1- آ) کلاس چندجمله‌ای (P) شامل تمام مسائل تصمیم‌گیری مانند Q است که برای حل آن‌ها الگوریتمی با مرتبه زمانی چندجمله‌ای مانند A که یک decider است وجود داشته باشد که به ازای هر ورودی x، اگر $x \in Q$ باشد در این صورت $A(x) = YES$ و اگر $x \notin Q$، در این صورت $A(x) = NO$ باشد.

ب) یک کاهش چندجمله‌ای از مسئله تصمیم‌گیری X به مسئله تصمیم‌گیری Y، یک الگوریتم (تابع) مانند f است که اولا یک نمونه از مسئله X مانند I_X را می‌گیرد، و نمونه I_Y از مسئله Y را به عنوان خروجی می‌دهد، دوما این الگوریتم در زمان چندجمله‌ای نسبت به $|I_X|$ اجرا می‌شود که نشان می‌دهد $|I_Y|$ نیز نسبت به $|I_X|$ چندجمله‌ای است و سوما اگر پاسخ مسئله X به ورودی I_X برابر با YES باشد، پاسخ مسئله Y نیز به ورودی I_Y مقدار YES خواهد بود. به عبارت دیگر می‌توان گفت:

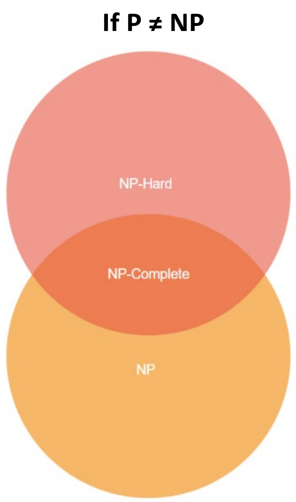
$I_X \in X \leftrightarrow f(I_X) = I_Y \in Y$

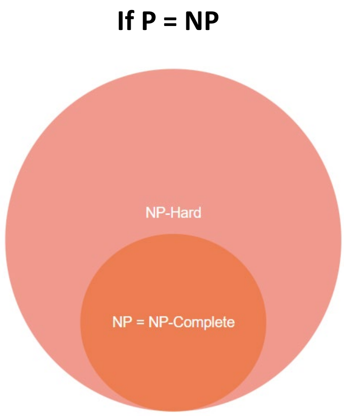
تبدیل ذکر شده را با عبارت $X \leq_P Y$ نشان می‌دهند.

ج) مسائل کلاس NP-Hard مسائلی مانند Q هستند که بتوانیم تمام مسائل کلاس NP را با استفاده از تابع کاهش چندجمله‌ای، به Q کاهش دهیم. با توجه به اینکه تمام مسائل کلاس NP به هر مسئله کلاس NP-Complete مانند Q’ قابل کاهش هستند، کافی‌ست Q’ را به Q کاهش دهیم. نکته مهم در مورد این کلاس مسائل این است که نیازی نیست مسئله Q در کلاس NP باشد. وجه تمایز این مسائل با کلاس NP این است که لزوما در زمان چندجمله‌ای verify نمی‌شوند و حتی ممکن است مسئله تصمیم‌گیری نباشند.

د) مسائل کلاس NP-Complete مانند Q زیرمجموعه‌ای از مسائل NP-Hard هستند که در کلاس مسائل NP قرار می‌گیرند. به عبارت دیگر می‌توان گفت مسئله Q اولا باید در کلاس NP باشد که این مورد به این معنی است که Q یک مسئله تصمیم‌گیری است که برای آن یک verifier مانند V وجود دارد که به ازای هر ورودی x اگر $x \in Q$ باشد، یک گواهی مانند y با اندازه چندجمله‌ای نسبت به ورودی مسئله Q وجود دارد که $V(x, y) = YES$ باشد و اگر $x \notin Q$، در این صورت به ازای هر گواهی مانند y، مقدار $V(x, y) = NO$ باشد، و دوما تمام مسائل کلاس NP باید این امکان را داشته باشند که توسط تابع کاهش چندجمله‌ای به مسئله Q کاهش یابند. مورد ذکر شده به این معنی است که کافی‌ست بتوانیم توسط یک تابع کاهش چندجمله‌ای، یک مسئله از کلاس NP-Complete مانند Q’ را به Q کاهش دهیم.

ه)





2- آ) این مورد صحیح است. فرض کنیم برای مسئله Q که در کلاس NP-Complete قرار دارد، یک الگوریتم با مرتبه زمانی چندجمله‌ای یافت شود و در نتیجه Q در کلاس P قرار بگیرد. در این صورت می‌دانیم هر مسئله Q’ که در کلاس NP قرار دارد، در زمان چندجمله‌ای می‌تواند به مسئله Q کاهش یابد. در نتیجه تمام مسائل این کلاس در زمان چندجمله‌ای حل می‌شوند و در کلاس P قرار می‌گیرند.

ب) این مورد صحیح نیست. همانطور که در مورد (د) سوال 1 گفته شد، برای اینکه یک مسئله در کلاس NP-Complete قرار بگیرد باید اولا یک مسئله NP باشد و دوما بتوانیم یک مسئله NP-Complete را در **زمان چندجمله‌ای** به آن کاهش دهیم که بخش کاهش چندجمله‌ای در سوال ذکر نشده است. در صورتی که منظور صورت سوال از کاهش را کاهش چندجمله‌ای در نظر بگیریم، این مورد صحیح است.

ج) این مورد صحیح نیست. طبق تعریف کلاس NP-Complete، تمامی مسائل کلاس NP قابل کاهش به مسائل این کلاس در زمان چندجمله‌ای هستند. اما مسائل کلاس NP-Hard لزوما در کلاس NP قرار نمی‌گیرند. از طرف دیگر مسائل NP-Hard لزوما حتی decidable هم نیستند. برای مثال مسئله Halting Problem در کلاس NP-Hard قرار می‌گیرد اما قابل کاهش به مسائل کلاس NP-Complete نیست.

د) این مورد نیز نادرست است. به عنوان مثال نقض می‌توان مسائل کلاس P را ذکر کرد که در زمان چندجمله‌ای حل می‌شوند و همگی در کلاس NP نیز قرار دارند، اما این مورد سبب نمی‌شود که بتوان تمام مسائل کلاس NP را در زمان چندجمله‌ای حل کرد.

ه) این مورد صحیح است. می‌دانیم تمام مسائل کلاس NP در زمان چندجمله‌ای قابل کاهش به هر مسئله کلاس NP-Hard مانند Q هستند. از طرف دیگر مسائل کلاس NP-Complete نیز در کلاس NP قرار می‌گیرند و در نتیجه آن‌ها نیز در زمان چندجمله‌ای قابل کاهش به مسئله Q هستند. پس اگر یک راه حل چندجمله‌ای برای مسئله Q یافت شود، تمامی مسائل کلاس NP از جمله کلاس NP-Complete نیز در زمان چندجمله‌ای حل می‌شوند.

3- برای اثبات قرار گرفتن مسئله ذکر شده در کلاس مسائل NP-Hard کافی‌ست بتوانیم مسئله Independent-Set که یک مسئله NP-Complete است را در زمان چندجمله‌ای به مسئله مورد نظر کاهش دهیم. مسئله Independent-Set به عنوان ورودی یک گراف و عدد k ($< G = (V, E),\ k >$) را می‌گیرد و مشخص می‌کند که آیا زیرمجموعه‌ای از رئوس گراف به اندازه k وجود دارد که بین هیچ کدام از آن‌ها یالی نباشد یا خیر. اگر تعداد رئوس G (|V|) را برابر با n در نظر بگیریم، در این صورت یک مجموعه جدید رئوس مانند U و با اندازه n + 1 به گراف اضافه می‌کنیم. از هر کدام از راس‌هایی که به گراف اضافه شده‌اند به تمام رئوس V یک یال وصل می‌کنیم. مجموعه یال‌های اضافه شده را F می‌نامیم. گراف جدید برابر با $G' = (V', E')$ است که $V' = V \cup U$ و $E' = E \cup F$ خواهند بود.

فرض کنیم M یک Independent-Set با اندازه k در گراف G باشد. با اضافه کردن تمام رئوس U به مجموعه M، مجموعه M’ ساخته می‌شود. تعداد رئوس این مجموعه برابر با n + k + 1 خواهد بود. به ازای هر مجموعه 3 راسی {a, b, c} که از M’ انتخاب کنیم، اگر بیش از 1 مورد آن از مجموعه U انتخاب شود، بدیهی‌ست که مثلثی بین این 3 راس تشکیل نمی‌شود زیرا بین 2 راسی که در مجموعه U قرار دارند، هیچ یالی وجود ندارد. از طرفی اگر بیش از یک راس از سه‌تایی ذکر شده در مجموعه M قرار بگیرند، در این حالت هم هیچ مثلثی تشکیل نمی‌شود زیرا رئوس M یک Independent-Set هستند و بین آن‌ها یالی وجود ندارد. در نتیجه با انتخاب هر 3 راس از مجموعه M’، هیچ مثلثی تشکیل نمی‌شود. **پس می‌توان گفت اگر G یک Independent-Set با اندازه k داشته باشد، G’ یک مجموعه بدون مثلث با حداقل اندازه n + k + 1 خواهد داشت.** در نتیجه گراف G’ را به همراه مقدار n + k + 1 به مسئله Triangle-Free می‌دهیم.

از طرفی اگر M’ یک مجموعه Triangle-Free با اندازه n + k + 1 در گراف G’ باشد، مجموعه‌ای مانند M با حداقل اندازه k در گراف G وجود دارد که یک Independent-Set است. برای نشان دادن این مورد کافی است مجموعه M را برابر با M’ \ U قرار دهیم. می‌دانیم که |U| = n + 1 است و در نتیجه اندازه مجموعه M حداقل k خواهد بود. همچنین می‌دانیم که $M' \cap U \neq \emptyset$ است. در نتیجه راس $v \in (M' \cap U)$ را در نظر می‌گیریم. به ازای هر 2 راس a و b که در مجموعه M در نظر بگیریم، با توجه به اینکه {v, a, b} مثلث تشکیل نمی‌دهند و از طرفی می‌دانیم $(v, a) \in E'$ و $(v, b) \in E'$، بدیهی‌ست که یال ab در E’ وجود ندارد. همچنین می‌دانیم که $E \subset E'$ است، در نتیجه یال ab در E هم وجود ندارد و رئوس a و b یک Independent-Set تشکیل می‌دهند.

بدیهی است که تبدیل ذکر شده در زمان چندجمله‌ای $O(|V|)$ قابل انجام است.

4- در ابتدا باید اثبات کنیم که این مسئله در کلاس مسائل NP قرار می‌گیرد. بدیهی‌ست که مسئله داده شده یک مسئله تصمیم‌گیری است. فرض کنیم که M یک گشت بسته از رئوس داده شده باشد. برای verify کردن پاسخ مسئله کافی‌ست یک verifier طراحی کنیم که M (به عنوان گواهی) و گراف اصلی و راس شروع را به عنوان ورودی می‌گیرد، ابتدا بررسی می‌کند که آیا راس شروع گشت با راس داده شده برابر باشد، سپس M را پیمایش می‌کند و در این پیمایش اولا بررسی می‌کند که یال‌های داده شده در گشت در گراف وجود داشته باشند و همچنین طول گشت داده شده را نیز محاسبه می‌کند و در نهایت برابری آن با k را بررسی می‌کند. مورد دیگری که بررسی می‌شود این است که گشت داده شده شامل یال تکراری نباشد و راس پایانی با راس شروع یکسان باشد. واضح است که مرتبه زمانی این کار چندجمله‌ای و از اردر $\mathcal{O}(|V|+|E|)$ است. در نتیجه مسئله داده شده در کلاس NP قرار دارد.

حال برای اینکه نشان دهیم این مسئله در کلاس NP-Complete قرار دارد، باید یکی از مسائل کلاس NP-Hard را به آن کاهش دهیم. برای این کار از مسئله Subset-Sum استفاده می‌کنیم.

ورودی مسئله Subset-Sum یک مجموعه مانند S و یک جمع مطلوب مانند k است. برای تبدیل این موارد به ورودی مسئله داده شده مراحل زیر را انجام می‌دهیم:

1) یک گراف با تعداد رئوس |S| در نظر می‌گیریم. در این بخش شماره رئوس گراف و اندیس اعداد مجموعه را از 0 شروع می‌کنیم. هر راس i در گراف متناظر با عنصر i-ام مجموعه S است.
2) از هر راس i به راس i+1 یک یال با وزن 0 قرار می‌دهیم. از راس آخر هم یک یال با وزن 0 به راس اول قرار می‌دهیم.
3) بر روی هر راس i، یک طوقه با وزن S[i] قرار می‌دهیم.
4) یک راس را به دلخواه انتخاب می‌کنیم و آن را راس شروع در نظر می‌گیریم. در اینجا برای سادگی می‌توانیم راس اول را انتخاب کنیم.
5) گراف بدست آمده را به همراه k و راس شروع انتخاب شده به مسئله گشت می‌دهیم.

بدیهی‌ست که مرتبه زمانی تبدیل ذکر شده چندجمله‌ای و از مرتبه $\mathcal{O}(|S|)$ خواهد بود.

می‌دانیم که گشت نمی‌تواند یال تکراری داشته باشد و در نتیجه هر طوقه حداکثر یک بار پیمایش می‌شود. خروجی مسئله گشت شامل تعدادی طوقه است که اگر طوقه روی راس i-ام انتخاب شده باشد، به این معنی است که مقدار S[i] باید در زیر مجموعه مطلوب انتخاب شود. واضح است که جمع عناصر انتخاب شده برای زیر مجموعه برابر با جمع وزن طوقه‌های انتخاب شده است که این مقدار برابر با k خواهد بود که همان پاسخ مطلوب مسئله Subset-Sum است.

از طرفی با داشتن یک زیر مجموعه مطلوب برای مسئله Subset-Sum، می‌توانیم گشتی به طول k در گراف ذکر شده بدست آوریم. در نتیجه می‌توانیم پاسخ هر مسئله را به مسئله دیگر تبدیل کنیم که نشان می‌دهد مسئله Subset-Sum به مسئله گشت کاهش پیدا کرده است. پس می‌توان گفت مسئله گشت در کلاس مسائل NP-Hard قرار می‌گیرد. از طرفی پیش‌تر اثبات کردیم که این مسئله در کلاس مسائل NP نیز قرار دارد پس در واقع مسئله گشت بسته با اندازه k در کلاس مسائل NP-Complete قرار می‌گیرد.

5- در ابتدا اگر اشتباه نکنم، ماتریس A باید $n \times m$ باشد، زیرا اگر این ماتریس مربعی باشد، با محاسبه ماتریس وارون A (A^{-1}) و ضرب آن در دو طرف معادله، بردار X بدست می‌آید. با توجه به اینکه محاسبه دترمینان ماتریس مربعی و محاسبه وارون آن و همچنین ضرب دو ماتریس در زمان چندجمله‌ای قابل محاسبه است، مسئله داده شده در کلاس P قرار می‌گیرد. در نتیجه ادامه سوال را با این فرض حل می‌کنم که ماتریس A یک ماتریس $n \times m$ باشد.

ابتدا باید اثبات کنیم که این مسئله در کلاس NP قرار می‌گیرد. بدیهی‌ست که مسئله داده شده یک مسئله تصمیم‌گیری است. فرض کنیم مسئله حل شده و پاسخ X برای آن بدست آمده است. در ادامه یک verifier برای این مسئله طراحی می‌کنیم. این verifier بردار X را به عنوان گواهی به همراه ماتریس A در ورودی دریافت می‌کند و با ضرب ماتریس A در بردار X که در زمان چندجمله‌ای قابل انجام است، برابری حاصل ضرب را با بردار تماما 1 بررسی می‌کند. در نتیجه این مسئله در کلاس مسائل NP قرار می‌گیرد.

حال برای اثبات اینکه مسئله داده شده در کلاس مسائل NP-Complete قرار می‌گیرد، باید یکی از مسائل کلاس NP-Hard را به آن کاهش دهیم. در این بخش از مسئله 3-Sat استفاده می‌کنیم. ابتدا باید مسئله 3-Sat را به مسئله 3D-Matching کاهش دهیم و سپس مسئله 3D-Matching را به مسئله داده شده (ZOE) کاهش دهیم. اثبات کاهش اول بسیار مفصل است و در این بخش نمی‌گنجد و به همین دلیل از نوشتن آن صرف نظر کردم =) اثبات ذکر شده در این لینک قابل دسترس است.

برای کاهش دوم به این شکل عمل می‌کنیم:

ورودی مسئله 3D-Matching شامل 3 مجموعه X، Y و Z است که هر کدام m عضو دارند. ورودی دیگر مسئله شامل n تا 3تایی مرتب (x_i, y_j, z_k) است. ماتریس A را یک ماتریس با 3m سطر و n ستون در نظر می‌گیریم. در ادامه مجموعه‌های X و Y و Z را در کنار هم قرار می‌دهیم تا یک مجموعه 3m عضوی به نام T را تشکیل دهند. حال در ماتریس A درایه a_{ij} را در صورتی برابر با 1 در نظر می‌گیریم که عضو i-ام مجموعه T در 3تایی j-ام وجود داشته باشد و در غیر این صورت درایه ذکر شده را برابر با 0 در نظر می‌گیریم. حال ماتریس A که یک ماتریس $3m \times n$ است را به مسئله ZOE می‌دهیم. پاسخ این مسئله بردار n عضوی X' خواهد بود که هر کدام از عناصر آن می‌تواند 0 و یا 1 باشد. اگر x'_i برابر با 1 باشد، به این معنی است که 3تایی i-ام را انتخاب می‌کنیم و اگر برابر با 0 باشد به این معنی است که 3تایی مورد نظر انتخاب نمی‌شود. همچنین بردار تماما 1 نیز شامل 3m عنصر 1 است که نشان می‌دهد تمام اعضای X و Y و Z باید انتخاب شوند. پس توانستیم مسئله 3D-Matching را به مسئله ZOE کاهش دهیم و با توجه به اینکه NP بودن این مسئله را پیش‌تر اثبات کردیم، مسئله ZOE در کلاس NP-Complete قرار می‌گیرد.

6- در ابتدا باید اثبات کنیم که مسئله داده شده در کلاس NP قرار می‌گیرد. بدیهی‌ست که مسئله داده شده از نوع تصمیم‌گیری است. فرض می‌کنیم مسئله حل شده و k راسی که باید حذف شوند را به صورت $i_1, i_2, \ldots, i_k$ در نظر می‌گیریم. در ادامه یک verifier برای این مسئله طراحی می‌کنیم. این verifier گراف ورودی مسئله را به همراه k راس حذف شده (گواهی) و عدد k به عنوان ورودی می‌گیرد، راس‌های ذکر شده را از گراف حذف می‌کند و با استفاده از الگوریتم‌های تشخیص دور، عدم وجود دور در گراف باقی‌مانده را بررسی می‌کند. برای این مورد می‌توان از الگوریتم DFS استفاده کرد. همچنین این verifier باید تعداد راس‌های حذف شده را هم بشمارد و برابری آن با k و همچنین تمایز دو به دوی این رئوس را نیز بررسی کند. انجام تمام این کارها در زمان چند جمله‌ای با مرتبه $O(|V| + |E|)$ امکان‌پذیر است. در نتیجه این مسئله در کلاس مسائل NP قرار می‌گیرد.

حال یکی از مسائل NP-Hard را به مسئله داده شده کاهش می‌دهیم. در این بخش از Vertex-Cover استفاده می‌کنیم. ورودی مسئله Vertex-Cover یک گراف G و عدد k است که مطلوب مسئله انتخاب k راس از رئوس G برای Vertex-Cover است. برای اینکه ورودی‌ها را به ورودی مسئله داده شده تبدیل کنیم، یک گراف جدید G’ با رئوس گراف G در نظر می‌گیریم. به ازای هر یال uv که در گراف G قرار دارد، یک یال از u به v و یک یال از v به u در گراف G’ قرار می‌دهیم. در این صورت به ازای هر یال یک دور ایجاد کرده‌ایم. حال اگر گراف G’ را به همراه مقدار k به ورودی مسئله FVS بدهیم، مسئله FVS باید به ازای هر یال (که برای آن یک دور ایجاد کرده‌ایم)، حداقل یک راس را حذف کند که دور ایجاد شده از بین برود. در نتیجه خروجی مسئله FVS دقیقا همان خروجی مطلوب برای مسئله Vertex-Cover است.

همچنین اگر نخواهیم بین هر 2 راس u و v دو یال قرار دهیم، می‌توانیم از یک راس dummy استفاده کرده که در این صورت یال اول از راس u به راس v و یال دوم از راس v به راس dummy و از راس dummy به راس u خواهد بود. در نهایت اگر راس dummy در رئوس حذف شده وجود داشت، آن را با یکی از رئوس اصلی یال (u یا v) جایگزین می‌کنیم.

بدیهی‌ست که مرتبه زمانی تبدیل ذکر شده، چندجمله‌ای و از مرتبه $O(|E| + |V|)$ خواهد بود.

برای اینکه نشان دهیم هر جواب برای مسئله FVS یک جواب برای مسئله Vertex-Cover نیز هست، از برهان خلف کمک می‌گیریم. فرض خلف می‌کنیم که پاسخ FVS نمی‌تواند یک پاسخ برای مسئله Vertex-Cover باشد. در این صورت یالی در گراف G وجود دارد که هیچ کدام از راس‌های دو طرف آن انتخاب نشده‌اند. طبق نحوه ساخت گراف G’، یک دور برای این یال ایجاد شده و با توجه به اینکه هیچ‌کدام از راس‌های این دور حذف نشده‌اند، این مورد یک پاسخ قابل قبول برای مسئله FVS نخواهد بود که این مورد با فرض تناقض دارد. پس فرض خلف باطل است و حکم اثبات می‌شود.

در نتیجه به راحتی می‌توانیم پاسخ‌های 2 مسئله را به یکدیگر تبدیل کنیم. پس با تبدیل ذکر شده توانستیم در زمان چندجمله‌ای مسئله Vertex-Cover را به مسئله FVS کاهش دهیم. با توجه به اینکه پیش‌تر اثبات کردیم این مسئله در کلاس مسائل NP قرار می‌گیرد، با کاهش یک مسئله NP-Hard به FVS، نشان دادیم که این مسئله در کلاس مسائل NP-Complete قرار می‌گیرد.