به نام خداوند بخشنده مهربان

طراحی و تحلیل الگوریتم‌ها

نیم‌سال اول سال تحصیلی ۹۷-۹۶

پاسخ تمرین شماره۱

مهر ماه ۱۳۹۶

۱. ابتدا آرایه را به دو نیمه‌ی $B = [15, 17, 9, 3]$ و $C = [1, 4, 93, 21, 2]$ تقسیم می‌کنیم.آرایه B به دونیمه‌ی $D = [15, 17]$ و $E = [9, 3]$ و آرایه D نیز به دونیمه‌ی $F = [15]$ و $G = [17]$ تقسیم می‌شود. حال دو آرایه‌ی F و G را ادغام می‌کنیم و به آرایه‌ی $D_{Sorted} = [15, 17]$ می‌رسیم. حال می‌خواهیم آرایه‌ی E را مرتب کنیم ، دو نیمه‌ی $[9]$ و $[3]$ را داریم ، از ادغام این دو به $E_{Sorted} = [3, 9]$ می‌رسیم. حال E_{Sorted} و D_{Sorted} را ادغام می‌کنیم ، در نتیجه آرایه‌ی $B_{Sorted} = [3, 9, 15, 17]$ به دست می‌آید.

حال می‌خواهیم آرایه C را مرتب کنیم.C به دو زیرآرایه‌ی $H = [1, 4]$ و $I = [93, 21, 2]$ تقسیم می‌شود. آرایه H خود به دوزیرآرایه‌ی $[1]$ و $[4]$ تقسیم می‌شود.در نتیجه $H_{Sorted} = [1, 4]$ است.آرایه I به دو زیر آرایه‌ی $J = [93]$ و $K = [21, 2]$ تقسیم می‌شود. همان طور که می‌دانیم از تقسیم و ادغام آرایه‌ی K ، آرایه‌ی $K_{Sorted} = [2, 21]$ به دست می‌آید و از ادغام K_{Sorted} و J ، $I_{Sorted} = [2, 21, 93]$ به دست می‌آید.از تقسیم و ادغام آرایه‌ی H نیز $H_{Sorted} = [1, 4]$ به دست می‌آید. حال دو آرایه‌ی H_{Sorted} و I_{Sorted} را ادغام می‌کنیم.در نتیجه $C_{Sorted} = [1, 2, 4, 21, 93]$ به دست می‌آید.

در انتها دو آرایه‌ی C_{Sorted} و B_{Sorted} را ادغام می‌کنیم تا $A_{Sorted} = [1, 2, 3, 4, 9, 15, 17, 21, 93]$ به دست آید.

۲. فرض کنید می‌خواهیم میانه‌ی آرایه‌ی حاصل از ادغام دو آرایه‌ی مرتب A و B را به دست آوریم. به طور کلی می‌خواهیم عضو kام آرایه‌ی 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 آرایه‌ی دیگر است. و اگر $k = 0$ باشد ، عضو kام 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 :

$$A_1 = [a_0, ..., a_{k/2-1}], A_2 = [a_{k/2}, ..., a_{m-1}]$$
$$B_1 = [b_0, ..., b_{k/2-1}], B_2 = [b_{k/2}, ..., b_{n-1}]$$

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 قطعاً می‌دانیم که عضو k ام آرایه‌ی ادغامی ، نمی‌تواند در B_1 حضور داشته باشد ، در نتیجه آرایه‌ی B_1 را دور می‌ریزیم و به دنبال عضو $k - k/2$ ام بین دو آرایه‌ی A و B_2 می‌گردیم.به صورت مشابه اگر $b_{k/2-1} > a_{k/2-1}$ باشد ، قطعاً می‌دانیم که عضو kام آرایه‌ی ادغامی ، نمی‌تواند در A_1 حضور داشته باشد ، در نتیجه آرایه‌ی A_1 را دور می‌ریزیم و به دنبال عضو $k-k/2$ ام بین دو آرایه‌ی A_2 و B می‌گردیم. این کار را تا جایی ادامه می‌دهیم که به یکی از شرط‌های بازگشتی برسیم. در هر بار اجرای این الگوریتم ، طول یکی از آرایه‌ها نصف می‌شود تا زمانی که یکی از آن‌ها صفر شود ، در نتیجه این زمان اجرایی این الگوریتم از مرتبه‌ی $O(logm + logn)$ خواهد بود.

۳. برای حل این سوال می‌توان از ایده‌ای مانند الگوریتم $QuickSort$ استفاده کرد.
۱- نقطه‌ای با کمترین مقدار x و نقطه با بیشترین مقدار x را پیدا می‌کنیم
۲- این دونقطه را با یک خط (L) به یکدیگر وصل می‌کنیم.در نتیجه مجموعه نقاط به دو قسمت بالای خط L و پایین خط L تقسیم می‌شوند.هر کدام از بخش‌ها را جداگانه با قدم‌های زیر حل می‌کنیم.
۳-در نقاط یک بخش ، نقطه‌ای را پیدا می‌کنیم که بیشترین فاصله با خط L را داشته باشد.(نقطه‌ی P).می‌دانیم که نقطه‌ی P جزو نقاطی است که در جواب نهایی وجود دارند. این نقطه را به دو سر پاره خط L وصل می‌کنیم ، یک مثلث تشکیل می‌شود ، واضح است که نقاطی که درون این مثلث هستند نمی‌توانند در جواب نهایی ما باشند.
۴- حال دو خطی که از P به دوسر پاره خط L کشیدیم مسئله را به دو زیر مسئله تقسیم می‌کنند. قدم شماره‌ی ۳ را برای این دو زیر مسئله تکرار می‌کنیم.
این کار را تا آنجایی ادامه می‌دهیم که هیچ نقطه‌ای در مرحله‌ی ۳ خارج از مثلث‌های تشکیل شده نماند.
پیچیدگی زمانی این مسئله نیز مانند $QuickSort$ در حالت متوسط $O(n * logn)$ است و در بدترین حالت $O(n^2)$ است.

۴. برای حل این مسئله از روش جستجوی دودویی استفاده می‌کنیم: برای پیدا کردن بزرگترین عنصر در آرایه $A =< a_1, a_2, ..., a_n >$ 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
$a_{n/2} < a_{n/2-1}$ آن‌گاه می‌دانیم پاسخ مسئله سمت چپ$a_{n/2}$ قرار دارد و برابر با بزرگترین عدد در $< a_1, ..., a_{n/2-1} >$ است که خود یک آرایه نافرم است و پاسخ آن را به صورت بازگشتی می‌توان یافت.اگر داشته باشیم $a_{n/2} < a_{n/2+1}$ آن‌گاه پاسخ مسئله سمت راست $a_{n/2}$ قرار دارد و برابر با بزرگترین عدد در $< a_{n/2+1}, ..., a_n >$ است که باز هم یک آرایه‌ی نافرم است و پاسخ آن را به صورت بازگشتی می‌توان یافت. زمان اجرای این الگوریتم از رابطه بازگشتی $T(n) = T(n/2) + O(1)$ پیروی می‌کند که با استفاده از قضیه اصلی می‌یابیم که $T(n) \in O(logn)$.

۵. برای حل این سوال از روشی مانند روش $mergesort$ استفاده می‌کنیم ، آرایه را به دونیمه A و B تقسیم می‌کنیم و فرض می‌کنیم مسئله برای هر نیمه حل شده است ، حال می‌خواهیم تعداد جفت‌هایی را بیابیم که یکی از آن‌ها در A و دیگری در B است. برای این کار نیز مانند $mergesort$ یک پوینتر روی هر آرایه در نظر می‌گیریم و روی اعضای دو آرایه حرکت می‌کنیم و جفت‌هایی که شرایط برنزی بودن داشته باشند را می‌شمریم.

زمان اجرای این الگوریتم همانند $mergesort$ از رابطه بازگشتی $T(n) = 2 * T(n/2) + O(n)$ پیروی می‌کند که با استفاده از قضیه اصلی می‌یابیم که $T(n) \in O(n * logn)$.

۶. برای حل این مسئله ، از $binarysearch$ روی تعداد صفحات استفاده می‌کنیم.در ابتدا مقدار کمینه برای تعداد صفحات را صفر و مقدار بیشینه را جمع تعداد تمام صفحات قرار می‌دهیم. حال بررسی می‌کنیم که آیا می‌توان مقدار میانگین این دو را به دانشجویان داد یا خیر. اگر امکان پذیر بود روی بازه صفر تا میانگین این کار را تکرار می‌کنیم و در غیر این صورت این کار را برای بازه‌ی میانگین تا بیشینه تکرار می‌کنیم.
برای بررسی این که آیا می‌توان مقداری را به دانشجویان داد ،به این صورت عمل می‌کنیم که به ترتیب صفحات را به دانشجویان $assign$ می‌کنیم تا زمانی که تعداد صفحات $assign$ شده از تعداد مورد نظر ما بیشتر نشود.در هنگام انجام این کار ، اگر نیاز باشد که تعداد دانشجویان بیشتر از m تا باشد به این نتیجه می‌رسیم که این تعداد صفحه قابل $assign$ کردن نیست.در غیر این صورت می‌توان این تعداد صفحه را به دانشجویان $assign$ کرد. زمان اجرای این الگوریتم از رابطه بازگشتی $T(n) = 2 * T(n/2) + O(n)$ پیروی می‌کند که با استفاده از قضیه اصلی می‌یابیم که $T(n) \in O(n * logn)$.