دانشکده‌ی علوم ریاضی

ساختمان داده‌ها

تمرین سری ۲

مدرّس: دکتر شهرام خزائی

مسأله‌ی ۱ (۳۵ نمره)

آ

الگوریتم Quick-Sort غیرتصادفی را بر روی آرایه زیر اجرا کنید و مقادیر آرایه را به ازای هربار اجرای Partition یادداشت کنید.(۱۰ نمره)

$$< ۴, ۳, ۷, ۱, ۵, ۲, ۶ >$$

پاسخ: پس از هر مرحله اجرای الگوریتم آرایه به ترتیب به صورت زیر در می‌آید:

$$< ۴, ۳, ۱, ۵, ۲, ۶, ۷ >$$

$$< ۱, ۲, ۴, ۵, ۳, ۶, ۷ >$$

$$< ۱, ۲, ۳, ۵, ۴, ۶, ۷ >$$

$$< ۱, ۲, ۳, ۴, ۵, ۶, ۷ >$$

ب

آرایه‌ای به طول ۷ را بیابید که بهترین حالت زمان اجرای الگوریتم مرتب سازی سریع بر روی آن رخ می‌دهد.(۱۰ نمره)

پاسخ: در آرایه‌ای بهترین حالت زمان اجرای الگوریتم مرتب‌سازی سریع رخ می‌دهد که پس از هر بار اجرای الگوریتم $Partition$ به دو زیر آرایه برابر تقسیم شود. برای آرایه به طول ۷ آرایه زیر این خاصیت را دارد:

$$< ۱, ۳, ۲, ۶, ۵, ۷, ۴ >$$

پ

چنین آرایه‌ای را برای هر n دلخواه بیابید.(۱۵ نمره)

پاسخ:
آرایه‌ی خواسته شده را بصورت استقرایی میسازیم. میدانیم که برای آرایه‌ی ۱ یا دو عضوی ترتیب چیده شدن اعداد مهم نیست و در صورت یک پارتیشن بندی انجام میشود. برای مجموعه ۳ عضوی نیز بهترین حالت این است که عضو وسطی اندیس پارتیشن بندی باشد تا طول آرایه را نصف کند و با یک پارتیشن بندی آرایه مرتب شود. (پایه استقرا)
حال برای $n > ۳$ آرایه را بدین صورت میسازیم که اگر فرض کنیم اعضای آرایه اعداد ۱ تا n هستند عضو با مقدار $[n/۲] + ۱$ را در انتهای آرایه قرار میدهیم و آرایه بصورت زیر در می آوریم: $A \qquad B \qquad [n/۲] + ۱$
که در آن طول زیر آرایه A برابر با $[(n-۱)/۲]$ و طول زیر آرایه B برابر با $n - ۱ - [(n-۱)/۲]$ است. طبق فرض استقرا اعداد ۱ تا $[(n-۱)/۲]$ را میتوانیم طوری قرار دهیم که تعداد پارتیشن بندی‌های آرایه مینمم شود. و این اعداد را در آرایه A قرار میدهیم و به مقدار هریک $[n/۲] + ۱$ واحد اضافه می‌کنیم. همچنین طبق فرض استقرا اعداد ۱ تا $n - ۱ - [(n-۱)/۲]$ را میتوانیم طوری در آرایه B قرار دهیم که تعداد پارتیشن بندی‌ها در بین این اعداد مینیمم شود. حال ادعا می‌کنیم در کل آرایه ایجاد شده تعداد پارتیشن‌ها تا مرتب شدن آرایه مینمم است. زیرا بعد از عملیات پارتیشن اول آرایه بصورت زیر در می آید: $B \qquad [n/۲] + ۱ \qquad A$ که در آن طول زیر آرایه A ، B تا حد ممکن بهم نزدیک است و در ادامه‌ی مراحل طبق فرض استقرا هر دو آرایه مستقل از هم با کم‌ترین تعداد پارتیشن بندی مرتب می‌شوند.

مسأله‌ی ۲ (۴۰ نمره)

اگر آرایه ی A از n عدد متمایز تشکیل شده باشند، یک وارونگی در این آرایه یک زوج i و j تعریف می‌شوند که $i < j$ و $A_i > A_j$ باشد.

آ

تعداد وارونگی‌های آرایه زیر را محاسبه کنید.(۵ نمره)

$$A = [۵, ۳, ۷, ۱, ۴, ۲, ۶]$$

پاسخ: زوج مرتب‌های زیر در آرایه بالا وارونگی دارند:

$$(۵,۳), (۵,۱), (۵,۴), (۵,۲), (۳,۱), (۳,۲), (۷,۱), (۷,۴), (۷,۲), (۷,۶), (۴,۲)$$

که تعداد آن‌ها برابر با ۱۱ می‌باشد.

ب

برای هر n دلخواه و زوج، اگر آرایه A به شکل زیر باشد تعداد وارونگی‌های آن را محاسبه کنید.(در جایگاه‌های فرد، اعداد زوج به صورت نزولی و در جایگاه‌های زوج، اعداد فرد به صورت صعودی قرار گرفته‌اند.)(۵ نمره)

$$A = [n, ۱, n - ۲, ۳, \ldots, , ۴, ۲, n - ۱]$$

پاسخ:
چون اعداد فرد بصورت صعودی قرار گرفته‌اند در بین خود آن‌ها وارونگی وجود ندارد و اعداد زوج چون بصورت نزولی قرار گرفته اند بین هر دو عضو آن‌ها یک وارونگی وجود دارد و تعداد وارونگی بین اعداد زوج برابر است با: $\frac{\frac{n}{۲} * (\frac{n}{۲} + ۱)}{۲}$. برای شمردن وارونگی بین اعداد زوج و فرد برای هر عدد زوج محاسبه میکنیم که با چند عدد فرد در آرایه ایجاد وارونگی میکند و به دنباله‌ی زیر می‌رسیم:

$$\frac{n}{۲} + \frac{n}{۲} - ۲ + \frac{n}{۲} - ۴ + \cdots + ۱ + ۱ + \cdots + \frac{n}{۲} - ۴ + \frac{n}{۲} - ۲$$

بنابراین تعداد کل وارونگی ها برابر است با:

$$\frac{n}{۲} + \frac{\frac{n}{۲} * (\frac{n}{۲} + ۱)}{۲} + \sum_{i=۱}^{[n/۴]} ۲ * (\frac{n}{۲} - ۲i)$$

پ

فرض کنید عناصر آرایه A جایگشتی تصادفی و یکنواخت از اعداد $[۱, ۲, \ldots, n]$ باشند. به کمک متغیر تصادفی نشانگر، امیدریاضی تعداد وارونگی‌های A را محاسبه کنید.(۱۵ نمره)

پاسخ: متغیر تصادفی نشانگر $X_{i,j}$ را بدین صورت تعریف می‌کنیم:
اگر a_i ، a_i تشکیل یک وارونگی دهند:

$X(i, j) = 1$

اگر a_i ، a_i تشکیل یک وارونگی ندهند:

$X(i, j) = 0$

و X را برابر با تعداد وارونگی‌ها تعریف می‌کنیم و داریم:

$$X = \sum_{i=1, j=1, i \neq j}^{n} X_{i,j}$$

اگر امیدریاضی طرفین تساوی بالا را محاسبه کنیم خواهیم داشت:

$$E[X] = E[\sum_{i=1, j=1, i \neq j}^{n} X_{i,j}]$$

و چون امیدریاضی یک رابطه خطی است میتوانیم سیگما را از درون امیدریاضی بیرون بیاوریم و داریم:

$$E[X] = \sum_{i=1,j=1,i\neq j}^{n} E[X_{i,j}]$$

و چون هر جفت به احتمال یک دوم تشکیل وارونگی میدهند و تعداد جفت‌ها برابر با

$$\frac{n(n-1)}{2}$$

است خواهیم داشت:

$$E[X] = \frac{1}{2} * n * \frac{(n-1)}{2}$$

ت

الگوریتمی ارائه دهید که تعداد وارونگی های یک آرایه را در بدترین حالت در مرتبه‌ی زمانی $\theta(n \log(n))$ محاسبه کند. (۱۵ نمره)

پاسخ: الگوریتم زیر الگوریتمی مشابه مرتب سازی ادغامی است. در این الگوریتم در هنگام پیدا کردن جای صحیح هر عدد محاسبه می‌کنیم که چند عضو هستند که در آرایه قبل از آن آمده اند و از آن بزرگ‌ترند و بدین ترتیب وارونگی‌های آن عدد محاسبه میشود. همچنین چون در این الگوریتم هر عدد درست یکبار در جای صحیح خود قرار میگیرد و همچنین همه‌ی اعدادی که بعد از آن در جای خود قرار می‌گیرند از آن بزرگ‌تر ژبودند میتوانیم تعداد این اعداد را با شمردن مجموع تعداد اعدادی که درجای اصلی خود قرار نگرفته اند و در آرایه قبل از آن آمده اند محاسبه کرد. همچنین این الگوریتم از مرتبه زمانی مرتب سازی ادغامی است و از $\theta(nlogn)$ است.

Algorithm 1 InvCount Function

Input: A, s, t
Output: $Number of inversions$

```
if s < t then
    mid = (s + t)/2
    return InvCount(A, s , mid) + InvCount(A, mid, t) + MergeCount(A, s, t)
end if
return 0
```

Algorithm 2 MergeCount Function

Input: s_1, t_2

Output: $CountedInversionInMerge$

$t_1 \leftarrow (s_1 + t_2)/2$
$B[s_1, \ldots, t_2] \leftarrow A[s_1, \ldots, t_2]$
$s_2 \leftarrow (s_1 + t_2)/2 + 1$
$sum \leftarrow 0$
$i \leftarrow s_1 - 1$
while $s_1 < t_1$ **and** $s_2 < t_2$ **do**
    $i \leftarrow i + 1$
    if $B_{s_1} < B_{s_2}$ **then**
        $A_i \leftarrow B_{s_1}$
        $s_1 \leftarrow s_1 + 1$
    else
        $A_i \leftarrow B_{s_2}$
        $s_2 \leftarrow s_2 + 1$
        $sum \leftarrow sum + t_1 - s_1$
    end if
end while
while $s_1 < t_1$ **do**
    $i \leftarrow i + 1$
    $A_i \leftarrow B_{s_1}$
    $s_1 \leftarrow s_1 + 1$
end while
while $s_2 < t_2$ **do**
    $i \leftarrow i + 1$
    $A_i \leftarrow B_{s_2}$
    $s_2 \leftarrow s_2 + 1$
end while
$return$ sum.

مسأله ۳ (۱۰ نمره)

فرض کنید یک آرایه‌ی مرتّب از اعداد طبیعی به طول n داریم که در آن هر عدد به غیر از یکی از آنها دقیقا دو بار ظاهر شده است و آن عدد استثنا فقط یک بار در آرایه ظاهر شده است. الگوریتمی از $O(\log(n))$ ارائه دهید که آن عدد استثنا را به عنوان خروجی چاپ کند.

پاسخ: چون در آرایه تنها یک عنصر وجود دارد که دوبار تکرار نشده است، طول آرایه فرد است. اگر عناصر آرایه را از ۰ تا $۲k + ۱$ شماره گذاری کنیم، تا قبل از دیده شدن عنصر تک در آرایه برای هر i داریم: $A_{۲i} = A_{۲i+۱}$ و بعد از دیده شدن عنصر تکین برای هر i داریم: $A_{۲i} = A_{۲i-۱}$. حال عضو وسط آرایه را در نظر میگیریم. چون طول آرایه فرد است حتما در یک جایگاه فرد آمده است. این عضو را با عضو بعدی و قبلی‌اش در آرایه مقایسه میکنیم. اگر با عضو قبلی‌اش برابر بود یعنی از ابتدای آرایه تا این عنصر،عنصر تک دیده نشده است و میتوان عنصر را برای نیمه‌ی دوم آرایه جست‌وجو کرد و اگر با عوض بعدی‌اش برابر بود یعنی عنصر تک در نیمه‌ی قبل آرایه دیده شده است و میتوان در نیمه‌ی اول آرایه آن عنصر را جست‌وجو کرد و اگر با هیچ یک برابر نبود خود عنصر خواسته شده است. چون مقایسه با عنصر قبلی و بعدی از $O(۱)$ امکان پذیر است و هربار نیز طول آرایه مورد جست‌وجو نصف میشود داریم: $T(n) = T(\frac{n}{۲}) + O(۱)$ و بنابراین مرتبه زمانی برنامه برابر با $O(\log(n))$ است.

مسأله‌ی ۴ (۳۰ نمره)

فرض کنید n نقطه‌ی متمایز در فضای دو بعدی $x - y$ داده شده است. به کمک روش تقسیم و حل الگوریتمی از مرتبه زمانی $O(n \log(n))$ ارائه دهید که دو نقطه با کمترین فاصله را به عنوان خروجی چاپ کند.

پاسخ: الگوریتم بصورت زیر است:
مرحله تقسیم: ابتدا با توجه به مولفه ی اول نقاط صفحه یعنی x آن ها، میانه ی این مقادیر را پیدا کرده و مقدار آن را m در نظر می گیریم.سپس خط $x = m$ را رسم می کنیم تا نقاط مان در صفحه به دو دسته مساوی تقسیم شوند.
حل: حال در هر دسته کمترین فاصله ی بین نقاط را می یابیم.فرض کنید این کمترین مقادیر در دسته اول و دوم به ترتیب برابر با $L_۱$ و $L_۲$ باشد.
حال کمترین فاصله ی بین جفت نقطه هایی که یکی از آنها در یک دسته و دیگری در دسته دیگر قرار دارند را پیدا می کنیم و در نهایت این سه مقدار را با هم مقایسه کرده و کمترین آن ها را بعنوان خروجی می دهیم.برای یافتن کمترین فاصله بین جفت نقطه هایی که یکی از آن ها در یک دسته قرار دارد و دیگری در دسته ی دیگر این گونه عمل می کنیم:
دو خط به فاصله ی $\delta = min\{L_۱, L_۲\}$ از خط میانی رسم می کنیم.یعنی دو خط $x = m + \delta$ و $x = m - \delta$.
حال نقاطی که بین دو خط $x = m + \delta$ و $x = m - \delta$ قرار دارند را در نظر می گیریم و آن ها را بر اساس ترتیب مقدرا y آن ها شماره گذاری می کنیم.حال فرض کنید s_i نقطه ای است که iامین مقدار کوچک مولفه ی y را در میان این نقاط داشته باشد.ادعا می کنیم اگر $|i - j| > ۸$ باشد آنگاه فاصله ی دو نقطه ی s_i و s_j از هم بیشتر از δ خواهد بود. زیرا اگر مربع های به ضلع $\frac{۱}{۲}\delta$ را در نظر بگیریم آنگاه مستطیل $۲\delta \times \delta$ که متشکل از ۸ مربع به ضلع $\frac{۱}{۲}\delta$ است حداکثر می تواند ۸ نقطه در خود داشته باشد.زیرا در هر مربع کوچک حداکثر یک نقطه می تواند قرار گیرد.بعبارتی اگر $|i - j| > ۸$ باشد آنگاه s_i و s_j در دو طرف این مستطیل که عرض آن δ است

قرار می گیرندو فاصله ی آن ها از δ بیشتر می شود.بنابراین کافیست هر نقطه در این ناحیه را تنها با ۸ نقطه ی بعد از خودش مقایسه کنیم تا کمترین فاصله در میان این مجموعه نقاط بدست آید.بعبارتی به زمانی از مرتبه ی $O(n)$ برای انجام این نیاز است.
ترکیب:در نهایت کم ترین فاصله هر یک از این سه دسته را مقایسه کرده و کمترین آنها را بعنوان خروجی درنظر می گیریم.
حال مرتبه زمانی این الگوریتم را بررسی می کنیم.از آنجا که در هر مرحله مسئله به دو زیر مسئله با اندازه ی $\lceil \frac{n}{۲} \rceil$ و $\lfloor \frac{n}{۲} \rfloor$ تقسیم می شود و زمان مرحله تقسیم و ترکیب هر کدام از مرتبه ی $O(n)$ هستند، مرتبه زمانی این الگوریتم بازگشتی بصورت زیر خواهد بود:

$$T(n) = ۲T(\frac{n}{۲}) + O(n)$$

که این رابطه همان رابطه ی بازگشتی زمان الگوریتم $merge - sort$ است که جواب آن برابر با $O(n \log(n))$ می باشد.

مسأله‌ی ۵ (۲۵ نمره)

الگوریتمی درجا (بدون هیچ حافظه اضافی) ارائه دهید که یک ماتریس $n * n$ در ورودی گرفته و آن را ۹۰ درجه به سمت چپ دوران دهد.
برای مثال اگر ورودی ماتریس زیر باشد:

$$\begin{matrix} ۱ & ۲ & ۳ \\ ۴ & ۵ & ۶ \\ ۷ & ۸ & ۹ \end{matrix}$$

خروجی الگوریتم باید به صورت زیر باشد:

$$\begin{matrix} ۳ & ۶ & ۹ \\ ۲ & ۵ & ۸ \\ ۱ & ۴ & ۷ \end{matrix}$$

پاسخ: ابتدا لمی ثابت میکنیم که میتوان عملیات $Swap$ کردن دو عدد را بدون استفاده از حافظه اضافی انجام داد. عدد a و b را در نظر بگیرید. عملیات‌های زیر دو عدد a و b را جا به جا می‌کنند:

$$a = a + b$$

$$b = a - b$$

$$a = a - b$$

حال با توجه به لم بالا الگوریتمی برای دوران ماتریس ارائه می‌دهیم:
ماتریس $n \times n$ را در نظر بگیرید.لایه های این ماتریس را این گونه درنظر میگیریم که لایه اول شامل درایه های سطر اول، سطر آخر،ستون اول و ستون آخر باشد.لایه دوم شامل درایه های سطر دوم(از درایه ی دوم این سطر تا درایه ی $n - ۱$ام این سطر)،سطر $n - ۱$ام(از درایه دوم تا درایه $n - ۱$ام این سطر) ،ستون دوم (از درایه دوم این ستون تا درایه $n - ۱$ام آن)و ستون $n - ۱$ام (از درایه دوم این ستون تا درایه $n - ۱$ام آن)باشد.و به همین

ترتیب الی آخر.لایه آخر نیز اگر n عددی فرد باشد همان درایه ی وسط ماتریس است و اگر n عددی زوج باشد همان زیرماتریس مربعی ۲×۲ وسط ماتریس اصلی است.
می دانیم بر اثر چرخش ۹۰ درجه ی ماتریس درایه های هر لایه دوباره روی همان لایه می افتند.بنابراین کافیست لایه به لایه این چرخش را اعمال کنیم تا در نهایت کل درایه های مارتیس ۹۰ درجه به سمت چپ بچرخند.
بنابراین از لایه اول آغاز می کنیم.برای چرخش ۹۰ درجه ماتریس به سمت چپ هر یک از درایه های این لایه بصورت زیر جابجا می شوند:

$$(۱, i) \rightarrow (n - i + ۱, ۱)$$
$$(i, ۱) \rightarrow (n, i)$$
$$(n, i) \rightarrow (n - i + ۱, n)$$
$$(i, n) \rightarrow (۱, i)$$

بعبارتی درایه های سطر اول به ستون اول، درایه های ستون اول به سطر آخر ، درایه های سطر آخر به ستون آخر و درایه های ستون آخر به سطر اول منتقل می شوند.حال مشابه لایه اول این روند را برای لایه های دیگر نیز تکرار می کنیم و به این ترتیب کل ماتریس ۹۰ درجه به سمت چپ دوران می یابد.برای اثبات این روند نیز می توان از استقرا استفاده کرد.به این صورت که فرض می کنیم با اعمال این الگوریتم به ازای kهای کوچکتر از n می توان ماتریس $k \times k$ را ۹۰ درجه به سمت چپ چرخش داد.حال ماتریس $n \times n$ را در نظر بگیرید.بدون در نظر گرفتن لایه اول طبق فرض استقرا می دانیم که می توان زیر ماتریس مربعی $(n - ۲) \times (n - ۲)$ در وسط ماتریس اصلی را ۹۰ درجه به سمت چپ چرخش داد.حال کافیست درایه های لایه اول را طبق همان روندی که در بالا گفتیم چرخش دهیم.(دقت کنید که حالت پایه استقرا یعنی $n = ۱$ به وضوح برقرار است.).به این ترتیب توانستیم ماتریس مان را با این روند ۹۰ درجه به سمت چپ چرخش دهیم.

مسأله‌ی ۶ (۱۵ نمره)

به کمک متغیر نشانگر محاسبه کنید، در مسئله ی Hire-Assistant با فرض این که کاندیداها به ترتیب تصادفی مصاحبه شوند، احتمال این که شما دقیقا دو بار استخدام کنید چقدر است؟

پاسخ:
برای اینکه دقیقا دو نفر انتخاب شوند باید تنها اولین نفر در دنباله و فرد با بیش‌ترین مقدار انتخاب شوند. اگر اولین نفر دنباله مقداری برابر با i داشته باشد باید در بین $n - i$ عدد بزرگ‌تر از آن اول عددn آمده باشد. که این به $۱/(n - i)$ رخ میدهد. همچنین احتمال آمدن نفر i در اول لیست $۱/n$ است. پس اگر احتمال اینکه دقیقا دو نفر انتخاب شوند و نفر اول مقدارش برابر با i باشد را با $P(X_i)$ نشان دهیم داریم:

$$P(X_i) = \frac{۱}{n} * \frac{۱}{(n - i)}$$

و اگر پیشامد اینکه دقیقا دو نفر انتخاب شوند را با متغیر تصادفی X نشان دهیم داریم:

$$P(X) = \sum_{i=۱}^{n} P(X_i) = \frac{۱}{n} * \sum_{i=۱}^{n} \frac{۱}{(n - i)}$$

که همانطور که میدانیم مقدار این سیگما از $\theta(logn)$ است و در نتیجه مقدار کل تابع از $\theta(logn/n)$ خواهد بود.

مسأله‌ی ۷ (۲۵ نمره)

فرض کنید تابع $RAND(a,b)$یک عدد صحیح کاملا تصادفی در بازه‌ی $[a,b]$ تولید می‌کند. با فرض داشتن $RAND(0,1)$ و استفاده از این تابع، بهترین الگوریتمی که می‌توانید را برای محاسبه تابع $RAND(a,b)$ ارائه دهید و مرتبه زمانی الگوریتم را بر حسب مقدار b و a محاسبه کنید.

پاسخ: عدد $c = b - a + 1$ را در نظر میگیریم. اگر الگوریتمی ارائه دهیم که عدد صحیح کاملا تصادفی در بازه $[0,c)$ تولید کند آنگاه جمع خروجی الگوریتم با عدد a عدد تصادفی در بازه مطلوب می‌شود.
برای ساختن عدد صحیح کاملا تصادفی در این بازه اولین عددی از توان‌های ۲ که بزرگتر یا مساوی با عدد c است را در نظر می‌گیریم. اگر این عدد برابر با 2^k باشد یک رشته ی k بیتی را در نظر میگیریم. با استفاده از تابع $RAND(0,1)$، k بیت تصادفی تولید میکنیم. این k بیت را یک عدد در مبنای ۲ در نظر می‌گیریم، اگر این عدد در بازه $[0,c)$ باشد میتوانیم عدد تصادفی در بازه مورد نظر را پیدا کنیم و الگوریتم را پایان بدهیم، در غیر این صورت الگوریتم پیدا کردن k بیت تصادفی را دوباره تکرار میکنیم.
چون در هر بار ایجاد کردن k بیت تصادفی الگوریتم به احتمال بیش از $\frac{1}{2}$ به پاسخ می‌رسد امیدریاضی تعداد تکرار اجرای الگوریتم کمتر از ۲ است. همچنین اگر مرتبه زمانی اجرای $RAND(0,1)$ را برابر با T در نظر بگیریم، چون k از $O(\log(c))$ است مرتبه زمانی اجرای الگوریتم برابر است با: $T * O(\log(b-a))$