به نام خدا

مرضیه علیداری - 810101236

تکلیف سوم الگوریتم پیشرفته

(1)

الف) غلط؛ مسائل NP-complete به صورت polynomial، قابل کاهش به هم هستند. اگر یک مسئله NP-complete به نام A در زمان خطی (یعنی $O(n^1)$؛ که n لزوماً توانش 1 است)، قابل حل باشد، این مسئله در زمان polynomial (یعنی $O(n^k)$؛ که k لزوماً 1 نیست)، قابل تبدیل به مسائل NP-complete دیگر است. پس مسائل NP-complete دیگر که A به آن‌ها تبدیل می‌شود، در polynomial قابل حل هستند.

ب) غلط؛ این قضیه برای NP-Complete ها بیان می‌شود. با توجه به اینکه مسائل NP-Complete در زمان polynomial قابل تبدیل به یکدیگر هستند، اگر یکی از آن‌ها در polynomial حل شود، تمام مسائل NP-Complete در زمان polynomial قابل حل هستند. پس این قضیه برای NP-C ها فقط برقرار است، که زیرمجموعه‌ای از NP هستند. اگر این قضیه را درباره‌ی NP ها بیان کنیم، با توجه به اینکه مسئله‌های P، polynomial هستند، و نوعی NP هستند؛ باید تمام NP ها هم قابل حل در polynomial باشند که اینطور نیست.

ج) غلط؛ درباره‌ی مسائل NP-Complete داشتیم: اگر یک مسئله‌ی NP-Complete به اسم A و یک مسئله‌ی NP به اسم B داشته باشیم؛ اگر بتوان مسئله‌ی A را در زمان polynomial به مسئله‌ی B، reduce کرد ← آنگاه B هم یک مسئله‌ی NP-Complete است. اگر در این سوال قید شده بود که این کاهش در polynomial انجام می‌شود، درست بود. ولی این شرط بیان نشده، و در حالت کلی این حرف برقرار نیست.

الف) اثبات NP بودن مسئله:

← فرضاً یک گراف به ما می‌دهند که k تا node از n تا node آن حذف شده است. باید بررسی شود که آیا در آن دور وجود دارد یا خیر.

← هزینه: $O(\underbrace{E}_{\text{تعداد یال‌ها} \leftarrow \text{حداکثر } (n-k)^2} + \underbrace{(n-k)}_{\text{تعداد node ها}}) = O((n-k)^2)$ → polynomial ✓

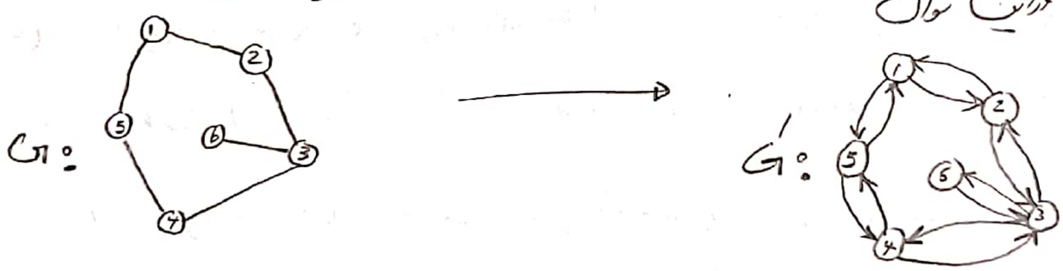
← توانستیم درستی این certificate را در polynomial time چک کنیم. پس این مسئله NP است. ✓

ب) در مسئله‌ی vertex cover (VC)، گرافی که داریم، بدون جهت است. ولی مسئله‌ی مورد تقاضای ما در اینجا، دارای گرافی جهت‌دار است. ← برای تبدیل مسئله‌ی VC به مسئله‌ی مورد تقاضا، هر یال در گراف بین دو node v و u را با دو یال جهت‌دار یکی از u به v و دیگری از v به u، جایگزین می‌کنیم.

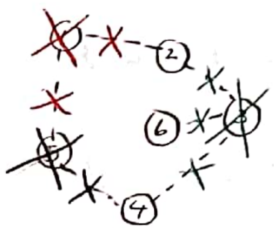
← مثلاً:

گراف مسئله‌ی VC

گراف مسئله‌ی مورد تقاضای‌مان در این سوال



⊕ فرضاً مسئله‌ی VC را داریم، که گراف G، گراف متناظر آن است. vertex cover این گراف، مجموعه node های {1,3,5} است. که با حذف این node ها، تمام یال‌های گراف حذف می‌شوند.



که در گراف متناظر (G')، با حذف همین node ها، تمام یال‌های G' نیز حذف می‌شوند. و دوری در این گراف باقی نمی‌ماند.

⊕ فرضاً مسئله‌ی مورد تقاضای‌مان در این سوال را داریم، که گراف G'، گراف متناظر آن است. در این گراف، هر دو node v و v' که بین شان یال است، یک دور به این شکل وجود دارد:

$((v \to v' \to v))$

که برای حذف این دور، باید یکی از node های v یا v' حذف شود. که با حذف یکی از آن‌ها، هر دو یال بین شان حذف می‌شود. پس برای حذف دورها در G'، تمام یال‌ها باید حذف شود. ← پس برای گراف متناظر (G)، با حذف همین node ها، یالی باقی نمی‌ماند. و همین مجموعه یال‌ها، vertex cover گراف G هستند.

← مشخص است که این reduction از VC به مسئله‌ی مورد تقاضای‌مان، در polynomial time قابل انجام است.

[1] ابتدا باید اثبات کنیم که آیا عضو NP است یا خیر:

اگر یک جواب از این مسئله به ما داده شود و گفته شود مجموع وزن یال های اعضای این زیر مجموعه یال های گراف (که دور) ، 0 است ، در Polynomial می‌توانیم آن ها را با هم جمع کنیم و چک کنیم که آیا برابر 0 می شود یا خیر.

[2] سپس باید NP-hard بودن آن را هم اثبات کنیم:

برای این اثبات از reduce کردن مسئله subset sum کمک می‌گیریم.

به این مسئله subset sum را داریم: یک مجموعه از اعداد مثبت و منفی داریم. می‌خواهیم ببینیم آیا زیر مجموعه ای از آن وجود دارد که مجموع اعضایش برابر 0 شود یا خیر. $\{x_1, x_2, \dots, x_n\}$

با استفاده از این مجموعه می‌خواهیم یک گراف تولید کنیم. به ازای هر عضو از این مجموعه، دو node در گراف تولید می‌کنیم. در کل گراف، 2n تا node خواهد داشت:

$x_1 \longrightarrow v_1 , v_1'$

$x_2 \longrightarrow v_2 , v_2'$

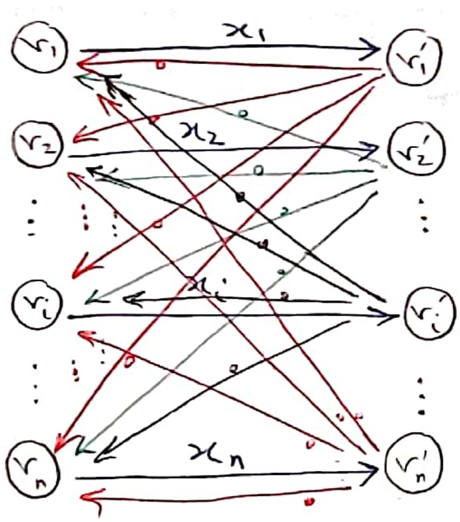
$\vdots$

$x_i \longrightarrow v_i , v_i'$

$\vdots$

$x_n \longrightarrow v_n , v_n'$

یال های گراف را به این صورت تولید می‌کنیم: [1] به ازای هر دو node v_i و v_i' ، یک یال با وزن x_i از v_i به v_i' با وزن x_i ایجاد می‌کنیم. [2] n تا یال از هر v_i' به تمام v_j ها (v_1 تا v_n) ، با وزن 0 ایجاد می‌کنیم. ← در نهایت، گراف تولید شده به این شکل خواهد بود:



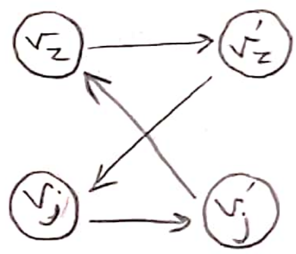
+ با استفاده از این یال های تولید شده، دورهایی به این شکل خواهیم داشت :

$$v_z \to v_z' \to v_j \to v_j' \to \dots \to v_k \to v_k'$$

که تمام یال های بین $v_i \to v_i'$ ها دارای وزن x_i هستند. و تمام یال های بین $v_i' \to v_j$ ها دارای وزن ۰ هستند.

و برای مثال، هزینه‌ی دور زیر، برابر با $x_z + x_j$ خواهد بود :

$$v_z \xrightarrow{W=x_z} v_z' \xrightarrow{W=0} v_j \xrightarrow{W=x_j} v_j' \xrightarrow{W=0} v_z$$



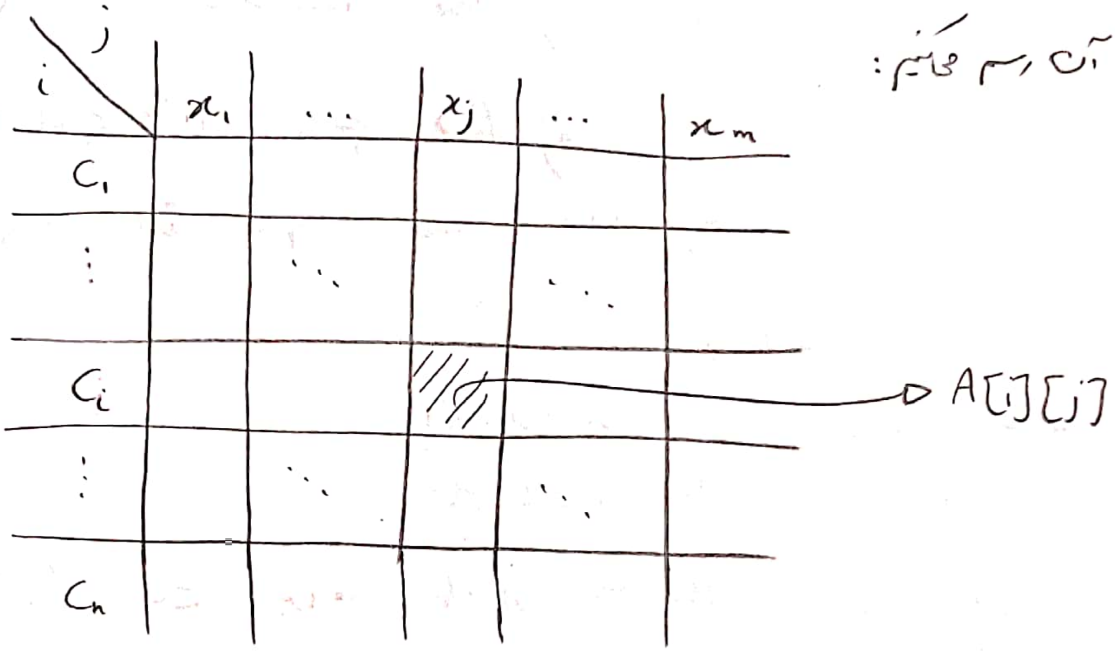
پس به راحتی می‌توان دید که :

⊕ اگر مجموع عناصر زیر مجموعه $\{x_z, x_j\}$ از مجموعه‌ی کلی‌مان در subset sum، برابر صفر باشد، هزینه‌ی دور متناظر در گراف‌مان هم برابر صفر است.

⊕ و اگر هزینه‌ی دور یافت شده در گراف (x_z+x_j) برابر ۰ باشد، زیر مجموعه‌ی $\{x_z, x_j\}$ در مجموعه‌ی کلی‌مان در subset sum هم برابر ۰ است.

⟸ پس در polynomial time توانستیم مسئله‌ی subset sum را به مسئله‌ی مورد نظرمان تبدیل کنیم. ⟸ پس این مسئله، NP-complete است. ✓

⊕ مسئله‌ی 3-CNF-SAT را درنظر می‌گیریم. می‌دانیم NP-Complete است.
تعداد variable ها را m و تعداد clause ها را برابر n درنظر می‌گیریم. جدول زیر را برای آن رسم می‌کنیم:



⟸ جدول را با این سه قانون پر می‌کنیم:

[1] اگر لیترال x_j در کلاز C_i وجود داشته باشد ⟸ $A[i][j] = O$

[2] اگر لیترال $\neg x_j$ در کلاز C_i وجود داشته باشد ⟸ $A[i][j] = X$

[3] درغیراین صورت ⟸ $A[i][j]$ را خالی نگه می‌داریم.

⟸ اگر 3-CNF-SAT ما satisfiable باشد، این پازل جواب دارد و بالعکس.

⊛ یک assignment true برای 3-CNF-SAT بدست می‌آوریم. و این عملیات را انجام می‌دهیم:

△1 اگر x_j=true باشد ⟸ X ها را از ستون j حذف می‌کنیم.

△2 اگر x_j=false باشد ⟸ O ها را از ستون j حذف می‌کنیم.

⟸ با توجه به اینکه هر variable حداقل در یک clause (خودش یا not اش) وجود دارد، هرستون، حداکثر یک نوع از علامت ها را دارد. + به علاوه، هر clause حداقل یک لیترال اش true است. بنابراین هر سطر حداقل یک علامت را دارد. ⟸ بنابراین، این پازل satisfiable است. ✓

ادامه‌ی ④

⊛ فرض می‌کنیم که این پازل قابل حل است. برای هر ستون j از n این موارد بررسی می‌شود:

△1 اگر ستون شامل O باشد ⟸ $x_j = true$

△2 اگر ستون شامل X باشد ⟸ $x_j = false$

△3 اگر ستون خالی باشد ⟸ x_j = false or true

به دلخواه ← اگر هنوز clause ای باشد که true نشده باشد، این متغیرهای باقی مانده را به طور مناسب مقدار دهی می‌کنیم تا همه‌ی clause ها true شوند

← هر سطر حداقل یک علامت دارد، پس هر clause در 3-CNF-SAT مان، حداقل یک لیترال اش true است. پس 3-CNF-SAT مورد نظر، Satisfiable است. ✓

⟸ واضح است که این reduction در polynomial time انجام می‌شود، پس با توجه به اینکه مسئله‌ی 3-CNF-SAT یک NP-Complete است، و در زمان polynomial به این مسئله تبدیل شود ← پس این مسئله، یک NP-hard است ✓

← مثال:

⊕→ 3-CNF-SAT: $(x_1 \vee \neg x_2 \vee x_4) \wedge (x_2 \vee x_3 \vee \neg x_4)$

← متغیرها: x_1, x_2, x_3, x_4

i \ j	x_1	x_2	x_3	x_4
c_1	O (c_1 در x_1)	X (c_1 در $\neg x_2$)		O (c_1 در x_4)
c_2		O (c_2 در x_2)	O (c_2 در x_3)	X (c_2 در $\neg x_4$)

⟹

i \ j	x_1	x_2	x_3	x_4
c_1	O			
c_2		O	O	X

← true assignment:

$x_1 = x_2 = true$

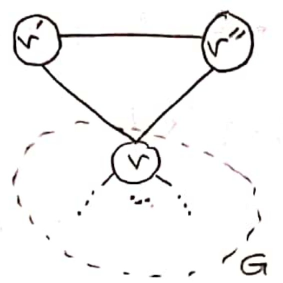
$x_3 = x_4 = false$

پس دیدیم که 3-CNF، satisfy شد.

⇕

پازل هم جوابش valid شد و شرایطش برقرار است. ✓

برای این اثبات، از مسئله HAM استفاده می‌کنیم. مسئله Ham را به این شکل تغییر می‌دهیم: به هر node v از گراف G به این شکل، دو node دیگر اضافه می‌کنیم:



یعنی به ازای هر گره، 2 گره و 3 یال جدید به این شکل اضافه می‌شود. و گراف جدید را G' می‌نامیم.

← باید اثبات کنیم که گراف G دارای یک دور همیلتونی است، اگر و تنها اگر، گراف G' دارای یک دور همیلتونی دوبل باشد.

⊕ فرضاً گراف G دارای یک دور همیلتونی به شکل $v_1 \to v_2 \to \dots \to v_i \to \dots \to v_n \to v_1$ باشد. در گراف G' با پیمایش بخش های جدید که اضافه شده، دور همیلتونی دوبل در گراف G' به این شکل خواهد بود:

$$((\underbrace{v_1 \to v_1' \to v_1'' \to v_1' \to v_1'' \to v_1}_{} \to v_2 \to \dots \to \underbrace{v_i \to v_i' \to v_i'' \to v_i' \to v_i'' \to v_i}_{} \to \dots \to$$

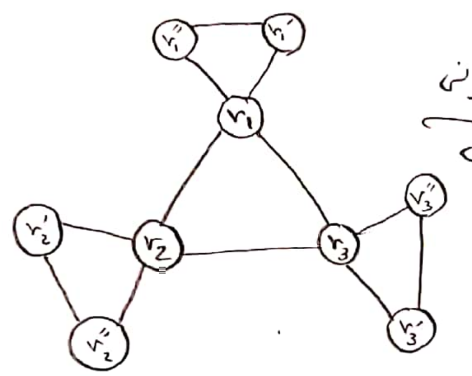
جایگزین v_1 جایگزین v_i

$$\underbrace{v_n \to v_n' \to v_n'' \to v_n' \to v_n'' \to v_n}_{} \to \underbrace{v_1 \to v_1' \to v_1'' \to v_1' \to v_1'' \to v_1}_{}))$$

جایگزین v_n جایگزین v_1

⊕ برای اثبات در جهت برعکس، گراف G' را در نظر می‌گیریم، که دور همیلتونی دوبل دارد:

G':



$\xrightarrow[\text{دوبل}]{\text{دور همیلتونی}}$ C': $((v_1\ v_1'\ v_1''\ v_1'\ v_1''\ v_1$
$v_2\ v_2'\ v_2''\ v_2'\ v_2''\ v_2$
$v_3\ v_3'\ v_3''\ v_3'\ v_3''\ v_3$
$v_1\ v_1'\ v_1''\ v_1'\ v_1''\ v_1))$

ادامه ی 5

که می‌خواهیم هر node در گراف اصلی G را در یک دور همیلتونی دقیقاً ۱ بار مشاهده کنیم. (یعنی v_1 و v_2 و v_3)

همانطور که مشخص هست، node های v_i' و v_i'' فقط از طریق node v_i به دیگر node ها متصل می‌شوند و برای رسیدن از این ها به دیگر node ها، بالاجبار از v_i گذشت.

اگر از ابتدا تا انتهای دور همیلتونی دوبل را بررسی کنیم، هر بار که به v_i جدید می‌رسیم، node های بعد از آن را پاک می‌کنیم، تا وقتی که دوباره آن v_i را مجدداً مشاهده کنیم.

$$((\ v_1\ \cancel{v_1'}\ \cancel{v_1''}\ \cancel{v_1}\ \cancel{v_1'}\ \cancel{v_1''}\ v_2\ \cancel{v_2'}\ \cancel{v_2''}\ \cancel{v_2}\ \cancel{v_2''}\ \cancel{v_2'}\ v_3\ \cancel{v_3''}\ \cancel{v_3'}\ \cancel{v_3}\ \cancel{v_3''}\ \cancel{v_3'}\ v_1\ \cancel{v_1'}\ \cancel{v_1}\ \cancel{v_1'}\ \cancel{v_1''}\ \cancel{v_1'}\ ))$$

بین v_1 و v_2 یال مستقیم هست.

بین v_2 و v_3 یال مستقیم هست.

بین v_1 و v_3 یال مستقیم هست.

که پس خواهیم داشت: $(v_1\ v_2\ v_3\ v_1)$ ← که یک دور همیلتونی است.

الف) پس توانستیم در polynomial time این reduction را از Ham به double Ham انجام بدهیم. پس اثبات می‌شود، با توجه به اینکه Ham، NP-complete است، که double Ham، NP-hard است.