یادآوری جلسه چهارم — **تحلیل مجانبی و روابط بازگشتی** — محمدهومان کشوری

در جلسه قبل به تحلیل مجانبی و روابط بازگشتی پرداخته شد. جلسه را با چند خاصیت ابتدایی از نمادهای O, Ω شروع کردیم:

$$f(n) = O(g(n)) \Leftrightarrow g(n) = \Omega(f(n)) \qquad (1)$$

$$\begin{matrix} f_1(n) = O(g_1(n)) \\ f_2(n) = O(g_2(n)) \end{matrix} \left\{ \begin{matrix} f_1(n) \times f_2(n) = O(g_1(n) \times g_2(n)) \\ f_1(n) + f_2(n) = O(Max(g_1(n), g_2(n))) \end{matrix} \right. \qquad (2)$$

در ادامه، راجع به این موضوع صحبت کردیم که در اکثر توابعی که ما در درس مورد بررسی قرار می دهیم، می توان با بزرگ کردن میزان c مقدار n_0 را یک در نظر گرفت. در این حالت، مثلا تعریف O به این شکل می‌شود:

$$f(n) = O(g(n)) : \quad \exists c \quad \forall n \quad f(n) \leq cg(n)$$

بنابراین، اگر تابعی برابر با $O(n)$ باشد، به این معنی است که یک مقدار c وجود دارد که مقدار آن کمتر از cn می‌باشد. از این تعریف در بخش روابط بازگشتی استفاده خواهیم کرد. سپس به تحلیل روابط بازگشتی پرداختیم. به طور کلی برای تحلیل روابط بازگشتی از ۴ روش استفاده می‌کنیم: **روش جایگذاری ، استقرا ، درخت بازگشتی و قضیه اصلی.**

در روش جایگذاری در واقع با بسط دادن رابطه بازگشتی سعی در به دست آوردن رابطه صریح می‌کنیم. به عنوان مثال:

$$\begin{aligned} T(1) = O(1) \qquad T(n) &= T(\tfrac{n}{2}) + O(1) \\ &= T(\tfrac{n}{2}) + O(1) \\ &= T(\tfrac{n}{4}) + O(1) + O(1) \\ &\dots \\ &= T(1) + \underbrace{O(1) + O(1) + \dots + O(1)}_{\log(n)} \Rightarrow \\ &= O(\log(n)) \times O(1) = O(\log(n)). \end{aligned}$$

همچنین نشان دادیم که برای تحلیل روابط بازگشتی که بخش عملیات آن مقدار ثابت نیست، ابتدا نیاز است صورت دقیق‌تر رابطه را بنویسیم. به عنوان نمونه اگر رابطه بازگشتی ما به صورت $T(n) = T(n/2) + O(n)$ و $T(1) < c_1$ باشد، ابتدا آن را با استفاده از تعریف نماد O بازنویسی می‌کنیم:

$$T(n) \leq T(n/2) + c_2(n), \quad T(1) \leq c_1$$

حال با استفاده از جایگذاری خواهیم داشت:

$$\begin{aligned} T(n) &\leq T(n/2) + c_2 n \\ &\leq T(n/4) + c_2 n/2 + c_2 n \\ &\leq T(n/8) + c_2 n/4 + c_2 n/2 + c_2 n \\ &\dots \\ &\leq c_1 + c_2 + 2c_2 + \dots + c_2 n/2 + c_2 n \\ &\leq c_1 + 2c_2 n = O(n). \end{aligned}$$

الگوریتم مرتب سازی ادغامی : این الگوریتم در واقع برای مرتب کردن یک آرایه استفاده می‌شود. برای توضیح این الگوریتم، ابتدا نشان دادیم که دو آرایه مرتب شده با مجموع اندازه n را می‌توان در زمان $O(n)$ تبدیل به یک آرایه مرتب شده با اندازه n کرد. بر این اساس الگوریتم مرتب سازی ادغامی را به این صورت تعریف می‌کنیم. $T(n)$ زمان اجرای این الگوریتم به ازای n عنصر است.

۱. آرایه را به دوقسمت مساوی A_1, A_2 تقسیم می‌کنیم. ($O(1)$)

۲. هر کدام از A_1, A_2 را به صورت بازگشتی مرتب می‌کنیم. ($2T(\frac{n}{2})$)

۳. دو آرایه تقسیم شده را با یکدیگر ادغام می‌کنیم. ($O(n)$)

زمان اجرا به صورت $T(n) = 2T(\frac{n}{2}) + O(n), T(1) = O(1)$ خواهد بود. بنا به تعریف خواهیم داشت: $T(1) \leq c_1$ و $T(n) \leq 2T(\frac{n}{2}) + c_2 n$. از استقرا استفاده کرده و نشان می‌دهیم $T(n) \leqslant c_3 n \log(n)$ که در آن $c_3 = c_1 + c_2$:

$$T(2) \leqslant 2T(1) + c_2(2) \leqslant 2(c_1) + 2(c_2) = 2(c_1 + c_2) \times \log 2 = 2c_3 \log 2 \Rightarrow base\ case \ \checkmark$$

$$T(n) \leqslant 2T(\tfrac{n}{2}) + c_2 n \leqslant 2c_3 \tfrac{n}{2} \log \tfrac{n}{2} + c_2 n \leqslant (\log(n) - 1)c_3 n + c_2 n \leqslant c_3 n \log(n) - c_3 n + c_2 n \xrightarrow{c_3 > c_2} \leqslant c_3 n \log(n) \Rightarrow T(n) = O(n \log(n)) \ \checkmark$$

پرسش: فرض کنید تابع $f(n) = n^{0/85}$ و $g(n) = n \log n$ و $h(n) = \frac{n}{\log n}$ داده شده است. به این سوالات پاسخ دهید:

۱. آیا یک مقدار $\epsilon > 0$ وجود دارد که به ازای آن $f(n) = O(n^{1-\epsilon})$؟

۲. آیا یک مقدار $\epsilon > 0$ وجود دارد که به ازای آن $g(n) = \Omega(n^{1+\epsilon})$؟

۳. آیا یک مقدار $\epsilon > 0$ وجود دارد که به ازای آن $h(n) = O(n^{1-\epsilon})$؟

پاسخ‌های خود را تا قبل از شروع کلاس به این آدرس ارسال کنید.