

به نام خدا

پاسخ نامه آزمون درس الگوریتم های پیشرفته

دانشجو:

فاطمه فتحی جم

استاد راهنما:

دکتر علی رضوی ابراهیمی

خرداد ۱۴۰۰

۱- در معنی و مفهوم علمی الگوریتم بحث نمایید. درس الگوریتم‌های مورد مطالعه برای مهندسين نرم افزار چه کاربرد و فایده‌ای دارند؟

مفهوم الگوریتم

تعاریف متعددی برای توصیف الگوریتم ارائه شده اما در واقع مفهوم علمی الگوریتم به معنای **روش حل سیستماتیک مسئله** است. الگوریتم (Algorithm) یک توالی صریح، دقیق، بدون ابهام و قابل اجرا به لحاظ مکانیکی از دستورات اولیه است که معمولاً برای انجام کار و هدف خاصی، تعریف می‌شوند. یک الگوریتم، روال یا فرمولی برای حل یک مسئله بر مبنای انجام یک توالی از فعالیت‌ها است.

به بیانی دیگر، می‌توان گفت که الگوریتم مجموعه‌ای از عملیات مرتب و محدود است که به منظور حل مسئله خاصی دنبال می‌شوند. این یعنی زنجیره‌ای از دستورات عمل‌های دقیق وجود دارد که باید به ترتیب خاصی دنبال شوند. هدف الگوریتم حل مسئله است و این یعنی الگوریتم‌ها دارای هدف از پیش تعریف شده هستند. در واقع، هدف از نوشتن یک الگوریتم، تولید یک خروجی است.

به طور غیررسمی، الگوریتم روش محاسباتی معین است که مقدار یا مجموعه‌ای از مقادیر را به عنوان **ورودی** گرفته و مقدار یا مجموعه‌ای از مقادیر را به عنوان **خروجی** تولید می‌کند؛ بنابراین می‌توان گفت الگوریتم دنباله‌ای از مراحل محاسباتی است که ورودی را به خروجی تبدیل می‌کند (مقدمه‌ای بر الگوریتم‌ها، توماس اچ کورمن و چارلز ای لایسرسان و رونالد ریوست و کلیفورد استین).

کاربرد الگوریتم برای مهندسين نرم افزار

همان‌طور که می‌دانیم کار مهندس نرم‌افزار، **تولید نرم‌افزار** است و الگوریتم دانش پایه و اساسی برای تولید یک نرم‌افزار است؛ بنابراین یادگیری الگوریتم و روش‌های حل مسئله برای مهندسين نرم‌افزار امری ضروری است. داشتن پایه قوی از تکنیک و دانش الگوریتمی خصوصیتی است که برنامه‌نویسان واقعاً ماهر را از مبتدیان مجزا می‌کند. درست است که امروزه با تکنولوژی محاسباتی جدید می‌توان بعضی از امور را بدون اینکه مطلب زیادی راجع به الگوریتم‌ها بدانید، به انجام رساند، اما این موضوع برای مهندسين نرم‌افزار کافی نمی‌باشد؛ مهندس نرم‌افزار و برنامه‌نویسان برای تولید نرم‌افزار مؤثر و کارآمد باید دانش زمینه‌ای خوبی در الگوریتم داشته باشند.

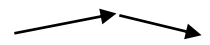
۲- نحوه عمل نیم پاک کننده در شبکه های مرتب ساز را شرح دهید.

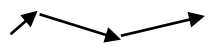
مرتب ساز ادغامی بایتونیک، الگوریتمی موازی برای مرتب سازی است که از آن برای ساخت شبکه های مرتب ساز نیز استفاده می شود. یک مرتب ساز بایتونیک از چندین مرحله تشکیل شده که به هر مرحله نیم پاک کننده گفته می شود. هر نیم پاک کننده یک شبکه مقایسه به عمق ۱ است که در آن خط ورودی i با خط $i + n/2$ برای $i=1,2,\dots,n/2$ مقایسه می شود.

شبکه مقایسه گر چیست؟ شبکه های مرتب ساز در واقع شبکه های مقایسه ای هستند که همیشه ورودی های خود را مرتب می کنند، بنابراین منطقی است که بحث ما با شبکه های مقایسه و ویژگی های آنها آغاز شود. یک شبکه مقایسه فقط از سیم ها و مقایسه کننده ها تشکیل شده است.

بایتونیک چیست؟ دنباله های Bitonic (دنباله های دورفتاره) دنباله ای از اعداد را یک دنباله Bitonic می نامیم، هرگاه این دنباله ابتدا به طور یکنوا صعودی بوده و سپس به طور یکنوا نزولی باشد و یا اینکه با یک چرخش دوری یا جایگشتی (Circularly Shifted) دارای خاصیت مذکور باشد.


مثال:


1, 4, 6, 8, 3, 2


6, 9, 4, 2, 3, 5

5, 6, 9, 4, 2, 3

چرخش دوری

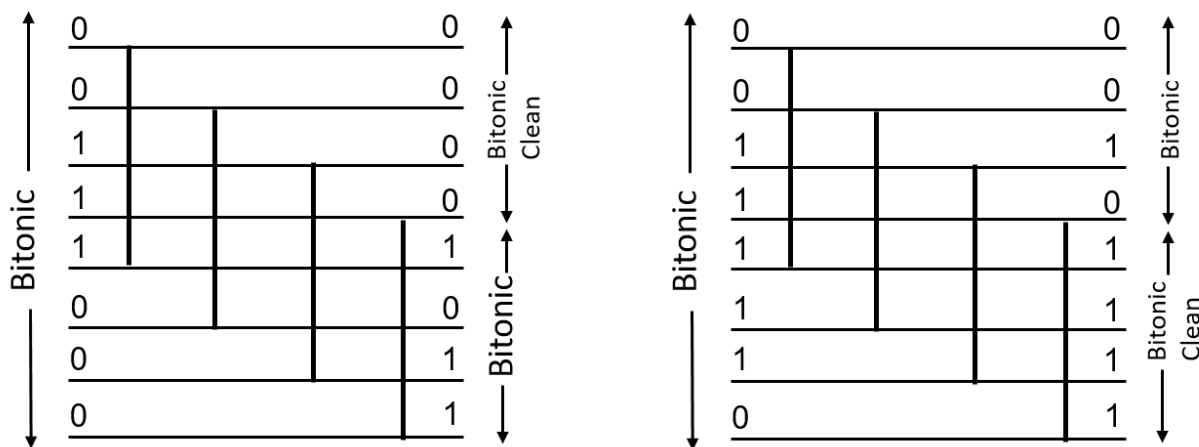

3, 5, 6, 9, 4, 2

نیم پاک کننده (Half Cleaner) که عمق آن همواره ۱ است، یک شبکه مقایسه گر با ساختار زیر است:

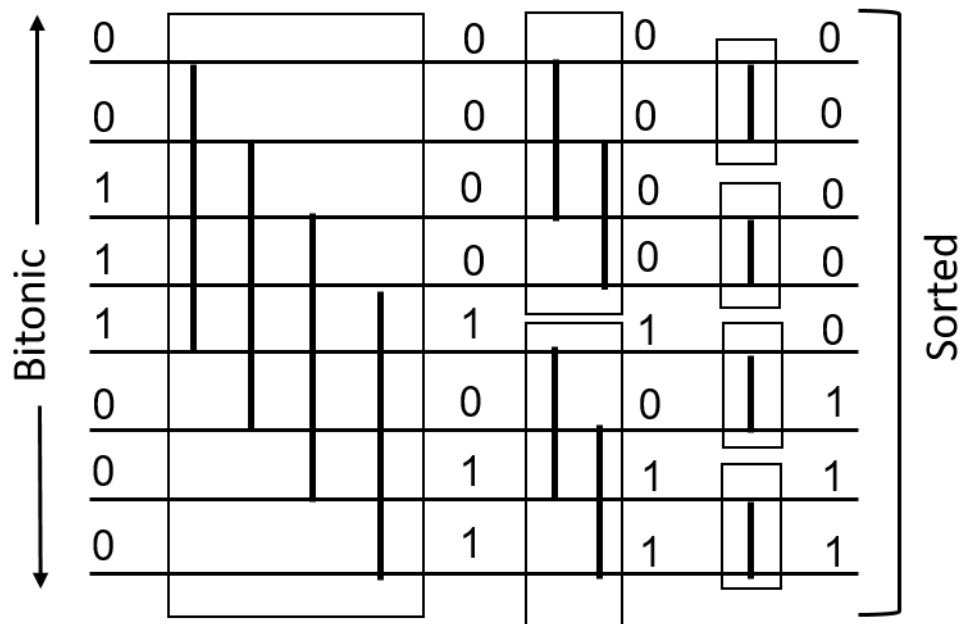
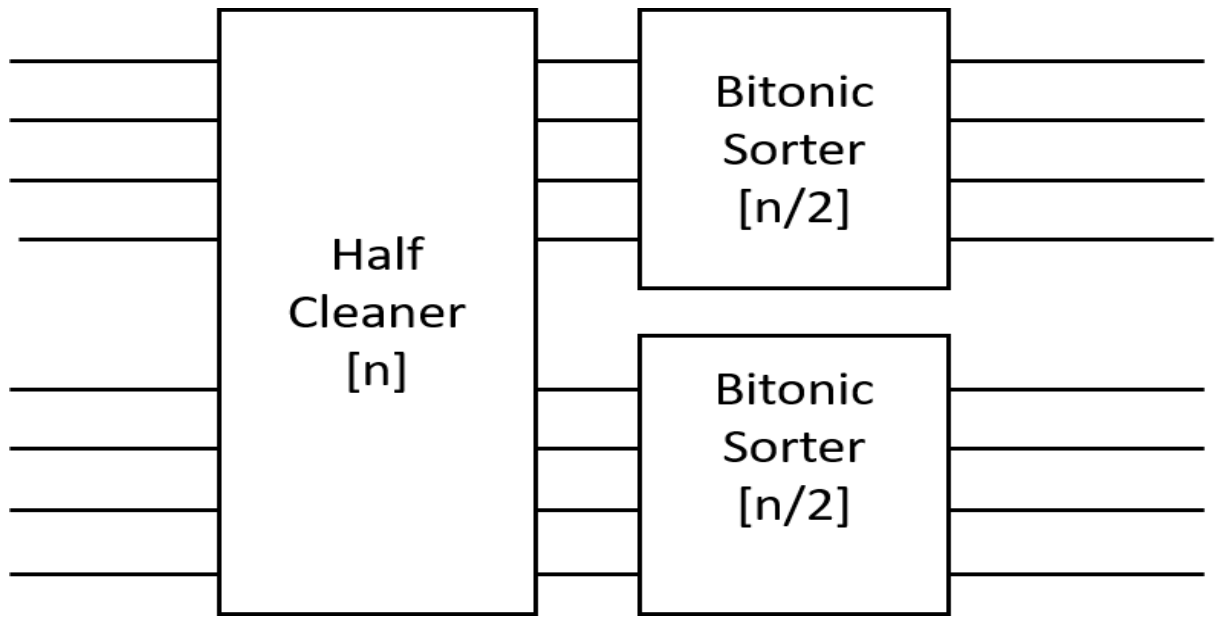
شامل n سیم و $n/2$ مقایسه‌گر است (معمولاً n توانی از ۲ است) بطوریکه هر مقایسه‌گر به این صورت عمل می‌کند:

- مقایسه‌گر اول، خط اول را به خط ۱ $n/2$ مرتبط می‌کند.
- مقایسه‌گر دوم، خط دوم را به خط ۲ $n/2$ مرتبط می‌کند.
- .
- .
- .
- مقایسه‌گر i ام، خط i ام را به خط $i+n/2$ مرتبط می‌کند.

بدین ترتیب یک نیم پاک‌کننده، یک دنباله Bitonic را به یک دنباله‌ای که نصف آن مرتب است، تبدیل می‌کند.



پس از عمل نیم پاک‌کننده همواره هر المان نیمه بالایی حداکثر به اندازه کوچک‌ترین المان نیمه پایینی خواهد بود و همواره یک نیمه مرتب شده است به این معنی که همه عناصر آن یا صفر است یا یک. به شکل زیر توجه کنید:



۱- نقش الگوریتم برای تأثیرگذاری مهندسی نرم افزار در تحقق شعار سال «تولید، پشتیبانی ها و مانع زدایی ها» چه می تواند باشد؟

همه ساله رهبر معظم انقلاب، متناسب با شرایط کلی حاکم بر کشور سال را به نامی مزین می کنند تا این مهم بتواند در سرلوحه برنامه ریزی های مسئولان و مردم قرار بگیرد. ایشان در نوروز سال ۱۴۰۰ نیز، این سال را به نام سال تولید، پشتیبانی ها و مانع زدایی ها نام گذاری کردند. در این بخش قصد داریم نقش درس مهندسی نرم افزار پیشرفته را برای تأثیرگذاری در تحقق شعار سال بررسی نماییم.

پیش از هر چیز بهتر است کمی در رابطه با معنا و مفهوم تولید که کلمه کلیدی شعار سال است، صحبت کنیم. تولید یا فرآوری، از اصطلاحات علم اقتصاد، به معنی تهیه کالا و خدمات مورد نیاز با استفاده از منابع و امکانات موجود است. تولید بر دو نوع است، تولید مداوم و تولید متناوب و فعالیت تولیدی سلسله اقداماتی است که برای تبدیل منابع به کالاهای مورد نیاز صورت می گیرد. نکته بسیار مهم اینجاست که نیاز به تولید، منجر به پیشرفت تمام رشته های مهندسی شده است. در مهندسی نرم افزار نیز تولید نرم افزار مؤثر و کارآمد می تواند تأثیر بسزایی در تحقق شعار سال داشته باشد.

همچنین می دانیم که الگوریتم، روش حل سیستماتیک مسئله است، پس اولین کاری که باید انجام شود این است که ما (مهندس نرم افزار) طرح مسئله کنیم و برای مسئله طرح شده راه حل ارائه دهیم. یعنی مشکلات و موانع موجود را با ارائه الگوریتم هایی مناسب بررسی کرده و نسبت به هزینه و کارایی یکی از الگوریتم های طراحی شده را انتخاب و به عنوان راه حل نهایی مسئله به مرحله اجرا در بیاوریم. به عنوان مثال می توان الگوریتمی طراحی و ارائه نمود که در آن به طور خاص، مسئله موانع موجود بر سر راه تولید را مورد بررسی قرار دهد.

بی شک الگوریتم و نرم افزاری که بر مبنای الگوریتم طراحی شده، ساخته می شود باید پیش از هر چیز نهاد، سازمان و یا کارخانه مذکور را به طور کامل تحلیل و واحدهای مرتبط را از نظر وجود یا عدم وجود مشکل بررسی نماید. در مرحله بعد مشکلات شناسایی شده را از نظر میزان اهمیت، میزان تخریب و سختی حل مسئله طبقه بندی کرد. سپس موانع و مشکلات طبقه بندی شده در مرحله قبل، مورد بررسی قرار داد و با بهره گیری از نظر و تخصص کارشناسان مربوطه نسبت به حل آن ها اقدام نمود.

بنابراین طراحی یک الگوریتم صحیح و تولید نرم افزار مبتنی بر آن، با تشخیص موانع موجود بر سر تولید کشور و همچنین مانع زدایی آن ها می تواند اولین قدم ها را در تحقق شعار سال بردارد.

۲- با ذکر مثال و ارائه یک الگوریتم اولیه (لزومی به اثبات درستی الگوریتم ارائه شده وجود ندارد) تلاش نمایید قدمی برای حل مشکل کارآمدی دانشجویان مهندسی نرم افزار برای مشارکت در تولید نرم افزار در دوران دانشجویی و یا پس از آن ارائه نمایید؟ (استفاده از الگوریتم های مطالعه شده در درس مورد استقبال خواهد بود)

عدم مشارکت دانشجویان در تولید نرم افزار در دوران دانشجویی از جمله مسائلی است که همواره مورد بحث بوده است و این در حالی است که تشکیل تیم های دانشجویی در رشته نرم افزار، تأثیر بسزایی هم در افزایش بار علمی دانشگاه و هم در توسعه توانایی و استعداد های دانشجویان و مهندسين نرم افزار آینده خواهد داشت. اما چگونه می توان قدمی برای حل مشکل کارآمدی دانشجویان مهندسی نرم افزار برای مشارکت در تولید نرم افزار در دوران دانشجویی و یا پس از آن برداشت؟

بدون شک الگوریتم در این جا می تواند کمک بسیاری برای پیدا کردن راه حل داشته باشد. چون همان طور که می دانیم، الگوریتم روش حل سیستماتیک مسئله است. یعنی هر زمانی که با مسئله خاصی روبه رو بودیم با طراحی و ارائه یک الگوریتم می توانیم برای حل آن مسئله چند راه حل پیدا کرده و بهینه ترین الگوریتم را برای حل مسئله انتخاب نماییم. قطعاً برای تمامی مشکلات ما الگوریتم نداریم اما الگوریتم ها موجود به ما ایده و استراتژی برای ایجاد الگوریتم های جدید از طریق گسترش و ساخت الگوریتم جدید را می دهد.

برای نوشتن یک الگوریتم باید سه عامل اصلی را شناسایی کنیم:

- ? مقادیر معلوم: اطلاعاتی که در اختیار ما قرار داده شده و باید به کمک آن ها به حل مسئله بپردازیم (داده ها)
- ? خواسته های مسئله: نتایجی که در اثر انجام محاسبات بر روی داده های مسئله حاصل می شود (مقادیر مجهول)
- ? عملیات محاسباتی: دستورات و روابط منطقی که برای رسیدن به خواسته های مسئله بر روی داده ها و مقادیر مجهول انجام می شود.

- ✓ در این مسئله مقادیر معلوم چه چیزهایی هستند؟ توانایی و استعداد یک دانشجوی مهندسی نرم افزار
 - ✓ خواسته های مسئله چه هستند؟ مشارکت در تولید نرم افزار در دوران دانشجویی و یا پس از آن
 - ✓ عملیات چه هستند؟ تحلیل و بررسی مشکلات، بهینه سازی، ایجاد فرآیند و پیش نیازها، دوام و گسترش
- زمانی که سه عامل ذکر شده را شناسایی کردیم اکنون می توانیم الگوریتم مورد نظر را طراحی کنیم. پیش از هر چیز با یک نظرسنجی آغاز می کنیم. یعنی ابتدا در قالب پرسش نامه ای از مهندسين نرم افزار در رابطه با دلیل عدم تمایل به همکاری و مشارکت در تولید نرم افزار سؤال می پرسیم. نتایج این پرسش نامه را مورد تجزیه و تحلیل و بحث و تفسیر قرار داده و مهم ترین عوامل مؤثر در میزان مشارکت دانشجویان شناسایی می شوند. بدین ترتیب ما

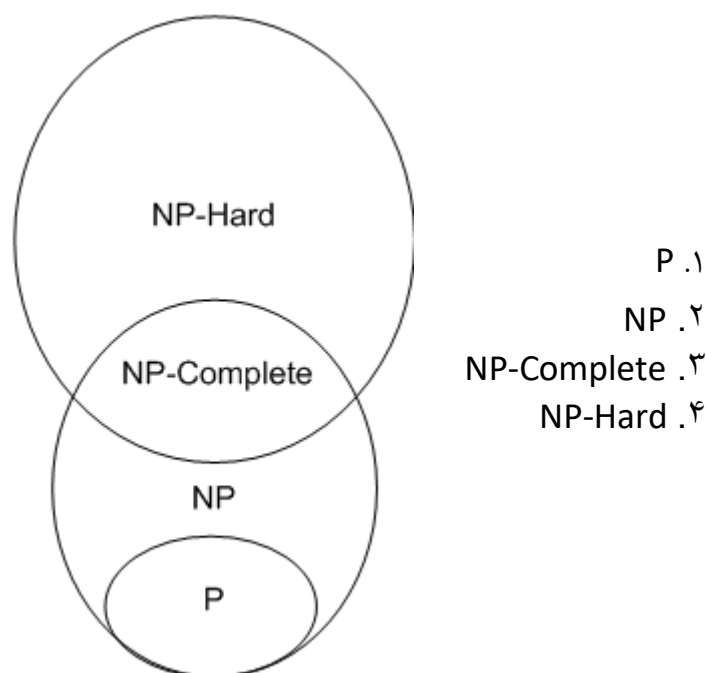
داده‌هایی خواهیم داشت که به ما می‌گویند که برای حل مشکل باید از کجا شروع کنیم. پس مشکلات شناسایی شده را طبق الگوریتم‌های اولیه موجود طبقه‌بندی کرده و موضوعاتی که بیشترین تأثیر را در عدم مشارکت دانشجویان دارند مورد بررسی قرار می‌دهیم.

اما الگوریتم موجود همچنان راه حلی برای حل مسئله ارائه نکرده است. از آن جایی که یکی از دلایل عدم مشارکت دانشجویان نرم‌افزار در تولید نرم‌افزار، نداشتن اطلاعات کافی در رابطه با دانش و توانایی‌های دیگر دانشجویان است، می‌توان بستری فراهم نمود تا دانشجویان در آن توانایی‌ها و دانش خود را در زمینه‌های مختلف ثبت نمایند (این امر نیز می‌تواند با طراحی یک آزمون و یا پرسش‌نامه دیگر پیاده‌سازی شود) و در نهایت می‌توان گروه‌هایی با توانایی و استعداد‌های مختلف تشکیل داد که هریک از اعضای گروه مطابق با توانایی‌های خود بخشی از پروژه را برای تولید نرم‌افزار برعهده بگیرند که این امر به تولید نرم‌افزار مؤثر در دوران دانشجویی و یا حتی پس از آن منجر خواهد شد.

۳- دسته‌های سختی مسائل و تفاوت آنها را تشریح نمایید.

نظریه پیچیدگی محاسباتی به بررسی دشواری حل مسائل به وسیله الگوریتم می‌پردازد. این نظریه بخشی از نظریه محاسباتی است که با منابع موردنیاز برای حل یک مسئله سروکار دارد. این نظریه روی پیچیدگی زمانی و فضایی الگوریتم‌ها بحث می‌کند. در واقع سختی و سادگی مسئله در ذات مسئله نهفته است. این نظریه می‌خواهد مسائل را برحسب دشواری آن‌ها دسته‌بندی کند.

توجه: کلاس دسته مسائلی است که توسط یک ماشین تورینگ قطعی در زمان چندجمله‌ای پذیرفته می‌شود و برای آن‌ها یک الگوریتم قطعی وجود دارد که در بدترین حالت در زمان چندجمله‌ای حل می‌شود. یعنی بتوان آن مسائل را در زمان $O(n^k)$ به‌ازای K ثابت حل کرد.



• کلاس P

هر مسئله‌ای که بتوان در زمان چندجمله‌ای به طور قطعی حل کرد، به کلاس P تعلق دارد. پس مرتب‌سازی سریع مسئله‌ای است در کلاس P چون زمانش حداکثر مربعی است. به بیان دیگر، اگر یک الگوریتم مانند یک ماشین تورینگ و یا یک برنامه رایانه‌ای با حافظه نامتناهی داشته باشیم که بتواند به‌ازای مقادیر ثابت k و c ، جواب درست را برای هر ورودی به طول n ، در حداکثر $c.nk$ مرحله به دست بیاورد، آنگاه می‌گوییم آن مسئله در زمان چندجمله‌ای قابل حل بوده و در کلاس P قرار دارد.

• کلاس NP

کلاس و دسته مسائل تصمیم پذیری است که اگر یک جواب از مسئله را داشته باشیم، می‌توانیم در زمان چندجمله‌ای بررسی کنیم که آیا این جواب متعلق به جواب مسئله است یا خیر. یا به عبارتی کلاس دسته مسائلی است که توسط یک ماشین تورینگ غیرقطعی در زمان چندجمله‌ای پذیرفته می‌شود.

• کلاس NP-Complete یا NPC

کلاس دسته مسائلی است که در دو شرط زیر صدق کند:

۱. $A \in NP$ یعنی مسئله NP باشد.

۲. برای هر مسئله B که آن هم NP است یعنی $B \in NP$ داشته باشیم: $B \leq_p A$
یک مسئله تصمیم‌گیری مانند L یک مسئله NPC است اگر در مجموعه مسائل نیز NP باشد و به‌سختی هیچ مسئله‌ای در NP نباشد، به‌طوری‌که هر راه‌حلی که برای مسئله تصمیم‌گیری می‌دهد دارای پیچیدگی زمانی چندجمله‌ای یا $O(n^k)$ باشد. به‌عبارتی دیگر پیچیده‌ترین مسائل کلاس NP در کلاس NPC وجود دارد. به‌طوری‌که هیچ الگوریتم شناخته شده و قابل اجرایی با زمان چندجمله‌ای برای آن‌ها وجود ندارد. مسائل NP کامل یا NPC جزء سخت‌ترین مسئله‌ها در کلاس NP هستند. ویژگی این مسائل، استفاده از مسئله‌های دیگر کلاس NP، برای حل آن با روش کاهش هست.

• کلاس NP-hard

مسئله A را NP-hard می‌نامیم هرگاه هر مسئله $B \in NP$ داشته باشیم که $B \leq_p A$.

کاهش پذیری‌های زمان چندجمله‌ای، ابزاری رسمی برای نمایش سخت بودن یک مسئله با وجود یک عامل زمان چندجمله‌ای به‌اندازه مسئله دیگر، فراهم می‌کند. بدین معنی که اگر $L_2 \leq L_1$ باشد، آنگاه L_1 بیش از یک عامل چندجمله‌ای، سخت از L_2 نخواهد بود. بنابراین می‌توان مجموعه زبان‌های NP کامل را تعریف نمود که سخت‌ترین زبان‌ها در NP هستند. زبان $L \subseteq \{0,1\}^*$ از نوع NP کامل است اگر در دو شرط زیر صدق کند:

الف) $L \in NP$

ب) $L_2 \leq L$ برای هر $L' \in NP$

اگر زبان L دارای ویژگی دوم باشد اما ویژگی اول را نداشته باشد، می‌گوییم، L از نوع NP سخت است. باید توجه داشت، NPC نیز جزء زبان‌های NP کامل به شمار می‌آیند.