دانشکده‌ی علوم ریاضی

داده‌ساختارها و الگوریتم‌ها — مهر ۱۳۹۳

جلسه‌ی ۷: مسأله محاسبه تعداد زوج معکوس

مدرّس: دکتر شهرام خزائی — *نگارنده: آرمیتا خواجه‌نصیری*

۱ مسأله محاسبه تعداد زوج معکوس

تعریف ۱ *فرض کنید $A[۱\cdots n]$ آرایه‌ای از n عدد متمایز باشد. اگر $i < j$ و $A[i] > A[j]$ آنگاه زوج مرتب (i,j) را یک زوج معکوس[۱] از A می‌نامند.*

هدف این جلسه بررسی مسأله محاسبه تعداد زوج معکوس‌ها است که به صورت زیر تعریف می‌شود.

- **ورودی:** آرایه‌ای از اعداد $A[۱\cdots n]$.
- **خروجی:** تعداد زوج معکوس‌های آرایه.

برای پیدا کردن درک بهتری از مفهوم زوج معکوس، مثال زیر را در نظر بگیرید.

مثال ۱ *زوج معکوس‌های موجود در آرایه زیر را بیابید.*

۱	۳	۵	۲	۴	۶

همانطور که گفته شد باید زوج مرتب‌هایی را بیابیم که وقتی از ابتدای آرایه به جلو می‌رویم، به ترتیب نباشند. پس زوج معکوس‌های این آرایه (۳,۲) و (۵,۲) و (۵,۴) *هستند.*

۱.۱ روش جستجوی کامل

مسأله محاسبه تعداد زوج معکوس را می‌توان با استفاده از روش بدیهی جستجوی کامل[۲] حل کرد. در این روش باید با استفاده از ۲ حلقه تمام زیرآرایه های ممکن را بررسی کنیم، که دراین صورت مرتبه زمانی $\Theta(n^۲)$ خواهد بود..پس برای داشتن پیچیدگی کمتر دنبال راهکار بهتری هستیم.

[۱] Inversion
[۲] brute-force

۲.۱ روش تقسیم و حل

مراحل مختلف این روش را به شکل زیر دسته‌بندی می‌کنیم:

- **زیرمسأله‌ها:** شمارش تعداد زوج معکوس ها از زیرآرایه $A[low \cdots high]$ که در فراخوانی اول $low = ۱$ و $high = n$ است.
- **تقسیم:** محاسبه نقطه میانی که آن را mid می‌نامیم و تقسیم کردن زیرآرایه به دو زیرآرایه با اندازه‌های تقریباً یکسان.
- **حل:** یافتن تعداد زوج‌های معکوس از $A[low \cdots mid]$ و $A[mid + ۱ \cdots high]$ به طور بازگشتی و تعداد زوج معکوس‌های جدا ازهم[۳]. یعنی زوج‌های معکوسی که اندیس یکی از اعضا آن در نیمه چپ آرایه و دیگری در نیمه راست آرایه است.
- **ترکیب:** محاسبه حاصل جمع ۳ مقدار به دست آمده در قسمت حل.

Algorithm 1 Count Inversions

```
function CountInv(A, low, high)
    if low == high then
      return 0
    else
        mid ← (low+high)/2
    l ← CountInv(A, low, mid)
    r ← CountInv(A, mid + 1, high)
    s ← SplitInv(A, low, mid, high)
      return l + r + s
```

در این تابع تعداد زوج‌های معکوس در نیمه چپ و راست آرایه را با فراخوانی تابع به صورت بازگشتی پیدا می‌کنیم. حال باید به دنبال یافتن راهی برای محاسبه تعداد زوج معکوس‌های جدا ازهم باشیم. یک روش بدیهی این است که همهٔ زوج‌هایی که یک اندیس سمت راست و یک اندیس سمت چپ دارند را دو به دو چک کنیم. اگر زمان اجرا را هنگامی که اندازه ورودی مساله n است، با $T(n)$ نشان دهیم، خواهیم داشت:

$$T(n) = ۲T(\frac{n}{۲}) + \Theta(n^۲) \Rightarrow T(n) = \Theta(n^۲)$$

حال به دنبال روشی هستیم که زوج معکوس‌های جدا ازهم را در زمان خطی $\mathcal{O}(n)$ محاسبه کند.

۳.۱ استفاده از مرتب سازی ادغامی برای یافتن زوج معکوس‌های جدا ازهم

ایده اصلی برای پیدا کردن تعداد زوج معکوس‌های جدا ازهم با پیچیدگی کمتر این است که تابع CountInversion ورودی را نیز مرتب کند. در این صورت آرایه از اول تا mid و از $mid + ۱$ تا انتها مرتب شده و به ترتیب در آرایه‌های B و C ذخیره می شود. انگیزه اصلی برای این کار این است که Merge به خودی خود زوج معکوس‌های جدا ازهم را نمایان می‌کند.

---

[۳] Split Inversion

Algorithm 2 SORT-AND-COUNT INVERSIONS

function SORT-AND-COUNTINV$(A, low, high)$
  if $low == high$ **then**
    return 0
  else
    $mid \leftarrow \frac{low+high}{2}$
  $(B, l) \leftarrow$ SORT-AND-COUNTINV(A, low, mid)
  $(C, r) \leftarrow$ SORT-AND-COUNTINV$(A, mid+, high)$
  $s \leftarrow$ COUNTSPLITINV$(A, low, mid, high)$
    return $l + r + s$

حال الگوریتم MERGE را روی دو زیرآرایه به دست آمده یعنی B و C اجرا می‌کنیم و فرض می‌کنیم آرایه خروجی D با طول n است.

Algorithm 3 MERGE

function MERGE $(A, low, mid, high)$
  $[assumes\ B[low \cdots mid]\ and\ C[mid+1 \cdots high]\ are\ sorted\ arrays\ of\ length\ \frac{n}{2}\ and\ D\ is\ the\ output\ array\ of\ length\ n\ ]$
$i = 1$
$j = 1$
  for $k = 1$ **to** n **do**
    if $B[i] < C[j]$ **then**
$D[k] = B[i]$
      $i \leftarrow i + 1$
    else
$D[k] = C[j]$
      $j \leftarrow j + 1$

لم ۱ *تعداد زوج معکوس‌های جدا از هم شامل یک عضو مانند y از آرایه دوم (C)، دقیقا برابر است با تعداد اعداد باقی‌مانده در آرایه اول (B) وقتی که y طی اجرا شدن الگوریتم $Merge$ داخل آرایهٔ خروجی (D) کپی می شود.*

برهان. *فرض کنید x یک عضو از آرایه اول یعنی B باشد.*

۱. *اگر x قبل از y در آرایه خروجی یعنی D چاپ شده باشد، یعنی $x < y$.است $\Leftarrow$ هیچ زوج معکوسی شامل x و y وجود ندارد.*

۲. *اگر y قبل از x در آرایه خروجی یعنی D چاپ شده باشد، یعنی $y < x$ است. $x \Leftarrow$ و y یک زوج معکوس جدا از هم هستند.*

مثال ۲ *فرض کنید قرار است دو آرایه زیر را ادغام کنیم.*

۱	۳	۵

۲	۴	۶

با اجرا کردن الگوریتم روی این دو آرایه به خروجی زیر می رسیم.

۱	۲	۳	۴	۵	۶

وقتی که ۲ در آرایه خروجی چاپ می شود، زوج معکوس‌های جدا از هم (۳,۲) و (۵,۲) نمایان می شوند. و وقتی ۴ وارد آرایه خروجی می شود، متوجه زوج معکوس جدا از هم (۵,۴) می‌شویم.

۴.۱ تحلیل زمان اجرای الگوریتم Sort-And-Count

پس باید هنگام ادغام کردن دو زیر آرایه مرتب شده، تعداد زوج معکوس‌های جدا از هم را نگه داریم. یعنی وقتی عضوی از آرایه دوم (C) در خروجی چاپ میشود، باید تعداد کّل زوج معکوس‌های جدا از هم را با تعداد اعضا باقی مانده در آرایه اول (B) جمع کنیم.
پس زمان اجرای زیر مساله یعنی Merge-And-CountSplitInv *برابر* $\mathcal{O}(n) = \mathcal{O}(n) + \mathcal{O}(n)$ *است و الگوریتم* Sort-And-Count *درست مانند مرتب سازی ادغامی در زمان* $\mathcal{O}(n \log n)$ *اجرا می شود.*