به نام خدا

مقدمه مولف

یک «ساختمان داده» راهی است برای ذخیره و سازماندهی داده‌ها به منظور تسهیل دسترسی و پیرایش آنها. هیچ ساختمان داده‌ای برای هر منظوری مناسب نیست پس نیاز است که نقاط قوت و ضعف ساختمان داده‌های مختلف را بدانیم.

کتاب حاضر، حاصل سال‌ها تدریس و مطالعه است. این کتاب چند بار ویرایش شده است و تلاش شده که مطالب به روز و مطابق با سرفصل‌های وزارت علوم باشد. این کتاب برای آمادگی کنکور کارشناسی ارشد مهندسی کامپیوتر، مهندسی فناوری اطلاعات و علوم کامپیوتر تألیف شده است و همچنین می‌توان از این کتاب به عنوان منبع درس ساختمان داده‌ها در دانشگاه استفاده کرد. این کتاب در ۸ فصل تنظیم شده است که توصیه می‌شود برای درک بهتر و عمیق‌تر فصل‌های ۱ تا ۷ کتاب طراحی الگوریتم همین مؤلف نیز مطالعه شود. در انتهای کتاب تست‌های کنکور ۹۱-۹۹ و دکتری ۹۵ به همراه پاسخ تشریحی و تست‌های تکمیلی آمده است. الگوریتم‌هایی که در کتاب آمده است از علامت‌ها و قواعد برخی زبان‌ها مثل پاسکال و C بهره برده‌اند. از آنجایی که در تست‌های کنکور معمولاً از شبه کد استفاده می‌شود که در آنها علامت‌های مختلفی بکار می‌رود، در این کتاب نیز چنین است. مثلاً در برخی الگوریتم‌ها علامت = به معنی انتساب و علامت == به معنی مقایسه است. و یا در برخی الگوریتم‌ها علامت := یا ← به معنی انتساب و علامت = به معنی مقایسه است.

این کتاب مطالب را به طور کامل پوشش داده است و اگر با دقت مطالعه شود، نیاز به منبع دیگری نیست. از دانشجویان گرامی تقاضا می‌کنم اگر پیشنهاد یا انتقادی در جهت بهبود کتاب دارند با اینجانب در میان گذارند.

دوستان عزیزی در آماده‌سازی کتاب تلاش کرده‌اند که از همه این عزیزان سپاسگزارم.

هادی یوسفی

پاییز ۱۳۹۹

Hadi_yusefi@yahoo.com

۰۹۱۲-۱۷۸۸۵۳۴

@yuseficlass

فهرست مطالب

فصل ۱ الگوریتم

الگوریتم

الگوریتم دنباله‌ای از دستورات خوش تعریف است که هیچ یا چند پارامتر ورودی دارد و حداقل یک خروجی دارد. برای حل یک مسئله ممکن است الگوریتم‌های مختلفی وجود داشته باشد که هر الگوریتم ویژگی منحصر به فردی دارد. در هر الگوریتم دو عامل حافظه مصرفی و زمان اجرا اهمیت دارد که زمان اجرا بسیار اهمیت بیشتری دارد.

در این بخش نحوه تحلیل زمان اجرای الگوریتم‌ها را بررسی می‌کنیم. برای این منظور نیازمند دانستن برخی خواص و قوانین ریاضی هستیم که آنها را یادآوری می‌کنیم.

قواعد ریاضی

هنگام محاسبه تعداد اجرای حلقه‌ها به دنباله‌ها برمی‌خوریم که برخی از آنها را یادآوری می‌کنیم.

۱) $\sum_{i=۱}^{n} i = ۱ + ۲ + ... + n = \frac{n(n+۱)}{۲} \sim \frac{۱}{۲} n^۲ = \theta(n^۲)$

(علامت $\sim$ به معنی هم ارزی است. علامت θ را تعریف خواهیم کرد. به عنوان مثال برای محاسبه حد $\lim_{n\to\infty} \frac{۱+۲+...+n}{۳n^۲+n+۱} = \frac{۱}{۶}$ از هم‌ارزی $\frac{۱}{۲} n^۲$ استفاده کردیم)

۲) $\sum_{i=۱}^{n} i^۲ = ۱^۲ + ۲^۲ + ... + n^۲ = \frac{n(n+۱)(۲n+۱)}{۶} \sim \frac{۱}{۳} n^۳ = \theta(n^۳)$

۳) $\sum_{i=۱}^{n} i^۳ = ۱^۳ + ۲^۳ + ... + n^۳ = (\frac{n(n+۱)}{۲})^۲ \sim \frac{۱}{۴} n^۴ = \theta(n^۴)$

۴) $\sum_{i=۱}^{n} i^k = ۱^k + ۲^k + ... + n^k \sim \frac{۱}{k+۱} n^{k+۱} = \theta(n^{k+۱})$ (۰ < k عدد ثابت)

۵) $\sum_{i=۱}^{n} \frac{۱}{i} = ۱ + \frac{۱}{۲} + \frac{۱}{۳} + ... + \frac{۱}{n} \sim Lnn = \theta(Lgn)$

(سری همسازه یا هارمونیک است که با Lnn تقریباً مساوی است. ثابت خواهیم کرد که پایه لگاریتم‌ها اگر ثابت باشد در رشد تابع تأثیری ندارد و به همین خاطر بجای Lnn نوشته‌ایم Lgn. در ضمن $Logn$ یا Lgn یعنی $Log_۲ n$)

۶) $\sum_{k=۰}^{n-۱} ax^k = a + ax + ax^۲ + ... + ax^{n-۱} = a\frac{۱-x^n}{۱-x} = a\frac{x^n - ۱}{x-۱}$

(سری هندسی با قدر نسبت x که اگر $n \to \infty$ و $| x | < ۱$ باشد داریم:

$$\sum_{k=۰}^{\infty} ax^k = a + ax + ax^۲ + ... = \frac{a}{۱-x}$$

۷) $\sum_{k=۰}^{n-۱} (a+kd) = a + (a+d) + (a+۲d) + ... + (a+(n-۱)d)$

$$= \frac{\text{تعداد جملات} \times (\text{جمله آخر} + \text{جمله اول})}{۲} = \frac{(a + a + (n-۱)d)n}{۲}$$

(تصاعد حسابی با قدر نسبت d است)

۸) $\sum_{i=m}^{n} a = a + a + ... + a = (n - m + ۱)a$

(a به i وابسته نیست)

۹) $\sum_{k=۰}^{\infty} kx^k = \frac{x}{(۱-x)^۲}$, $| x | < ۱$ (با مشتق‌گیری از سری هندسی اثبات می‌شود)

۱۰) $\sum_{k=۱}^{n} (a_k - a_{k-۱}) = a_n - a_۰$

($\sum_{k=۱}^{n-۱} \frac{۱}{k(k+۱)} = \sum_{k=۱}^{n-۱} (\frac{۱}{k} - \frac{۱}{k+۱}) = ۱ - \frac{۱}{n}$ (سری تلسکوپی گویند به عنوان مثال

لگـاریتم در ایـن درس اهمیـت زیـادی دارد. لگـاریتم معکـوس تـابع تـوان اسـت یعنـی اگـر $f(x) = a^x$ آنگاه $f^{-1}(x) = Log_a^x$. به عبـارتی اگـر $a^b = c$ آنگـاه $Log_a^c = b$ و بـرعکس. برخی خواص لگاریتم عبارت است از:

۱) $Log_c^{(ab)} = Log_c^a + Log_c^b$

۲) $Log_c^{(\frac{a}{b})} = Log_c^a - Log_c^b$

۳) $Log_{b^n}^{a^m} = \frac{m}{n} Log_b^a$

۴) $Log_b^a = \frac{Log_c^a}{Log_c^b}$

۵) $Log_b^a = \frac{۱}{Log_a^b}$

۶) $a^{Log_a^b} = b$

۷) $a^{Log_c^b} = b^{Log_c^a}$

رشد توابع

می‌خواهیم بررسی کنیم وقتی n را به ∞ می‌بـریم $f(n) = ۴n^۲$ زودتـر بـه ∞ مـی‌رود یـا $g(n) = n^۳$. درواقع می‌خواهیم بررسی کنـیم بـه ازای nهـای خیلـی بـزرگ کـدام تـابع بزرگتـر می‌شود. برای مقایسه رشد توابع می‌توان از حد کمک گرفت.

رشد $۴n^۲$ از $n^۳$ کمتر است. $\lim_{n\to\infty} \frac{۴n^۲}{n^۳} = \lim_{n\to\infty} \frac{۴}{n} = ۰ \Rightarrow$

به طور کلی حد $\frac{f(n)}{g(n)}$ وقتی $n \to \infty$ سه حالت دارد:

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \begin{cases} ۰ \leftrightarrow & \text{رشد } f(n) < \text{رشد } g(n) \\ \infty \leftrightarrow & \text{رشد } f(n) > \text{رشد } g(n) \\ ۰ < k < \infty \leftrightarrow & \text{رشد } f(n) = \text{رشد } g(n) \end{cases}$$

لازم به ذکر است که توابع مورد بررسی مثبت هستند $(g(n)\ ,\ f(n) > ۰)$ ولی می‌توانند نزولی[۱] باشند که در این صورت گوییم رشد منفی دارند. مثلاً برای مقایسه رشد توابع نزولی $f(n) = \frac{۱}{n}$ و $g(n) = \frac{۱}{n^۲}$ داریم:

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{\frac{۱}{n}}{\frac{۱}{n^۲}} = \lim_{n\to\infty}(n) = \infty$$

پس رشد $\frac{۱}{n}$ از $\frac{۱}{n^۲}$ بیشتر است.

با کمک حد می‌توان به نتایج زیر رسید:

۱) ضریب ثابت تأثیری در رشد ندارد مثلاً رشد $۴n^۲$ با $n^۲$ مساوی است.

۲) رشد یک چند جمله‌ای با رشد جمله با بیشترین درجه‌اش مساوی است یعنی:

$$a_p n^p + a_{p-۱} n^{p-۱} + ... + a_۱ n + a_۰ \sim a_p n^p = \theta(n^p)$$

۳) پایه ثابت لگاریتم تأثیری در رشد ندارد یعنی $Log_a n$ و Log_b^n هم رشد هستند ($a,b > ۱$ ثابت)

۴) رشد توابع نمایی از چند جمله‌ای بیشتر است یعنی رشد a^n از n^b بیشتر است (a,b ثابت و $a > ۱$)

۵) رشد a^n از $Log^b n = (Logn)^b$ همیشه بیشتر است (a,b ثابت و $a > ۰$)

۶) رشد $(Logn)^a$ از $Log^b Logn = (LogLogn)^b$ همیشه بیشتر است (a,b ثابت و $a > ۰$)

۷) رشد n^n از رشد $n!$ بیشتر است.

۸) رشد $Lg(n!)$ با رشد $n.Lgn$ مساوی است.

(اثبات این گزاره کمی مشکل است. یک راه اثبات، استفاده از تقریب استرلینگ است که می‌گوید $n! = \sqrt{۲\pi n}(\frac{n}{e})^n(۱ + \theta(\frac{۱}{n}))$ از طرفین Lg بگیرید)

بنابراین ترتیب رشد زیر قابل اثبات است:

$$\frac{۱}{n^۲} < \frac{۱}{n} < ۱ < LgLgn < \sqrt{Lgn} < Lgn < \sqrt{n} < n < nLgn$$

---

۱ - تابع f را صعودی گویند اگر $m < n$ آنگاه $f(m) \le f(n)$. تابع f را نزولی گویند هرگاه $m < n$ آنگاه $f(m) \ge f(n)$. تابع f را صعودی اکید گویند هرگاه $m < n$ آنگاه $f(m) < f(n)$. تابع f را نزولی اکید گویند هرگاه $m < n$ آنگاه $f(m) > f(n)$.

$< n^۲ < n^۳ < ... < ۲^n < ۳^n ... < n! < n^n$

رشد ۱ یعنی رشد ثابت. یعنی اصلاً زیاد نمی‌شود و به n وابسته نیست. مثلاً حلقه‌ای که همیشه ۱۰۰۰ بار اجرا می‌شود، رشدش از مرتبه $\theta(۱)$ است.

نکته: اگر رشد $Lg(f(n))$ از رشد $Lg(g(n))$ کمتر باشد آنگاه حتماً رشد $f(n)$ از $g(n)$ کمتر است. مثلاً می‌خواهیم $f(n) = (Lgn)^n$ را با $g(n) = n^{\lg n}$ مقایسه کنیم $Lg(f(n)) = n \cdot LgLgn$ و $Lg(g(n)) = Lg^۲ n$. واضح است که رشد $Lg(g(n))$ از $Lg(f(n))$ کمتر است پس رشد $g(n) = n^{Lgn}$ از $f(n) = (Lgn)^n$ کمتر است.

نکته: اگر رشد $Lg(f(n))$ از رشد Lgn کمتر یا مساوی باشد آنگاه گویند $f(n)$ محدود به چند جمله‌ای است ($f(n) \le n^c$ که c ثابت است، یعنی رشد $f(n)$ از رشد چند جمله‌ای‌ها بیشتر نیست)

مثال: آیا $\lceil Lgn \rceil !$ (یا $\lfloor Lgn \rfloor !$) محدود به چند جمله‌ای است؟

پاسخ: لگاریتم $\lceil Lgn \rceil !$ را گرفته و بررسی می‌کنیم.

$$Lg(\lceil Lgn \rceil !) \sim \lceil Lgn \rceil \cdot Lg \lceil Lgn \rceil \sim Lgn \cdot Lg(Lgn) > Lgn$$

پس $\lceil Lgn \rceil !$ محدود به چند جمله‌ای نیست. (دقت کنید با توجه به اینکه $Lgn!$ با $n.Lgn$ هم‌ارز است پس $Lg \lceil Lgn \rceil !$ با $\lceil Lgn \rceil \cdot Lg \lceil Lgn \rceil$ هم‌ارز است)

مثال: آیا $\lceil LgLgn \rceil !$ محدود به چند جمله‌ای است؟

پاسخ: بررسی کنید لگاریتم این تابع رشد کمتری از Lgn دارد پس محدود به چند جمله‌ای است.

نمادهای مجانبی *(asymptotic notations)*

برای بیان زمان اجرای الگوریتم‌ها و رشد توابع از نمادهای O و Ω و θ و o و w استفاده می‌شود.

نماد $O(big \cdot oh)$: گویند تابع $f(n)$ از مرتبه $O(g(n))$ است و می‌نویسیم $f(n) \in O(g(n))$ یا $f(n) = O(g(n))$ هرگاه رشد $f(n)$ از رشد $g(n)$ کمتر یا مساوی باشد. مثلاً $۳n^۲ + n + ۴ \in O(n^۲)$ یا $۳n^۲ + n + ۴ \in O(n^۳)$ ولی $۳n^۲ + n + ۴ \notin O(n)$.

تعریف دقیق ریاضی نماد O عبارت است از:

$$O(g(n)) = \{f(n) \mid \exists c, n_۰ > ۰ \ , \ \forall n \ge n_۰ : f(n) \le c \cdot g(n)\}$$

یعنی $O(g(n))$ شامل همه توابعی است که رشدشان از $g(n)$ کمتر یا مساوی است. مثلاً $O(n^۲)$ شامل توابعی است که رشدشان از $n^۲$ کمتر یا مساوی است مثل $۳n^۲ + ۱$، $۵n$، $Lg^۴n$ و $\sqrt{n}$ و $LgLgn$ و ...

توجه: اگر $f(n)$ از مرتبه $O(g(n))$ باشد بهتر است بنویسیم $f(n) \in O(g(n))$. زیرا $O(g(n))$ یک مجموعه است و نوشتن $f(n) = O(g(n))$ درست نیست ولی به دلیل استفاده خیلی زیاد، این اشتباه نادیده گرفته شده است و در این کتاب نیز استفاده می‌شود.

نماد Ω (اُمگای بزرگ): گویند تابع $f(n)$ از مرتبه $\Omega(g(n))$ است و می‌نویسیم $f(n) \in \Omega(g(n))$ یا $f(n) = \Omega(g(n))$ هرگاه رشد $f(n)$ از رشد $g(n)$ بیشتر یا مساوی باشد. مثلاً $۳n^۲ + n + ۴ \in \Omega(n^۲)$ یا $۳n^۲ + n + ۵ \in \Omega(n)$ ولی $۳n^۲ + n + ۴ \notin \Omega(n^۳)$.

تعریف ریاضی دقیق Ω عبارت است از:

$$\Omega(g(n)) = \{f(n) \mid \exists c, n_۰ > ۰ \ , \ \forall n \geq n_۰ : f(n) \geq c \cdot g(n)\}$$

یعنی $\Omega(g(n))$ شامل همه توابعی است که رشدشان از $g(n)$ بیشتر یا مساوی است. مثلاً $\Omega(n^۲)$ شامل توابعی است که رشدشان از $n^۲$ بیشتر یا مساوی است مثل $n^۲$، $۳n^۲ + n$، $n^۳ - n$، ۲^n و $n!$ و ...

نماد θ (تتا): گویند تابع $f(n)$ از مرتبه $\theta(g(n))$ است و می‌نویسیم $f(n) \in \theta(g(n))$ یا $f(n) = \theta(g(n))$ هرگاه $f(n)$ و $g(n)$ هم رشد باشند. مثلاً $۳n^۲ + n + ۴ \in \theta(n^۲)$ و $۳n^۲ + n + ۴ \notin \theta(n)$ و $۳n^۲ + n + ۴ \notin \theta(n^۳)$

تعریف ریاضی دقیق θ عبارت است از:

$$\theta(g(n)) = \{f(n) \mid \exists c_۱, c_۲, n_۰ > ۰ \ , \ \forall n \geq n_۰ : c_۱ \cdot g(n) \leq f(n) \leq c_۲ \cdot g(n)\}$$

به راحتی می‌توان دریافت که اشتراک $O(g(n))$ با $\Omega(g(n))$ برابر $\theta(g(n))$ است.

نماد o (little.oh): گویند تابع $f(n)$ از مرتبه $o(g(n))$ است و می‌نویسیم $f(n) \in o(g(n))$ یا $f(n) = o(g(n))$ هرگاه رشد $f(n)$ از رشد $g(n)$ کمتر باشد مثلاً $۳n^۲ + n + ۴ \in o(n^۳)$ ولی $۳n^۲ + n + ۴ \notin o(n^۲)$.

تعریف ریاضی دقیق نماد o عبارت است از:

$$o(g(n)) = \{f(n) \mid \exists n_۰ > ۰ \ , \ \forall c > ۰ \ , \ \forall n \geq n_۰ : f(n) < c.g(n)\}$$

توجه کنید تفاوت این تعریف با تعریف $(big)O$ این است که اینجا $\forall c > ۰$ ولی در تعریف $(big)O$ داشتیم $\exists c > ۰$. به هر حال این تعریف می‌گوید، $o(g(n))$ شامل توابعی است که رشدشان از $g(n)$ کمتر است. مثلاً $o(n^۲)$ شامل توابعی مثل $۳n + ۴$ و Lgn و $Lg^۵n$ و $\sqrt{n}$ و $\frac{۱}{n}$ و ... است.

نماد ω ***(اُمگای کوچک):*** گویند تابع $f(n)$ از مرتبه $\omega(g(n))$ است و می‌نویسیم $f(n) \in \omega(g(n))$ یا $f(n) = \omega(g(n))$ هرگاه رشد $f(n)$ از رشد $g(n)$ بیشتر باشد. مثلاً $۳n^۲ + n + ۴ \in \omega(n)$ ولی $۳n^۲ + n + ۴ \notin \omega(n^۲)$.

تعریف ریاضی دقیق نماد ω عبارت است از:

$$\omega(g(n)) = \{f(n) \big| \exists n_۰ > ۰ \ , \ \forall c > ۰ \ , \ \forall n \geq n_۰ : f(n) > c \cdot g(n)\}$$

یعنی $\omega(g(n))$ شامل توابعی است که رشدشان از $g(n)$ بیشتر است. مثلاً $\omega(n^۲)$ شامل توابعی مثل $۳n^۳ + ۴$ و ۲^n و $n!$ و n^n و ... است.

واضح است که $\omega(g(n))$ با $o(g(n))$ هیچ اشتراکی ندارد.

خلاصه تعاریف نمادها را می‌توان به شکل زیر بیان کرد:

$f(n) \in O(g(n)) \leftrightarrow f$ رشد $\leq g$ رشد

$f(n) \in \Omega(g(n)) \leftrightarrow f$ رشد $\geq g$ رشد

$f(n) \in \theta(g(n)) \leftrightarrow f$ رشد $= g$ رشد

$f(n) \in o(g(n)) \leftrightarrow f$ رشد $< g$ رشد

$f(n) \in \omega(g(n)) \leftrightarrow f$ رشد $> g$ رشد

همچنین می‌توان با توجه به تعریف نمادها و مفهوم حدگیری، روابط زیر را ارائه داد:

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = \begin{cases} ۰ \leftrightarrow f(n) \in o(g(n)) \rightarrow f(n) \in O(g(n)) \\ \infty \leftrightarrow f(n) \in \omega(g(n)) \rightarrow f(n) \in \Omega(g(n)) \\ ۰ < k < \infty \leftrightarrow f(n) \in \theta(g(n)) \end{cases}$$

دقت کنید اگر $f(n) \in o(g(n))$ می‌توان نتیجه گرفت $f(n) \in O(g(n))$ ولی عکس این مطلب لزوماً صحیح نیست.

خواص زیر برای نمادها به راحتی قابل بررسی هستند:

a) بازتابی *(reflexive)*: نمادهای O و Ω و θ بازتاب هستند یعنی

$$f(n) \in O(f(n)) \ , \ f(n) \in \Omega(f(n)) \ , \ f(n) \in \theta(f(n))$$

b) تقارنی $(symmetric)$: نماد θ متقارن است یعنی:

$$f(n) \in \theta(g(n)) \leftrightarrow g(n) \in \theta(f(n))$$

c) تقارنی ترانهاده $(transpose\ \ symmetric)$: نمادهای O و Ω و همچنین o و ω دارای خاصیت تقارنی ترانهاده هستند یعنی:

$$f(n) \in O(g(n)) \leftrightarrow g(n) \in \Omega(f(n))$$
$$f(n) \in o(g(n)) \leftrightarrow g(n) \in \omega(f(n))$$

d) تعدی $(transitive)$: همه نهادها متعدی هستند یعنی:

$$f(n) \in O(g(n)) \ \ , \ \ g(n) \in O(h(n)) \rightarrow f(n) \in O(h(n))$$

بجای نماد O می‌توان نمادهای دیگر را قرار داد.

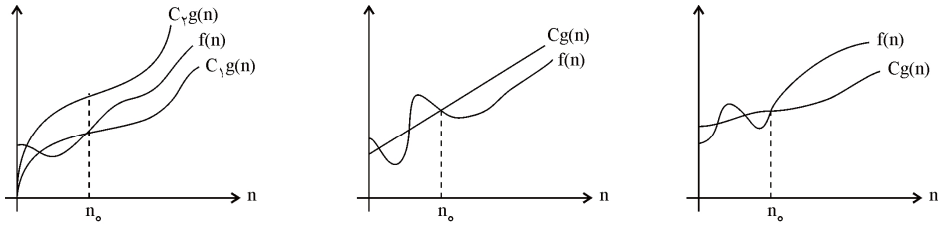
e) اگر $f(n) = O(g(n))$ و $f(n) = \Omega(g(n))$ آنگاه $f(n) = \theta(g(n))$ و برعکس.

مثال: آیا $O(f(n)) - \Omega(f(n))$ و $o(f(n))$ دو مجموعه مساوی هستند؟

پاسخ: هرچند ظاهراً باید مساوی باشند ولی نیستند. مثلاً تابع $g(n) = \begin{cases} n & ,\text{زوج } n \\ ۱ & ,\text{فرد } n \end{cases}$ را در نظر بگیرید. واضح است که $g(n) \in O(n)$ زیرا $g(n)$ کمتر مساوی n است. $g(n) \notin \Omega(n)$ زیرا به ازای nهای فرد، $g(n)$ برابر ۱ است که از n بیشتر مساوی نیست. پس $g(n) \in O(n) - \Omega(n)$. از طرفی $g(n) \notin o(n)$ زیرا به ازای nهای زوج، $g(n)$ برابر n است که از n کمتر نیست. پس نشان دادیم توابعی وجود دارد که عضو $O(n) - \Omega(n)$ هستند ولی عضو $o(n)$ نیستند. می‌توان نشان داد همیشه:

$$o(f(n)) \subseteq O(f(n)) - \Omega(f(n))$$

توجه: نمودارهای زیر، نمادهای O و Ω و θ را نشان می‌دهند.



f(n)=θ(g(n)) f(n)=O(g(n)) f(n)=Ω(g(n))

محاسبه زمان اجرا

زمان اجرای یک الگوریتم به عوامل مختلفی بستگی دارد مثل نوع سخت‌افزار، نوع کامپایلر، اندازه ورودی و ... ما می‌خواهیم زمان اجرا را طوری بیان کنیم که مستقل از سخت‌افزار و نرم‌افزار باشد. زمان اجرا را برابر تعداد اجرای همه دستورات تعریف می‌کنیم.

مثال: زمان اجرا (تعداد اجرای همه دستورات، گام شماری) تکه برنامه‌های زیر را بدست می‌آوریم:

۱) $for(i = ۱ \quad to \quad n)$ // بار $n+۱$

$write("* ");$ // بار n

زمان اجرا $= ۲n + ۱ = \theta(n)$

حلقه for و حلقه $while$ خودشان از جمله داخلشان یک بار بیشتر اجرا می‌شوند البته به شرطی که دستوری مثل $break$ استفاده نشود.

۲) $for(i = a \quad to \quad b)$ // بار $b - a + ۲$

$write("* ");$ // بار $b - a + ۱$

زمان اجرا $= ۲(b-a) + ۳ = \theta(b-a)$

۳) $for(i = ۱ \quad to \quad n)$ // بار $n+۱$

$for(j = ۱ \quad to \quad n)$ // بار $n(n+۱)$

$write("* ")$; // بار $n^۲$

زمان اجرا $= ۲n^۲ + ۲n + ۱ = \theta(n^۲)$

۴) $for(i = ۱ \quad to \quad n)$ // $n+۱$

$for(j = ۱ \quad to \quad n)$ // $n(n+۱)$

$for(k = ۱ \quad to \quad n)$ // $n^۲(n+۱)$

$write("* ")$; // $n^۳$

زمان اجرا $= ۲n^۳ + ۲n^۲ + ۲n + ۱ = \theta(n^۳)$

لازم به ذکر است که اگر زمان اجرا به طور کامل پرسش نشود و فقط مرتبه زمان اجرا را برحسب نماد θ بخواهیم، لازم نیست تعداد اجرای همه خطوط را بیابیم بلکه کافی است فقط تعداد اجرای دستور اصلی (دستور $write$ در مثال‌های فوق) را بیابیم.

مثال: در تکه برنامه‌های زیر تعداد اجرای دستور $write$ را محاسبه می‌کنیم:

۱)
```
for(i = ۱  to  n)
    for(j = ۱  to  i)
        write("* ");
```

حلقه دوم به حلقه اول وابسته است. به ازای $i = ۱$ دستور $write$ یک بار اجرا می‌شود. به ازای $i = ۲$ دو بار و به همین ترتیب به ازای $i = n$ دستور $write$ به تعداد n بار اجرا می‌شود پس تعداد اجرای دستور $write$ برابر $۱ + ۲ + ... + n = \frac{n(n+۱)}{۲}$ است.

برای محاسبه تعداد اجرای دستور $write$ می‌توان به ازای هر حلقه یک $\sum$ نوشت و به شکل زیر نیز محاسبه کرد:

$$\sum_{i=۱}^{n}\sum_{j=۱}^{i} ۱ = \sum_{i=۱}^{n} i = \frac{n(n+۱)}{۲} = \theta(n^۲)$$

۲)
```
for(i = ۱  to  n)
    for(j = ۱  to  i)
    for(k = ۱  to  j)
        write("* ");
```

$$\text{تعداد اجرای } write = \sum_{i=۱}^{n}\sum_{j=۱}^{i}\sum_{k=۱}^{j} ۱ = \sum_{i=۱}^{n}\sum_{j=۱}^{i} j = \sum_{i=۱}^{n} \frac{i(i+۱)}{۲} = \frac{۱}{۲}\left[\sum_{i=۱}^{n} i^۲ + \sum_{i=۱}^{n} i\right]$$

$$= \frac{۱}{۲}\left[\frac{n(n+۱)(۲n+۱)}{۶} + \frac{n(n+۱)}{۲}\right] = \frac{۱}{۲} \times \frac{n(n+۱)(۲n+۴)}{۶}$$

$$= \frac{n(n+۱)(n+۲)}{۶} = \theta(n^۳)$$

۳)
```
i = n;
while(i > ۱)
{
    write("* "); // بار اجرا می‌شود ⌊n/۲⌋ = θ(n)
    i = i − ۲;
}
```

از i هر بار ۲ واحد کم می‌شود. مثلاً اگر $n = ۱۰$ باشد دستور $write$ به ازای $i = ۱۰,۸,۶,۴,۲$ اجرا می‌شود. یا اگر $n = ۱۱$ باشد دستور $write$ به ازای $i = ۱۱,۹,۷,۵,۳$ اجرا می‌شود. پس دستور $write$ همیشه $\left\lfloor \frac{n}{۲} \right\rfloor$ بار اجرا می‌شود.

توجه: علامت $\lfloor\ \rfloor$ کف *(floor)* است و علامت $\lceil\ \rceil$ سقف *(ceiling)* است. $\lfloor x \rfloor$ برابر بزرگترین عدد صحیح کوچکتر یا مساوی x است و $\lceil x \rceil$ برابر کوچکترین عدد صحیح بزرگتر یا مساوی x است.

۴)
$i = n;$
$while(i > ۱)$
{
$write("* ");$ // بار $\lfloor Lgn \rfloor = \theta(Lgn)$
$i = \left\lfloor \frac{i}{۲} \right\rfloor;$
}

دستور $write$ به ازای $i = \frac{n}{۲^۰}, \frac{n}{۲^۱}, \frac{n}{۲^۲}, \ldots, \frac{n}{۲^{k-۱}}$ اجرا می‌شود (یعنی k بار) حال اگر $\frac{n}{۲^k} = ۱$ $(k = \lg n)$ حلقه پایان می‌یابد. و اگر n توانی از ۲ نباشد به راحتی می‌توان بررسی کرد که علامت $\lfloor\ \rfloor$ لازم است.

۵)
$i = ۱;$
$while(i < n)$
{
$write("* ");$ // بار $\lfloor Lgn \rfloor = \theta(Lgn)$
$i = i * ۲;$
}

همانند قبلی می‌توان بررسی کرد. و یا می‌توان به ازای $n = ۸$ و $n = ۷$ بررسی کرد که ۳ بار اجرا می‌شود و نتیجه گرفت که علامت $\lfloor\ \rfloor$ لازم است.

۶)
```
i = ۲;
while(i < n)
{
write("* "); // بار ⌈LgLgn⌉ = θ(LgLgn)
i = i * i;
}
```

دستور $write$ به ازای $i = ۲^{۲^۰}, ۲^{۲^۱}, ۲^{۲^۲}, ۲^{۲^۳}, \ldots, ۲^{۲^{k-۱}}$ یعنی k بار اجرا می‌شود و وقتی $i = ۲^{۲^k} = n$ حلقه پایان می‌یابد پس: $۲^{۲^k} = n \rightarrow ۲^k = Lgn \rightarrow k = LgLgn$

می‌توان بررسی کرد اگر $LgLgn$ عدد صحیح نباشد علامت $\lceil\ \rceil$ لازم است.

در برخی الگوریتم‌ها زمان اجرا به طور دقیق قابل محاسبه نیست بلکه در حالت‌های مختلف، متفاوت است. سه حالت در تحلیل الگوریتم‌ها مطرح می‌شود. بهترین $(Best)$، متوسط $(Average)$ و بدترین $(worst)$ حالت.

به عنوان مثال فرض کنید می‌خواهیم مقدار x را در آرایه $c[۱..n]$ جستجوی خطی کنیم. عمل اصلی در جستجو، مقایسه است. اگر x در $c[۱]$ باشد بهترین حالت است و ۱ مقایسه می‌خواهد $(B(n) = ۱)$. اگر x در $c[n]$ باشد و یا اصلاً نباشد بدترین حالت است و n مقایسه انجام می‌شود $(\omega(n) = n)$. بررسی حالت متوسط معمولاً دشوار است و باید تمام حالات را در نظر گرفت و سپس میانگین گرفت. فعلاً فرض کنید x حتماً در آرایه هست در این صورت احتمال وجود x در خانه $c[i]$ $(i = ۱,۲,\ldots,n)$ برابر $\frac{۱}{n}$ است و زمان اجرای حالت متوسط برابر است با:

$$A(n) = \frac{۱}{n}\sum_{i=۱}^{n} i = \frac{۱}{n}\frac{n(n+۱)}{۲} = \frac{n+۱}{۲}$$

دقت کنید اگر x در $c[۱]$ باشد ۱ مقایسه، در $c[۲]$ باشد ۲ مقایسه و ...

حال فرض کنید x با احتمال p در آرایه هست پس با احتمال $\frac{p}{n}$ در $c[i]$ است و با احتمال $p-۱$ در آرایه نیست در این صورت زمان اجرای حالت متوسط برابر است با:

$$A(n) = \frac{p}{n}\times ۱ + \frac{p}{n}\times ۲ + \ldots + \frac{p}{n}\times n + (۱-p)\times n = \frac{p(n+۱)}{۲} + (۱-p)n$$

مثال: فرض کنید با احتمال ٪۵۰ عنصر x در $c[۱..n]$ هست. به طور متوسط چه تعداد عنصر آرایه برای یافتن x بررسی می‌شوند؟

پاسخ: $p = \frac{۱}{۲}$ پس $A(n) = \frac{n+۱}{۴} + \frac{n}{۲} = \frac{۳n+۱}{۴}$ یعنی به طور متوسط $\frac{۳}{۴}$ عناصر آرایه بررسی می‌شوند.

مسائل:

۱- تابع $Lg^* n$ برابر است با تعداد باری که از n لگاریتم (پایه ۲) بگیریم که حاصل ۱ یا کمتر از ۱ شود. مثلاً $Lg^* ۱۶ = ۳ (LgLgLg ۱۶ = ۱)$ و یا $Lg^* ۶۵۵۳۶ = ۴$ و $Lg^* ۲^{۶۵۵۳۶} = ۵$. تابع $Lg^* n$ رشد بسیار ضعیفی دارد و از تمام توابع صعودی که در این بخش دیدیم رشد کمتری دارد. در هر قسمت بررسی کنید ترتیب داده شده صحیح است.

$a)\quad n^{\frac{۱}{Lgn}} < Lg^۲ n < Lg(n!) < n^۳ < (\frac{۳}{۲})^n < ۲^{۲^n}$

(ابتدا بررسی کنید $n^{\frac{۱}{Lgn}} = ۲$ ثابت است)

$b)\quad ۲^{Lg^* n} < (\sqrt{۲})^{Lgn} < n^۲ < n! < ۲^{۲^{n+۱}}$

(بررسی کنید $\sqrt{۲}^{Lgn} = \sqrt{n}$)

$c)\quad Lg^* n < LgLgn < \sqrt{Lgn} < Lgn < n^{LgLgn} < ۲^n < n \cdot ۲^n$

(بررسی کنید $n^{LgLgn} = (Lgn)^{Lgn}$)

$d)\quad \sqrt{Lgn} < ۲^{Lgn} < ۴^{Lgn} < e^n < (n+۲)!$

$e)\quad Lg^*(n^n) < ۲^{\sqrt{۲Lgn}} < n < nLgn < n^{۲+Lgn} < (\frac{۳}{۲})^n$

$f)\quad ۴^{(Lgn+LgLgn)} = n^۲ Lg^۲ n < n^۲ Lg^۳ \sqrt{n}$

۲- درستی یا نادرستی عبارات زیر را مشخص کنید.

$a)\ f(n) + g(n) = \theta(\max\{f(n), g(n)\})$

$b)$ اگر $f(n) = \Omega(n^۲)$ و $g(n) = \Omega(n)$، در این صورت $f(g(n)) = \Omega(n^۳)$

$c)\ f(g(n)) = \theta(g(f(n)))$

$d)\ f(n) = \theta(f(\frac{n}{۲}))$

$e)$ اگر $T_۱(n) = O(f)$ و $T_۲(n) = O(f)$ آنگاه

$i)\ T_۱(n) + T_۲(n) = O(f)$

$ii)\ T_۱(n) * T_۲(n) = O(f)$

$iii)\ T_۱(n) = O(T_۲(n))$

iv) $\frac{T_۱(n)}{T_۲(n)} = O(۱)$

f) اگر $f(n) = \Omega(g(n))$ و $g(n) = O(f(n))$ آنگاه $f(n) = \theta(g(n))$

پاسخ: a) درست است.

b) نادرست. مثلاً $f(n) = n^۲$ و $g(n) = n$ آنگاه $f(g(n)) = n^۲$ که $\Omega(n^۳)$ نیست.

c) نادرست. مثلاً $f(n) = Lgn$ و $g(n) = n^۲$ آنگاه $f(g(n)) = ۲Lgn$ و $g(f(n)) = (Lgn)^۲$

d) نادرست. مثلاً $f(n) = ۴^n$ آنگاه $f(\frac{n}{۲}) = ۲^n$

e) فقط قسمت i درست است. برای نقض قسمت ii فرض کنید $T_۱(n) = T_۲(n) = f = n$ آنگاه $T_۱(n) * T_۲(n) = n^۲ \neq O(n)$. برای نقض قسمت iii و iv فرض کنید $f = n^۳$ و $T_۱(n) = n^۲$ و $T_۲(n) = n$

f) نادرست. $f(n) = \Omega(g(n))$ با $g(n) = O(f(n))$ معادل است و نتیجه نمی‌شود $f(n) = \theta(g(n))$

۳- آیا می‌توان ادعا کرد که برای هر دو تابع $f(n)$ و $g(n)$ یا $f(n) = O(g(n))$ یا $g(n) = O(f(n))$ یا هر دو؟

پاسخ: خیر. فرض کنید $f(n) = n$ و $g(n) = n^{۱+\sin n}$ بررسی کنید $f(n) \neq O(g(n))$ و $g(n) \neq O(f(n))$

۴- پیچیدگی تکه برنامه‌های زیر را بیابید.

```
a)  for(i = ۱ to n)
        for(j = ۱ to i^۲)
            if(j mod i == o)
                for(k = ۱ to j)
                    write("* ");

b)  for(i = ۱ to n)
        if(odd(i))   for(j = i to n)  x = x + ۱;
          else       for(j = ۱ to i)  y = y + ۱;
```

c) $i = ۳;$
$if(n == ۲ \ \ or \ \ \ n == ۳)return \ \ true;$
$if(n \ \ \mathrm{mod}\ ۲ == ۰)return \ \ false;$
$while(i^۲ \leq n)$
$if(n \ \ \mathrm{mod}\, i == ۰)return\ false;$
$\quad else \ \ i = i + ۲;$
$return\, true;$

d) $for(i = ۱ \ ; i \leq n \ ; \ i = i * ۲)$
$\quad for(j = ۱ \ ; \ \ j \leq n \ \ ; \ j + +)$
$\quad\quad write("* ");$

e) $for(i = ۱ \ ; \ i \leq n \ \ ; \ i + +)$
$\quad for(j = ۱ \ ; \ j \leq n \ ; \ j = j * ۲)$
$\quad\quad write("* ");$

پاسخ: a) به ازای هر i ، داخلی‌ترین حلقه i بار اجرا می‌شود. در واقع به ازای $j = i, ۲i, ۳i, ..., i * i$ شرط if درست است و حلقه داخلی اجرا می‌شود پس حلقه داخلی به تعداد $i + ۲i + ... + i * i = i\frac{i(i+۱)}{۲}$ بار اجرا می‌شود که از مرتبه $\theta(i^۳)$ است و چون خود i، n بار تکرار می‌شود، مرتبه کل $\theta(n^۴)$ است.

b) چه i زوج باشد و چه فرد مرتبه $\theta(n^۲)$ می‌شود.

c) با توجه به شرط حلقه $while$ یعنی $i^۲ \leq n$ مرتبه $O(\sqrt{n})$ است.

۵- اگر f_w و f_a به ترتیب درجه‌های پیچیدگی زمان اجرای یک الگوریتم در بدترین حالت و حالت میانگین باشند، کدام عبارت $f_a = \theta(f_w)$ ، $f_a = o(f_w)$، $f_a = O(f_w)$ همواره صحیح است؟

پاسخ: زمان در حالت متوسط کمتر یا مساوی زمان در بدترین حالت است پس همواره $f_a = O(f_w)$

۶- بعضی منابع نماد Ω را با کمی تغییر تعریف می‌کنند. نام این نماد جدید را $\overset{\infty}{\Omega}$ (امگای بی‌نهایت) می‌گذاریم. گوییم $f(n) = \overset{\infty}{\Omega}(g(n))$ هرگاه ثابت مثبت c وجود داشته باشد که برای تعداد بی‌شماری n داشته باشیم $f(n) \geq cg(n)$. نشان دهید برای هر دو تابع مثبت $f(n)$ و $g(n)$ یا $f(n) = O(g(n))$ یا $f(n) = \overset{\infty}{\Omega}(g(n))$ (یا هر دو)

۷- نماد $\tilde{O}(soft-oh)$ همانند O است که فاکتورهای لگاریتمی را چشم‌پوشی می‌کند.

$$\tilde{O}(g(n)) = \{f(n) : \exists c,k,n_{\circ} > \circ\ ,\ \forall n \geq n_{\circ} : f(n) \leq cg(n)Lg^{k}(n)\}$$

بـه همـین ترتیـب $\tilde{\Omega}$ و $\tilde{\theta}$ نیـز تعریـف مـی‌شـوند. نـشان دهیـد $f(n) = \tilde{O}(g(n))$ و $f(n) = \tilde{\Omega}(g(n))$ اگر و فقط اگر $f(n) = \tilde{\theta}(g(n))$.

۸- فرض کنید $f(n)$ تابع صعودی است. f_c^* را تعریف می‌کنیم:

$$f_c^*(n) = \min\{i \geq \circ : f^{(i)}(n) \leq c\}$$

$f^{(i)}(n)$ را نیز به صورت مقابل تعریف می‌کنیم:

$$f^{(i)}(n) = \begin{cases} n & if\ \ i = \circ \\ f(f^{(i-۱)}(n)) & if\ i > \circ \end{cases}$$

مثلاً اگر $f(n) = ۲n$ آنگاه $f^{(i)}(n) = ۲^i n$ است.

با توجه به جدول مقابل، درستی مقادیر ستون $f_c^*(n)$ را بررسی کنید.

به عنوان مثال فرض کنید $f(n) = \frac{n}{۲}$ و $c = ۱$:

$$f^{(۲)}(n) = \frac{n}{۲^۲}\ ,\ \ ...,\ f^{(i)}(n) = \frac{n}{۲^i} \leq ۱$$

در نتیجه کمترین i برابر Lgn می‌شود.

f(n)	c	$f_c^*(n)$
$n-۱$	۰	n
Lgn	۱	Lg^*n
$\frac{n}{۲}$	۱	$\lceil \lg n \rceil$
$\frac{n}{۲}$	۲	$\lceil \lg n \rceil - ۱$
$\sqrt{n}$	۲	$\geq \lg\lg n$
$\sqrt{n}$	۱	∞
$n^{\frac{۱}{۳}}$	۲	$Log_{۳}(Lgn)$
$\frac{n}{Lgn}$	۲	$O(\lg n)$

[سؤال‌های طبقه‌بندی شده فصل اول]

۱- مجموع مراحل خطوط در برنامه زیر چند است؟

```
float sum (int num[ ] , int n)
{
   int i , temp=0 ;
   for (i= 0 ; i<n ; i++)
      temp += num [i] ;
   return temp ;
}
```

۱) $۲n+۳$
۲) $۲n+۱$
۳) تعداد مراحل بستگی به n دارد و نامشخص است.
۴) $n+۱$

۲- در برنامه زیر تعداد دفعات تکرار دستورالعمل ۳ برابر است با... **(علوم کامپیوتر ۸۰)**

۱) $for\,(k=۰;k<=n-۱;k++)$
۲) $for\,(i=۱;i<=n-k;i++)$
۳) $a[i][i+k]=k\ ;$

۱) $n^۲$
۲) $\frac{n(n+۱)}{۲}$
۳) $\frac{n^۲}{۲}$
۴) $\frac{n(n-۱)}{۲}$

۳- مقدار محاسبه شده برای $TOTAL$ را تعیین کنید. **(علوم کامپیوتر ۷۹)**

```
TOTAL:= 0 ;
for i:=1  to  N  do
begin
   K:=N ;
   while (K<>1) do begin
      K:=K div 2 ;
      TOTAL := TOTAL +  ۱;
   end
end ;
```

۱) $N(\log_۲^N-۱)$
۲) $N\log_۲^N$
۳) $N(\log_۲^N+۱)$
۴) $\log_۲^N+۱$

۴- با توجه به تکه برنامه مقابل مقدار $f(n)$ عبارت است از:

```
i = ۱
while (i<=n)
{
j = ۱
while (j<=n)
{
j = j* ۲
}
i = i + ۱
}
```

۱) $\log_{۲}^{n}$

۲) $\log_{۲}^{n}+۱$

۳) $n\left(\frac{n+۱}{۲}\right)$

۴) $n\left(\log_{۲}^{n}+۱\right)$

۵- روال بازگشتی مقابل را در نظر بگیرید مطلوبست (۵ و ۳۰) M

```
M(A , B)
P ← ۰
while A<> 0
   Do {if A mod ۲ = ۱ then P ← P + B
   A ← ⌊A/۲⌋
   B ← ۲B;}
Return P
```

۱) ۲۰

۲) ۴۰

۳) ۷۰

۴) ۱۵۰

۶- اگر M واحد، زمان اجرای $Modul\ A$ شود و N تعداد داده‌های ورودی باشد، تابع پیچیدگی زمانی الگوریتم زیر کدام است؟

```
I:=N
while I > ۱
for j := ۱ to N
   Modul A
end for
I := I / ۵
end while
```

۱) $MNLog_{۵}^{N}$

۲) $MN^{۲}$

۳) $NLog_{۵}^{M}$

۴) $MN^{۲}Log_{۲}^{N}$

۷- زمان مصرفی قطعه برنامه زیر در بهترین و بدترین حالت چه مقداری است و مربوط به چه مواردی می‌باشد؟

```
for i ← ۲ to n do
    if k mod i = ۰ then
        for j ← i to n do
            ℓ ← ℓ* k
```

۱) در بهترین حالت زمان خطی است و این مربوط به ورودی است که k به n بخش پذیر نباشد. و بدترین حالت زمانی است که k به کلیه اعداد ۱ تا n بخش پذیر باشد در اینصورت زمان $۲n$ است.

۲) در بهترین حالت زمان ضریب ثابتی از n است و این حالت مربوط به مواردی است که k مضربی از n باشد و بدترین زمان $n^۲$ است که مربوط به باقی حالات می‌شود.

۳) در هر حال و در کلیه موارد زمان مصرفی $n^۲$ است.

۴) اگر k ضریبی از $n!$ باشد در بدترین حالت قرار می‌گیریم که زمان مصرفی مربوطه $n^۲$ است و بهترین حالت زمانی است که k به هیچکدام از اعداد ۲ تا n بخش پذیر نباشد در این صورت زمان مصرفی خطی است.

۸- مرتبه زمانی شبه کد زیر چیست؟ **(IT ۸۴)**

```
for (i = ۱ ; i <= n ; i++)
{
    for (j = ۱; j <= n; j++)
        x++;
    n-- ;
}
```

۱) n

۲) $n^۲$

۳) $logn$

۴) $n\ logn$

۹- مرتبه زمانی شبه کد زیر چیست؟ **(IT ۸۶)**

```
for (i = ۱; i <= n ; i++)
    for (j = ۱; j <= n; j++)

    {
        x++;
        n-- ;
    }
```

۱) n

۲) $n \hat{} ۲$

۳) $log\ n$

۴) $n\ log\ n$

۱۰- پس از اجرای قطعه کد زیر، مقدار نهایی x چه مقداری خواهد بود؟ **(IT ۸۵)**

```
x=0;
for (i = ۱; i <= n; i++){
    for (j = ۱; j <= n; j++) x++;
    j = ۱;
    while (j<n){
        x++; j = j * ۲;
    }
}
```

۱) $n.\lceil \log_۲ n \rceil$

۲) $n^۲ + \lceil \log_۲ n \rceil$

۳) $n^۲ + n\lceil \log_۲ n \rceil$

۴) $n\left(۱ + \lfloor \log_۲ n \rfloor\right)$

۱۱- کدامیک از مجموعه توابع زیر برحسب افزایش مرتبه ***(Order)*** از چپ به راست مرتب هستند؟ **(علوم کامپیوتر ۷۹)**

۱) $n!$ و $n^{۱۰۰۰}$ و $(۱/۰۰۵)^n$

۲) $n!$ و $(۱/۰۰۵)^n$ و $n^{۱۰۰۰}$

۳) $n^{۱۰۰۰}$ و $n!$ و $(۱/۰۰۵)^n$

۴) $n^{۱۰۰۰}$ و $(۱/۰۰۵)^n$ و $n!$

۱۲- مرتبه بزرگی ***(Order of magnitude)*** قطعه کد زیر چیست؟ **(آزاد ۸۰)**

```
k := ۰ ;
for i := ۱ to n do begin
    for j := ۱ to m do
        k := k + ۱;
    j := ۱ ;
    while j<n do begin
        k := k + ۱;
        j := ۲j;
    end
end ;
```

۱) $O(n^۲)$

۲) $O(nm)$

۳) $O(nm + n^۲)$

۴) $O\ (nm+nlogn)$

۱۳- هزینه اجرای تابع زیر چه اندازه است؟

```
i := n ;
while (i > ۱) do
    begin
        i := i mod ۲;
        write (i) ;
    end;
```

۱) $O\left(\log_۲^n\right)$

۲) $O(n)$

۳) $O\left(n \log_۲^n\right)$

۴) $O(۱)$

۱۴- گزینه درست را انتخاب کنید. **(علوم کامپیوتر ۸۰)**

۱) $f(n)\in O\big(g(n)\big), h(n)\in \Omega\big(g(n)\big) \Rightarrow f(n)\in \theta\big(h(n)\big)$

۲) $f(n)\in \theta\big(g(n)\big), g(n)\in O\big(h(n)\big) \Rightarrow h(n)\in O\big(f(n)\big)$

۳) $f(n)\in \Omega\big(g(n)\big), h(n)\in O\big(g(n)\big) \Rightarrow f(n)\in \theta\big(g(n)\big)$

۴) $f(n)\in \Omega\big(g(n)\big), g(n)\in \theta\big(h(n)\big) \Rightarrow f(n)\in \Omega\big(h(n)\big)$

۱۵- کدام یک از عبارات زیر صحیح است؟ **(علوم کامپیوتر ۸۱)**

۱) $\sum_{i=0}^{n} i^2 = O(n^3)$ ۲) $3^n = O(2^n)$

۳) $n^2 \log n = O(n^2)$ ۴) $\frac{n^2}{\log n} = O(n^2)$

۱۶- کدام یک از روابط ذیل درست است؟ **(آزاد ۸۱)**

۱) $O(Log\, n) < O(\sqrt{n})$ ۲) $O(Log\, n) > O(\sqrt{n})$

۳) $O(n!) < O(a^n)$ ۴) $O(\sqrt{n^3}) < O(n)$

۱۷- فرض کنید f و g، دو تابع دلخواه به شکل $f, g : N \to R^+$، به علاوه فرض کنید $\lim_{n\to\infty} \frac{f(n)}{g(n)} = +\infty$ کدام یک از گزاره‌های زیر درست است؟ **(مهندسی کامپیوتر دولتی ۸۲)**

۱) $g(n)\notin \Omega\big(f(n)\big)$ و $f(n)\in O\big(g(n)\big)$ ۲) $f(n)\in O\big(g(n)\big)$ و $g(n)\in O\big(f(n)\big)$

۳) $f(n)\notin \theta\big(g(n)\big)$ و $g(n)\notin \Omega\big(f(n)\big)$ ۴) $f(n)\in \Omega\big(g(n)\big)$ و $f(n)\in \theta\big(g(n)\big)$

۱۸- کدام عبارت صحیح است؟ **(علوم کامپیوتر سراسری ۸۲)**

۱) $(n+1)(n^2-2n+1)\in O(2^n)$ ۲) $(n+1)(n^2-2n+1)\in \theta(n)$

۳) $(n+1)(n^2-2n+1)\in \Omega(n^4)$ ۴) $(n+1)(n^2-2n+1)\in O(n^2 \log n)$

۱۹- کدام گزینه صحیح می‌باشد؟ **(علوم کامپیوتر سراسری ۸۴)**

۱) $n^3 \log n = O(n^{3+\xi}) \quad 0 < \xi < 0/1$

۲) $\sqrt{n} = (\log n)$

۳) $n^{1+\xi} = O(n \log n) \quad 0 < \xi < 0/1$

۴) $n^2 = O\left(\frac{n^2}{\log n}\right)$

۲۰- کدام یک از گزینه‌های زیر صحیح است؟ **(طراحی الگوریتم IT ۸۴)**

۱) $n^۲ \sin n \in \Omega(n)$

۲) $n^۲ \sin n \in O(n)$

۳) $n \in \Theta(n^۲ \sin n)$

۴) $n \in \Omega(n^۲ \sin n)$

۲۱- در رشد توابع زیر کدام ترتیب صحیح می‌باشد؟ $(\varepsilon > ۰)$ **(علوم کامپیوتر ۸۵)**

۱) $O\left(\frac{n^۲}{\log n}\right)$ و $O(۱+\varepsilon)^n$ و $O(n \log n)$

۲) $O\left(\frac{n^۲}{\log n}\right)$ و $O(n \log n)$ و $O(۱+\varepsilon)^n$

۳) $O(۱+\varepsilon)^n$ و $O(n \log n)$ و $O\left(\frac{n^۲}{\log n}\right)$

۴) $O(۱+\varepsilon)^n$ و $O\left(\frac{n^۲}{\log n}\right)$ و $O(n \log n)$

۲۲- کدام یک از عبارات زیر درست‌اند: **(دولتی ۸۵)**

$e^{c\sqrt{n}} = O(e^{\sqrt{n}}).I \quad C \geq ۱$

$n^۲ = O(n \lg n).II$

$n = O(n \lg n).III$

۱) فقط II

۲) فقط III

۳) I و II

۴) I و III

۲۳- کدام یک از عبارت‌های زیر نادرست است؟ **(IT ۸۵)**

۱) $O(n!) = O(n^n)$

۲) $O(۱۰^۶) < O(n) < O(n.\log_۲ n)$

۳) هر الگوریتم از مرتبه $\theta(n)$ از مرتبه $O(n^۲)$ نیز هست.

۴) حذف عنصر آخر در یک لیست زنجیره‌ای یک طرفه، با داشتن اشاره گرهای $first$ و $last$ از مرتبه $O(1)$ است.

۲۴- کامپیوتری در واحد زمان مسأله‌ای به اندازه ۱۶ را که الگوریتم آن از مرتبه زمانی $n۲^n$ است حل می‌کند. اگر سرعت کامپیوتر ۱۳۱۰۷۲ برابر گردد این کامپیوتر همان مسأله را با چه اندازه‌ای در واحد زمان حل خواهد کرد؟ **(دولتی ۸۶)**

۱) $۱۶+۱۷ + log۱۷$ ۲) $۱۶\times۱۷\times\log۱۷$

۳) ۳۲ ۴) $۱۶ + \log۱۳۱۰۷۲$

۲۵- کدام گزینه صحیح است؟ (۱/۲ یعنی 1.2) **(علوم کامپیوتر ۸۹)**

۱) $\frac{n}{\lg n} = O(n^{۱-x}), ۳^n = \Omega(n.۲^n)$

۲) $n\left(\log_۳^n\right)^۵ = \Omega(n^{۱/۲}), \sqrt{n} = O\left((\lg n)^۵\right)$

۳) $\frac{n}{\lg n} = \Omega(n^{۱-x}) \quad ۰ < x < ۱ \, , ۳^n = O(n۲^n)$

۴) $n\left(\log_۳^n\right)^۵ = O(n^{۱/۲}) \, , \sqrt{n} = \Omega\left((\lg n)^۵\right)$

۲۶- میزان زمان لازم برای اجرای قطعه برنامه زیر چگونه برآورد می‌شود؟ **(IT ۸۳)**

```
for i ← ۱ to n
    for j ← n to i
        for k ← ۱ to n^۲
            sum ← sum + A
```

۱) $O(n^۲)$

۲) $\theta(n^۳)$

۳) $\omega(n^۴)$

۴) $\theta(n^۴)$

۲۷- کدام گزینه مرتبه زمانی کد زیر را نشان می‌دهد؟ **(IT - ۸۷ و ۹۰)**

```
for(i = ۱; i <= n; i++)
    for(j = ۱; j <= n; j = j + i)
    x++ ;
```

۱) $\theta(n)$ ۲) $\theta(n^۲)$

۳) $\theta(n\log n)$ ۴) $\theta(n^۲\log n)$

۲۸- در کد زیر دستور $x++$ چند بار تکرار می‌شود فرض کنید $n >= ۳$ است. **(IT - ۸۷)**

```
for(i = ۳; i <= n; i = i*۲)
    x++;
```

۱) $\left\lfloor\log\frac{n}{۳}\right\rfloor$ ۲) $\left\lceil\frac{(\log n)}{۳}\right\rceil$

۳) $\left\lfloor\log\frac{n}{۳}+۱\right\rfloor$ ۴) $\left\lfloor\frac{(\log n+۱)}{۳}\right\rfloor$

۲۹- مرتبه زمانی شبه کد زیر چیست؟ **(IT - ۸۸)**

```
for( i = ۱; i ≤ n ; i = i* ۲)
    for (j = ۱; j ≤ n; j = j* ۲)
        for (k = ۱; k ≤ j; k++)
            x++;
```

۱) $\theta(n)$

۲) $\theta(n^۲)$

۳) $\theta(n \log n)$

۴) $\theta\left(n(\log n)^۲\right)$

۳۰- در یک زمستان سرد، خرس قطبی n قطعه گوشت دقیقاً به اندازه‌های ۱، ۲ تا n را در غاری ذخیره کرده است. او هر روز یکی از این قطعه گوشت‌ها را به صورت تصادفی انتخاب می‌کند. اگر اندازه‌ی گوشت عدد فردی بود، آن را کاملاً می‌خورد. اگر زوج بود، آن را دقیقاً نصف می‌کند، یک نصف آن را می‌خورد و نصف دیگر را مجدداً در غار قرار می‌دهد. اگر گوشتی موجود نباشد، خرس می‌میرد. با این الگوریتم، برای nهای خیلی بزرگ روزهای باقیمانده از عمر خرس ما تابع کدام یک از گزینه‌ها خواهد بود؟ **(مهندسی کامپیوتر ۸۹)**

۱) $\theta(n)$ ۲) $\theta(\log n)$ ۳) $\theta(n \log n)$ ۴) $\theta(n^۲)$

۳۱- توابع $f(n) = ۴^{\lg n}$ ، $g(n) = l g^{\lg n} n$ و $h(n) = \lg^۲ n$ را در نظر بگیرید. کدام یک از گزاره‌های زیر صحیح است؟ **(مهندسی کامپیوتر ۸۹)**

۱) $f(n) \in O(g(n)),\ f(n) \in \Omega(h(n))$

۲) $f(n) \in \Theta(h(n)),\ g(n) \in \Omega(f(n))$

۳) $g(n) \in \Omega(h(n)),\ h(n) \in \Omega(f(n))$

۴) $h(n) \in O(g(n)),\ f(n) \in \Theta(g(n))$

[پاسخنامه سؤال‌های طبقه‌بندی شده فصل اول]

۱) گزینه (۱). لازم به ذکر است که تعریف متغیر به شرطی جمله اجرایی است که مقدار اولیه گرفته باشد:

جمله	تعداد اجرا
$temp=0$	۱
for	$n+۱$
$temp+=num[i]$	n
$return$	۱
	جواب $=۲n+۳$

۲) گزینه (۲).

$$\sum_{k=۰}^{n-۱}\sum_{i=۱}^{n-k}۱=\sum_{k=۰}^{n-۱}(n-k)=\sum_{k=۰}^{n-۱}n-\sum_{k=۰}^{n-۱}k=n^۲-\frac{n(n-۱)}{۲}=\frac{n(n+۱)}{۲}$$

۳) گزینه (۲). حلقه for، N بار و حلقه $while$، $\log_۲^N$ بار اجرا می‌شود. می‌توان به ازای $N=۱$ بررسی کرد که در این صورت گزینه ۲ بدست می‌آید. البته باید فرض کنیم که N توانی از ۲ است.

۴) گزینه (۴). حلقه $while$ داخلی $\log_۲^n+۱$ بار اجرا می‌شود و حلقه بیرونی n بار اجرا می‌شود.

۵) گزینه (۴).

A	۳۰	۱۵	۷	۳	۱	۰
B	۵	۱۰	۲۰	۴۰	۸۰	۱۶۰
P	۰	۰	۱۰	۳۰	۷۰	۱۵۰
$A\ mod ۲$	۰	۱	۱	۱	۱	

۶) گزینه (۱). حلقه $while$ دارای مرتبه $\log_۵^N$، حلقه for، N و داخل حلقه for، M است.

۷) گزینه (۴). اگر k بر اعداد ۲ تا n بخش پذیر باشد آنگاه حلقه for داخلی نیز هر بار اجرا می‌شود و چون هر دو حلقه به n وابسته اند، زمان $n^۲$ می‌شود ولی اگر k بر هیچ یک از اعداد ۲ تا n بخش پذیر نباشد فقط حلقه بیرون اجرا می‌شود که مرتبه آن n است.

۸) گزینه (۲). به ازای $i=۱$، جمله $x++$ (جمله اصلی)، n بار اجرا می‌شود. به ازای $i=۲$، $n-۱$ بار (دقت کنید n یک واحد کم شده است). به همین ترتیب به ازای هر

افزایش i مقدار n کم می‌شود وقتی به $\frac{n}{۲}$ برسیم کار تمام است. بنابراین تعداد اجرای جمله $x++$ حدوداً برابر است با: $n+(n-۱)+(n-۲)+\ldots+(\frac{n}{۲})=\theta(n^۲)$

۹) گزینه (۱). به ازای $i=۱$، جمله $x++$ به تعداد $\frac{n}{۲}$ بار اجرا می‌شود زیرا به ازای هر $j++$ یکبار $n--$ انجام می‌شود. به ازای $i=۲$ جمله اصلی به تعداد $\frac{n}{۴}$ بار اجرا می‌شود و به همین ترتیب: $\frac{n}{۲}+\frac{n}{۴}+\frac{n}{۸}+\ldots=n(\frac{\frac{۱}{۲}}{۱-\frac{۱}{۲}})=\theta(n)$

۱۰) گزینه (۳). حلقه for داخلی، n بار. حلقه $while$، $\lfloor \log n \rfloor$ و حلقه for بیرونی n بار اجرا می‌شوند.

۱۱) گزینه (۲). توابع نمایی مثل $(۱/۰۰۵)^n$ از چند جمله‌ای مثل $n^{۱۰۰۰}$ رشد بیشتری دارند.

۱۲) گزینه (۴). حلقه $while$ دارای مرتبه $O(\log_۲^n)$ و حلقه for داخلی دارای مرتبه $O(m)$ می‌باشد و چون این دو حلقه متوالی هستند دارای مرتبه $O(m+\log_۲^n)$ می‌باشند بنابراین کل برنامه دارای مرتبه $O(n(m+\log_۲^n))$ می‌باشد.

۱۳) گزینه (۴). حلقه $while$ به ازای هر n، تعداد محدودی اجرا می‌شود، چون حاصل $mod۲$ یا ۰ یا ۱ است.

۱۴) گزینه (۴). $f\in\Omega(g)$ یعنی رشد f بیشتر یا مساوی g است. $g\in\theta(h)$ یعنی g با h هم رشد است. پس رشد f بیشتر یا مساوی h است یعنی $f\in\Omega(h)$

۱۵) گزینه (۴ و ۱). رشد $\frac{n^۲}{\log n}$ از $n^۲$ کمتر است زیرا $n^۲$ از $n^۲\log n$ کمتر است پس $\frac{n^۲}{\log n}=O(n^۲)$ ولی اگر O را به معنی θ فرض کنیم گزینه ۱ صحیح است.

۱۶) گزینه (۱). نرخ رشد تابع $\sqrt{n}$ سریع‌تر از $log\ n$ است. زیرا n^a از $(\log n)^b$ رشد بیشتری دارد $(a,b>۰)$

۱۷) گزینه (۳). توجه کنید فرض مسئله نشان می‌دهد نرخ رشد تابع $f(n)$ سریعتر از تابع $g(n)$ است. پس $f(n)\in\Omega(g(n))$ بوده ولی $f(n)\notin\theta(g(n))$.

۱۸) گزینه (۱). تابع $(n+۱)(n^۲-۲n+۱)$ از مرتبه $\theta(n^۳)$ است پس می‌توان گفت از مرتبه $O(۲^n)$ است.

۱۹) گزینه (۱). می‌دانیم لگاریتم به هر توان ثابتی از n به هر توان مثبتی، رشد کمتری دارد یعنی $(Lgn)^a < n^b$ پس $(\varepsilon > ۰) Lgn < n^{\varepsilon}$ حال طرفین را در $n^۳$ ضرب می‌کنیم $n^۳Lgn < n^{۳+\varepsilon}$ در نتیجه $n^۳Lgn = O(n^{۳+\varepsilon})$. در مورد سایر گزینه‌ها دقت کنید:

$$\sqrt{n} > Lgn \quad , \quad n^{۱+\varepsilon} > nLgn \quad , \quad n^۲ > \frac{n^۲}{Lgn}$$

۲۰) گزینه (۱). اگر n طبیعی فرض شود آنگاه $n^۲ \sin n \geq Cn$ به ازای حداقل یک C و $n_۰$ مثبت و همه $n \geq n_۰$ صحیح است. البته $\sin n$ باید داخل قدر مطلق باشد.

۲۱) گزینه (۴). رشد $n \log n$ کمتر از $\frac{n^۲}{\log n}$ و کمتر از $(۱+\varepsilon)^n$ است (از حد استفاده کنید)

۲۲) گزینه (۲). مرتبه $e^{c\sqrt{n}}$ از $e^{\sqrt{n}}$ بیشتر است و $n^۲$ از $nlogn$ بیشتر است.

۲۳) گزینه (۴). گزینه ۴ نیاز به پیمایش دارد تا آدرس گره ماقبل آخر بدست آید که مرتبه آن $O(n)$ می‌شود. البته گزینه ۱ هم اشکال دارد و باید به صورت $n! = O(n^n)$ نوشته شود.

۲۴) گزینه (۳). (دقت کنید $۲^{۱۷} = ۱۳۱۰۷۲$) وقتی گفته می شود مرتبه زمانی $n۲^n$ است می توان فرض کرد زمان اجرا $cn۲^n$ است.

$$c \times ۱۶ \times ۲^{۱۶} = ۱ \Rightarrow c = \frac{۱}{۲^{۲۰}} \quad , \quad \frac{۱}{۲^{۲۰}} \times \frac{۱}{۱۳۱۰۷۲} \times n \times ۲^n = ۱ \Rightarrow n = ۳۲$$

۲۵) گزینه (۴).

رشد $\sqrt{n}$ از $\lg n$ به هر توانی بیشتر است.

رشد $\log_۳^n$ به هر توانی از $n^{۰/۲}$ کمتر است.

۲۶) گزینه (۴). حلقه اول و دوم به n و حلقه داخلی به $n^۲$ وابسته‌اند. در حلقه دوم to به معنی $down\ to$ است!

۲۷) گزینه (۳).

i	j	تعداد اجرای $x++$
۱	n و ... و ۲ و ۱	n
۲	... و ۳ و ۱	$\frac{n}{۲}$
۳	... و ۴ و ۱	$\frac{n}{۳}$
:		
n	۱	$\frac{n}{n}$

$$= n + \frac{n}{۲} + \frac{n}{۳} + ... + \frac{n}{n} = n(۱ + \frac{۱}{۲} + \frac{۱}{۳} + ... + \frac{۱}{n}) \simeq n.Lnn = \theta(n.\lg n)$$

توجه: ثابت می‌شود $\frac{۱}{۲} + ... + \frac{۱}{n} < Lnn < ۱ + \frac{۱}{۲} + ... + \frac{۱}{n-۱}$

۲۸) گزینه (۳). به ازای n=۳ ، یک بار اجرا می‌شود. و به ازای $n = ۶$ دوبار که فقط در گزینه ۳ صدق می‌کنند.

۲۹) گزینه (۳). حلقه i را نادیده بگیرید آنگاه تعداد اجرای دو حلقه دیگر برابر $(۲^m = n)$، $۲^۰ + ۲^۱ + ۲^۲ + ... + ۲^m = ۲^{m+۱} - ۱ = \theta(n)$ می‌باشد. پس مرتبه کل الگوریتم $\theta(n.\log n)$ است زیرا حلقه i از مرتبه $\theta(\log n)$ است.

۳۰) گزینه (۱). n قطعه گوشت، خرس را n روز زنده نگه می‌دارد. $\frac{n}{۲}$ از این قطعات زوج است که این قطعات مجدداً ذخیره می‌شوند که در نتیجه $\frac{n}{۲}$ روز دیگر خرس زنده می‌ماند. $\frac{n}{۴}$ از این قطعات زوج است که در نتیجه باقی می‌مانند و $\frac{n}{۴}$ روز دیگر خرس زنده می‌ماند. به همین ترتیب تعداد روزهای باقیمانده از عمر خرس برابر است با:

$$n + \frac{n}{۲} + \frac{n}{۴} + \frac{n}{۸} + ... = n\left(۱ + \frac{۱}{۲} + \frac{۱}{۴} + ...\right) = n \times \frac{۱}{۱ - \frac{۱}{۲}} = ۲n$$

۳۱) گزینه (۱). ترتیب رشد توابع به شکل زیر است:

$$\lg^۲ n < ۴^{\lg n} = n^۲ < \lg^{\lg n} n$$

پس $۴^{\lg n} \in O(\lg^{\lg n} n) \quad , \quad ۴^{\lg n} \in \Omega(\lg^۲ n)$

توجه: اگر لگاریتم توابع گفته شده را بگیرید، به همین نتیجه می‌رسید. ولی دقت کنید که اگر تابع f از تابع g رشد کمتری داشته باشد ممکن است lgf با lgg هم رشد شود.

فصل ۲ الگوریتم‌های بازگشتی

الگوریتم بازگشتی خود را فراخوانی می‌کند. با یک مثال وارد بحث می‌شویم.

مثال: با توجه به تابع بازگشتی f

```
f(n)
{
    if(n ≤ ۱)return   n * n;
    return   f(n − ۱) + f(n − ۱) + n;
}
```

الف) $f(۴)$ چند است؟

ب) $f(۴)$ چه تعداد فراخوانی دارد؟

ج) $f(۴)$ چه تعداد عمل + را انجام می‌دهد؟

د) $f(۴)$ چه تعداد عمل * انجام می‌دهد؟

هـ) اگر $T(n)$ تعداد فراخوانی $f(n)$ باشد برای $T(n)$ فرمول بازگشتی بنویسید.

و) اگر $T(n)$ تعداد عمل + برای $f(n)$ باشد برای $T(n)$ فرمول بازگشتی بنویسید.

ز) اگر $T(n)$ تعداد عمل * برای $f(n)$ باشد برای $T(n)$ فرمول بازگشتی بنویسید.

ح) اگر $T(n)$ زمان اجرای تابع $f(n)$ باشد برای $T(n)$ فرمول بازگشتی بنویسید.

پاسخ: الف)

$$f(۱) = ۱ \ , \ f(۲) = f(۱) + f(۱) + ۲ = ۴,$$
$$f(۳) = f(۲) + f(۲) + ۳ = ۱۱ \ , \ f(۴) = f(۳) + f(۳) + ۴ = ۲۶$$

ب) $f(۴)$ دو بار $f(۳)$ را صدا می‌زند. هر $f(۳)$ دو بار $f(۲)$ را صدا می‌زند. هر $f(۲)$ دو بار $f(۱)$ را صدا می‌زند پس تعداد فراخوانی جمعاً $۱۵ = ۸ + ۴ + ۲ + ۱$ است.

ج) به ازای هر $n > ۱$ دو بار عمل + انجام می‌شود. پس تعداد عمل + برابر $۱۴ = ۸ + ۴ + ۲$ است.

د) به ازای هر $n \leq ۱$ یک بار عمل * انجام می‌شود. و چون ۸ بار $f(۱)$ فراخوانی می‌شود پس ۸ بار عمل * انجام می‌شود.

هـ) هر فراخوانی $f(n)$ دو بار $f(n-۱)$ را فراخوانی می‌کند پس $T(n) = ۲T(n-۱) + ۱$. $T(n-۱)$ بخاطر $f(n-۱)$ و عدد ۱ همان فراخوانی اول است. این رابطه بازگشتی مرتبه یک است (یعنی $T(n)$ فقط به $T(n-۱)$ وابسته است) و یک شرط اولیه می‌خواهد که $T(۱) = ۱$ چون به ازای $n = ۱$ یک فراخوانی انجام می‌شود.

و) هر فراخوانی $f(n)$ که $n > ۱$ دو بار عمل + دارد و دو بار فر اخوانی $f(n-۱)$ پس $T(n) = ۲T(n-۱) + ۲$ و شرط اولیه $T(۱) = ۰$ چون به ازای $n = ۱$ اصلاً عمل + ندارد.

ز) هر فراخوانی $f(n)$ که $n > ۱$ اصلاً عمل * ندارد ولی دو فراخوانی $f(n-۱)$ دارد پس $T(n) = ۲T(n-۱)$ و شرط اولیه $T(۱) = ۱$ چون به ازای $n = ۱$ یک عمل * انجام می‌شود.

ح) زمان اجرا برابر تعداد اجرای همه دستورات است ولی می‌توان زمان اجرا را برابر تعداد اجرای عمل اصلی دانست. عمل اصلی در تابع f می‌تواند تعداد انجام عمل + باشد یا تعداد اجرای if یا تعداد فراخوانی‌ها. اگر زمان را برابر تعداد اجرای if فرض کنیم می‌توان نوشت: $T(n) = ۲T(n-۱) + ۱$ و $T(۱) = ۱$. می‌تون ثابت کرد که $T(n) = ۲^n - ۱$ می‌شود و از مرتبه $\theta(۲^n)$ است.

نکته: اگر $T(n) = a \cdot T(n-b) + c$ که a, b, c ثابت هستند آنگاه اگر $a \neq ۱$ آنگاه $T(n) = \theta(a^{\frac{n}{b}})$ و اگر $a = ۱$ و $c \neq ۰$ آنگاه $T(n) = \theta(n)$. برای اثبات این نکته، حل روابط بازگشتی را بخوانید)

مثال: زمان اجرای تابع مقابل از چه مرتبه‌ای است؟

```
g(n)
{
    if(n ≤ ۱)return  n * n;
    return  ۲ * g(n − ۱) + n;
}
```

پاسخ: اگر $T(n)$ را برابر تعداد اجرای if فرض کنیم آنگاه $T(n) = T(n-1) + 1$ و $T(1) = 1$ که $T(n) = \theta(n)$ یعنی مرتبه خطی است. دقت کنید خروجی این تابع با تابع مثال قبل همواره مساوی است ولی زمان اجرای این دو تابع متفاوت است.

حل روابط بازگشتی

برای یافتن مرتبه الگوریتم‌های بازگشتی ابتدا باید عمل اصلی آن الگوریتم را مشخص کنیم سپس برای آن رابطه بازگشتی بنویسیم و سپس رابطه بازگشتی را حل کنیم (البته خیلی مواقع با نکات می‌توان بدون حل مرتبه را یافت) و مرتبه اجرایی را بیابیم. روش‌های حل روابط بازگشتی متنوع هستند که در درس ساختمان گسسته تشریح می‌شوند. در این درس خلاصه‌ای از روش‌های حل را ارائه می‌دهیم. ابتدا حل روابط بازگشتی به فرم

$$a_n = c_1 \cdot a_{n-1} + c_2 \cdot a_{n-2} + ... + c_k \cdot a_{n-k} + b_1^n \cdot p_1(n) + b_2^n \cdot p_2(n) + ...$$

که رابطه مرتبه k است را بررسی می‌کنیم. در این رابطه b_iها ثابت هستند و $p_i(n)$ ها چند جمله‌ای درجه d_i ($p_1(n)$ چند جمله‌ای درجه d_1 است. $p_2(n)$ چند جمله‌ای درجه d_2 است و ...) در ضمن همه b_iها متمایز هستند. برای حل این رابطه معادله مشخصه تشکیل می‌دهیم که عبارت است از:

$$(x^k - c_1x^{k-1} - c_2x^{k-2} - ... - c_k)(x-b_1)^{d_1+1}(x-b_2)^{d_2+1}... = 0$$

ریشه‌های این معادله را می‌یابیم. اگر دو ریشه حقیقی متمایز x_1 و x_2 وجود داشت. شکل کلی جواب $a_n = \alpha_1(x_1)^n + \alpha_2(x_2)^n$ است. اگر سه ریشه حقیقی متمایز x_1 و x_2 و x_3 وجود داشت، شکل کلی جواب $a_n = \alpha_1(x_1)^n + \alpha_2(x_2)^n + \alpha_3(x_3)^n$. اگر ریشه مضاعف $x_1 = x_2$ داشت، شکل جواب $a_n = (\alpha_1 + \alpha_2 n)(x_1)^n$ است. اگر ریشه ۳ گانه $x_1 = x_2 = x_3$ داشت شکل جواب $a_n = (\alpha_1 + \alpha_2 n + \alpha_3 n^2)(x_1)^n$ است و به همین ترتیب (α_iها ثابت هستند که با توجه به شروط اولیه محاسبه می‌شوند)

مثال: رابطه بازگشتی فیبوناچی $F_n = F_{n-1} + F_{n-2}$، $F_0 = 0$ و $F_1 = 1$ یک رابطه مرتبه ۲ است. این رابطه همگن است (یعنی $b_i^n p_i(n)$ ندارد) برای حل، معادله مشخصه تشکیل می‌دهیم که $x^2 - x - 1 = 0$ است. ریشه‌های این معادله $x_1 = \frac{1+\sqrt{5}}{2}$ و $x_2 = \frac{1-\sqrt{5}}{2}$

می‌باشد پس شکل کلی جواب $F_n = \alpha_1(\frac{1+\sqrt{5}}{2})^n + \alpha_2(\frac{1-\sqrt{5}}{2})^n$ است. برای یافتن α_1 و α_2، شروط اولیه را اعمال می‌کنیم:

$$\begin{cases} F_0 = \alpha_1(\frac{1+\sqrt{5}}{2})^0 + \alpha_2(\frac{1-\sqrt{5}}{2})^0 = 0 \\ F_1 = \alpha_1(\frac{1+\sqrt{5}}{2})^1 + \alpha_2(\frac{1-\sqrt{5}}{2})^1 = 1 \end{cases}$$

از معادله اول $\alpha_1 + \alpha_2 = 0$ پس $\alpha_2 = -\alpha_1$ که در معادله دوم جایگزین می‌کنیم:

$$\alpha_1(\frac{1+\sqrt{5}}{2}) - \alpha_1(\frac{1-\sqrt{5}}{2}) = 1 \rightarrow \alpha_1 = \frac{1}{\sqrt{5}}$$

پس $\alpha_2 = \frac{-1}{\sqrt{5}}$ در نتیجه:

$$F_n = \frac{1}{\sqrt{5}}(\frac{1+\sqrt{5}}{2})^n - \frac{1}{\sqrt{5}}(\frac{1-\sqrt{5}}{2})^n = \theta((\frac{1+\sqrt{5}}{2})^n)$$

مثال: رابطه $a_n = a_{n-1} + n$ و $a_0 = 0$ مرتبه یک و ناهمگن است. برای حل ابتدا دقت کنید این رابطه به شکل $a_n = a_{n-1} + 1^n(n)$ است که $b_1 = 1$ و $d_1 = 1$ پس معادله مشخصه $(x-1)(x-1)^2 = 0$ است. سه ریشه مساوی $x_1 = x_2 = x_3 = 1$ دارد پس $a_n = (\alpha_1 + \alpha_2 n + \alpha_3 n^2)(1)^n$. برای یافتن α_1 و α_2 و α_3 نیاز به ۳ شرط اولیه داریم. از رابطه $a_n = a_{n-1} + n$ و شرط $a_0 = 0$ می‌توان a_1 و a_2 را محاسبه کرد:

$a_1 = a_0 + 1 = 1$ و $a_2 = a_1 + 2 = 3$

حال a_0 و a_1 و a_2 را اعمال می‌کنیم:

$$\begin{cases} a_0 = (\alpha_1 + 0 + 0)(1)^0 = 0 \\ a_1 = (\alpha_1 + \alpha_2 + \alpha_3)(1)^1 = 1 \\ a_2 = (\alpha_1 + 2\alpha_2 + 4\alpha_3)(1)^2 = 3 \end{cases}$$

با حل این دستگاه $\alpha_1 = 0$ و $\alpha_2 = \alpha_3 = \frac{1}{2}$ پس $a_n = \frac{n(n+1)}{2} = \theta(n^2)$

مثال: با توجه به رابطه بازگشتی $T(n) = 4T(n-2) + 2^n + n$ فقط مرتبه $T(n)$ را با نماد θ بیان کنید.

پاسخ: این رابطه ناهمگن مرتبه ۲ است و به شکل

$$T(n) = ۰T(n-۱) + ۴T(n-۲) + ۲^n(۱) + ۱^n(n)$$

است پس $b_۱ = ۲$ و $d_۱ = ۰$ و $b_۲ = ۱$ و $d_۲ = ۱$ بنابراین معادله مشخصه عبارت است از:

$$(x^۲ - ۴)(x-۲)^۱(x-۱)^۲ = ۰$$

ریشه‌های این معادله ۲- و ۲ و ۲و ۱و ۱ می‌باشد پس شکل کلی جواب

$$T(n) = \alpha_۱(-۲)^n + (\alpha_۲ + \alpha_۳n)(۲)^n + (\alpha_۴ + \alpha_۵n)(۱)^n$$

اگر $\alpha_۳ \neq ۰$ مرتبه این رابطه $T(n) = \theta(n \cdot ۲^n)$ است.

مثال: زمان اجرای تابع زیر از چه مرتبه‌ای است؟

```
f(n)
{
   if(n ≤ ۱)return n;
   return f(n-۱) * f(n-۲) + n;
}
```

پاسخ: زمان اجرای تابع f را برابر تعداد اجرای if در نظر می‌گیریم و با $T(n)$ نشان می‌دهیم پس $T(n) = T(n-۱) + T(n-۲) + ۱$ و $T(۰) = ۱$ و $T(۱) = ۱$. که با حل این رابطه $T(n) = \theta((\frac{۱+\sqrt{۵}}{۲})^n)$ حاصل می‌شود که نمایی است درواقع $۲^{\frac{n}{۲}} < T(n) < ۲^n$ می‌باشد.

تاکنون روابط بازگشتی خطی را حل کردیم یعنی روابطی که a_n به $a_{n-۱}$ و $a_{n-۲}$ و ... وابسته هست. حال می‌خواهیم نحوه بیان مرتبه سایر روابط بازگشتی را ارائه دهیم.

قضیه Master: فرض کنید $a \geq ۱$ و $b > ۱$ ثابت هستند و $f(n)$ یک تابع است. با توجه به رابطه بازگشتی $T(n) = aT(\frac{n}{b}) + f(n)$ که $\frac{n}{b}$ می‌تواند به شکل $\lfloor \frac{n}{b} \rfloor$ یا $\lceil \frac{n}{b} \rceil$ یا $\frac{n}{b} \pm c$ باشد. برای یافتن مرتبه $T(n)$ سه حالت پیش می‌آید:

حالت ۱: اگر $f(n) = O(n^{Log_b^{a-\varepsilon}})$ که $\varepsilon > ۰$ ثابت است آنگاه $T(n) = \theta(n^{\log_b^a})$

حالت ۲: اگر $f(n) = \theta(n^{\log_b^a})$ آنگاه $T(n) = \theta(f(n) \cdot \lg n) = \theta(n^{\log_b^a} \cdot \lg n)$

حالت ۳: اگر $f(n) = \Omega(n^{\log_b^{a+\varepsilon}})$ که $\varepsilon > ۰$ ثابت است و اگر $af(\frac{n}{b}) \leq cf(n)$ برای ثابت $c < ۱$ و nهای به اندازه کافی بزرگ آنگاه $T(n) = \theta(f(n))$.

توجه: اگر $f(n)$ چند جمله‌ای باشد، قضیه $Master$ را می‌توان به زبان ساده اینطور بیان کرد که مرتبه $f(n)$ را با $n^{\log_b^a}$ مقایسه کنید اگر $f(n)$ رشد بیشتری داشت آنگاه $T(n) = \theta(f(n))$. اگر $n^{\log_b^a}$ رشد بیشتری داشت آنگاه $T(n) = \theta(n^{\log_b^a})$ و اگر هم‌رشد بودند آنگاه $T(n) = \theta(f(n) \cdot \lg n)$.

مثال:

$a)\quad T(n) = ۴T(\frac{n}{۲}) + n \quad , \quad a = ۴, b = ۲, f(n) = n = O(n^{\log_۲^{۴-\varepsilon}})$

که به ازای مثلاً $\varepsilon = ۱$ درست است پس $T(n) = \theta(n^۲)$

$b)\quad T(n) = ۲T(\frac{n}{۲}) + n \quad , \quad a = b = ۲, f(n) = n = O(n^{\log_۲^۲})$

حالت ۲ برقرار است پس $T(n) = \theta(n \cdot \lg n)$

$c)\quad T(n) = ۲T(\frac{n}{۲}) + n^۲ \quad , \quad a = b = ۲, f(n) = n^۲ = \Omega(n^{\log_۲^{۲+\varepsilon}})$

که به ازای مثلاً $\varepsilon = ۱$ درست است. حال شرط $af(\frac{n}{b}) \le cf(n)$ را بررسی می‌کنیم:

$$۲f(\frac{n}{۲}) = ۲(\frac{n^۲}{۴}) = \frac{۱}{۲}n^۲ \le cn^۲$$

که به ازای مثلاً $c = \frac{۱}{۲}$ برقرار است پس $T(n) = \theta(n^۲)$ البته چون $f(n) = n^۲$ چند جمله‌ای است نیاز به بررسی شرط فوق نیست.

$d)\quad T(n) = ۲T(\frac{n}{۲}) + n \cdot \lg n \quad , \quad a = b = ۲, f(n) = n \cdot \lg n$

ظاهراً $n \cdot \lg n$ از $n^{\lg n_۲^۲} = n$ رشد بیشتری دارد و باید حالت ۳ برقرار باشد ولی $n \cdot \lg n = \Omega(n^{\log_۲^{۲+\varepsilon}})$ به ازای هیچ $\varepsilon > ۰$ درست نیست. پس نمی‌توان از قضیه $Master$ کمک گرفت.

$e)\quad T(n) = ۴T(\frac{n}{۲}) + n \cdot \lg n \quad , \quad a = ۴, b = ۲, f(n) = n \cdot \lg n = O(n^{\log_۲^{۴-\varepsilon}})$

به ازای مثلاً $\varepsilon = ۱$ درست است پس $T(n) = \theta(n^۲)$