به نام خدا

طراحی الگوریتم‌ها

آرش شفیعی

مسئله‌های ان‌پی کامل

مسئله‌های ان‌پی کامل

- بسیاری از الگوریتم‌هایی که در مبحث طراحی الگوریتم‌ها بررسی کردیم، الگوریتم‌های زمان [1] چند جمله‌ای بودند بدین معنی که به ازای ورودی با اندازهٔ n ، زمان اجرای آنها در بدترین حالت به ازای ثابت k برابر است با $O(n^k)$.
- اما همهٔ مسئله‌ها در زمان چند جمله‌ای قابل محاسبه نیستند.
- برخی از مسئله‌ها مانند مسئلهٔ توقف [2] (توقف‌پذیری یک برنامه بر روی یک ورودی) یا مسئلهٔ دهم هیلبرت (حل‌پذیری معادلات سیاله [3])، نه تنها در زمان چند جمله‌ای قابل محاسبه نیستند، بلکه توسط هیچ الگوریتمی قابل محاسبه نیستند.
- معمولاً به مسائلی که در زمان چند جمله‌ای قابل حل هستند، مسائل آسان و به مسائلی که در زمان چند جمله‌ای قابل محاسبه نیستند، مسائل سخت [4] یا غیر قابل کنترل [5] می‌گوییم.

[1] polynomial-time algorithm
[2] halting problem
[3] diophantine equations
[4] hard problem
[5] intractable

مسئله‌های ان‌پی کامل

- در این قسمت می‌خواهیم دسته‌ای از مسائل را معرفی کنیم که به آنها مسائل ان‌پی کامل [1] می‌گوییم. پیچیدگی محاسباتی مسائل ان‌پی کامل نامعلوم است. هیچ الگوریتم چند جمله‌ای برای این دسته از مسائل تاکنون پیدا نشده است و کسی نتوانسته اثبات کند که هیچ الگوریتم چند جمله‌ای برای آنها وجود ندارد.
- مسئلهٔ پی در برابر ان‌پی [2] یکی مسائل حل نشدهٔ مهم در علوم کامپیوتر است که در مورد آن صحبت خواهیم کرد.
- برخی از مسائل محاسباتی بسیار شبیه یکدیگر هستند و با وجود شباهت ظاهری‌شان برای یکی از آنها الگوریتم چند جمله‌ای وجود دارد و دیگری در دستهٔ مسائل ان‌پی کامل است. در اینجا به برخی از این مسائل اشاره می‌کنیم.

---

[1] NP-complete problems
[2] P vs. NP

مسئله‌های ان‌پی کامل

- مسئله کوتاهترین مسیر در مقابل مسئله بلندترین مسیر : با این که این دو مسئله بسیار شبیه یکدیگرند، اما برای مسئله کوتاهترین مسیر در گراف یک الگوریتم چند جمله‌ای وجود دارد که در زمان $O(|V||E|)$ اجرا می‌شود، اما مسئله بلندترین مسیر یک مسئلهٔ ان‌پی کامل است.
- مسئلهٔ دور اویلری و دور همیلتونی : دور اویلری [1] در یک گراف دوری است که از هر یال دقیقا یک بار عبور می‌کند. این دور می‌تواند از هر رأس چندبار عبور کند. این مسئله در زمان $O(|E|)$ برای گراف $G(V, E)$ قابل حل است. دور همیلتونی [1] دوری است که از هر رأس دقیقا یک‌بار عبور می‌کند. این مسئله یک مسئله ان‌پی کامل است.

---

[1] Eulerian cycle
[1] Hamiltonian cycle

مسئله‌های ان‌پی کامل

- مسئلهٔ صدق‌پذیری ۲ تایی و صدق پذیری ۳ تایی : عبارات منطقی شامل متغیرهای منطقی هستند که مقدار آنها می‌تواند صفر یا یک باشد. این متغیرها به عنوان عملوند با تعدادی عملگر منطقی عبارات منطقی را می‌سازند. عملگرهای منطقی شامل عملگر عطف [1] ($\wedge$) ، عملگر فصل [2] ($\vee$) و عملگر نقیض [3] ($\sim$) می‌شوند.
- یک عبارت صدق پذیر [4] است اگر با انتساب مقادیر صفر و یک به متغیرهای عبارت، مقدار عبارت برابر با ۱ شود. یک عبارت به صورت فرم نرمال عطفی k تایی است اگر آن عبارت از عطف چندین عبارت تشکیل شده باشد که هر یک از آن عبارات از فصل k متغیر یا نقیض متغیر تشکیل شده باشند.
- مسئلهٔ صدق پذیری ۲ تایی در واقع تعیین صدق پذیری یک عبارت به صورت فرم نرمال ۲ تایی است. گرچه برای مسئلهٔ صدق پذیری ۲ تایی الگوریتم چندجمله‌ای وجود دارد، اما صدق‌پذیری ۳ تایی ان‌پی کامل است.

---

[1] conjunction
[2] disjunction
[3] negation
[4] safisfiable

مسئله‌های ان‌پی کامل

- سه دسته از مسائل مهم محاسباتی را در اینجا به صورت غیر رسمی تعریف می‌کنیم : مسائل پی، مسائل ان‌پی و مسائل ان‌پی کامل.

- کلاس پی [1] شامل مسائلی است که در زمان چند جمله‌ای [2] قابل حل هستند. این دسته از مسائل در زمان $O(n^k)$ به ازای ثابت k قابل حل هستند، جایی که n اندازهٔ ورودی مسئله است. بسیاری از مسائلی که تا اینجا مورد بررسی قرار دادیم در کلاس پی قرار دارند.

- کلاس ان‌پی [3] شامل مسائلی است که در زمان چند جمله‌ای قابل تصدیق [4] هستند، بدین معنی که اگر یک مقدار به عنوان جواب داده شود، در زمان چند جمله‌ای می‌توان بررسی کرد که آیا مقدار داده شده یک جواب درست است یا خیر. برای مثال اگر برای یک گراف یک دور داده شود، می‌توان در زمان چند جمله‌ای بررسی کرد آیا دور داده شده همیلتونی است یا خیر.

---

[1] class P
[2] polynomial time
[3] class NP
[4] verifiable

مسئله‌های ان‌پی کامل

- هر مسئله در کلاس پی به کلاس ان‌پی نیز متعلق دارد، زیرا اگر یک مسئله در زمان چند جمله‌ای قابل حل باشد، در زمان چند جمله تصدیق‌پذیر نیز هست. بنابراین می‌توان بگوییم $\mathrm{P} \subseteq \mathrm{NP}$.

- مسئله مشهور پی در مقابل ان‌پی [1] می‌پرسد آیا کلاس پی و ان‌پی برابرند یا خیر. به عبارت دیگر آیا مسائلی که در زمان چند جمله‌ای قابل تصدیق هستند، در زمان چند جمله‌ای قابل حل هستند یا خیر؟

- یک مسئله به دستهٔ مسائل ان‌پی کامل [2] تعلق دارد اگر عضو کلاس ان‌پی باشد و همچنین به سختی همهٔ مسائل ان‌پی است. به عبارت دیگر اگر هر یک از مسائل ان‌پی کامل در زمان چندجمله‌ای حل شوند، آنگاه همهٔ مسائل ان‌پی در زمان چندجمله‌ای حل می‌شوند.

[1] P vs NP
[2] NP-complete

مسئله‌های ان‌پی کامل

- بسیاری از دانشمندان علوم کامپیوتر بر این باورند که مسائل ان‌پی هیچ‌گاه در زمان چندجمله‌ای حل نخواهند شد.
- به عنوان یک طراح الگوریتم، اگر بتوانید ثابت کنید که یک مسئله ان‌پی کامل است، به احتمال زیاد به راحتی برای آن الگوریتم چندجمله‌ای پیدا نخواهید کرد، بنابراین بهتر است تمرکز خود را بر روی پیدا کردن یک الگوریتم تقریبی خوب بگذارید یا مسئله را برای حالت‌های خاص حل کنید.
- اگر ثابت کنید یک مسئله ان‌پی کامل است، در واقع می‌توانید سختی آن مسئله را نشان دهید و نشان دهید جستجو برای یک الگوریتم چندجمله‌ای به احتمال بسیار زیاد بی ثمر خواهد بود.

مسئله‌های ان‌پی کامل

- بسیاری از مسئله‌ها، مسائل بهینه سازی [1] هستند و هدف از حل این مسائل یافتن مقداری است که از جواب‌های دیگر بهتر (کوچکتر، بزرگتر، ...) است. برای مثال مسئله کوتاهترین مسیر، یک مسئلهٔ بهینه سازی است، زیرا در میان همهٔ مسیرهای ممکن، هدف پیدا کردن مسیری است که در طول آن از همه کمتر است.
- مسائل ان‌پی کامل مسائل بهینه سازی نیستند، بلکه مسائل تصمیم‌گیری [2] هستند. در این‌گونه مسائل جواب بله و خیر است.
- با این حال مسائل بهینه‌سازی قابل تبدیل به مسائل تصمیم‌گیری هستند. برای مثال به ازای یک گراف و دو رأس u و v و مقدار k می‌توانیم بپرسیم آیا مسیری با طول k از u به v وجود دارد یا خیر.
- پس اگر نشان دهیم یک مسئلهٔ تصمیم‌گیری ان‌پی کامل است، مسئلهٔ بهینه‌سازی متناظر با آن نیز ان‌پی کامل خواهد بود.

[1] optimization problem
[2] decision problem

مسئله‌های ان‌پی کامل

- حال می‌خواهیم نشان دهیم که یک مسئله از مسئلهٔ دیگر سخت‌تر نیست.

- معمولاً یک مسئله در حالت کلی توسط تعدادی پارامتر بیان می‌شود. وقتی پارامترهای یک مسئله را مقداردهی می‌کنیم در واقع یک نمونه [1] از مسئله را به دست آورده‌ایم.

- برای مثال مسئلهٔ کوتاهترین مسیر می‌پرسد به ازای یک گراف دلخواه و دو رأس u و v چگونه کوتاهترین مسیر را در حالت کلی پیدا کنیم. یک نمونه از مسئله درواقع یک گراف معین و دو رأس ورودی و خروجی معین است.

[1] intance

مسئله‌های ان‌پی کامل

- مسئلهٔ تصمیم‌گیری A را در نظر بگیرید که می‌خواهید آن را در زمان چندجمله‌ای حل کنید.

- فرض کنید می‌دانید چگونه مسئله B را در زمان چندجمله‌ای حل کنید.

- همچنین فرض کنید می‌دانید چگونه هر نمونه α از مسئله A رابه نمونهٔ β از مسئلهٔ B تبدیل کنید به طوری‌که :
  ۱. تبدیل نمونه α به نمونه β در زمان چندجمله‌ای انجام شود.
  ۲. جواب نمونه مسئله تصمیم‌گیری α بله باشد اگر و تنها اگر جواب نمونه مسئلهٔ تصمیم‌گیری β بله باشد.

- چنین روندی برای تبدیل مسئله‌های تصمیم‌گیری به یکدیگر را الگوریتم کاهش در زمان چندجمله‌ای [1] می‌نامیم.

- اگر B در زمان چندجمله‌ای قابل حل باشد و A در زمان چندجمله‌ای قابل کاهش به B باشد، آنگاه برای حل مسئلهٔ A در زمان چندجمله‌ای کافی است آن را در زمان چندجمله‌ای به B کاهش دهیم و سپس B را در زمان چندجمله‌ای حل کنیم.

[1] polynominal time reduction algorithm

مسئله‌های ان‌پی کامل

- الگوریتم کاهش در زمان چندجمله‌ای در واقع روشی برای حل مسئله A در زمان چندجمله‌ای است.

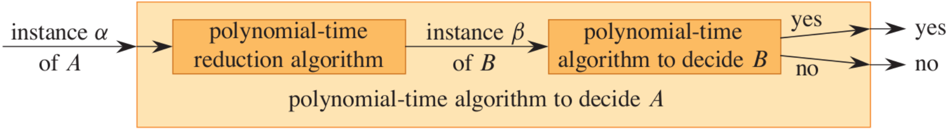
۱. به ازای نمونه α از مسئله A ، از الگوریتم کاهش در زمان چندجمله‌ای برای تبدیل آن به نمونه β از مسئله B استفاده می‌کنیم.

۲. مسئله تصمیم‌گیری B را برای نمونه β در زمان چندجمله‌ای حل می‌کنیم.

۳. جواب نمونه β را به عنوان جواب نمونه α در نظر می‌گیریم.

مسئله‌های ان‌پی کامل

- شکل زیر الگوریتم کاهش برای حل یک مسئله را نشان می‌دهد.



مسئله‌های ان‌پی کامل

- پس با استفاده از روش کاهش برای حل مسئله A از الگوریتم چندجمله‌ای مسئله B استفاده می‌کنیم.
- از همین ایده استفاده می‌کنیم برای اینکه نشان دهیم یک مسئله به سختی یک مسئلهٔ دیگر است.

مسئله‌های ان‌پی کامل

- فرض کنید ثابت شده‌است که برای مسئله A الگوریتم چندجمله‌ای وجود ندارد. حال فرض کنید یک الگوریتم کاهش در زمان چندجمله‌ای برای تبدیل یک نمونه از مسئله A به یک نمونه از مسئله B وجود دارد. می‌توانیم با استفاده از برهان خلف نشان دهیم که هیچ الگوریتم چندجمله‌ای برای B وجود ندارد.

- برای اثبات این قضیه فرض کنید برای B یک الگوریتم چندجمله‌ای وجود داشته باشد، در آن صورت می‌توانستیم با استفاده از کاهش، یک الگوریتم چندجمله‌ای برای A پیدا کنیم، اما می‌دانیم ثابت شده است که الگوریتم چندجمله‌ای برای A وجود ندارد. پس به تناقض می‌رسیم و در نتیجه فرض اولیه نادرست بوده و الگوریتم چندجمله‌ای برای B وجود ندارد.

مسئله‌های ان‌پی کامل

- برای اینکه نشان دهیم B ان‌پی کامل است از همین روند استفاده می‌کنیم. اگرچه نمی‌توانیم به طور قطعی بگوییم که هیچ الگوریتم چندجمله‌ای برای A وجود ندارد، اما می‌توانیم اثبات کنیم B ان‌پی کامل است با فرض اینکه A ان‌پی کامل است. پس برای اثبات ان‌پی کامل بودن یک مسئله باید یکی از مسائل ان‌پی‌کامل را در زمان چندجمله‌ای به آن مسئله کاهش دهیم. می‌گوییم مسئلهٔ B به سختی مسئلهٔ A است.

- اولین مسئله‌ای که ان‌پی کامل بودن آن اثبات شده است، مسئلهٔ صدق پذیری است. برای اثبات اولین مسئله اثبات شد که همهٔ مسائل ان‌پی در زمان چندجمله‌ای قابل کاهش به مسئلهٔ صدق‌پذیری هستند. بنابراین اگر مسئلهٔ صدق‌پذیری در زمان چندجمله‌ای حل شود، همهٔ مسائل ان‌پی در زمان چندجمله‌ای حل خواهند شد. به عبارت دیگر مسئلهٔ صدق‌پذیری به سختی همهٔ مسائل ان‌پی است.

- در واقع هر یک از مسائل ان‌پی‌کامل به سختی همهٔ مسائل ان‌پی است، زیرا حل یکی از آنها در زمان چندجمله‌ای منجر به حل همهٔ مسائل ان‌پی می‌شود.

الگوریتم‌های تقریبی

- بسیاری از مسائل محاسباتی کاربردی ان‌پی کامل هستند و با این حال با توجه به اهمیت زیادی که دارند نیاز داریم جوابی برای آنها پیدا کنیم گرچه پیدا کردن جواب دقیق برای اینگونه مسائل در زمان چندجمله‌ای امکان‌پذیر نیست.

- وقتی یک مسئله ان‌پی کامل است، برای حل آن سه راه پیش رو داریم : (۱) اگر ورودی نسبتاً کوچک باشد، می‌توان یک جواب بهینه در زمان نمایی به سرعت برای آن پیدا کرد. (۲) می‌توان یک حالت خاص از مسئله را در زمان چند جمله‌ای حل کرد. (۳) می‌توان یک جواب نزدیک به جواب بهینه در زمان چند جمله‌ای برای آن پیدا کرد. در بسیاری از کاربردها جواب نزدیک به جواب بهینه [1] نیز کافی است. به چنین الگوریتم‌هایی که جواب نزدیک به بهینه تولید می‌کنند، الگوریتم‌های تقریبی [2] می‌گوییم. برای بسیاری از مسائل ان‌پی کامل می‌توان یک الگوریتم تقریبی در زمان چندجمله‌ای پیدا کرد.

---
[1] near-optimal solution
[2] approximation algorithm

الگوریتم‌های تقریبی

- فرض کنید بر روی مسئلهٔ بهینه‌سازی کار می‌کنید که در آن هر یک از جواب‌های بالقوه [1] دارای یک هزینه است و می‌خواهید یک جواب نزدیک به بهینه پیدا کنید. بسته به نوع مسئله، ممکن است مسئله بیشینه سازی [2] یا کمینه سازی [3] باشد. می‌توانید یک جواب بهینه با هزینه حداکثر یا هزینه حداقل پیدا کنید.

[1] potential solution
[2] maximization
[3] minimization

الگوریتم‌های تقریبی

- می‌گوییم یک الگوریتم دارای ضرب تقریب $\rho(n)$ [1] است اگر به ازای هر ورودی با اندازهٔ n ، هزینهٔ C جواب تولید شده توسط الگوریتم نسبت به هزینهٔ C^* مربوط به جواب بهینه از مقدار $\rho(n)$ کمتر باشد. به عبارت دیگر :

$$\left\{\frac{C}{C^*}, \frac{C^*}{C}\right\} \leqslant \rho(n)$$

- اگر یک الگوریتم دارای ضریب تقریب $\rho(n)$ باشد، به آن الگوریتم تقریبی $\rho(n)$ می‌گوییم.

[1] approximation ratio

الگوریتم‌های تقریبی

- از الگوریتم‌های تقریبی $\rho(n)$ هم برای مسائل کمینه سازی و هم برای مسائل بیشینه سازی استفاده می‌شود.
- در یک مسئله بیشینه سازی، داریم $0 < C \leqslant C^*$ و بنابراین مقدار C^*/C مقدار بزرگ‌تری است که در آن هزینهٔ جواب بهینه از هزینهٔ جواب تقریبی بزرگ‌تر است.
- در یک مسئله کمینه سازی، داریم $0 < C^* \leqslant C$ و بنابراین مقدار C/C^* مقدار بزرگ‌تری است که در آن هزینهٔ جواب تقریبی از هزینهٔ جواب بهینه بزرگ‌تر است.
- با فرض اینکه همهٔ هزینه‌ها مقادیر مثبت هستند، ضریب تقریب در یک الگوریتم تقریبی هیچ‌گاه کمتر از ۱ نیست.
- بنابراین یک الگوریتم تقریبی با ضریب ۱ جوابی بهینه تولید می‌کند و هر چه ضریب تقریب الگوریتم تقریبی بیشتر باشد، جواب به دست آمده از جواب بهینه دورتر است.

الگوریتم‌های تقریبی

- برای بسیاری از مسائل، الگوریتم‌های تقریبی چند جمله‌ای با ضریب تقریب کوچک وجود دارد و برای برخی دیگر از مسائل، الگوریتم‌های تقریبی دارای ضریب تقریبی هستند که با مقدار n افزایش پیدا می‌کند.
- در برخی از الگوریتم‌های تقریبی چندجمله‌ای، هرچه الگوریتم در زمان بیشتری اجرا شود، ضریب تقریب بهتری به دست می‌آید. در چنین مسائلی می‌توان با افزایش زمان محاسبات ضریب تقریب را بهبود داد.

مسئلهٔ پوشش رأسی

- مسئله پوشش رأسی [1] یک مسئلهٔ ان‌پی کامل است.

- پوشش رأسی یک گراف بدون جهت $G = (V, E)$ زیر مجموعه‌ای از رئوس گراف $V' \subseteq V$ است به طوری‌که اگر (u, v) یک یال از گراف G باشد، آنگاه $u \in V'$ یا $v \in V'$ یا هردو. اندازه پوشش رأسی تعداد رأس‌های زیر مجموعهٔ V' است.

- در مسئلهٔ پوشش رأسی می‌خواهیم کمترین تعداد رئوس در یک گراف را پیدا کنیم که یک پوشش رأسی تشکیل می‌دهند یا به عبارت دیگر همهٔ یال‌ها را پوشش می دهند. به چنین پوشش رأسی یک پوشش رأسی بهینه [2] می‌گوییم.

- اگرچه هیچ الگوریتم چند جمله‌ای برای مسئلهٔ پوشش رأسی یافته نشده است، اما یک الگوریتم تقریبی چندجمله‌ای برای پیدا کردن جواب نزدیک به بهینه وجود دارد.

---

[1] vertex cover problem

[2] optimal vertex cover

مسئلهٔ پوشش رأسی

- الگوریتم تقریبی زیر یک گراف بدون جهت را دریافت می‌کند و یک پوشش رأسی باز می‌گرداند که اندازهٔ آن کمتر از دو برابر پوشش رأسی بهینه است.

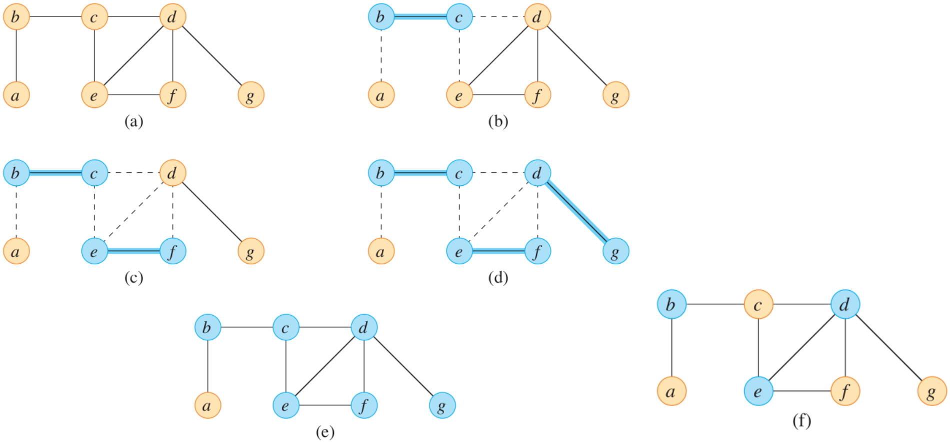
Algorithm Approx-Vertex-Cover

```
function APPROX-VERTEX-COVER(G)
C = ∅
E' = G.E
while E' ≠ ∅ do
    let (u,v) be an arbitrary edge of E'
    C = C ∪ {(u,v)}
    remove from E' edge (u,v) and every edge incident on either u or v
return C
```

- متغیر C پوشش رأسی تشکیل شده را در بر می‌گیرد. این الگوریتم در زمان $O(|V| + |E|)$ توسط نمایش گراف با لیست مجاورت اجرا می‌شود.

مسئلهٔ پوشش رأسی

- شکل زیر نشان می‌دهد این الگوریتم چگونه برروی یک گراف عمل می‌کند. در پایان این الگوریتم تقریبی ۶ رأس را برای پوشش رأسی پیدا می‌کند در صورتی که جواب بهینه برای این نمونه مسئله ۳ است.



مسئلهٔ پوشش رأسی

- قضیه : الگوریتم تقریبی پوشش رأسی یک الگوریتم چند جمله‌ای تقریبی با ضریت ۲ است.
- اثبات : مجموعهٔ C یک پوشش رأسی است زیرا الگوریتم در حلقه تا وقتی تکرار می‌شود که همهٔ یال‌های G.E با یکی از رئوس C پوشش داده شده‌اند.
- حال می‌خواهیم ثابت کنیم این الگوریتم یک الگوریتم تقریبی با ضریب ۲ است.
- فرض کنید A مجموعه‌ای از یال‌ها باشد که در خط ۴ الگوریتم انتخاب شده‌اند. برای اینکه یال‌های مجموعه A پوشش داده شوند، هر پوشش رأسی باید حداقل یکی از دو رأس هریال در A را شامل شود. هیچ دو یالی در A رأس مشترک ندارند، زیرا وقتی یک یال در خط ۴ انتخاب شد، همهٔ یال‌هایی که با آن یال رأس مشترک دارند از مجموعه E' در خط ۶ حذف می‌شوند.

مسئلهٔ پوشش رأسی

- بنابراین هیچ دویالی در A با یک رأس از C^* پوشش داده نشده‌اند و این بدین معنی است که به ازای هر رأس در C^* ، حداکثر یک یال در A وجود دارد و بنابراین داریم :

$$|C^*| \geqslant |A|$$

- هر اجرای خط ۴ یک یال را انتخاب می‌کند که هیچ‌کدام از دو رأس مجاور آن در C نیستند و بنابراین داریم :

$$|C| = 2|A|$$

- با استفاده از دو رابطهٔ به دست آمده خواهیم داشت :

$$|C| = 2|A| \leqslant 2|C^*|$$

و قضیه بدین ترتیب اثبات می‌شود.

نتیجه‌گیری

- یک الگوریتم روشی است گام‌به‌گام برای حل یک مسئلهٔ محاسباتی.
- روش‌های گوناگون برای طراحی یک الگوریتم برای یک مسئله محاسباتی وجود دارد.
- برای یک مسئله ممکن است الگوریتم‌های متفاوت با رویکردهای متفاوت وجود داشته باشد. دو معیار مهم سنجش الگوریتم‌ها زمان اجرا و میزان حافظه استفاده شده توسط آنها است. ممکن است یک الگوریتم زمان اجرای بسیار بالایی داشته باشد ولی میزان حافظهٔ مورد نیاز آن نیز بسیار بالا باشد. چنین الگوریتمی در موقعیت‌هایی کاربرد دارد که زمان اجرا مهم‌ترین معیار سنجش است. برای مثال در سیستم‌های بلادرنگ نیاز به زمان پاسخ پایین وجود دارد. الگوریتم دیگری می‌تواند زمان حافظهٔ بسیار کمی استفاده کند ولی زمان اجرای آن نیز نسبتا پایین باشد چنین الگوریتمی در موقعیتی کاربرد دارد که میزان حافظه بسیار حائز اهمیت است. برای مثال در سیستم‌های نهفته حافظه محدود است. همچنین زمان اجرا و میزان حافظه مورد نیاز یک الگوریتم ممکن است حد وسط باشد. چنین الگوریتمی وقتی استفاده می‌شود که هر دو معیار زمان و حافظه به یک اندازه اهمیت داشته باشند.

نتیجه‌گیری

- روش‌ها و رویکردهای متفاوتی را برای حل مسائل محاسباتی از جمله روش تقسیم و حل، برنامه‌ریزی پویا، حریصانه و جستجوی فضای حالت را بررسی کردیم.
- روش‌های تقسیم و حل و برنامه‌ریزی پویا و حریصانه برای حل مسئله در زمان چندجمله‌ای به کار می‌روند. اما مسئله‌های زیادی وجود دارند که هنوز الگوریتم چندجمله‌ای برای آنها دریافت نشده است و بنابراین برای حل اینگونه مسائل باید همهٔ جواب‌های احتمالی بررسی شوند تا جواب مورد نظر پیدا شود.
- چنین الگوریتم‌هایی در دستهٔ الگوریتم‌های جستجوی فضای حالت یا جستجوی ترکیبیاتی قرار می‌گیرند.

نتیجه‌گیری

- اگر فضای حالت بسیار بزرگ باشد ممکن است حتی برای مسئله‌های نسبتاً کوچک رویکرد جستجوی ترکیبیاتی در زمان معقول پاسخگو نباشد. روش پسگرد روشی است که برای بررسی فضای حالت به طور منظم استفاده می‌شود و تعدادی از حالات حذف می‌شوند. همچنین در مسائل بهینه‌سازی به منظور کاهش تعداد حالات از روش شاخه و کران برای کوچک‌کردن فضای حالت استفاده می‌کنیم.

- مسائلی وجود دارند که برای آنها الگوریتم چند جمله‌ای یافته نشده است، اما الگوریتم‌های تقریبی وجود دارد که در زمان چند جمله‌ای جواب نزدیک به جواب دقیق یا به عبارت دیگر جوابی تقریبی برای مسئله پیدا می‌کنند. در الگوریتم‌های تقریبی اثبات می‌شود جواب تقریبی یافته شده توسط الگوریتم چه میزان انحراف از جواب دقیق مسئله خواهد داشت.