به نام خدا

نظریهٔ زبان‌ها و ماشین‌ها

آرش شفیعی

پیچیدگی محاسباتی

پیچیدگی

- در مبحث محاسبه‌پذیری تصمیم‌پذیری مسائل محاسباتی را بررسی کردیم. به عبارت دیگر بررسی کردیم آیا برای یک مسأله الگوریتمی وجود دارد یا خیر.
- حتی اگر مسأله‌ای تصمیم‌پذیر باشد، ممکن است الگوریتمی که برای آن وجود دارد نیاز به زمان و حافظه‌ای بسیار زیاد داشته و الگوریتم در عمل غیرقابل استفاده باشد.
- در مبحث پیچیدگی بررسی می‌کنیم که محاسبات در عمل نیازمند چه میزان زمان و حافظه هستند. به عبارت دیگر بررسی می‌کنیم در عمل محاسبات چقدر کارآمد هستند.
- کارآمدی محاسبات را با توجه به میزان زمان و حافظهٔ مورد نیاز آنها توسط پیچیدگی زمانی و پیچیدگی حافظه[1] می‌سنجیم. در اینجا تنها در مورد پیچیدگی زمانی صحبت خواهیم کرد.

[1] time and space complexity

پیچیدگی

- برای محاسبهٔ پیچیدگی زمانی یک مسأله چند نکته را باید در نظر بگیریم: (۱) مدل محاسباتی که در آن محاسبه انجام می‌شود ماشین تورینگ است. (۲) اندازهٔ مسأله را با n نشان می‌دهیم. (۳) معمولا می‌خواهیم کارآمدی یک الگوریتم را به ازای n های بسیار بزرگ بسنجیم.
- می‌توانیم فرض کنیم که ماشین تورینگ یک حرکت در واحد زمان انجام می‌دهد، بنابراین می‌خواهیم محاسبه کنیم که برای انجام یک محاسبات معین برای مسأله‌ای با اندازهٔ n ماشین تورینگ چند حرکت انجام می‌دهد و چه اتفاقی می‌افتد وقتی n به سمت اعداد بسیار بزرگ میل می‌کند.
- پیچیدگی زمانی یک تابع از n است، بنابراین می‌گوییم پیچیدگی مسأله $T(n)$ است یا به عبارت دیگر برای مسأله با اندازهٔ n ماشین تورینگ $T(n)$ حرکت انجام می‌دهد.

پیچیدگی

- فرض کنید M یک ماشین تورینگ قطعی باشد که بر روی همهٔ ورودی‌ها متوقف می‌شود. پیچیدگی زمانی M تابعی است به صورت $f : \mathbb{N} \rightarrow \mathbb{N}$، به طوری که $f(n)$ حداکثر تعداد گام‌هایی است که M برای توقف بر روی یک ورودی با اندازهٔ n نیاز دارد.
- می‌گوییم زمان اجرای ماشین M (یا الگوریتم M) برای ورودی‌های با اندازهٔ n برابر است با $f(n)$.

پیچیدگی

- معمولا برای اندازه‌گیری زمان اجرای الگوریتم‌ها از تحلیل مجانبی [1] استفاده می‌کنیم. تحلیل مجانبی (تحلیل حدی) روشی برای توصیف حدی توابع است.
- در تحلیل مجانبی الگوریتم‌ها، می‌خواهیم زمان اجرای الگوریتم را به ازای مقادیر بسیار بزرگ n بدانیم.
- فرض کنید f و g دو تابع باشند. می‌گوییم $f(n) = O(g(n))$ اگر عدد صحیح c و $n_۰$ وجود داشته باشند به طوری که به ازای $n > n_۰$ داشته باشیم: $f(n) \leq cg(n)$.
- وقتی $f(n) = O(g(n))$ می‌گوییم $g(n)$ کران بالای مجانبی $f(n)$ است.
- برای مثال اگر داشته باشیم $f(n) = ۳n^۲ + ۲n + ۱$ آنگاه $f(n) = O(n^۲)$.

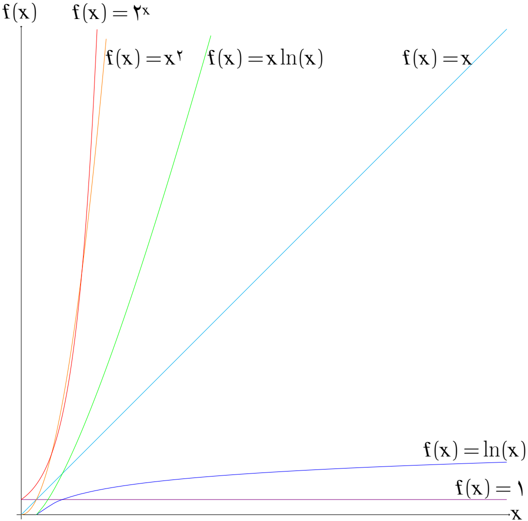
[1] asymptotic analysis

پیچیدگی

- فرض کنید $t : \mathbb{N} \rightarrow \mathbb{R}^+$ یک تابع باشد. کلاس پیچیدگی زمانی [1] $\mathrm{TIME}(t(n))$ مجموعهٔ همهٔ زبان‌هایی است که در زمان $O(t(n))$ قابل محاسبه هستند.

[1] time complexity class

مقایسه رشد توابع



مقایسه رشد توابع پیچیدگی

اگر هر گام فقط یک میکروثانیه زمان ببرد:

اندازهٔ n	۲۰	۴۰	۶۰
تابع پیچیدگی f(n)			
n	۰/۰۰۰۰۲ ثانیه	۰/۰۰۰۰۴ ثانیه	۰/۰۰۰۰۶ ثانیه
$n^۲$	۰/۰۰۰۴ ثانیه	۰/۰۰۱۶ ثانیه	۰/۰۰۳۶ ثانیه
$n^۳$	۰/۰۰۸ ثانیه	۰/۰۶۴ ثانیه	۰/۲۱۶ ثانیه
$n^۵$	۳/۲ ثانیه	۱/۷ دقیقه	۱۳ دقیقه
۲^n	۱ ثانیه	۱۲/۷ روز	۳۶۶ قرن
۳^n	۵۸ دقیقه	۳۸۵۵ قرن	$۱/۳ \times ۱۰^{۱۳}$ قرن

پیچیدگی

- از مقایسهٔ رشد توابع می‌توانیم نتیجه بگیریم رشد چندجمله‌ای برای پیچیدگی یک الگوریتم مطلوب و رشد نمایی بسیار نامطلوب است، چراکه حتی برای اندازه‌های بسیار کوچک (برای مثال ۶۰)، الگوریتمی با رشد نمایی به چندین قرن زمان نیاز دارد.

کلاس پی

- کلاس پی [1] رده‌ای از زبان‌ها است که در زمان چندجمله‌ای توسط ماشین تورینگ قطعی تصمیم‌پذیر هستند.
- به عبارت دیگر $P = \bigcup_k TIME(n^k)$.
- کلاس پی کلاس مسائلی است که در عمل توسط کامپیوترها قابل حل شدن هستند.
- وقتی مسأله‌ای در کلاس پی باشد، آن مسأله در زمان چندجمله‌ای n^k به ازای عدد ثابت k قابل حل شدن است.

[1] class P

کلاس پی

- مسألهٔ پیدا کردن یک مسیر از یک رأس به رأس دیگر در یک گراف جهت‌دار در کلاس پی است.
- $\text{PATH} = \{\langle G, s, t\rangle :$ یک گراف جهت‌دار است و یک مسیر از s به t وجود دارد $G\}$
- $\text{PATH} \in P$ زیرا می‌توان از الگوریتم جستجوی سطح-اول [1] برای حل آن استفاده کرد که پیچیدگی آن از مرتبه $O(|V| + |E|)$ است. می‌توانیم اندازهٔ گراف را مجموع تعداد رأس‌ها و یال‌ها در نظر بگیریم، و بنابراین پیچیدگی الگوریتم برابر است با $O(n)$.

[1] Breadth-First Search (BFS)

کلاس پی

- آیا الگوریتمی در زمان چندجمله‌ای وجود دارد که به ازای دو عدد داده شده تعیین کند آیا آن دو عدد نسبت به هم اول‌اند یا خیر؟
- به عبارت دیگر زبان $\text{RELPRIME} = \{\langle x, y \rangle :$ x و y نسبت به هم اول‌اند$\}$ را در نظر بگیرید. آیا $\text{RELPRIME} \in P$ ؟
- الگوریتم اقلیدس، الگوریتمی است که به ازای دو عدد x و y در زمان $\log b$ بزرگترین مقسوم علیه مشترک دو عدد را می‌یابد، به طوری که $b = \min(x, y)$. بنابراین این زبان در کلاس پی است.

کلاس پی

- چند زبان (مسألهٔ) دیگر در کلاس پی:
- $\text{CONNECTED} = \{\langle G \rangle :$ یک گراف همبند است $G\}$
- $\text{EULERIAN\_CIRCUIT} = \{\langle G \rangle :$ یک گراف است که شامل دور اویلری است $G\}$
  دور اویلری یک دور ساده است که در آن هیچ یالی تکرار نشده است.
- $\text{PRIMES} = \{\langle x \rangle :$ یک عدد صحیح اول است $x\}$
  تا سال ۲۰۰۲ هیچ الگوریتم چندجمله‌ای برای این مسأله وجود نداشت، تا اینکه الگوریتم AKS در این سال توسط سه دانشمند علوم کامپیوتر در هند (آگراوال، کایال، و ساکسنا) ابداع شد.

کلاس ان‌پی

- برای برخی از مسأله‌ها هیچ الگوریتمی در زمان چند جمله‌ای یافت نشده است. یک احتمال این است که چنین الگوریتمی وجود دارد ولی هنوز کسی آن را نیافته است. احتمال دیگر این است که چنین الگوریتمی وجود ندارد و مسأله در زمان چندجمله‌ای قابل حل نیست.

- مسأله مسیر همیلتونی [1] را در نظر بگیرید. این مسأله می‌پرسد به ازای گراف داده شدهٔ G آیا گراف دارای یک مسیر همیلتونی از رأس s به رأس t است؟

- $\mathrm{HAMPATH} = \{\langle G, s, t\rangle :$ دارای یک مسیر همیلتونی از رأس s به رأس t است G$\}$

- مسیر همیلتونی مسیری است که از هر رأس گراف دقیقا یک بار عبور می‌کند.

- گرچه برای مسأله مسیر همیلتونی الگوریتمی در زمان چندجمله‌ای وجود ندارد، ولی به ازای یک مسیر داده شده می‌توانیم در زمان چندجمله‌ای بررسی کنیم آیا مسیر همیلتونی است یا خیر. پس بررسی (راستی آزمایی) [2] جواب مسألهٔ همیلتونی آسان‌تر از پیدا کردن جواب آن است.

---

[1] Hamiltonian path

[2] verifying

کلاس ان‌پی

- فرض کنید الگوریتم A ورودی x را دریافت می‌کند و تصمیم می‌گیرد که x عضو A است و سپس خروجی y را تولید می‌کند. یک تصدیق‌کنند [1] برای الگوریتم A الگوریتمی است که x و y را به عنوان ورودی دریافت می‌کند و بررسی می‌کند آیا A با ورودی x خروجی y را تولید می‌کند یا خیر.
- یک تصدیق کننده را تصدیق‌کنندهٔ چندجمله‌ای [2] می‌نامیم اگر به ازای ورودی x و y در زمان چندجمله‌ای نسبت به طول x تعیین کند آیا y جواب x است یا خیر.
- یک زبان را تصدیق‌پذیر چندجمله‌ای [3] می‌نامیم اگر یک تصدیق‌کنندهٔ چندجمله‌ای داشته باشد.
- برای مثال در مسألهٔ مسیر همیلتونی ورودی تصدیق کننده، $\langle G, s, t \rangle$ است و یک مسیر y . الگوریتم تصدیق کننده باید تصمیم بگیرد آیا y یک مسیر همیلتونی در گراف G از s به t است یا خیر.

[1] verifier
[2] polynomial time verifier
[3] polynomially verifiable

کلاس ان‌پی

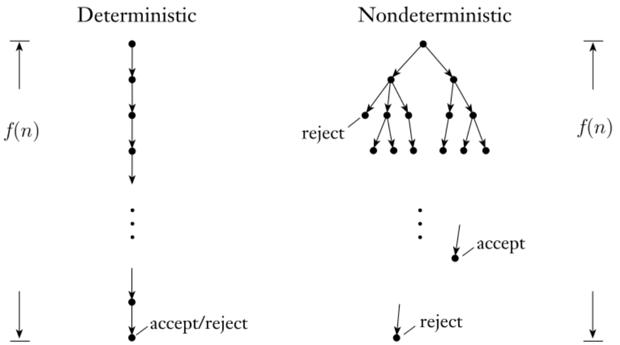
- کلاس ان‌پی [1] مجموعه‌ای است شامل همهٔ زبان‌های تصدیق‌شوندهٔ چندجمله‌ای.
- این دسته از زبان‌ها را بدین دلیل ان‌پی می‌نامیم که توسط یک ماشین تورینگ غیرقطعی [2] در زمان چندجمله‌ای [3] می‌توان آنها را تصمیم گرفت.
- فرض کنید یک ماشین تورینگ غیرقطعی در یک گام همهٔ جواب‌ها را حدس بزند، آنگاه در زمان چندجمله‌ای می‌توان جواب را بررسی کرد.

---
[1] class NP
[2] Nondeterministic Turing machine
[3] Polynomial time

کلاس ان‌پی

- دقت کنید که پیچیدگی زمانی برای ماشین غیرقطعی از شروع آغاز به کار ماشین است تا وقتی که همهٔ کپی‌های ماشین در همهٔ مسیرها به پایان برسند.

- در شکل زیر پیچیدگی زمانی برای هر دو ماشین قطعی و غیرقطعی f(n) است. گرچه ماشین غیرقطعی محاسبات بیشتری انجام می‌دهد اما در زمان f(n) به پایان می‌رسد.



کلاس ان‌پی

- کلاس پیچیدگی زمانی غیرقطعی [1] $\mathrm{NTIME}(t(n))$ مجموعهٔ همهٔ زبان‌هایی است که در زمان $O(t(n))$ توسط یک ماشین تورینگ غیرقطعی تصمیم‌پذیر باشند.

- بنابراین تعریف می‌کنیم: $\mathrm{NP} = \bigcup_k \mathrm{NTIME}(n^k)$

- برای اثبات ان‌پی بودن یک مسأله باید نشان دهیم که یک الگوریتم تصدیق‌کننده چندجمله‌ای برای آن وجود دارد.

- برای مثال مسأله پیدا کردن کلیک در یک گراف در کلاس ان‌پی است. یک کلیک مجموعه‌ای از رئوس در یک گراف است که هر جفت از آنها توسط یک یال به یکدیگر متصل هستند.
$\mathrm{CLIQUE} = \{\langle G, k\rangle :$ یک گراف بی‌جهت است و شامل یک کلیک با k رأس است $\}$

[1] nondeterministic time complexity class

مسألهٔ پی در مقابل ان‌پی

- مسائل پی مسائلی هستند که برای تصمیم‌گیری آنها یک الگوریتم در زمان چندجمله‌ای وجود دارد.
- مسائل ان‌پی مسائلی هستند که برای تصدیق جواب آنها یک الگوریتم در زمان چندجمله‌ای وجود دارد.
- برای مسائل ان‌پی الگوریتم چندجمله‌ای تاکنون پیدا نشده است و هیچ‌کس نیز اثبات نکرده است که برای آنها الگوریتم چندجمله‌ای هرگز وجود نخواهد داشت.
- مسأله پی در مقابل ان‌پی [1] می‌پرسد آیا کلاس پی و ان‌پی برابر هستند یا خیر؟ به عبارت دیگر آیا $\mathrm{P} \stackrel{?}{=} \mathrm{NP}$ ؟

[1] P versus NP problem

مسألهٔ پی در مقابل ان‌پی

- آیا برای مسائل ان‌پی، که برای آنها تصدیق‌کنندهٔ چندجمله‌ای وجود دارد، تصمیم‌گیرندهٔ چندجمله‌ای نیز پیدا خواهد شد؟
- این مسأله یکی از هفت مسأله جایزهٔ هزاره [1] است.
- در سال ۲۰۱۰ یکی از مسائل جایزهٔ هزاره به نام مسألهٔ حدس پوانکاره [2] توسط ریاضیدان روسی به نام گریگوری پرلمان [3] حل شده است.

[1] Millennium prize problems
[2] Poincaré conjecture
[3] Grigori Perelman

ان‌پی کامل

- در دههٔ ۱۹۷۰ استیون کوک [1] و لئونید لوین [2] که بر روی مسألهٔ پی در مقابل ان‌پی کار می‌کردند، متوجه شدند تعدادی از مسائل ان‌پی قابل تبدیل کردن به یکدیگرند، چنانچه اگر الگوریتمی چندجمله‌ای برای یکی از آنها پیدا شود، بقیهٔ آنها نیز توسط یک الگوریتم چندجمله‌ای حل خواهند شد. این دسته از مسائل ان‌پی‌کامل [3] نامیده شدند.

[1] Stephen Cook
[2] Leonid Levin
[3] NP-complete problems

ان‌پی کامل

- فرض کنید تابع f تابعی است از جملات زبان A به جملات زبان B به طوری که $w \in A$ اگر و تنها اگر $f(w) \in B$.

- زبان A کاهش‌پذیر در زمان چندجمله‌ای به زبان B است، اگر الگوریتمی در زمان چندجمله‌ای برای محاسبه تابع f وجود داشته باشد.

- زبان B ان‌پی‌کامل است، اگر (۱) در کلاس ان‌پی باشد و (۲) همهٔ زبان‌های A در ان‌پی در زمان چندجمله‌ای قابل کاهش به B باشند.

- اولین مسأله‌ای که اثبات شد ان‌پی‌کامل است، مسألهٔ صدق‌پذیری [1] بود.

---

[1] satisfiability problem (SAT)

مسألهٔ صدق‌پذیری

- یک متغیر بولی متغیری است که مقدار آن درست [1] یا نادرست [2] باشد. می‌توانیم مقدار درست را با ۱ و مقدار نادرست را با ۰ نشان دهیم.
- با استفاده از متغیرهای بولی [3] و عملگرهای منطقی مانند عطف و فصل و نقیض [4] می‌توانیم یک عبارت منطقی [5] بسازیم [6].

[1] true
[2] false
[3] Boolean variables
[4] Boolean operations
[5] conjunction (AND)، disjunction (OR)، negation (NOT)
[6] Boolean expression or Boolean formula

مسألهٔ صدق‌پذیری

- یک عبارت منطقی، در فرم نرمال عطفی [1] است اگر عطف چندین عبارت فصلی به صورت $e = (p_{1,1} \vee \cdots \vee p_{1,m_1}) \wedge (p_{2,1} \vee \cdots \vee p_{2,m_2}) \wedge \cdots \wedge (p_{n,1} \vee p_{n,2} \vee \cdots \vee p_{n,m_n})$ باشد.

- مسألهٔ صدق‌پذیری بدین صورت بیان می‌شود: برای یک عبارت منطقی e در فرم نرمال عطفی، تعیین کنید آیا مقادیری از متغیرها وجود دارند به طوری که به ازای آن مقادیر، مقدار عبارت منطقی e درست باشد.

- برای مثال مقدار عبارت $e_1 = (\overline{x_1} \vee x_2) \wedge (x_1 \vee x_3)$ به ازای $x_1 = 0$، $x_2 = 1$، $x_3 = 1$ درست است، اما عبارت $e_2 = (x_1 \vee x_2) \wedge \overline{x_1} \wedge \overline{x_2}$ به ازای هیچ مقداری صدق‌پذیر نیست.

[1] conjunctive normal form (CNF)

مسألهٔ صدق‌پذیری

- مسألهٔ صدق‌پذیری بدین صورت بیان می‌شود: به ازای عبارت منطقی دادهٔ شدهٔ e در فرم نرمال عطفی، آیا e صدق‌پذیر است؟
- $\text{SAT} = \{\langle e \rangle :$ یک عبارت منطقی صدق‌پذیر در فرم نرمال عطفی است e$\}$
- می توانستیم زبان SAT را بدین صورت نیز تعریف کنیم:
- $\text{SAT} = \{\langle e \rangle :$ یک عبارت منطقی صدق‌پذیر است e$\}$
- از عبارت منطقی در فرم نرمال عطفی بدین دلیل استفاده می‌کنیم که به یک فرم ساده و استاندارد است و همهٔ عبارت‌های منطقی می‌توانند در زمان چندجمله‌ای بدین فرم تبدیل شوند.

مسائل ان‌پی‌کامل

- یک حالت خاص مسأله SAT هنگامی است که عبارت منطقی داده شده در فرمال نرمال عطفی سه‌تایی به صورت $e = (p_{۱,۱} \vee p_{۱,۲} \vee p_{۱,۳}) \wedge (p_{۲,۱} \vee p_{۲,۲} \vee p_{۲,۳}) \wedge \cdots \wedge (p_{n,۱} \vee p_{n,۲} \vee p_{n,۳})$ باشد.
- زبان ۳SAT یک حالت خاص از زبان SAT و بنابراین ان‌پی‌کامل است.
- $۳SAT = \{\langle e \rangle :$ یک عبارت منطقی صدق‌پذیر در فرم نرمال عطفی سه‌تایی است $e\}$

مسائل ان‌پی‌کامل

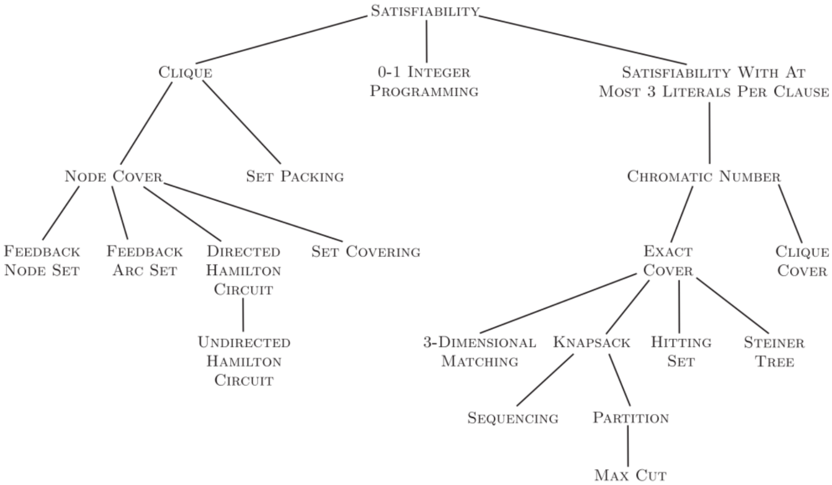
- می‌توان در زمان چندجمله‌ای مسألهٔ ۳SAT را به CLIQUE کاهش داد. بنابراین مسأله CLIQUE نیز ان‌پی‌کامل است.

- یک پوشش رأسی با k رأس مجموعه‌ای از k رأس است که همهٔ یال‌ها را در یک گراف پوشش می‌دهد.
VERTEX-COVER =
{⟨G, k⟩ : G یک گراف بدون‌جهت است که شامل یک پوشش رأسی با k رأس است}
می‌توان ۳SAT را به VERTEX-COVER نیز کاهش داد و بنابراین VERTEX-COVER نیز ان‌پی‌کامل است.

- همچنین مسألهٔ ۳SAT قابل کاهش به HAMPATH است و بنابراین مسألهٔ پیدا کردن مسیر همیلتونی نیز ان‌پی‌کامل است.

مسائل ان‌پی‌کامل

- در سال ۱۹۷۲ ریچارد کارپ بیست مسأله را کاهش داد و نشان داد ۲۱ مسألهٔ محاسباتی ان‌پی‌کامل هستند.



مثال

- اثبات کنید مسألهٔ CLIQUE ان‌پی کامل است.
- یک کلیک مجموعه‌ای از رئوس در یک گراف است که هر جفت از آنها توسط یک یال به یکدیگر متصل هستند.
  $\mathrm{CLIQUE} = \{\langle \mathrm{G}, \mathrm{k}\rangle :$ یک گراف بی‌جهت است و شامل یک کلیک با k رأس است $\}$

مثال

- اثبات کنید مسألهٔ CLIQUE ان‌پی کامل است.
- برای اثبات ان‌پی کامل بودن مسألهٔ CLIQUE ، مسأله ۳SAT را با یک الگوریتم چندجمله‌ای به آن کاهش می‌دهیم.
- فرض کنید ϕ یک عبارت منطقی در فرم نرمال عطفی سه‌تایی به صورت $\phi = (a_1 \vee b_1 \vee c_1) \wedge (a_2 \vee b_2 \vee c_2) \wedge \cdots \wedge (a_k \vee b_k \vee c_k)$ باشد.
- الگوریتم کاهش f در زمان چندجمله‌ای، با استفاده از عبارت ϕ، رشتهٔ $\langle G, k \rangle$ را تولید می‌کند به طوری G یک گراف بی‌جهت باشد.
- به ازای هر یک از متغیرها در یک گروه $(a_i \vee b_i \vee c_i)$ یک رأس در نظر می‌گیریم. بنابراین با استفاده از عبارت منطقی ϕ یک گراف دارای ۳k رأس تولید می‌کنیم.

مثال

- دقت کنید یک عبارت منطقی بر روی تعدادی متغیر $x_{۱}, \cdots, x_n$ تعریف شده است. بنابراین متغیرها به صورت x_m یا $\overline{x_m}$ هستند.
- همچنین توجه کنید برای گراف‌هایی که دارای $۳k$ رأس نیستند، عبارت $(a_i \vee b_i \vee c_i)$ می‌تواند شامل دو یا سه متغیر تکراری باشد. به طور مثال در یک عبارت می‌توانیم داشته باشیم $(x_{۱} \vee x_{۱} \vee \overline{x_{۳}})$ و یا $(x_{۲} \vee x_{۲} \vee x_{۲})$.
- متغیر x_p در گروه i را به صورت $(x_p)_i$ نشان می‌دهیم.

مثال

- حال برای رسم یال‌ها، اولا هیچ یالی از یک رأس در یک گروه به رأسی دیگر در همان گروه متصل نمی‌کنیم.
- دوما، از هر یک از رئوس گروه i به هر یک از رئوس گروه j یال‌های زیر را متصل می‌کنیم:

۱. $(x_p)_i \bullet\!\!-\!\!\bullet (x_p)_j$ ، $(\overline{x_p})_i \bullet\!\!-\!\!\bullet (\overline{x_p})_j$

۲. $(x_p)_i \bullet\!\!-\!\!\bullet (x_q)_j$ ،$(\overline{x_p})_i \bullet\!\!-\!\!\bullet (\overline{x_q})_j$ ،$(\overline{x_p})_i \bullet\!\!-\!\!\bullet (x_q)_j$ ،$(x_p)_i \bullet\!\!-\!\!\bullet (\overline{x_q})_j$ به طوری که $p \neq q$

- پس همهٔ یال‌ها به غیر از یال‌های $(x_p)_i \bullet\!\!-\!\!\bullet (\overline{x_p})_j$ ، $(\overline{x_p})_i \bullet\!\!-\!\!\bullet (x_p)_j$ را رسم می‌کنیم.

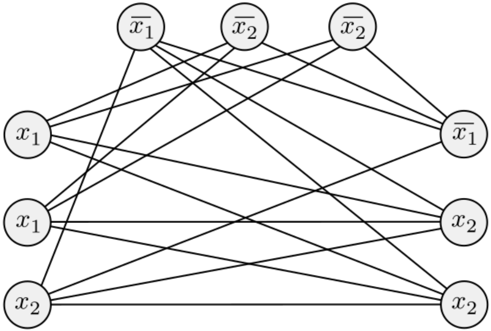
مثال

- حال توجه کنید که اگر عبارت ϕ تصدیق‌پذیر باشد، آنگاه مجموعه‌ای از k متغیر در k گروه وجود دارند که مقدار آنها درست است و این مجموعه که شامل k متغیر است، شامل هیچ دو متغیر x_m و $\overline{x_m}$ نیست، چون x_m و $\overline{x_m}$ نمی‌توانند همزمان هر دو درست باشند.
- این مجموعه که عبارت ϕ را تصدیق‌پذیر می‌کند، یک کلیک با k رأس را در گراف تولید شدهٔ G نشان می‌دهد. بدین ترتیب مسأله ۳SAT را به مسألهٔ CLIQUE کاهش دادیم. پس CLIQUE ان‌پی‌کامل است.

مثال

- عبارت $\phi = (x_۱ \vee x_۱ \vee x_۲) \wedge (\overline{x_۱} \vee \overline{x_۲} \vee \overline{x_۲}) \wedge (\overline{x_۱} \vee x_۲ \vee x_۲)$ را می‌توان به صورت گراف زیر درآورد.

- عبارت ϕ به ازای $x_۲ = ۱$ و $\overline{x_۱} = ۱$ صدق پذیر است. همچنین در گراف تولید شده کلیک $(x_۲, \overline{x_۱}, \overline{x_۱})$ وجود دارد.



مسائل ان‌پی‌سخت

- یک مسأله ان‌پی‌سخت [1] است اگر بتوان همهٔ مسائل ان‌پی را به آن مسأله کاهش داد، گرچه آن مسأله خود در ان‌پی نباشد.
- بنابراین ممکن است برای یک مسألهٔ ان‌پی‌سخت هیچ تصدیق‌کننده‌ای در زمان چندجمله‌ای وجود نداشته باشد.

---
[1] NP-hard

مسائل بهینه‌سازی

- تاکنون تنها در مورد مسائل تصمیم‌گیری [1] صحبت کردیم. پاسخ مسائل تصمیم‌گیری بلی یا خیر است. به عبارت دیگر می‌پرسیم آیا الگوریتمی وجود دارد که تعیین کند یک ورودی متعلق به یک زبان است یا خیر. الگوریتم‌های تصمیم‌گیری را از نظر درجهٔ سختی بر اساس زمان مورد نیاز برای تصمیم‌گیری دسته‌بندی کردیم.

- دسته‌ای دیگر از مسائل به نام مسائل بهینه‌سازی [2] وجود دارند که در اینگونه مسائل به دنبال بهترین یا بهینه‌ترین پاسخ از بین مجموعه‌ای از پاسخ‌ها می‌گردیم.

- برای مثال مسألهٔ VERTEX-COVER یک مسألهٔ تصمیم‌گیری است که در آن می‌پرسیم آیا به ازای یک گراف داده شدهٔ G یک مجموعه از k رأس وجود دارد که همهٔ یال‌ها را پوشش دهد یا خیر. در مسألهٔ MIN-VERTEX-COVER که یک مسألهٔ بهینه‌سازی است به دنبال کوچک‌ترین مجموعه از رئوس می‌گردیم که همهٔ یال‌ها را پوشش دهند.

[1] decision problem
[2] optimization problems

مسائل بهینه‌سازی

- هر مسألهٔ بهینه‌سازی یک مسألهٔ تصمیم‌گیری متناظر آن دارد که از لحاظ درجهٔ سختی با آن در یک کلاس قرار دارند.
- در مسائل تصمیم‌گیری می‌پرسیم به ازای یک مقدار k برای یک مسأله پاسخی وجود دارد یا خیر. در مسائل بهینه‌سازی می‌پرسیم به ازای همهٔ مقادیر k بهینه‌ترین (کوچک‌ترین، بزرگ‌ترین، کوتاهترین، بلندترین، ...) مقدار چیست؟
- برای مثال در مسألهٔ تصمیم‌گیری رنگ‌آمیزی گراف می‌پرسیم آیا رئوس یک گراف با ۴ رنگ قابل رنگ‌آمیزی هستند به طوری که هیچ دو رأس مجاوری هم‌رنگ نباشند. در مسألهٔ بهینه‌سازی رنگ‌آمیزی گراف می‌پرسیم کمترین تعداد رنگ‌هایی که با آن می‌توان یک گراف را رنگ‌آمیزی کرد چیست؟
- از آنجایی که برای حل یک مسألهٔ بهینه‌سازی ممکن است به زمانی از مرتبه نمایی احتیاج داشته باشیم، لذا معمولا در اینگونه مسائل به دنبال یک پاسخ تقریبی توسط یک الگوریتم تقریبی [1] می‌گردیم.

---
[1] approximation algorithm