دانشکده‌ی علوم ریاضی

داده‌ساختارها و الگوریتم‌ها — ۲۳ مهر ۹۲

جلسه‌ی ۷: مسأله زیرآرایه بیشینه و الگوریتم‌های تصادفی

مدرّس: دکتر شهرام خزائی — *نگارنده: معین زمانی و امیر عزیزی*

۱ مسأله زیرآرایه بیشینه

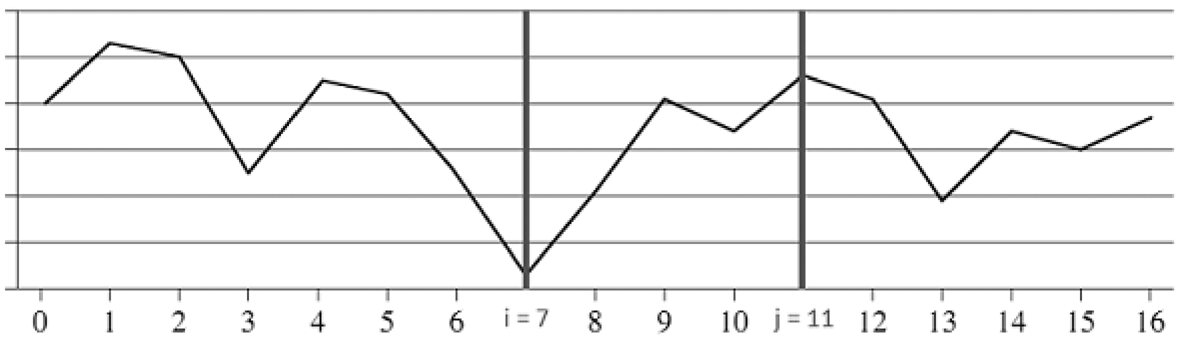
در این مسأله هدف پیدا کردن بزرگترین زیر مجموعه پیوسته از اعداد یک آرایه است، که بیشترین مقدار ممکن را داشته باشد. فرض بر این است که تعدادی از اعداد منفی می‌باشند، زیرا درصورت مثبت بودن اعداد مسأله بدیهی است.

- **ورودی:** آرایه‌ای از اعداد $A[1 \cdots n]$.
- **خروجی:** اندیس‌های i, j که زیرآرایه $A[i \cdots j]$ دارای بزرگترین مجموع در بین همه زیرآرایه‌های ممکن است.

برای آشنایی با کاربرد عملی این مسأله، مثال زیر را در نظر بگیرید.

مثال ۱ *فرض کنیم قیمت‌های سهام معامله شده‌ای را در n روز متوالی داریم. در این حالت سوال‌هایی پیش می‌آید، از قبیل اینکه سهام را باید در چه روزی بخریم؟ سهام خریداری شده را در چه روزی بفروشیم؟ برای حل این مسأله، آن را به مسأله زیرآرایه بیشنه تبدیل می‌کنیم:*

$$A[i] = (\text{قیمت سهام در روز } i\text{اُم}) - (\text{قیمت سهام در روز } (i-1)\text{اُم})$$



شکل ۱: نمودار قیمت سهام

آنگاه زیرآرایه ناتهی دارای مجموع بیشینه، نشان دهنده روزهایی خواهد بود که باید صاحب سهام می‌بودیم. اگر زیرآرایه بیشینه $A[i \cdots j]$ باشد، آنگاه باید سهام را در روز $(i-1)$اُم خرید و در روز jاُم فروخت.

۱.۱ روش جستجوی کامل

مسأله زیرآرایه بیشینه را می‌توان با استفاده از روش بدیهی جستجوی کامل[۱] حل کرد. در این روش باید تمام $\binom{n}{2}$ زیرآرایه ممکن را بررسی کنیم، که دراین صورت مرتبه زمانی $\Theta(n^2)$ خواهد بود.

۲.۱ روش تقسیم و حل

مراحل مختلف این روش را به شکل زیر دسته‌بندی می‌کنیم:

- **زیرمسأله‌ها:** زیرآرایه بیشینه از $A[low \cdots high]$ که در فراخوانی اول $low = 1$ و $high = n$ است.
- **تقسیم:** محاسبه نقطه میانی که آن را mid می‌نامیم و تقسیم کردن زیر آرایه به دو زیرآرایه با اندازه‌های تقریباً یکسان.
- **حل:** یافتن زیرآرایه‌ای بیشینه از $A[low \cdots mid]$ و $A[mid + 1 \cdots high]$.
- **ترکیب:** یافتن زیرآرایه‌ای بیشینه که از نقطه میانی می‌گذرد و انتخاب بهترین جواب از میان این ۳ راه حل.

با استفاده از تابع FMCS می‌توان زیرآرایه‌ای بیشینه که شامل نقطه میانی می‌باشد یافت:

Algorithm 1 Find Maximum Crossing Subarray

```
function FMCS(A, low, mid, high)
    sum_l ← ∞
    sum ← 0
    for i = mid downto low do
        sum ← sum + A[i]
        if sum > sum_l then
            sum_l ← sum
            max_l ← i
    sum_r ← ∞
    sum ← 0
    for j = mid + 1 to high do
        sum ← sum + A[j]
        if sum > sum_r then
            sum_r ← sum
            max_r ← j
    return (max_l, max_r, sum_l + sum_r)
```

این تابع در حلقه اول که $mid - low + 1$ بار اجرا می‌شود، زیرآرایه‌ای بیشینه که شامل عنصر میانی و عناصری که در سمت چپ آن قرار دارد می‌یابد.
حلقه دوم هم که $high - (mid + 1) + 1$ بار اجرا می‌شود، زیرآرایه‌ای بیشینه که شامل اولین عنصر بعد از عنصر میانی و عناصری که در سمت راست آن قرار دارد می‌یابد. پس مرتبه اجرای این الگوریتم $\Theta(n)$ است که $n = high - low + 1$.
روند حل و تقسیم برای مسأله زیرآرایه بیشینه توسط تابع FMS انجام می‌شود:

[۱] brute-force

Algorithm 2 Find Maximum Subarray

```
function FMS(A, low, high)
    if high == low then
        return (low, high, A[low])
    else
        mid ← ⌊(low+high)/2⌋
        (low_l, high_l, sum_l) ← FMS(A, low, mid)
        (low_r, high_r, sum_r) ← FMS(A, mid + 1, high)
        (low_c, high_c, sum_c) ← FMCS(A, low, mid, high)
        if sum_l ≥ sum_r and sum_l ≥ sum_c then
            return (low_l, high_l, sum_l)
        else
            if sum_r ≥ sum_l and sum_r ≥ sum_c then
                return (low_r, high_r, sum_r)
            else
                return (low_c, high_c, sum_c)
```

۳.۱ آنالیز زمان اجرای الگوریتم FMS

برای سادگی تحلیل زمان اجرای الگوریتم فرض می‌کنیم n توانی از ۲ باشد. زمان اجرا را هنگامی که اندازه ورودی مسأله n است را با $T(n)$ نشان می‌دهیم. برای حالت پایه یعنی وقتی که $n = ۱$ است، خط دوم اجرا خواهد شد که زمان ثابتی طول می‌کشد پس:

$$T(۱) = \Theta(۱)$$

حالت بازگشتی زمانی رخ خواهد داد که $n > ۱$ باشد، خطوط ۱و۳ زمان ثابتی طول می‌کشند. همچنین زمان اجرای خطوط چهارم و پنجم $۲T(n/۲)$ می‌باشد. همچنین می‌دانیم زمان اجرای FMCS از مرتبه $\Theta(n)$ است، پس:

$$\begin{aligned} T(n) &= \Theta(۱) + ۲T(n/۲) + \Theta(n) + \Theta(۱) \\ &= ۲T(n/۲) + \Theta(n) \end{aligned}$$

در نتیجه رابطه کلی $T(n)$ به شکل زیر خواهد بود:

$$T(n) = \begin{cases} \Theta(۱) & n = ۱ \\ ۲T(n/۲) + \Theta(n) & n > ۱ \end{cases}$$

که همان رابطه بازگشتی MergeSort می‌باشد، پس:

$$T(n) = \Theta(n \log n)$$

۲ الگوریتم‌های تصادفی

الگوریتم‌های تصادفی در مواردی به‌کار می‌روند که ما از توزیع احتمال داده‌های ورودی‌ها اطلاعی نداشته‌باشیم یا نتوانیم آن را به صورت محاسباتی مدل‌سازی کنیم. در اینگونه موارد با تصادفی‌سازی در الگوریتم، یک توزیع روی ورودی‌ها (که معمولاً یکنواخت است) تحمیل می‌کنیم. یک الگوریتم تصادفی است، اگر بخشی از رفتار آن تحت تاثیر مقدار دریافت شده از یک مولد عدد تصادفی [۲] باشد. به عبارت دیگر خروجی یک الگوریتم تصادفی بر خلاف الگوریتم

[۲] random-number generator

قطعی[۳] فقط به داده ورودی بستگی ندارد، بلکه هر بار اجرای الگوریتم روی یک داده ورودی ثابت منجر به اجراهای متفاوتی می‌شود. در بعضی موارد اجرای‌های متفاوت منجر به خروجی‌های مختلفی برای الگوریتم می‌شود، در بعضی موارد منجر به زمان‌های اجرای مختلفی برای الگوریتم می‌شود، و در برخی دیگر هر دو با هم اتفاق می‌افتد.

۱.۲ مسأله استخدام

شرکتی می‌خواهد هموراه بهترین فرد را برای یک موقعیت کاری حساس استخدام کند. این شرکت با یک آژانس کاریابی قرارداد دارد به طوری که آژانس ابتدا یک لیست شامل n فرد را در اختیار شرکت قرار می‌دهد. سپس، این شرکت هر روز با یکی از افراد لیست به عنوان کاندیدا مصاحبه می‌کند. شرکت باید پس از مصاحبه بلافاصله تصمیم استخدام کارمند جدید را بگیرد؛ در صورت مثبت بودن تصمیم باید کارمند استخدام‌شده قبلی خود را اخراج کند و کارمند جدید را استخدام کند که هزینه‌ی ثابتی را برای شرکت به دنبال دارد. شرکت متعهد است که همیشه بهترین کاندیدا را از بین افراد استخدام کند. از آنجا که شرکت باید همیشه کسی را در استخدام خود داشته باشد همیشه اولین فرد استخدام می‌شود. فرض کنیم کاندیداها از ۱ تا n شماره گزاری شوند و شرکت به ترتیب همان لیست ارسالی آژانس افراد را برای مصاحبه فراخوانی کند. شبه‌کد[۴] حل این مسأله به صورت زیر است.

Algorithm 3 Algorithm: HIRE

```
function HIRE(list of n candidates)
    hire candidate 1
    best ← 1
    for i = 2 to n do
        interview candidate i
        if candidate i is better than candidate best then
            hire candidate i
            best ← i
    return list of hired candidates
```

حال اگر تعداد افراد استخدامی m نفر باشد، هزینه استخدام $O(m)$ خواهد بود. مقدار m بستگی به لیست دریافتی از آژانس دارد. در بهترین حالت شرکت فقط نفر اول را استخدام می‌کند و در بدترین حالت شرکت همه‌ی n کاندیدا را استخدام می‌کند. بدترین حالت زمانی اتفاق می‌افتد که کاندیداها به ترتیب صعودی از لحاظ کیفیت در لیست آژانس قرار گیرند و بهترین حالت زمانی اتفاق می‌افتد که نفر اول لیست بهترین گزینه باشد. اگر فرض کنیم که آژانس لیست خود را با ترتیب کاملا تصادفی ارسال می‌کند، بهترین و بدترین حالت به‌ترتیب با احتمال $\frac{1}{n}$ و $\frac{1}{n!}$ رخ می‌دهند. برای مقابله با هر گونه سوءاستفاده احتمالی آژانس و جلوگیری از تحمیل بیشترین هزینه $O(n)$ به شرکت توسط آژانس، شرکت تصمیم می‌گیرد که ترتیب افراد را خود به تصادف انتخاب کند. بنابراین از الگوریتم تصادفی زیر استفاده می‌کند:

[۳] deterministic algorithm
[۴] pseudocode

Algorithm 4 Algorithm: RANDOMIZED-HIRE

```
function RANDOMIZED-HIRE(list of n candidates)
    randomly permute list of candidates
    hire candidate 1
    best ← 1
    for i = 2 to n do
        interview candidate i
        if candidate i is better than candidate best then
            hire candidate i
            best ← i
    return list of hired candidates
```

در این صورت، در هر اجرا به تناسب ترتیب مصاحبه با کاندیداها هزینه تغییر می‌کند. هزینه‌ای که به شرکت تحمیل می‌شود توسط متوسط تعداد افراد استخدام شده تعیین می‌گردد. ادامه این مبحث به محاسبه متوسط هزینه شرکت اختصاص دارد.

۲.۲ متغیر تصادفی شاخص

متغیر تصادفی شاخص، تکنیکی قدرتمند برای محاسبه امید ریاضی یک متغیر تصادفی است. این تکنیک در شرایطی که عدم استقلال پیشامدها وجود دارد بسیار مفید می‌باشد.

تعریف ۱ *فرض کنید A یک رخداد در یک فضای نمونه‌ای داده شده با توزیع احتمال معلوم باشد. متغیر تصادفی شاخص رخداد A، که با $\mathrm{I}\{A\}$ نشان داده می‌شود، اینگونه تعریف می‌شود:*

$$\mathrm{I}\{A\} = \begin{cases} ۱ & \mathtt{if\ A\ occurs} \\ ۰ & \mathtt{if\ A\ does\ not\ occur} \end{cases}$$

لم ۱ *برای هر رخداد A، فرض کنیم $X_A = \mathrm{I}\{A\}$. آنگاه* $\mathrm{E}[X_A] = \Pr\{A\}$.

برهان. فرض می‌کنیم $\bar{A}$، متمم A باشد، داریم:

$$\mathrm{E}[X_A] = \mathrm{E}[\mathrm{I}\{A\}] = ۱ \times \Pr\{A\} + ۰ \times \Pr\{\bar{A}\} = \Pr\{A\}$$

■

مثال ۲ *مشخص کنید در یک بار پرتاب سکه سالم، امید ریاضی تعداد شیرها چقدر است.*

جواب. فضای نمونه این مسأله $\{H, T\}$ است. همچنین داریم $\Pr\{H\} = \Pr\{T\} = \frac{۱}{۲}$. متغیر تصادفی شاخص را به صورت $X_H = \mathrm{I}\{H\}$ تعریف می‌کنیم که تعداد شیرها را در یک پرتاب می‌شمارد. از آن جایی که $\Pr\{H\} = \frac{۱}{۲}$، از لم ؟؟ نتیجه می‌گیریم که $\mathrm{E}[X_H] = \frac{۱}{۲}$.

مثال ۳ *تعداد شیرهای مورد انتظار در n پرتاب سکه را مشخص کنید.*

جواب. فرض می‌کنیم X متغیر تصادفی تعداد شیرها در n پرتاب سکه باشد. برای $i = ۱, ۲, \ldots, n$، متغیر تصادفی X_i را متغیر تصادفی شاخص رخداد آمدن شیر در پرتاب iام تعریف می‌کنیم. یعنی:

$$X_i = \mathrm{I}\{\mathtt{the\ outcome\ of\ the\ }i\mathtt{th\ throw\ is\ head}\}\ .$$

پس داریم $X = \sum_{i=۱}^{n} X_i$. طبق لم ؟؟، داریم $\mathrm{E}[X_i] = \Pr\{H\}$ برای $i = ۱, ۲, \ldots, n$. در نتیجه امید ریاضی تعداد شیرها به صورت $\mathrm{E}[X] = \mathrm{E}[\sum_{i=۱}^{n} X_i]$ است. از آن جا که می‌دانیم امید ریاضی، خطی است و اینکه مقدار هر $\mathrm{E}[X_i]$ را می‌دانیم می‌توانیم مسأله را حل کنیم.

$$
\begin{aligned}
\mathrm{E}[X] &= \mathrm{E}[\textstyle\sum_{i=1}^{n} X_i] \\
&= \textstyle\sum_{i=1}^{n} \mathrm{E}[X_i] \\
&= \textstyle\sum_{i=1}^{n} \frac{1}{2} \\
&= \frac{n}{2}
\end{aligned}
$$

دقت کنید که اگر با استفاده از توزیع احتمال متغیر تصادفی X می‌خواستیم $\mathrm{E}[X]$ را بدست آوریم باید ابتدا مقدار احتمال $\Pr\{X = k\}$ (برای $k = 0, 1, \ldots, n$) و سپس مجموع $\sum_{k=0}^{n} k \Pr\{X = k\}$ را محاسبه می‌کردیم. با استفاده از توزیع دوجمله‌ای همان نتیجه قبلی بدست می‌آید:

$$
\mathrm{E}[X] = \sum_{k=0}^{n} k \frac{\binom{n}{k}}{2^n} = \frac{1}{2} \ .
$$

۳.۲ آنالیز الگوریتم استخدام تصادفی

در مسأله استخدام فرض می‌کنیم که کاندیداها به ترتیب تصادفی برسند. متغیر تصادفی X را تعداد دفعات افراد استخدام شده در نظر می‌گیریم. برای $i = 1, \ldots, n$ متغیر تصادفی X_i را متغیر تصادفی شاخص رخداد استخدام شدن کارمند iام تعریف می‌کنیم. یعنی:

$$
X_i = \mathrm{I}\{\texttt{the } i\texttt{th candidate is hired}\} \ .
$$

کاندیدا i تنها در صورتی انتخاب می‌شود که بهتر از هر یک از کاندیداهای $1, 2, \ldots, i-1$ باشد. از آن جا که کاندیداها به ترتیب کاملا تصادفی برای مصاحبه حضور می‌یابند، هر یک از این i کاندیدا اول، دارای شانس برابر برای بهترین بودن دارند. بنابراین احتمال اینکه کاندیدا i ام بهترین کاندیدا در بین i کاندیدای اول باشد برابر $\frac{1}{i}$ است. با استفاده از لم ؟؟ داریم $\mathrm{E}[X_i] = \frac{1}{i}$. حال به محاسبه $\mathrm{E}[X]$ می‌پردازیم

$$
\begin{aligned}
\mathrm{E}[X] &= \mathrm{E}[\textstyle\sum_{i=1}^{n} X_i] \\
&= \textstyle\sum_{i=1}^{n} \mathrm{E}[X_i] \\
&= \textstyle\sum_{i=1}^{n} \frac{1}{i} \\
&= \ln n + O(1)
\end{aligned}
$$

بنابراین هزینه مورد انتظار استخدام $O(\log n)$ است، که بسیار بهتر از هزینه بدترین حالت، یعنی $O(n)$ است.

۴.۲ نحوه‌ی استفاده از مقادیر تصادفی در الگوریتم‌ها

در شبه‌کدهای خود از تابع $\textsc{Random}(a, b)$ برای تولید یک عدد صحیح کاملاً تصادفی از مجموعه اعداد صحیح $\{a, a+1, \ldots, b\}$ استفاده می‌کنیم که a و b اعداد صحیح هستند؛ یعنی همه $b - a + 1$ مقدار ممکن r، که $a \le r \le b$، شانس انتخاب یکسانی دارند. در این‌صورت اصطلاحاً گفته می‌شود که الگوریتم از $\log(b - a + 1)$ بیت تصادفی استفاده می‌کند.

سؤال ۱ *چگونه می‌توان با استفاده از تابع* $\textsc{Random}$ *یک آرایه را تصادفی کرد؟ به عبارت دیگر چگونه می‌توان یک جایشگشت کاملاً تصادفی روی ترتیب عناصر یک آرایه اعمال کرد؟*