نظریه علوم کامپیوتر

نیم‌سال دوم ۱۳۹۹-۱۴۰۰

مدرس: دکتر جواد ابراهیمی، دستیار آموزشی: ساجد کریمی

دانشکده علوم ریاضی

تمرین سری پنجم

نام و نام‌خانوادگی: امید یعقوبی

شماره دانشجویی:

پرسش ۱

می‌خواهیم نشان دهیم $SPACE(f(n)) \subseteq TIME(2^{O(f(n))})$ برقرار است.
فرض کنیم $A \in SPACE(f(n))$ آن‌گاه برای زبان A یک ماشین تورینگ قطعی با نام M وجود دارد که روی ورودی w آن‌را در فضای $O(f(n))$ که $n = |w|$ است، اجرا می‌کند. در این صورت هر کانفیگوریشن از این TM روی ورودی w اندازه‌ی حداکثر $c.f(n)$ برای یک ثابت c دارد. هر کانفیگوریشن، که یک "snapshot" از محاسبه ماشین M روی ورودی w است و به شکل (q_x, α, i) می‌باشد که در آن q_x حالت فعلی و α محتوای active نوار و i پوزیشن فعلی هد روی نوار است. پس تعداد کل کانفیگوریشن‌های ممکن M روی ورودی w برابر $|Q|.|\Gamma_m|^{c.f(n)}.c.f(n)$ برای یک ثابت c می‌باشد که به روشنی برابر $2^{O(f(n))}$ است.

حال گراف configuration را این‌گونه می‌سازیم که هر نود آن یک configuration ممکن از ماشین M روی ورودی w باشد. بین هر دو راس یال جهت دار به شکل (C_i, C_j) داریم اگر که $C_i \vdash C_j$ برقرار باشد. یا به عبارتی C_j بتواند یک configuration به دست آمده از C_i، configuration بوسیله‌ی تابع δ_m باشد. چک کردن وجود یال جهت دار بین دو کانفیگوریشن ممکن از مرتبه‌ی $O(max\{|C_i|, |C_j|\}) = O(f(n))$ می‌باشد و توسط الگوریتم *One-Step-Yieldability* که در ادامه توضیح می‌دهم انجام می‌شود.
برای دو کانفیگوریشن دلخواه $C_i = (q_x, \alpha, i)$ و $C_j = (q_y, \alpha', i')$ می‌گوییم $C_i \vdash C_j$ اگر شرایط زیر برقرار باشد:

$$(q_y, a, D) = \delta(q_x, \alpha_i) \; s.t$$

$$\alpha'_i = a \; and \; (\forall k)(k \neq i \Rightarrow \alpha'_k = \alpha_k) \; and \; (D = R \Rightarrow i' = i+1) \; and \; (D = L \Rightarrow i' = max\{i-1, 1\})$$

به روشنی اگر $C_i \vdash C_j$ آن‌گاه تنها تفاوت بین دو محتوای نوار در پوزیشن i می‌باشد. پس در این الگوریتم در یک حلقه‌ی for چک می‌شود که بقیه‌ی محتوای نوار بی تفاوت باشد. این از مرتبه‌ی $O(f(n))$ می‌باشد. حال چک می‌کنیم $\alpha'_i = a$ باشد چراکه $\delta(q_x, \alpha_i) = (q_y, a, D)$ همچنین چک می‌کنیم پوزیشن هِد به درستی به روز شده باشد. یعنی اگر $D = R$ باشد آن‌گاه هِد باید یک واحد به راست برود پس $i' = i + 1$ در غیر این صورت دو حالت داریم، یا هِد در سمت چپ‌ترین پوزیشن است و از آن چپ‌تر نمی‌تواند برود، در این صورت $i' = i = 1$ باقی می‌ماند یا اینکه در سمت چپ‌ترین پوزیشن نیست که در این حالت $i' = i - 1$ یا به عبارتی برای هر دوی این حالت‌ها داریم $i' = max\{i - 1, 1\}$. پس بررسی $C_i \vdash C_j$ از مرتبه‌ی $O(f(n))$ است.

پس گراف configuration که آن را با $G_{m,w}$ نشان می‌دهیم دارای $2^{O(f(n))}$ راس است و برای آن که ببینیم بین هر دو راس یالی وجود دارد یا نه $O(f(n))$ زمان نیاز داریم. پس ساخت $G_{m,w}$ از مرتبه‌ی $2^{O(f(n))}$ زمانی است.

حال می‌گوییم $w \in L(M)$ است اگر و تنها اگر که مسیری از Initial configuration به یکی از Accepting Configuration ها وجود داشته باشد. حال برای آن‌که Accepting Configuration را یکتا کنیم، M را بدون آن‌که زبان آن تغییر پیدا کند و زمان و فضای آن از نظر مجانبی تغییری پیدا کنند این گونه تغییر می‌دهیم که در هر حالت *Accept* قبلی ابتدا نوار را پاک کند و سپس هِد را روی سمت چپ‌ترین خانه‌ی نوار قرار دهد. به این ترتیب، Accepting Configuration یکتا خواهد شد. نام این کانفیگوریشن را C_f می‌گذاریم و منظورمان از C_0 همان Initial Configuration است. حال می‌گوییم رشته در زبان است اگر و فقط اگر در گراف کانفیگوریشن مسیری از C_0 به C_f وجود داشته باشد.

بر اساس قضیه‌ی 7.14 منبع [۱] می‌دانیم که $PATH \in P$، پس اجرای $PATH(\langle G_{m,w}, C_0, C_f\rangle)$ از مرتبه‌ی زمانی $O(|G_{m,w}|) = O(2^{O(f(n))})$ می‌باشد.

هر مرحله‌ی گفته شده مستقل از دیگری و هرکدام از مرتبه‌ی $O(2^{O(f(n))})$ زمانی بود. پس برای $A \in SPACE(f(n))$ یک الگوریتم در زمان $O(2^{O(f(n))})$ معرفی کردیم. از آن‌جا که A یک زبان دلخواه بود، پس

$$SPACE(f(n)) \subseteq TIME(O(2^{O(f(n))}))$$

بر قرار است.
■

پرسش ۲

می‌خواهیم نشان دهیم چگونه با داشتن یک اوراکل برای زبان SAT می‌توانیم برای هر نمونه $\phi \in SAT$ یک Satisfying Assignment در زمان Polynomial پیدا کنیم. به عبارتی نشان می‌دهیم چگونه می‌توان با دسترسی به جواب برای ورژن Decision این مساله ورژن Search آن را نیز حل کرد.
توجه شود که منظور از نماد $\phi[x_i/\top]$ جایگزینی True با متغیر x_i در فرمول ϕ بوده و منظور از $\phi[x_i/\bot]$ جایگزینی False با متغیر x_i در فرمول ϕ می‌باشد. همچنین منظور از ماشین M^{SAT} ماشین M است که به اوراکلی برای SAT دسترسی دارد.

حال در ماشین M^{SAT} ابتدا یک Query به اوراکل SAT می‌دهیم تا درستی $\phi \in SAT$ را بررسی کنیم. اگر درست بود $\phi[x_1/\top]$ را از روی ϕ در زمان Polynomial می‌سازیم. دوباره یک Query به اوراکل SAT می‌دهیم تا درستی $\phi[x_1/\top] \in SAT$ را بررسی کنیم. از آن‌جا که اگر $\phi \in SAT$ آن‌گاه یکی از $\phi[x_1/\top] \in SAT$ یا $\phi[x_1/\bot] \in SAT$ درست است، پس زمانی که برای Query داده شده، یعنی $\phi[x_1/\top] \in SAT$ جواب accept آمد آن‌گاه مقدار x_1 را برابر $\top$ قرار می‌دهیم و در غیراین صورت می‌دانیم که باید مقدار x_1 را برابر $\bot$ قرار دهیم. همچنین زمانی که $\phi[x_1/\top] \notin SAT$ برای مرحله‌ی بعدی، فرمول $\phi[x_1/\bot]$ را در زمان Polynomial می‌سازیم. اگر بدون از دست دادن عمومیت (wlog) فرض کنیم $\phi[x_1/\top] \in SAT$ آن‌گاه به سراغ متغیر بعدی، یعنی x_2 می‌رویم. به عبارتی، یک Query به اوراکل جهت بررسی $\phi[x_1/\top, x_2/\top] \in SAT$ می‌دهیم. به همین ترتیب اگر برای Query داده شده، یعنی $\phi[x_1/\top, x_2/\top] \in SAT$ جواب accept آمد، آن‌گاه x_2 را با $\top$ مقداردهی می‌کنیم و در غیر این صورت آن را با $\bot$ مقدار دهی می‌کنیم. اگر تعداد متغیرهای ϕ را n بگیریم، بار اول ϕ را با n متغیر بررسی می‌کنیم، بار دوم آن را با $n-1$ متغیر بررسی می‌کنیم و به همین ترتیب پس از n مرحله و n عدد Query یک Satisfying Assignment برای ϕ پیدا می‌کنیم. در هر یک از این n مرحله، در بدترین حالت جایگزینیِ $x_i = \top$ درست نیست و در مجموع دو مرحله جایگزینی داریم که Polynomial است. تعداد کل مراحل نیز دقیقا n است و $n = O(|\phi|)$ پس مرتبه‌ی زمانی این الگوریتم، با در نظر گرفتن هزینه‌ی $O(1)$ برای هر Query برابر $Poly(n).Poly(n) = Poly(n)$ است. یعنی چون هر مرحله چندجمله‌ای انجام می‌شود و تعداد مراحل هم چندجمله‌ای‌اند پس کل زمان اجرا نیز چند جمله‌ای است. حتا اگر هزینه‌ی هر Query را نیز به جای ثابت چندجمله‌ای می‌گرفتیم بازهم هزینه‌ی زمانی چند جمله‌ای بود. به این ترتیب در زمان چند جمله‌ای نسبت به اندازه‌ی فرمول ϕ بوسیله‌ی اوراکل برای SAT ماشین M^{SAT} ورژن Search مساله‌ی SAT را حل می‌کند. به عبارتی برای هر نمونه از SAT در زمان چند جمله‌ای با دسترسی به اوراکل SAT یک Satisfying Assignment پیدا کردیم.
$\blacksquare$

پرسش ۳

آ) می‌خواهیم نشان دهیم $P \subseteq NP \cap coNP$ برقرار است. فرض می‌کنیم $L \in P$ و بعد نشان می‌دهیم $L \in NP$ و همچنین $L \in coNP$ می‌باشد. اگر $L \in P$ آن‌گاه برای آن یک TM در زمان $Poly(n)$ وجود دارد. بر اساس تعریف برای هر TM قطعی یک NTM وجود دارد که از قابلیت nondeterministic خود استفاده نمی‌کند. به عبارتی هر مقدار تعریف شده از δ در NTM مجموعه‌ای به اندازه‌ی 1 است که از روی تابع انتقال ماشین قطعی ساخته شده است. پس $L \in NP$ برقرار است. حال اگر $L \in P$ نشان می‌دهیم که $\overline{L} \in P$ برقرار است. یا به عبارتی نشان می‌دهیم که P تحت مکمل بسته است. اگر $L \in P$ پس برای آن یک TM با نام M وجود دارد که آن را در زمان چند جمله‌ای تصمیم می‌گیرد. حال M' را این‌گونه می‌سازیم که M را روی ورودی‌اش اجرا کند، اگر M روی ورودی accept کرد او reject کند و برعکس. از آن‌جا که شبیه‌سازی سربار زمانی چندجمله‌ای دارد و خود M نیز در زمان $Poly(n)$ اجرا می‌شود. پس M' در زمان چند جمله‌ای اجرا خواهد شد. از آن‌جا که M' خروجی M را reverse می‌کند پس $L(M') = \overline{L(M)} = \overline{L}$ در نتیجه $\overline{L} \in P$ می‌باشد. براساس نتیجه‌ی بخش اول اثبات، $\overline{L} \in NP$ می‌باشد. با توجه به تعریف $coNP$ داریم $L \in coNP$ همچنین از بخش اول اثبات می‌دانیم که $L \in NP$ پس $L \in NP \cap coNP$ و از آن‌جا که L دلخواه بود خواهیم داشت $P \subseteq NP \cap coNP$ و اثبات تمام می‌شود.

■

ب) می‌خواهیم نشان دهیم اگر $P = NP$ آن‌گاه $NP = coNP$ خواهد بود. برای این‌کار ابتدا نشان می‌دهیم با داشتن این فرض داریم $coNP \subseteq P$ سپس بوسیله‌ی این حکم به سادگی $NP = coNP$ اثبات می‌شود. فرض $P = NP$ را داریم. حال برای هر زبان $L \in coNP$ بر اساس تعریف $coNP$ خواهیم داشت $\overline{L} \in NP$ و از آن‌جا که $P = NP$ پس $\overline{L} \in P$ و همچنین در بخش الف همین سوال نشان دادیم که کلاس P نسبت به مکمل بسته است، با نتیجه‌ی همان بخش از سوال داریم $L \in P$ و از آن‌جا که L دلخواه بود، حکم $coNP \subseteq P$ اثبات می‌شود.

برای هر زبان $L \in coNP$ از آن‌جا که $coNP \subseteq P$ پس $L \in P$ و چون فرض $P = NP$ را داریم، پس $L \in NP$ و از آن‌جا که L دلخواه بود داریم $coNP \subseteq NP$؛ حال جهت خلاف آن: برای هر زبان $L \in NP$ با توجه به فرض $P = NP$ خواهیم داشت $L \in P$ و همچنین با توجه به آن که در بخش الف این سوال نشان دادیم که $P \subseteq NP \cap coNP$ پس $L \in coNP$ و در نهایت از آن‌جا که L دلخواه بود نتیجه می‌گیریم که $NP \subseteq coNP$ و در نهایت خواهیم داشت $NP = coNP$ و به این شکل حکم اثبات می‌شود.

■

پ) می‌دانیم که اگر $A \leq_p B$ و $B \in NP$ آن‌گاه $A \in NP$، چراکه اگر A به B در زمان چندجمله‌ای کاهش پیدا کند و برای B بر اساس تعریف NP یک NTM در زمان چندجمله‌ای داشته باشیم، آن‌گاه برای A هم یک NTM در زمان چند جمله‌ای داریم. به این شکل که ورودی را توسط تابع کاهش محاسبه‌پذیر در زمان چندجمله‌ای قطعی f ابتدا به ورودی متناظر آن در زبان B می‌بریم، سپس الگوریتم چندجمله‌ای nondeterministic مربوط به B را اجرا می‌کنیم و بر اساس تعریف *Polynomial Reduction* جواب هرچه باشد جواب درستی نسبت به زبان A خواهد بود.

لِم ۱: نشان می‌دهیم اگر $A \leq_p B$ آن‌گاه $\overline{A} \leq_p \overline{B}$ می‌باشد. اگر f یک Polynomial Reduction از A به B باشد، آن‌گاه f همچنین یک Polynomial Reduction از $\overline{A}$ به $\overline{B}$ است. براساس تعریف داریم:

$$w \in A \Leftrightarrow f(w) \in B$$

از تعریف بالا داریم:

$$w \notin A \Leftrightarrow f(w) \notin B$$

پس:

$$w \in \overline{A} \Leftrightarrow f(w) \in \overline{B}$$

در نتیجه $\overline{A} \leq_p \overline{B}$ برقرار است.

حال ادعا می‌کنیم که اگر $A \leq_p B$ و $B \in coNP$ در نتیجه $A \in coNP$ خواهد بود. اگر $B \in coNP$ در نتیجه $\overline{B} \in NP$ از آن‌جا که $A \leq_p B$ براساس لِم ۱ داریم $\overline{A} \leq_p \overline{B}$ حال براساس بخش اول این اثبات داریم $\overline{A} \in NP$ و در

نهایت براساس تعریف $coNP$ داریم $A \in coNP$ برقرار است.

حال می‌خواهیم نشان دهیم اگر زبان A وجود داشته باشد به شکلی که $A \in NPC \cap coNPC$ آن‌گاه $NP = coNP$ خواهد بود. اگر $A \in NPC$ باشد آن‌گاه هر زبان در NP به زبان A در زمان چندجمله‌ای کاهش پیدا می‌کند یا به عبارتی $Polynomial\ reducible$ به A می‌باشد. از آن‌جا که $A \in coNP$ پس براساس نتیجه‌ی بالا هر زبان در NP درون $coNP$ می‌افتد، پس $NP \subseteq coNP$ خواهد بود. حال برای هر زبان دلخواه $L \in coNP$ بر اساس تعریف می‌دانیم که $\overline{L} \in NP$ و از آن‌جا که $NP \subseteq coNP$ پس $\overline{L} \in coNP$ و براساس تعریف $coNP$ داریم $L \in NP$ و در نتیجه $coNP \subseteq NP$ که نشان می‌دهد در صورت داشتن این فرض $NP = coNP$ خواهد بود و حکم ثابت می‌شود.
■

پرسش ۴

آ) می‌خواهیم نشان دهیم NP نسبت به اجتماع بسته است. اگر $L_1 \in NP$ و $L_2 \in NP$ نشان می‌دهیم $L_1 \cup L_2 \in NP$ برقرار است. اگر این دو شرط برقرار باشد پس برای L_1 و L_2 ماشین‌هایی با نام‌های M_1 و M_2 وجود دارد که آن‌ها را به صورت nondeterministic در زمان Polynomial تصمیم می‌گیرد. ما از این ماشین‌ها برای ساخت ماشین nondeterministic اجتماع این دو زبان استفاده می‌کنیم. نام ماشین اجتماع را $M_{L_1 \cup L_2}$ می‌گذاریم:

$$M_{L_1 \cup L_2}(w) := \begin{cases} \text{1- Run } M_1 \text{ on } w. \text{ If } M_1 \text{ accepts, Accept} \\ \\ \text{2- Run } M_2 \text{ on } w. \text{ If } M_2 \text{ accepts, Accept} \\ \\ \text{3- Otherwise, Reject} \end{cases}$$

در هر دو خط شماره‌ی 1 و 2 ماشین $M_{L_1 \cup L_2}$ از خاصیت nondeterministic بودن خود برای اجرای دو ماشین M_1 و M_2 روی ورودی w استفاده می‌کند. ماشین $M_{L_1 \cup L_2}$ رشته‌ی w را accept می‌کند اگر و تنها اگر یکی از ماشین‌های M_1 یا M_2 رشته را accept کنند. در نتیجه $M_{L_1 \cup L_2}$ به راستی $L_1 \cup L_2$ را به صورت nondeterministic تصمیم می‌گیرد. از آن‌جا که اجرای خط شماره‌ی 1 و 2 هر دو زمان چندجمله‌ای نیاز دارند پس اجرای کل $M_{L_1 \cup L_2}$ روی ورودی w برابر $Poly(n) + Poly(n)$ بوده و در نتیجه $Poly(n)$ است. پس $M_{L_1 \cup L_2}$ مرتبه‌ی زمانی $Poly(n)$ دارد. پس $L(M_{L_1 \cup L_2}) \in NP$ و از آن‌جا که نشان دادیم $L(M_{L_1 \cup L_2}) = L_1 \cup L_2$ در نهایت داریم $L_1 \cup L_2 \in NP$ پس NP تحت اجتماع بسته است.

■

ب) حال می‌خواهیم نشان دهیم NP نسبت به اشتراک هم بسته است. برای نشان دادن آن از همان نام‌ها و تعریف‌های بخش قبلی سوال استفاده می‌کنیم. حال از دو ماشین M_1 و M_2 جهت ساخت یک NTM برای $L_1 \cap L_2$ استفاده می‌کنیم. نام این NTM را $M_{L_1 \cap L_2}$ می‌گذاریم:

$$M_{L_1 \cap L_2}(w) := \begin{cases} \text{1- Run } M_1 \text{ on } w. \text{ If } M_1 \text{ rejects, Reject} \\ \\ \text{2- Run } M_2 \text{ on } w. \text{ If } M_2 \text{ rejects, Reject} \\ \\ \text{3- Otherwise, Accept} \end{cases}$$

با همان استدلال بخش قبل، این NTM از خاصیت nondeterministic بودن خود برای اجرای ماشین‌های M_1 و M_2 استفاده می‌کند. ماشین $M_{L_1 \cap L_2}$ رشته‌ی w را accept می‌کند اگر و تنها اگر که هیچ‌کدام از دو ماشین w را reject نکنند. در این صورت به روشنی $w \in L(M_1)$ و $w \in L(M_2)$ می‌باشد. پس $w \in L_1 \cap L_2$ می‌باشد. همچنین از آن جا که هر دو ماشین در زمان Polynomial اجرا می‌شوند و جمع دو Polynomial بازهم Polynomial است، پس مرتبه‌ی زمانی $M_{L_1 \cap L_2}$ نیز $Poly(n)$ می‌باشد. پیش‌تر نشان دادیم که $L(M_{L_1 \cap L_2}) = L_1 \cap L_2$ پس $L_1 \cap L_2 \in NP$ و در نتیجه NP تحت اشتراک هم بسته است.

■

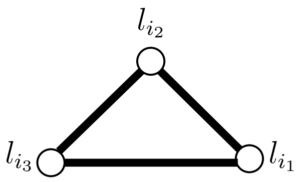
پرسش ۵

می‌خواهیم نشان دهیم مساله‌ی $INDSET$ در کلاس NPC است. در این مساله به دنبال زیرمجموعه‌ای از راس‌ها به اندازه‌ی k هستیم که بین هیچ دو راسی در این زیرمجموعه یال نباشد.

ابتدا ادعا می‌کنیم که $INDEST \in NP$ برای این کار نشان می‌دهیم برای آن یک Verifier در زمان چندجمله‌ای وجود دارد و طول هر Certificate برای این Verifier نیز چندجمله‌ای نسبت به ورودی $\langle G, k\rangle$ است. یک Certificate قبول شده توسط Verifier برای این مساله به سادگی همان زیرمجموعه‌ی k عضوی از $V(G)$ است به شکلی که بین هیچ‌کدام از دو راس آن یال نباشد. اگر $n = |V(G)|$ آن‌گاه هر راس از G با $lg(n)$ بیت قابل نمایش است و از آن‌جا که بزرگترین زیرمجموعه از $V(G)$ خود آن است، پس اندازه‌ی هر Certificate از مرتبه‌ی $O(nlg(n))$ بوده و در نتیجه $Poly(n)$ است. حال برای Verify کردن هر Certificate ابتدا بررسی می‌کنیم به ترتیب هر عضو آن عضوی از $V(G)$ باشد، سپس اگر آن راس، راسی از G نبود آن را reject می‌کنیم در غیر این صورت شمارنده‌ای را که برای چک کردن تعداد این اعضا داریم یک واحد افزایش می‌دهیم. افزایش یک واحدی شمارنده به روشنی مرتبه‌ی زمانی $lg(n)$ داشته و بررسی عضویت هر عضو در $V(G)$ به روشنی با Search ، خطی و چندجمله‌ای است. حال به ازای هر دو راس u و v در این زیرمجموعه چک می کنیم که $(u, v) \in E(G)$ اگر چنین یالی پیدا شد reject می‌کنیم، از آن‌جا که $E(G)$ از مرتبه‌ی $O(|\langle G, k\rangle|)$ است پس هر بار چک کردن از مرتبه‌ی $O(|\langle G, k\rangle|)$ بوده و از آن‌جا که تعداد جفت‌های ممکن حداکثر $O(n^2) = Poly(|\langle G, k\rangle|)$ است، این مرحله نیز در زمان $Poly(|\langle G, k\rangle|).Poly(|\langle G, k\rangle|)$ یعنی $Poly(|\langle G, k\rangle|)$ طول می‌کشد. پس $INDSET \in NP$ ثابت می‌شود.

$\blacksquare$

حال می‌خواهیم نشان دهیم $INDSET \in NPH$ برقرار است. برای این‌کار یک مساله NPH را به آن کاهش چندجمله‌ای می‌دهیم. می‌خواهیم نشان دهیم که $3SAT \leq_p INDSET$ برقرار است. برای این‌کار تابع کاهش f را در ادامه توصیف می‌کنیم.
تابع $f : \phi \mapsto \langle G, k\rangle$ برای هر clause در ϕ به شکل $C_i = (l_{i_1}, l_{i_2}, l_{i_3})$ یک Clause gadget به شکل مثلث تشکیل می‌دهد:



به روشنی Variable gadget ها همان راس‌های این مثلث هستند. حال می‌خواهیم بین Clause gadget ها بر اساس فرمول ϕ ارتباط برقرار کنیم. اگر لیترال l_{i_x} در یک Clause gadget انتخاب شود، می‌خواهیم که انتخاب Negation آن لیترال در زیرمجموعه منجر به این شود که زیرمجموعه‌ی ساخته شده مستقل نباشد. اگر بین هر لیترال و Negation آن یک یال بکشیم، هر مجموعه‌ی مستقل تنها می‌تواند یکی از دو عضو $\{l_{i_x}, \neg l_{i_x}\}$ داشته باشد و نه هردوی آن‌ها را. پس به ازای هر دو لیترال l_{i_x} و l_{j_y} یال (l_{i_x}, l_{j_y}) وجود دارد اگر و تنها اگر که شرط $l_{i_x} = \neg l_{j_y}$ برقرار باشد. برای مثال اگر $l_{i_x} = x_{i'}$ و $l_{j_y} = \neg x_{i'}$ آن‌گاه یال (l_{i_x}, l_{j_y}) را به $E(G)$ اضافه می‌کنیم.

حال ادعا می‌کنیم که f در زمان چندجمله‌ای انجام می‌شود. می‌دانیم که ϕ شامل m عدد Clause است. هر Clause دارای 3 لیترال است. پس G شامل $3m$ نود است. به ازای هر Clause یک K_3 یا مثلث داریم و بین برخی از راس‌های Clause gadget ها نیز یال داریم. در بدترین حالت اگر گرافی با $3m$ نود داشته باشیم تعداد یال‌های آن از مرتبه‌ی $O(m^2)$ است. پس گراف اندازه‌ی $Poly(|\phi|)$ دارد. ابتدا برای هر Clause در زمان چندجمله‌ای یک مثلث ایجاد می‌کنیم. حال برای هر دو نود با برچسب‌های l_{i_x} و l_{j_y} شرط $l_{i_x} = \neg l_{j_y}$ را بررسی می‌کنیم و در صورت برقرار بودن این شرط یال (l_{i_x}, l_{j_y}) را اضافه می‌کنیم. می‌دانیم که به تعداد $O(|\phi|^2) = Poly(|\phi|)$ جفت لیترال ممکن داریم. بررسی این شرط هم که چندجمله‌ای است. پس ساخت یال‌های G نیز در زمان چندجمله‌ای انجام می‌شود. همچنین k در این‌جا همان تعداد Clause ها یعنی m می‌باشد. نگاشت $m \mapsto k$ نیز در زمان چندجمله‌ای انجام می‌شود. پس f در زمان چندجمله‌ای انجام می‌شود.

حال می‌خواهیم نشان دهیم که عبارت زیر برقرار است:

$$\phi \in 3SAT \Leftrightarrow \langle G, m\rangle \in INDSET$$

ابتد نشان می‌دهیم که $\langle G, m\rangle \in INDSET \Rightarrow \phi \in 3SAT$ برقرار است. اگر فرض کنیم که G دارای یک مجموعه‌مستقل (IS) به اندازه‌ی m با نام S است، پس هیچ دو راسی که برچسب آن‌ها به شکل x_i و $\neg x_i$ باشد در مجموعه وجود ندارد. چراکه نحوه ساخت G به شکلی بود که بین این دو راس یال قرار می‌دهد. برای هر متغیر $x_i \in \phi$ سه حالت مجزا وجود دارد. ۱: $x_i \in S, \neg x_i \notin S$ ۲: $\neg x_i \in S, x_i \notin S$ ۳: $x_i \notin S, \neg x_i \notin S$. حال می‌خواهیم از روی S یک Satisfying assignment برای ϕ بسازیم. برای هر $x_i \in \phi$ برای هر سه حالت یک assignment تعیین می‌کنیم و سپس نشان می‌دهیم که این assignment برای همه‌ی متغیرها یک Satisfying assignment است همچنین نام این assignment را c می‌گذاریم. اگر $x_i \in S$ آن‌گاه مقدار x_i را برابر $\top$ قرار می‌دهیم. اگر $\neg x_i \in S$ مقدار آن را برابر $\bot$ قرار می‌دهیم. در غیر این صورت، یعنی $x_i \notin S$ و $\neg x_i \notin S$ در این حالت، جای x_i هرمقداردهی $\top$ یا $\bot$ منجر به Satisfy شدن هر Clause می‌شود. از آن‌جا که $|S| = m$ پس نودهای S از m، Gadget متمایز انتخاب شدند که هر یک متناظر یک Clause است. چراکه بین هر دو راس در یک Gadget که یک مثلث است، یال وجود دارد. پس از هر Clause دلخواه مثل C_i یک لیترال به عنوان برچسب نود در S وجود دارد. برای هر $l_{i_y} \in S$ اگر l_{i_y} فرم Positive داشت، یعنی $l_{i_y} = x_{i'}$ بود، از آن‌جا که $x_{i'} \in C_i$ پس C_i، Satisfy می‌شود. چراکه در c متغیر $x_{i'}$ را با $\top$ مقداردهی کردیم. اگر l_{i_y} فرم Negative داشت، یعنی $l_{i_y} = \neg x_{i'}$ بود، از آن‌جا که $\neg x_{i'} \in C_i$ پس C_i، Satisfy می‌شود. چراکه در c متغیر $x_{i'}$ را با $\bot$ مقداردهی کردیم. پس هر Clause به این شکل Satisfy می‌شود و در نتیجه ϕ هم Satisfy می‌شود. پس $\phi \in 3SAT$ برقرار است. به این ترتیب حکم ثابت می‌شود.

■

حال می‌خواهیم نشان دهیم که $\phi \in 3SAT \Rightarrow \langle G, m\rangle \in INDSET$ برقرار است. اگر $\phi \in 3SAT$ باشد، پس برای فرمول ϕ یک Satisfying assignment با نام c وجود دارد. از آن‌جا که ϕ با مقداردهی c، Satisfy می‌شود، پس هر Clause آن به شکل مستقل Satisfy می‌شود. از آن‌جا که هر Clause دلخواه با نام C_i، Satisfy شده است، پس دست کم درون آن یک لیترال با نام l_{i_y} وجود دارد که با مقداردهی c ارزش $\top$ خواهد داشت. حال مجموعه‌ی مستقل (IS) با نام S را از روی c و ϕ این‌گونه می‌سازیم که از هر Clause با نام C_i یک لیترال $l_{i_y} \in C_i$ را که با مقداردهی c ارزش $\top$ خواهد داشت را انتخاب می‌کنیم و نود متناظر آن را در S قرار می‌دهیم. روشن است که $|S| = m$ چرا که از هر Clause یک لیترال برداشته‌ایم. حال ادعا می‌کنیم که بین هیچ‌کدام از راس‌های S یال وجود ندارد. برای هر دو راس انتخابی l_{i_x} و l_{j_y} در S برچسب این دو راس از دو Clause مجزا می‌باشد، چرا که از هر Clause دقیقا یک لیترال در S وجود دارد. پس یال (l_{i_x}, l_{j_y}) وجود دارد اگر و تنها اگر که $l_{i_x} = \neg l_{j_y}$ برقرار باشد. ولی از آن‌جا که این Assignment یک Satisfying Assignment است پس امکان ندارد این دو انتخاب را باهم داشته باشیم. چراکه ما به ازای هر Clause لیترالی را انتخاب کردیم که با مقداردهی c ارزش $\top$ داشت و اگر هر دوی l_{i_x} و l_{j_y} ارزش $\top$ داشته باشند و شرط $l_{i_x} = \neg l_{j_y}$ نیز برقرار باشد، یعنی متغیر x وجود دارد که c به آن دو ارزش مختلف $\top$ و $\bot$ را نسبت داده است که این با تعریف Assignment هم‌خوانی ندارد. پس یال (l_{i_x}, l_{j_y}) وجود ندارد. در نتیجه S یک مجموعه‌مستقل (IS) با اندازه‌ی m است. در نهایت نشان دادیم که $\langle G, m\rangle \in INDSET$ می‌باشد و حکم اثبات می‌شود.

■

پرسش ۶

مساله‌ی $Exact\ Spanning\ Tree$ (به اختصار EST) این‌گونه تعریف می‌شود که گراف G با وزن آن w و دو عدد l و u داده شده‌اند، می‌خواهیم ببینیم آیا درخت فراگیری از G وجود دارد به شکلی که وزن آن بین l و u باشد. می‌خواهیم نشان دهیم که $EST \in NPC$ برقرار است. این نشان می‌دهد که اگر برای نسخه‌های Minimum و Maximum الگوریتم چندجمله‌ای داشته باشیم نمی‌توان نتیجه گرفت که برای نسخه‌ی Interval آن هم الگوریتم چندجمله‌ای وجود دارد.

ابتدا نشان می‌دهیم که $EST \in NP$ برقرار است. یک $Certificate$ برای این مساله به سادگی همان زیردرخت فراگیر T از G است. از آن‌جا که T زیردرختی فراگیر از G است، پس $V(G) = V(T)$ و هم‌چنین $E(T) \subseteq E(G)$ پس $|T| = Poly(|G|)$ و در نتیجه Certificate اندازه‌ای چندجمله‌ای نسبت به ورودیِ $\langle G, w, l, u\rangle$ دارد.

برای Verify کردن این Certificate ابتدا چک می‌کنیم $V(T) = V(G)$ برقرار باشد. این چک به روشنی از مرتبه‌ی زمانی $Poly(|G|)$ است. حال می‌خواهیم بررسی کنیم که T درخت باشد، یا به عبارتی همبند باشد و دور نداشته باشد. هر دوی این‌ها به آسانی توسط پیمایش DFS قابل فهمیدن است و می‌دانیم که زمان اجرای DFS چندجمله‌ای است. T، Connected است اگر و تنها اگر که در یک بار اجرای DFS همه‌ی نودها ملاقات شوند. همچنین T، Acyclic است اگر و تنها اگر که در DFS-Tree یال back-edge نداشته باشیم. اگر back-edge مثل (u, v) داشته باشیم، آن‌گاه u از نوادگان v است. پس از خاصیت Connected بودن DFS-tree استفاده می‌کنیم، از v به u می‌رویم و توسط یال (u, v) باز می‌گردیم. پس Cycle داریم. برای جهت خلاف آن، اگر Cycle داشته باشیم. اگر اولین نود این Cycle را که در DFS کشف می‌شود u بگیریم، از آن‌جا که از u به همه‌ی راس‌ها در Cycle مسیر است، پس همه‌ی راس‌های ملاقات شده‌ی بعد از آن از نوادگان u خواهند . از آن‌جا که u در یک Cycle است، دست کم یک راس v از نوادگان آن وجود دارد که $(v, u) \in E(G)$ پس براساس تعریف این یال یک back-edge است. به این ترتیب با DFS دو شرط Acyclic بودن و Connected بودن را در زمان چندجمله‌ای چک می‌کنیم. در آخرین مرحله‌ی Verification وزن درخت فراگیر را با l و u مقایسه می‌کنیم. به دست آوردن وزن از روی w محاسبه‌ی یک زیگمای ساده و به روشنی از مرتبه‌ی $Poly(|\langle G, w, l, u\rangle|)$ است. بررسی کوچکترمساوی بودن این وزن از u و بزرگترمساوی بودن آن از l نیز به روشنی چندجمله‌ای است. این مراحل مستقل از هم بوده و هر کدام از مرتبه‌ی $Poly(|\langle G, w, l, u\rangle|)$ می‌باشد. پس این Verifier، Cerificate را در زمان چندجمله‌ای Verify می‌کند. نتیجه می‌گیریم که $EST \in NP$ برقرار است.

■

حال می‌خواهیم نشان دهیم که $EST \in NPH$ است. برای این‌کار یک مساله‌ی NPH را به آن کاهش چندجمله‌ای می‌دهیم. مساله‌ی مورد نظر ما برای کاهش به EST مساله‌ی $SubsetSum$ است که آن را به اختصار SS می‌نامیم. می‌خواهیم نشان دهیم که $SS \leq_p EST$ برقرار است.

حال ماشین محاسبه‌گر تابع f به شکل $f : \langle S, t\rangle \mapsto \langle G, w, l, u\rangle$ را با نام M_f تعریف می‌کنیم:

$$M_f(\langle S, t\rangle) := \begin{cases} V(G) \leftarrow \{v_1, v_2\} \cup S \\ \text{Append } (v_1, v_2) \text{ to } E(G) \\ \text{Set } w[v_1, v_2] = 0 \\ l \leftarrow t \\ u \leftarrow t \\ \\ \text{For each } x_i \in S : \begin{cases} \text{1- Append } (v_1, x_i) \text{ to } E(G) \\ \text{3- Set } w[v_1, x_i] = x_i \\ \text{3- Append } (v_2, x_i) \text{ to } E(G) \\ \text{4- Set } w[v_2, x_i] = 0 \end{cases} \end{cases}$$

الگوریتم محاسبه‌ی f به این شکل است که برای ساخت G ابتدا برچسب‌های راس‌های G را برابر $S \cup \{v_1, v_2\}$ قرار می‌دهیم. دو راس v_1 و v_2 را خودمان اضافه کردیم و برچسب بقیه‌ی راس‌ها اعضای S خواهند بود. یال (v_1, v_2) را با وزن 0 به یال‌ها اضافه می‌کنیم، حال u را برابر l و برابر t قرار می‌دهیم تا هر سه مقدار یکی باشند. حال به ازای هر عضو در S مانند x_i دو یال اضافه می‌کنیم. یال (v_1, x_i) با وزن x_i و یال بعدی یال (v_2, x_i) با وزن 0 خواهد بود.

ادعا می‌کنیم M_f در زمان $Poly(|\langle S, t\rangle|)$ اجرا می‌شود. ساخت گراف در دو مرحله انجام می‌شود. ۱: اضافه کردن $|S| + 2$ راس به گراف، که شامل دو راس کمکی v_1 و v_2 و $|S|$ راس به ازای هر عضو در S می‌باشند. به روشنی ساخت $|S| + 2$ راس از مرتبه‌ی $O(|S|)$ و در نتیجه $Poly(|\langle S, t\rangle|)$ می‌باشد. ۲: اضافه کردن $2|S| + 1$ یال به گراف، به این شکل که ابتدا یک یال با وزن 0 بین v_1 و v_2 قرار دادیم و سپس به ازای هر $x_i \in S$ دو عدد یال اضافه کردیم؛ یکی با وزن 0 و بین دو راس v_2 و x_i و دیگری با وزن x_i و بین دو راس v_1 و x_i بود. اضافه کردن این $2|S| + 1$ یال و وزنشان نیز به روشنی در زمان $Poly(|\langle S, t\rangle|)$ انجام می‌شود. همچنین دو مقداردهی $l \leftarrow t$ و $u \leftarrow t$ نیز در زمان $Poly(|\langle S, t\rangle|)$ انجام می‌شوند. پس M_f در زمان $Poly(|\langle S, t\rangle|)$ محاسبه‌ی خود را انجام می‌دهد.

حال می‌خواهیم نشان دهیم که عبارت زیر برقرار است:

$$\langle S, t\rangle \in SS \Leftrightarrow \langle G, w, l, u\rangle \in EST$$

برای این‌کار ابتدا نشان می‌دهیم که $\langle S, t\rangle \in SS \Rightarrow \langle G, w, l, u\rangle \in EST$ برقرار است. اگر $\langle S, t\rangle \in SS$ پس وجودا دارد $\alpha \subseteq S$ به شکلی که $sum(\alpha) = t$ می‌باشد. حال می‌خواهیم از روی α یک زیردرخت فراگیر از G را با وزن $l = u = t$ بسازیم. نام این گراف را که بعدتر نشان می‌دهیم یک درخت است T می‌گذاریم. یال (v_1, v_2) را با وزن 0 به T اضافه می‌کنیم. حال برای هر راس با برچسب $x_i \in S$ دو حالت وجود دارد. اگر $x_i \in \alpha$ یال (v_1, x_i) را با وزن x_i به T اضافه می‌کنیم و اگر $x_i \notin \alpha$ یال (v_2, x_i) را با وزن 0 به T اضافه می‌کنیم. ادعا می‌کنیم که T درخت است. برای این‌کار ابتدا اثبات می‌کنیم که T، Connected است. برای این‌کار کافیست نشان دهیم بین هر دو راس انتخابی مسیر وجود دارد. اگر دو راس از T انتخاب کنیم این دو راس ۴ حالت دارند:

۱. این دو راس v_1 و v_2 هستند که بین آن‌ها مسیری با وزن 0 با یال (v_1, v_2) وجود دارد.

۲. یکی از این دو راس v_1 و دیگری x_i است. در این حالت دو زیرحالت داریم، اگر $x_i \in \alpha$ پس یال (v_1, x_i) با وزن x_i وجود دارد پس مسیر داریم. اگر هم $x_i \notin \alpha$ پس یال (v_2, x_i) با وزن 0 وجود دارد. مسیر از v_1 به x_i به روشنی برابر (v_1, v_2, x_i) می‌باشد.

۳. یکی از این دو راس v_2 و دیگری x_i است. در این‌جا نیز دو زیرحالت داریم، اگر $x_i \notin \alpha$ پس یال (v_2, x_i) با وزن 0 وجود دارد پس مسیر داریم. اگر هم $x_i \in \alpha$ ار آن‌جا که یال (v_1, x_i) و یال (v_2, v_1) وجود دارد، مسیر از v_2 به x_i وجود دارد.

۴. هر دوی آن‌ها راس‌های ساخته شده از S به شکل x_i و x_j هستند. این حالت چهار زیرحالت دارد. اگر $x_i \in \alpha$ و $x_j \in \alpha$ پس دو یال (v_1, x_i) و (v_1, x_j) وجود دارند. پس به روشنی بین این دو مسیر است. اگر یکی از آن ها عضوی از α باشد و دیگری نباشد. این خود دو زیرحالت است. بدون از دست دادن عمومیت (wlog) فرض کنیم که $x_i \in \alpha$ و $x_j \notin \alpha$ در این زیرحالت دو یال (v_1, x_i) و (v_2, x_j) را داریم. می‌دانیم که یال (v_1, v_2) را نیز داریم. پس بین این دو مسیر است. اگر $x_i \notin \alpha$ و $x_j \notin \alpha$ پس دو یال (v_2, x_i) و (v_2, x_j) وجود دارد. پس بین این دو مسیر است.

در نتیجه T، Connected است.

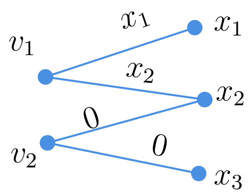
حال می‌خواهیم نشان دهیم که T، Acyclic است. می‌دانیم که هیچ راس با درجه‌ی 1 نمی‌تواند بخشی از یک Cycle باشد. اگر نشان دهیم برای هر راس $x_i \in S$ در T داریم $deg(x_i) = 1$ آن‌گاه نشان دادیم که هیچ راس x_i نمی‌تواند بخشی از یک دور باشد. یال‌هایی که برای هر راس $x_i \in S$ اضافه می‌شوند دو حالت دارند. اگر $x_i \in \alpha$ آن‌گاه یال (v_1, x_i) را داریم و اگر $x_i \notin \alpha$ آن‌گاه یال (v_2, x_i) را داریم. امکان ندارد هر دو شرط باهم درست باشند. پس برای هر $x_i \in S$ داریم $deg(x_i) = 1$ پس اگر T دارای Cycle باشد، این Cycle تنها می‌تواند دارای دو راس v_1 و v_2 باشد یا یک Cycle با یکی از این دو راس باشد. این هم ممکن نیست. چراکه برای Cycle به اندازه‌ی 1 باید $Loop$ داشته باشیم که در گراف وجود

ندارد و برای Cycle تنها شامل v_1 و v_2 باید بیش از یک یال بین v_1 و v_2 وجود داشته باشد. این گراف هم Multi-Edge ندارد. پس T نمی‌تواند Cycle داشته باشد. پس T یک درخت است. همچنین نحوه‌ی ساخت G به شکلی بود که همه‌ی راس‌های $S \ni x_i$ و v_1 و v_2 را شامل شود. پس T شامل همه‌ی راس‌های G است. پس T یک Spanning Tree است.

حال نشان می‌دهیم که وزن T برابر $l = u = t$ است. مشاهده شود که یال‌های با وزن ناصفر T به شکل (v_1, x_i) با وزن x_i هستند. هم‌چنین یال (v_1, x_i) را در صورتی داریم که $x_i \in \alpha$ باشد. پس وزن T برابر است با $\sum_{x_i \in \alpha} x_i = sum(\alpha) = t$ در نتیجه T یک Spanning Tree با وزن $l = u = t$ است. در نهایت $\langle G, w, l, u \rangle \in EST$ را نشان دادیم و حکم اثبات می‌شود.

■

حال می‌خواهیم نشان دهیم $\langle G, w, l, u \rangle \in EST \Rightarrow \langle S, t \rangle \in SS$ برقرار است. اگر $\langle G, w, l, u \rangle \in EST$ پس وجود دارد زیردرختی فراگیر *(Spanning Tree)* از G که وزن آن دقیقا t است نام آن را T می‌گذاریم. توجه شود که در این زیر درخت لزومی ندارد که یال (v_1, v_2) وجود داشته باشند و ممکن است برای یک $x_i \in S$ دو یال (v_1, x_i) و (v_2, x_i) وجود داشته باشد:



همانطور که مشاهده می‌شود درخت بالا دارای وزن $x_1 + x_2$ است ولی راس x_2 درجه‌ی 2 دارد و هر دو یال (v_1, x_2) و (v_2, x_2) در درخت وجود دارند. در حقیقت ما اصلا از این فرض اشتباه که هر دو یال (v_1, x_i) و (v_2, x_i) باهم وجود ندارند، استفاده نمی‌کنیم و ادامه‌ی اثبات بدون این فرض اشتباه انجام می‌شود.

از آن‌جا که $Weight(T) = t$ است و تنها یال‌های ناصفر T به شکل (v_1, x_i) با وزن x_i هستند پس داریم:

$$\left(\sum_{(v_1, x_i) \in E(T)} w[v_1, x_i] \right) = Weight(T) = t$$

حال α را این‌گونه تعریف می‌کنیم:

$$\alpha := \{ w[v_1, x_i] | (v_1, x_i) \in E(T) \}$$

ادعا می‌کنیم که $S \supseteq \alpha$ و $sum(\alpha) = t$ می‌باشد. برای هر عضو $w[v_1, x_i] \in \alpha$ یال $(v_1, x_i) \in E(T)$ وجود دارد و براساس تعریف G می‌دانیم که $w[v_1, x_i] = x_i$ هست و T زیردرختی از G است، پس $(v_1, x_i) \in E(G)$ نیز می‌باشد. براساس تعریف f یال (v_1, x_i) در G است اگر و تنها اگر $x_i \in S$ باشد. پس $x_i \in S$ برقرار است و از آن‌جا که x_i دلخواه بود به گزاره‌ی $\alpha \subseteq S$ خواهیم رسید. هم‌چنین ادعا می‌کنیم که $sum(\alpha) = t$ برقرار است. چون $Weight(T) = t$ می‌باشد و تنها یال‌های با وزن ناصفر G به شکل (v_1, x_i) با وزن x_i هستند، پس براساس تعریف α به عبارت زیر می‌رسیم:

$$sum(\alpha) = \left(\sum_{(v_1, x_i) \in E(T)} w[v_1, x_i] \right) = \left(\sum_{x_i \in \alpha} x_i \right) = Weight(T) = t$$

در نتیجه $\alpha \subseteq S$ و $sum(\alpha) = t$ و درنهایت $\langle S, t \rangle \in SS$ و این‌گونه حکم اثبات می‌شود.

■

پرسش ۷

می‌خواهیم نشان دهیم که اگر $EXP \neq NEXP$ آن‌گاه $P \neq NP$ برقرار است. برای این‌کار عکس‌نقیض *(contra-positive)* آن را اثبات می‌کنیم. [۲] فرض می‌کنیم $P = NP$ است و نشان می‌دهیم که $EXP = NEXP$ خواهد بود. برای زبان دلخواه L اگر $L \in NEXP$ پس $L \in NTIME(2^{n^c})$ برای یک ثابت c خواهد بود. حال زبان L_{pad} را این‌گونه می‌سازیم:

$$L_{pad} = \{\langle x, 1^{2^{|x|^c}}\rangle | x \in L\}$$

ادعا می‌کنیم که $L_{pad} \in NP$ است. برای این‌کار یک NTM برای L_{pad} معرفی می‌کنیم که در زمان $Poly(|\langle x, 1^{2^{|x|^c}}\rangle|)$ این زبان را Decide می‌کند. از آن‌جا که $L \in NTIME(2^{n^c})$ پس برای آن یک NTM با نام M وجود دارد که آن را در زمان 2^{n^c} برای یک ثابت c، Decide می‌کند. حال به کمک M که L را Decide می‌کند، برای L_{pad} یک NTM با نام M' در زمان Polynomial نسبت به ورودی معرفی می‌کنیم:

$$M'(w) := \begin{cases} \text{1- If } w \neq \langle x, 1^{2^{|x|^c}}\rangle \text{ Reject} \\ \\ \text{2- Run } M \text{ on } x \text{ for } 2^{|x|^c} \text{ steps } : \begin{cases} \text{1- } M \text{ accepts, Accept} \\ \text{2- } M \text{ rejects, Reject} \end{cases} \end{cases}$$

می‌دانیم که $|w| = O(2^{|x|^c})$ برقرار است. در ابتدا چک می‌کنیم ورودی به فرم $\langle x, 1^{2^{|x|^c}}\rangle$ باشد. این چک به روشنی در زمان $O(|\langle x, 1^{2^{|x|^c}}\rangle|)$ یعنی چندجمله‌ای نسبت به اندازه‌ی ورودی انجام می‌شود. می‌دانیم که اجرای M روی x به اندازه‌ی $2^{|x|^c}$ زمان می‌برد. شبیه‌سازی نیز سربار لگاریتمی دارد که در مرتبه‌ی M' بی تاثیر است. پس شبیه‌سازی M روی x نیز در زمان چندجمله‌ای نسبت به اندازه‌ی ورودی انجام می‌شود. از آن‌جا که M در زمان $2^{|x|^c}$ اجرا می‌شود. پس از اجرای شبیه‌سازی تا $2^{|x|^c}$ قدم، اجرای M روی ورودی x تمام می‌شود. نتیجه‌ی M روی ورودی x را به عنوان خروجی M' برمی‌گردانیم. دیدیم که $L(M') = L_{pad} \in NP$ می‌باشد.

اگر $P = NP$ پس $L_{pad} \in P$ می‌باشد. در نتیجه برای L_{pad} یک TM در زمان $Poly(|\langle x, 1^{2^{|x|^c}}\rangle|)$ وجود دارد. نام این TM را N می‌گذاریم. حال TM با نام N' را به کمک N به گونه‌ای طراحی می‌کنیم که L را در زمان $O(2^{|n|^c})$ برای یک ثابت c، Decide کند:

$$N'(w) := \begin{cases} \text{1- Construct } \langle w, 1^{2^{|w|^c}}\rangle \\ \\ \text{2- Run } N \text{ on } \langle w, 1^{2^{|w|^c}}\rangle : \begin{cases} \text{1- } N \text{ accepts, Accept} \\ \text{2- } N \text{ rejects, Reject} \end{cases} \end{cases}$$

همانطور که مشاهده می‌شود ماشین N' ابتدا از روی w رشته‌ی $\langle w, 1^{2^{|w|^c}}\rangle$ را در زمان $O(2^{|w|^c})$ می‌سازد. یعنی $2^{|w|^c}$ عدد 1 را به آن Pad می‌کنیم. (*تکنیک استفاده شده در اثبات هم‌Padding نام دارد.*) سپس از TM قطعی N که رشته را در زمان $O(2^{|w|^c})$ اجرا می‌کند استفاده می‌کنیم تا $w \in L$ را Decide کنیم. شبیه‌سازی هم سربار لگاریتمی دارد. پس N' یک TM قطعی است که $w \in L$ را در زمان $O(2^{|w|^c})$، Decide می‌کند. در نتیجه $L \in EXP$ می‌باشد و از آن‌جا که L یک زبان دلخواه در $NEXP$ بود، در نهایت می‌رسیم به $NEXP \subseteq EXP$ که بخش غیربدیهی قضیه را اثبات می‌کند.
■

حال می‌خواهیم نشان دهیم که $EXP \subseteq NEXP$ اگر $L \in EXP$ پس برای آن TM قطعی وجود دارد که می‌تواند آن را در زمان 2^{n^c} برای یک ثابت c، Decide کند. می‌دانیم که برای هر TM قطعی یک NTM وجود دارد که از خاصیت

غیرقطعی بودن خود استفاده نمی‌کند. (*تابع انتقال آن از روی تابع انتقال ماشین قطعی ساخته می‌شود، با این تفاوت که هر عضو از هم‌دامنه‌ی این تابع انتقال به مجموعه‌یی با اندازه‌ی یک تبدیل می‌شود.*) زمان آن هم به روشنی تغییر نمی‌کند. پس $EXP \subseteq NEXP$ و در نهایت حکم اثبات می‌شود.
■

مراجع

[۱] Sipser, M. (2012), *Introduction to the Theory of Computation* ,Cengage learning

[۲] Arora, S. and Barak, B. , *Computational complexity: a modern approach* ,Cambridge University Press