نظریه‌ی زبان‌ها و ماشین‌ها

درس‌نامه‌ی ۱۴

کاظم فولادی
http://kazim.fouladi.ir

ویراست اول: ۱۳۸۵
ویراست دوم: ۱۳۸۷
ویراست سوم: ۱۳۹۳

فهرست مطالب

۱۴ مقدمه‌ای بر پیچیدگی محاسباتی ۱

۱۴-۱ کارآمدی محاسبات ۱

۱۴-۲ ماشین تورینگ و پیچیدگی ۱

۱۴-۳ خانواده‌های زبان‌ها و کلاس‌های پیچیدگی ۳

۱۴-۴ کلاس‌های پیچیدگی چندجمله‌ای قطعی و غیرقطعی (P و NP) ۵

۱۴-۵ مسائل NP: چند نمونه ۶

۱۴-۶ کاهش زمان-چندجمله‌ای ۶

۱۴-۷ NP-کامل بودن و مسائل باز ۶

۱۴ مقدمه‌ای بر پیچیدگی محاسباتی

An Introduction to Computational Complexity

برای محاسبات واقعی، نه تنها نیاز داریم که بدانیم مسئله قابل حل است، بلکه باید قادر باشیم الگوریتم کارآمدی را برای آن بسازیم.

به مسائلی که می‌توان آنها را به صورت کارآمد حل کرد، رام‌شدنی (tractable) می‌گویند.

مهم‌ترین جنبه‌های کارآیی، نیازهای فضا و زمان است:

نظریه‌ی پیچیدگی: پیچیدگی فضایی و پیچیدگی محاسباتی

۱۴-۱ کارآمدی محاسبات

فرضیات زیر برای بحث در خصوص پیچپدگی محاسباتی لازم است:

۱) مدل محاسباتی مورد مطالعه‌ی ما، ماشین تورینگ (چندنواره) است.

۲) اندازه‌ی مسئله با n نمایش داده می‌شود.

۳) در تحلیل الگوریتم، معمولاً به عملکرد آن در شرایط خاص کمتر از رفتار کلی علاقه‌مندیم. معمولاً رفتار الگوریتم در زمانی که اندازه‌ی مسئله بزرگ می‌شود برای ما اهمیت دارد (افزایش منابع مورد نیاز با رشد n).

هدف اصلی: به دست آوردن نیاز زمانی مسئله به صورت تابعی از اندازه‌ی مسئله با استفاده از ماشین تورینگ به عنوان مدل.

مفهوم زمان در ماشین تورینگ: هر حرکت ماشین تورینگ را یک واحد زمانی در نظر می‌گیریم.

پس زمان یک محاسبه، تعداد حرکاتی است که انجام می‌شود.

معمولاً بدترین حالت را در نظر می‌گیریم.

وقتی می‌گوییم محاسبه‌ای دارای پیچیدگی زمانی $T(n)$ است، یعنی محاسبه‌ی هر مسئله‌ای با اندازه‌ی n با انجام $T(n)$ حرکت توسط یک ماشین تورینگ به انجام می‌رسد.

۱۴-۲ ماشین تورینگ و پیچیدگی

زمانی که صحبت از پیچیدگی می‌شود، دیگر هم‌ارزی بین انواع ماشین‌های تورینگ قابل طرح نیست.

مثال

ماشین تورینگ (یک نواره) برای زبان $L = \{a^nb^n : n \geq ۱\}$ را در نظر بگیرید.

برای $w = a^n b^n$ حدود $2n$ گام جهت تطبیق هر a با b متناظر لازم است.
بنابراین کل محاسبه به زمان $O(n^2)$ نیاز دارد.
با ماشین تورینگ دونواره:
ابتدا همه‌ی aها را روی نوار دوم کپی می‌کنیم،
سپس آنها را با bهای روی نوار اول تطبیق می‌دهیم.
هر دو عمل کپی و تطبیق در زمان $O(n)$ انجام می‌شود،
پس کل محاسبه با ماشین تورینگ دو نواره به اندازه‌ی $O(n)$ زمان نیاز دارد.

مثال

مسئله‌ی عضویت در زبان‌های مستقل از متن:
اگر w را طول رشته و اندازه‌ی مسئله را n در نظر بگیریم،

- جستجوی جامع دارای پیچیدگی $O(n^M)$ است (که M وابسته به گرامر است)،
- الگوریتم CYK دارای پیچیدگی $O(n^3)$ است

که هر دو الگوریتم‌های **قطعی** هستند.
یک الگوریتم **غیرقطعی** برای این مسئله، قواعدی که باید برای اشتقاق w استفاده شوند را حدس (حدس صحیح) می‌زند. اگر گرامر فاقد قواعد تهی و یکه باشد، آن وقت طول اشتقاق حداکثر $|w| = n$ و الگوریتم غیرقطعی دارای پیچیدگی $O(n)$ خواهد بود.

مثال

مسئله‌ی ارضاپذیری (صدق‌پذیری) (satisfiability).
اگر عبارت e در فرم نرمال عطفی (CNF) باشد، آیا مقادیری از لیترال‌های x_1، x_2، ...، x_n وجود دارد که به ازای آنها e درست باشد؟
مثلاً اگر $e_1 = (\bar{x_1} \vee x_2) \wedge (x_1 \vee x_3)$ آن‌گاه $x_1 = 0$، $x_2 = 1$ و $x_3 = 1$ مقدار e_1 را درست می‌کنند. اما $e_2 = (x_1 \vee x_2) \wedge (\bar{x_1} \vee \bar{x_2})$ ارضاناپذیر است (تمام انتساب‌ها e_2 را نادرست می‌کنند).
الگوریتم قطعی برای بررسی ارضاپذیری، 2^n حالت ممکن برای متغیرها را بررسی می‌کند بنابراین پیچیدگی زمانی آن $O(2^n)$ خواهد بود که نسبت به n نمایی است.
الگوریتم غیرقطعی برای این مسئله به این ترتیب عمل می‌کند که اگر e ارضاپذیر باشد، آن‌گاه مقدار هر یک از x_iها را حدس زده، e را ارزیابی می‌کند. این یک الگوریتم $O(n)$ است.

نتیجه پاسخ به پرسش‌های پیچیدگی متاثر از نوع ماشین تورینگ مورد استفاده است.
به علاوه، مسئله‌ی قطعیت در مقابل عدم قطعیت بسیار مهم است.

الگورتیم مربوط به ماشین چندنواره ممکن است بسیار نزدیک به آن چیزی باشد که در زبان‌های برنامه‌سازی داریم. بنابراین، از یک ماشین تورینگ چندنواره به عنوان مدل استفاده می‌کنیم.

۳-۱۴ خانواده‌های زبان‌ها و کلاس‌های پیچیدگی

در سلسله‌مراتب چامسکی، خانواده‌های زبان‌ها با کلاس‌های آتوماتا مرتبط می‌شوند و هر کلاس آتوماتا بر اساس طبیعت حافظه‌ی موقت آن تعریف می‌شود.
راه دیگر دسته‌بندی زبان‌ها، استفاده از یک ماشین تورینگ از یک نوع خاص با در نظرگرفتن **پیچیدگی** به عنوان عامل تمایز است.

تعریف **پیچیدگی زمانی یک زبان.** می‌گوییم که یک ماشین تورینگ M، زبان L را در زمان $T(n)$ می‌پذیرد، اگر هر $w \in L$ با $|w| \leq n$ در $O(T(n))$ حرکت پذیرفته شود.
اگر M غیرقطعی باشد، این بدان معناست که برای هر $w \in L$، حداقل یک دنباله از حرکات به طول $O(T(|w|))$ وجود دارد که منجر به پذیرش رشته می‌شود.

تعریف زبان L را عضو کلاس $DTIME(T(n))$ می‌گوییم، هرگاه یک ماشین تورینگ قطعی چندنواره موجود باشد که L را در زمان $T(n)$ بپذیرد.

تعریف زبان L را عضو کلاس $NTIME(T(n))$ می‌گوییم، هرگاه یک ماشین تورینگ غیرقطعی موجود باشد که L را در زمان $T(n)$ بپذیرد.

مشخص است که

$$DTIME(T(n)) \subseteq NTIME(T(n))$$

و

$$T_1 \in O(T_2(n)) \qquad \text{iff} \qquad DTIME(T_1(n)) \subseteq DTIME(T_2(n))$$

هرچه $T(n)$ بزرگتر شود، زبان‌های بیشتری را دربر می‌گیرد.

قضیه برای هر عدد صحیح مثبت k داریم

$$DTIME(n^k) \subset DTIME(n^{k+1})$$

نتیجه زبان‌هایی وجود دارند که می‌توانند در زمان $n^۲$ پذیرفته شوند ولی برای آنها هیچ الگوریتم عضویت خطی وجود ندارد.
زبان‌هایی وجود دارند که در $DTIME(n^۳)$ هستند اما در $DTIME(n^۲)$ نیستند و ...
تابع پیچیدگی $T(n)$ هر چه قدر سریع هم رشد کند، همواره چیزی خارج از $DTIME(T(n))$ وجود دارد.

قضیه تابع تام محاسبه‌پذیر تورینگ $f(n)$ به طوری که هر زبان بازگشتی در $DTIME(f(n))$ باشد، وجود ندارد.

مثال

هر زبان منظم می‌تواند توسط یک آتوماتون متناهی قطعی در زمانی متناسب با طول ورودی پذیرفته شود، پس

$$L_{REG} \subseteq DTIME(n)$$

اما $DTIME(n)$ زبان‌های متعددی را فراتر از L_{REG} می‌پذیرد.
مانند زبان مستقل از متن $\{a^n b^n : n \geq ۰\}$ که می‌تواند در زمان $O(n)$ تشخیص داده شود.

مثال

زبان غیر مستقل از متن $L = \{ww : w\{a, b\}^*\}$ در $NTIME(n)$ است.
زیرا با الگوریتم غیرقطعی زیر می‌توان رشته‌های متعلق به این زبان را شناسایی کرد:

۱) ورودی را از فایل ورودی روی نوار ۱ کپی کنید. به طور غیر قطعی وسط رشته را حدس بزنید.

۲) قسمت دوم را روی نوار ۲ کپی کنید.

۳) علایم روی نوار ۱ را با علایم نوار ۲ یک به یک مقایسه کنید.

روشن است که تمامی قدم‌های فوق را می‌توان در زمان $O(|w|)$ انجام داد، پس $L \in NTIME(n)$.
اگر بتوانیم الگوریتمی قطعی برای پیدا کردن وسط رشته در زمان $O(n)$ بسازیم، می‌توان نشان داد که $L \in DTIME(n)$ است. این کار را می‌توان با نگاه کردن به هر علامت روی نوار ۱ و ثبت شمارش روی نوار ۲ با شمارش یک در میان نمادها انجام داد.

مثال

با تعریف L_{CF} به عنوان خانواده‌ی زبان‌های مستقل از متن و L_{CS} به عنوان خانواده‌ی زبان‌های حساس

به متن، داریم:

$$L_{CF} \subseteq DTIME(n^3)$$
$$L_{CF} \subseteq NTIME(n)$$

در خانواده‌ی زبان‌های حساس به متن، تجزیه با جستجوی جامع نیز ممکن است.
چون در هر قدم تعداد محدودی از قواعد قابل اعمال است، هر رشته به طول n می‌تواند در زمان n^M پویش شود (M وابسته به گرامر است). البته **نمی‌توان** حد بالایی بر روی M قایل شد و گفت $L_{CS} \in DTIME(n^M)$.

▲

هر چه $T(n)$ افزایش یابد، مقدار بیشتری از خانواده‌های زبان‌های منظم، مستقل از متن و حساس به متن پوشش داده می‌شود.

ارتباط میان سلسله‌مراتب چامسکی و کلاس‌های پیچیدگی، چندان واضح نیست.

۴-۱۴ کلاس‌های پیچیدگی P و NP

کلاس‌های پیچیدگی $DTIME(n)$ و $DTIME(n^2)$ تفاوت اساسی ندارند، زیرا به مدل ماشین تورینگ مورد استفاده وابسته‌اند (تعداد نوارها و ...) و به وضوح مشخص نیست که کدام‌یک برای کامپیوتر واقعی مناسب‌تر است.
این بحث به بررسی کلاس پیچیدگی **P** منجر می‌شود:

$$\mathbf{P} = \bigcup_{i \geq 1} DTIME(n^i)$$

این کلاس شامل تمام زبان‌هایی است که با یک ماشین تورینگ قطعی در زمان چندجمله‌ای (بدون توجه به درجه‌ی چندجمله‌ای) پذیرفته می‌شوند. پس

$$L_{REG} \subset \mathbf{P}, \qquad L_{CF} \subset \mathbf{P}$$

از آنجا که تمایز بین کلاس‌های پیچیدگی قطعی و غیرقطعی اساسی است، کلاس پیچیدگی

$$\mathbf{NP} = \bigcup_{i \geq 1} NTIME(n^i)$$

را معرفی می‌کنیم. بدیهی است که

$$\mathbf{P} \subseteq \mathbf{NP}$$

اما آنچه شناخته نشده است، این است که آیا این رابطه‌ی شمول، محض است یا خیر.

◄ **نکته** هر مسئله‌ای که در **P** باشد، اصطلاحاً رام‌شدنی (tractable) خوانده می‌شود. **(تز کوک - کرپ)**

۱۴-۵ مسائل NP: چند نمونه

مسائل **NP** آنهایی هستند که با یک الگوریتم غیرقطعی در زمان چندجمله‌ای حل می‌شوند.

- مسئله‌ی ارضاپذیری
- مسئله‌ی مسیرهای همیلتونی در یک گراف
- مسئله‌ی Clique در گراف

مسائل **NP** متعددی وجود دارند که برخی از آنها شبیه مثال‌های فوق هستند و برخی دیگر کاملاً متفاوت هستند، اما همگی در دو نکته مشترک هستند:

۱) همه‌ی مسائل **NP** دارای راه‌حل‌های ساده‌ی غیرقطعی هستند.

۲) همه‌ی این مسائل دارای راه‌حل‌هایی با پیچیدگی زمانی نمایی هستند، اما شناخته نشده است که آیا رام‌شدنی هستند یا خیر (یعنی الگوریتم قطعی چندجمله‌ای دارند یا خیر).

۱۴-۶ کاهش زمان-چندجمله‌ای

تعریف زبان L_1 کاهش‌پذیر چندجمله‌ای *(polynomial-time reducible)* به زبان L_2 خوانده می‌شود اگر یک ماشین تورینگ قطعی موجود باشد که هر w_1 از الفبای L_1 بتواند در زمان چندجمله‌ای به w_2 در الفبای L_2 تبدیل شود، به طوری که

$$w_1 \in L_1 \quad \Leftrightarrow \quad w_2 \in L_2$$

از این تعریف نتیجه می‌شود که اگر L_1 در زمان چندجمله‌ای کاهش پذیر به L_2 باشد و $L_2 \in \mathbf{P}$، آنگاه $L_1 \in \mathbf{P}$.
به طور مشابه اگر $L_2 \in \mathbf{NP}$ آنگاه $L_1 \in \mathbf{NP}$.

۱۴-۷ NP-کامل بودن و مسائل باز

تعریف زبان L، **NP**-*complete* نام دارد، اگر $L \in \mathbf{NP}$ باشد و هر $L' \in \mathbf{NP}$ در زمان چندجمله‌ای کاهش‌پذیر به L باشد.

از این تعریف به دست می‌آید که اگر یک L_1 موجود باشد که $L_1 \in \mathbf{NP}-\text{complete}$ و در زمان چندجمله‌ای کاهش‌پذیر به L_2 باشد، آنگاه L_2 هم **NP**-complete است.

از این تعریف نتیجه می‌شود که اگر بتوانیم یک الگوریتم زمان ـ چندجمله‌ای برای یک زبان **NP**-complete بیابیم، آن‌گاه هر زبان متعلق به **NP** همچنین در **P** خواهد بود و در این صورت $\mathbf{NP} = \mathbf{P}$.

مثال

مسئله‌ی ارضاپذیری می‌تواند به عنوان مسئله‌ی یک زبان دیده شود. نمونه‌های مشخص رشته‌ها را به گونه‌ای کد می‌کنیم که یک رشته در صورتی پذیرفته شود که آن عبارت ارضاپذیر باشد. این مسئله **NP**-complete است.

قضیه **قضیه‌ی کوک.** مسئله‌ی ارضاپذیری **NP**-*complete* است.

علاوه بر مسئله‌ی ارضاپذیری، تعداد زیادی مسئله‌ی **NP**-complete وجود دارد. برای هر یک از این مسائل می‌توان الگوریتم‌های زمان ـ نمایی یافت، اما برای هیچ یک از آنها کسی الگوریتم زمان ـ چندجمله‌ای پیدا نکرده است. این شکست‌ها ما را به این اعتقاد رسانده است که احتمالاً $\mathbf{P} \neq \mathbf{NP}$، اما تا وقتی که کسی یک زبان واقعی در **NP** ایجاد کند که در **P** نباشد یا تا وقتی که کسی ثابت نکند هیچ زبانی با این شرط وجود ندارد، این مسئله باز می‌ماند.

مراجع

[1] P. Linz, **An Introduction to Formal Languages and Automata**, 5th Ed., Jones and Bartletts, 2012.

[2] M. Sipser, **Introduction to the Theory of Computation**, 3rd Ed., Cengage Learning, 2013.