اختلاف را دارند و ماکسیمم و مینیمم هر تکه را نگه داشته و بااستفاده از این عناصر عملیات مرج کردن را انجام می‌دهیم. برای حل مثال نیز در شکل زیر بیان شده است.



۲. (15 نمره) یک آرایه را $unimodal$ می‌نامیم هرگاه از ابتدای آرایه تاجایی اعداد به صورت صعودی باشند و از آنجا به بعد به صورت نزولی باشند. حال ادعا می‌کنیم این تعبییر جهت در چنین آرایه‌ای را می‌توان در $O(log\ n)$ بدست آورد. ابتدا عنصر میانی را نگاه می‌کنیم و اگر از عدد قبل بزرگتر و از عدد بعد کوچکتر باشد در قسمت صعودی قرار دارد و می‌توانیم تغییر را در نیمه دوم آرایه که آن هم $unimodal$ است، جستوجو کنیم، همین طور اگر از عدد قبل کوچکتر و از عدد بعدی بزرگتر بود در قسمت نزولی قرار دارد و پاسخ در نیمه ابتدایی می‌باشد در صورتی که در هر دو عنصر کناری خود بزرگتر باشد نیز عنصر مد نظر ماست. به همین صورت مانند جستوجو دودویی ادامه می‌دهیم تا عنصر را پیدا کنیم. حال توجه می‌کنیم که این نقاط بر حسب مولفه x، $unimodal$ هستند و نقطه با بیشترین x هم همان نقطه تغییر جهت می‌باشد پس در $O(log\ n)$ می‌شود آن نقطه را بدست آورد. سپس اگر از اول نقاط تا این نقطه با بیشترین x را در نظر بگیریم می‌توانیم نقطه با بیشینه y را محاسبه کرد.

۳. (15 نمره) روش حل تقسیم و حل: کافی است همانند ایده‌ای که برای کوییک سورت بکار بردیم عنصری را به عنوان شاخص انتخاب نماییم و همه عناصر بزرگتر و عناصر کوچکتر از آن را به دو بخش تقسیم نماییم اگر تعداد عناصر مساوی با آن بیشتر از نصف عناصر بود که به تمام و عنصر مربوطه را یافتیم و در غیر این صورت بین دو بخش، بخش کوچکتر را حذف کرده و این کار را ادامه می‌دهیم تا یا به عنصر مربوطه برسیم یا عناصر باقی مانده از نیمی از عناصر کوچکتر باشد که در این صورت عنصر غالب نداریم. در نهایت به طور میانگین این مساله را $O(n)$ حل می‌کند.
روش غیر تقسیم و حل: فرض کنید دو متغیر $indexMajority$ و $count$ داریم که به ترتیب با 0 و 1 اینیشیالایز شده اند. روی عناصر آرایه و با شروع از عنصر دوم (A_1) شروع به پیمایش می‌کنیم و اگر به عنصر با اندیس i رسیدیم اگر مقدار آن با $indexMajority$ برابر بود مقدار $count$ را یکی اضافه کرده و در غیر این صورت مقدار $count$ را یکی کم می‌کنیم در صورتی که مقدار $count$ صفر شد مقدار $indexMajority$ و $count$ را به i و 1 آپدیت می‌کنیم و به سراغ عنصر بعدی می‌رویم بعد از اتمام پیمایش اگر تعداد تکرار های

عنصر $indexMajority$ از $\frac{n}{2}$ بیشتر بود به عنوان عنصر غالب شناخته می‌شود در غیر این صورت چنین عنصری موجود نیست.

۴. (20 نمره) فرض کیند می‌خواهیم میانه‌ی آرایه‌ی حاصل از ادغام دو آرایه‌ی مرتب A, B را به دست بیاوریم به طور کلی می‌خواهیم عنصر k ام آرایه حاصل از ادفام را پیدا کنیم که در ابتدا $k = (|A| + |B|)$ است و اگر طول یکی از از دو آرایه صفر باشد، عنصر k ام ادغام عنصر kام آرایه دیگر است. و اگر $k = 0$ باشد عنصر k ام برابر عنصر اول ادغام دو آرایه است. (اینها شروط بازگشت می‌باشند.)
در بقیه موارد $\frac{k}{2}$ عضو اول از هر آرایه را جدا می‌کنیم تا ۴ بخشی به صورت زیر ایجاد گردد. $A_{1:\frac{k}{2}-1}, A_{\frac{k}{2}:m-1}, B_{1:\frac{k}{2}-1}, B_{\frac{k}{2}:m-1}$
اگر $A_{\frac{k}{2}-1} = B_{\frac{k}{2}-1}$ باشد عضو k همان $A_{\frac{k}{2}-1}$ است. اگر $A_{\frac{k}{2}-1} > B_{\frac{k}{2}-1}$ باشد می‌دانیم که عضو k ام آرایه A و نیمه راست B است و در نهایت اگر $A_{\frac{k}{2}-1} < B_{\frac{k}{2}-1}$ باشد عنصر k ام در ادغام آرایه B در نیمه راست A می‌باشد. این کار را تاجایی ادامه می‌دهیم که به یکی از شرطهای بازگشت برسیم و در هر گام مقدار k نیز آپدیت می‌شود و چون در هر گام حداقل نیمی از یکی از دو آرایه حذف می‌شود لذا اردر زمانی این الگوریتم $O(log\ n + log\ m)$ می‌باشد.

۵. (20 نمره) به سادگی می توان الگوریتمی را به صورت حریصانه طراحی کنیم. اعداد را مرتب کنیم و از عدد بزرگتر شروع به برداشتن اعداد کنیم و اولین جایی که مجموع اعداد مان بزرگتر مساوی k شود متوقف شویم به یک مجموعه خواهیم رسید که اثبات می شود کمترین عضو را دارد. (البته ممکن است یکتا نباشد و با همین تعداد بتوانیم به مجموع حداقل k برسیم) اما به خاطر مرتب سازی که مرتبه زمانی $nlgn$ دارد درست نخواهد بود.
از ایده $quick\ sort$ استفاده می کنیم و می توانیم همین مجموعه اعداد (از بزرگترین اعداد مطابق الگوریتم حریصانه) را پیدا کنیم.
مساله را به صورت بازگشتی حل می کنیم. یک عدد تصادفی انتخاب می کنیم و طبق آن اعداد را به دسته کوچکتر و بزرگتر مساوی آن تقسیم می کنیم. مجموع دسته بزرگتر را محاسبه می کنیم (sum) و اگر مجموع آن بزرگتر مساوی k بشود یعنی مجموعه جواب کاملا در دسته بزرگتر است. پس مساله را به صورت بازگشتی برای آن حل می کنیم. (با همین مقدار k) اما اگر مجموع آن از k کمتر شود یعنی همه اعداد دسته بزرگتر در جواب ما هستند و علاوه بر آن اعداد دیگری هم وجود دارد که باید پیدا کنیم. همه آن ها را ذخیره می کنیم و برای پیدا کردن بقیه، مساله را با $k - sum$ حل می کنیم (به عنوان پارامتر k مساله).
تفاوتی که با $quick\ sort$ وجود دارد این است که بعد از تقسیم فقط به سراغ یک سمت ایجاد شده برای غلبه می رویم و در سمت دیگر پردازشی انجام نمی دهیم. به همین خاطر در تحلیل مرتبه زمانی زمان اجرای متوسط آن $O(n)$ است و مطلوب سوال است.
اگر در هر مرحله سایز مجموعه را n در نظر بگیریم میانگین سایز مجموعه ای که به سراغش می رویم n۲ است. پس اگر درخت اجرا را در نظر بگیریم به مجموع $n + \frac{n}{2} + + \frac{n}{4} + ... = O(n)$ می رسیم در حالی که میدانیم زمان متوسط اجرای $quick\ sort$ از $O(n\,lgn)$ است. و در بدترین حالت هر دوی الگوریتم ها مرتبه $O(n2)$ دارند.

۶. (20 نمره) روش حل بدیهی این سوال به این صورت می‌باشد که می‌توان دانه دانه مستطیل های سیاه را اضافه کرد و اشتراکات آن‌ها را در صورت وجود پیدا کرد و در نهایت شکل نهایی را بدست آورد. که این روش از اردر زمانی $O(n^2)$ می‌باشد. در اینجا می‌توان ابتکار را به کار برد و همانند الگوریتم مرج ـ سورت عمل کرد یعنی شکل دو نیمه را بدست بیاوریم و در نهایت شکل ها را با هم مرج کرده و شکل نهایی را به عنوان خروجی بدهیم. اما به چه صورت مرج کردن انجام بگیرد ؟ فقط کافیست همانند الگوریتم مرج سورت برحسب x سورت کرده و برای ادغام نیز با مقایسه ارتفاع دو عنصر هر کدام را اضافه می‌کنیم. در صورتی که مقدار x هر دویکی بود یا داخل شکل نهایی قرار می‌گرفت آن را اضافه نمی‌کنیم.