جواب تمرین تئوری اول
طراح: سینا کچویی

۱. برای حل این سوال از روشی مانند mergesort استفاده میکنیم، آرایه را به دو نیمه ی A, B تقسیم می‌کنیم و فرض میکنیم مسئله برای هر نیمه حل شده است، حال می خواهیم تعداد جفت هایی را بیابیم که یکی از آن ها در A و دیگری در B است. برای این کار نیز مانند mergesort یک پوینتر رو هر آرایه در نظر میگیریم و روی اعضای دو آرایه حرکت می‌کنیم و جفت هایی که شرایط برنزی بودن داشته باشند را می‌شماریم.
زمان اجرای این الگوریتم همانند mergesort از رابطه بازگشتی $T(n) = 2 * T(n/2) + O(n)$ پیروی میکند که با استفاده از قضیه ی اصلی می‌یابیم که $T(n) \in O(n * log\ n)$

۲. فرض کنید می‌خواهیم میانه ی آرایه ی حاصل از ادغام دو آرایه ی مرتب A, B را به دست آوریم. به طور کلی می خواهیم عضو kام آرایه ی حاصل از ادغام را پیدا کنیم. که در ابتدا $k = (len(A) + len(B))/2$ اگر طول یکی از دو آرایه صفر باشد ، عضو kام ادغام این دو ، برابر عضو kام آرایه ی دیگر است. و اگر $k = 0$ باشد ، عضو4ام برابر اولین عضو A یا B است. (شروط بازگشت) در بقیه موارد $k/2$ عضو اول از هر آرایه را جدا می‌کنیم. تا ۴ بخش به این صورت تشکیل شود:

$$A_1 = [a_0, a_1, \dots, a_{k/2-1}],\ A_2 = [a_{k/2}, \dots, a_{m-1}]$$
$$B_1 = [b_0, b_1, \dots, b_{k/2-1}],\ B_2 = [b_{k/2}, \dots, b_{m-1}]$$

اگر $b_{k/2-1} = a_{k/2-1}$ باشد ، عضو k ام برابر $a_{k/2-1}$است. اگر $b_{k/2-1} < a_{k/2-1}$ باشد قطع می‌دانیم که عضو kام آرایه‌ی ادغامی ، نمی تواند در B حضور داشته باشد ، در نتیجه آرایه‌ی B_1 را دور می‌ریزیم و به دنبال عضو$k - k/2$ام بین دو آرایه ی A, B_2 می گردیم. به صورت مشابه اگر $b_{k/2-1} > a_{k/2-1}$ باشد ، قطعا می دانیم که عضو kام آرایه ی ادغامی ، نمی‌تواند در A_1حضور داشته باشد ، در نتیجه آرایه ی A_1 را دور می‌ریزیم و به دنبال عضو $k - k/2$ ام بین دو آرایه ی B, A_2 می گردیم. این کار را تا جایی ادامه می‌دهیم که به یکی از شرطهای بازگشتی برسیم. در هر بار اجرای این الگوریتم ، طول یکی از آرایه ها نصف می‌شود تا زمانی که یکی از آنها صفر شود، در نتیجه این زمان اجرایی این الگوریتم از مرتبه ی $O(log\ m + log\ n)$ خواهد بود.

۳. اگر بانوي شماره i را با a_i و همسرش را با b_i نشان دهیم یک آرایه به صورت $< a_1, a_2, \dots, a_n, b_1, b_2, \dots, b_n >$ خواهیم داشت که می خواهیم این آرایه را به صورت $< a_1, b_1, a_2, b_2, \dots, a_n\ b_n >$ در آوریم. با استفاده از ایده حل و تقسیم عمل می کنیم و در هر مرحله آرایه ورودي را از وسط نصف می‌کنیم(به عنوان مثال به دو زیر آرایه $left$ و right تقسیم می کنیم) و عناصر نیمه دوم $left$ را با عناصر نیمه اول $right$ به ترتیب عوض می کنیم ($swap$) حال چون هر کدام از زیر آرایه هاي $left$ و $right$ شبیه آرایه اولیه می باشند می توانیم همین کار را به صورت بازگشتی روي هر کدام از آنها انجام دهیم تا موقعی که طول آرایه ورودي دو شود.

- توجه شود که چون تعداد زوج ها به صورت توانی از دو می باشد، در هر مرحله که آرایه را نصف می‌کنیم طول هر دو زیر آرایه چپ و راست یکسان خواهد بود. زمان اجرای الگوریتم از رابطه بازگشتی $T(n) = 2T(n/2) + O(n)$ پیروي می کند که با استفاده از قضیه اصلی در می‌یابیم که زمان اجرای الگوریتم $O(nlog\ n)$ می باشد.

۴. فرض کنیم که اعداد نوشته شده روي هر کارت یک آرایه اي از اعداد به صورت $< a_1, \dots, a_n >$ 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.

- 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 می‌ابیم که زمان اجرای الگوریتم $O(nlog\ n)$ می‌باشد.

- 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 آرایه و با شروع از عنصر دوم a_2 شروع به پیمایش می کنیم و به هر عنصر با اندیس i که می‌رسیم مقدار آن عنصر را با مقدار $index-majority$ مقایسه می کنیم. اگر با هم برابر بودند مقدار $count$ را یک واحد زیاد می کنیم و به سراغ عنصر بعدي می رویم و در غیر اینصورت مقدار $count$ را یک واحد کم می کنیم و اگر مقدار $count$ صفر شد، $index - majority$ را برابر با i و مقدار $count$ را برابر با 1 می کنیم. بعد از اتمام این پیمایش، تعداد تکرار $index - majority$ را در آرایه اعدادي که داریم، می شماریم. اگر این مقدار بزرگتر از $n/2$ بود به عنوان عنصر پرتکرار شناخته می شود و در غیر این صورت عنصر پرتکرار نداریم.

۵. برای حل این مسئله از روش جستجوی دودویی استفاده می کنیم: برای پیدا کردن بزرگترین عنصر در آرایه $A = < a_1 , a_2 , \ldots , a_n >$ مقدار $a_{n/2}$ را مورد بررسی قرار می‌دهیم ، اگر داشته باشیم $a_{n/2} > a_{n/2-1}$ و $a_{n/2} > a_{n/2+1}$ آنگاه $a_{n/2}$ جواب مسئله است. اگر داشته باشیم $a_{n/2} < a_{n/2-1}$ آن گاه می دانیم پاسخ مسئله سمت چپ $a_{n/2}$ قرار دارد و برابر با بزرگترین عدد در $< a_1 , a_2 , \ldots , a_{n/2-1} >$ است که خود یک آرایه نافرم است و پاسخ آن را به صورت بازگشتی می توان یافت. اگر داشته باشیم $a_{n/2} < a_{n/2+1}$ آن گاه پاسخ مسئله سمت راست $a_{n/2}$ قرار دارد و برابر با بزرگترین عدد در $< a_{n/2+1} , \ldots , a_n >$ است که باز هم یک آرایه ی نافرم است و پاسخ آن را به

صورت بازگشتی می توان یافت. زمان اجرای این الگوریتم از رابطه بازگشتی $T(n) = T(n/2) + O(1)$ پیروی می کند که با استفاده از قضیه اصلی می یابیم که $T(n) \in O(logn)$

۶. $title(i_s,j_s,i_w,j_w)$ را تابعی تعریف میکنیم که مربعی که شروع آن خانه (i_s,j_s) است، طول ضلع آن n است و خانه (i_w,j_w) در آن نباید پوشانده شود را تزیین میکند، در حالتی که $n = 2$ است نحوه پیاده سازی این تابع واضح است. در حالات دیگر، مربع مورد نظر را به چهار مربع تقسیم میکنیم، مربعی که شامل (i_w,j_w) است خود یک نمونه کوچکتر از این مسئله است و به صورت بازگشتی حل میشود، برای سه مربع دیگر به این شکل عمل می کنیم: یک صفحه سنگی را در مرکز مربع به گونه ای قرار میدهیم که سه خانه ای که می‌پوشاند در سه مربعی قرار بگیرند که (i_w,j_w) در آن واقع نشده‌اند. حال این سه مربع نیز سه نمونه کوچکتر همین مسئله هستند و به صورت بازگشتی حل میشوند.