תרגיל בית 1 סיבוכיות

שאלה 1

3. נגדיר באופן רקורסיבי אלגוריתם שבהינתן מעגל $C(x_1, \dots x_n)$ מייצר מעגל חדש $C'(x_1, \dots x_n)$ כך שב$C'(x_1, \dots x_n)$ כל שערי הNOT מחוברים ישירות לקלט ומתקיים $(size(C') = O(size(C))$.
המעגל הראשוני יהיה C, והאלגוריתם יוגדר כסט פעולות שמבצעים על C כדי להפוך אותו לC'.
נתחיל מהשורש של C. עבור צומת כללי במעגל: אם הצומת נוצר על ידי האלגוריתם עצמו, נסיים (כלומר נדלג על הצומת). אם הצומת אינו שער NOT, נסמן את הצומת בp. נסמן את סוג הצומת בAND, OR, NOT ב$type(p)$. אם $type(p) = NOT$, וצומת הקלט של p הוא שער NOT, נמחק את p ונחבר את מה שהיה מחובר לp לצומת הקלט של צומת הקלט של p (כלומר צומת הקלט של שער הNOT האחר). אם $type(p) = NOT$, וצומת הקלט של p אינו קלט אינו שער NOT, נפעיל את האלגוריתם רקורסיבית על צומת הקלט של p. אם $type(p) = NOT$ וצומת הקלט של p הוא צומת קלט, נסיים. אחרת, $type(p) \in \{AND, OR\}$. אם לא קיים צומת NOT שמקבל קלט מp, נפעיל את האלגוריתם רקורסיבית על צמתי הקלט של p. אחרת, נשכפל את p (ניצור צומת q עם $type(q) = type(p)$ שמחובר לצמתי הקלט של p). ננתק מp את צמתי הNOT שמחוברים אליו ונחבר אותם לq. נוסיף צמתי NOT לכל הקשתות שנכנסות לq, כלומר נהפוך את כל הקלטים שלו. נהפוך את q לצומת OR אם $type(p) = AND$ ולצומת AND אם $type(p) = OR$. נמחק את כל צמתי הNOT שיש קשת שיוצאת מq אליהם, ונחבר את הצמתים שמחוברים לצמתי הNOT האלו לq. נפעיל את האלגוריתם באופן רקורסיבי על כל צמתי הקלט של p, q.

הוכחת נכונות: נוכיח באינדוקציה שהמעגל C' מחשב פונקציה זהה לC. נאמר שהאלגוריתם פועל על תת מעגל של C בעומק m (m העומק של תת המעגל עצמו, לא העומק של השורש שלו ביחס לשורש של C) אם הצומת הראשון שהוא מופעל עליו הוא השורש של תת מעגל זה. נגדיר צומת כשורש של תת מעגל אם ורק אם תת המעגל כולל את כל המסלולים מהשורש לקלטים.
הטענה המדויקת תהיה: הפעלת האלגוריתם הרקורסיבי על תת מעגל של C בעומק $k \in \mathbb{N}$ לא תשנה את הפונקציה שC מחשב.

בסיס: הטענה נכונה עבור כל תת מעגל בעומק 1.

צעד: נניח שהטענה נכונה עבור כל תתי המעגלים של C בעומק $\leq k$. נוכיח אותה עבור $k+1$. נניח שהאלגוריתם מופעל על צומת בC בעומק $k+1$. לפי כללי דה-מורגן, והעובדה ש$NOT\big(NOT(x)\big)=$ x, המעגל $C'^{(0)}$ שיווצר בסיום השינויים שיעשו למעגל C בסוף הפעולות שנעשות בקריאה הראשונה לאלגוריתם עצמה, כלומר עד הקריאות הבאות לאלגוריתם לא כולל, יחשב את אותה פונקציה כמו C. הקריאות הרקורסיביות ייעשו עבור תתי מעגלים של C בעלי עומק k ומטה (מהגדרת עומק). לכן לפי הנחת האינדוקציה הקריאות הרקורסיביות לא ישנו את הפונקציה ש$C'^{(0)}$ מחשב, שהיא גם הפונקציה שC מחשב, ולכן לאחר ההפעלה של האלגוריתם הפונקציה שC מחשב לא תשתנה.

לכן המעגל C' מחשב את אותה פונקציה שC מחשב (נובע מהפעלת ההוכחה באינדוקציה עבור הפעלה של האלגוריתם על השורש של C, שהיא ההגדרה של C').

הוכחת גודל: נשים לב שעבור כל צומת NOT שהאלגוריתם הרקורסיבי נתקל בו במהלך הריצה, האלגוריתם אינו מגדיל את המעגל, ועבור כל צומת AND,OR שהאלגוריתם נתקל בו במהלך הריצה, האלגוריתם חוזר מגדיל את המעגל בלכל היותר צומת אחד ו2 קשתות (עבור הקלטים של הצומת). מאחר והאלגוריתם לא יפגוש שער OR,AND פעמיים (כי הוא ממשיך לעומק המעגל ולא הופך שערי AND, OR קיימים לעמוקים יותר במעגל, וידלג על שערים שהוא יצר בעצמו מהגדרה), הוא לכל היותר יכפיל את גודל המעגל פי 3, ולכן $size(C') = O(size(C))$.

הוכחה שכל שערי הNOT מחוברים לקלט: נוכיח באינדוקציה על עומק תת המעגל שהאלגוריתם פועל עליו.

הטענה המדויקת: הפעלת האלגוריתם על תת מעגל של C בעומק k ששורשו הוא שער AND או OR או על תת מעגל בגובה $k+1$ ששורשו הוא שער NOT תסיים כאשר כל שערי הNOT ששייכים למעגל לאחר מכן יהיו מחוברים לקלטים וכנ"ל לגבי כל שערי הNOT שהאלגוריתם ייצור בתהליך.

בסיס: עבור מעגל בעומק 0,1 שערי הNOT מלכתחילה מחוברים לקלט וריצת האלגוריתם על המעגל לא משנה את זה.

צעד: נניח שהטענה נכונה עבור כל תתי המעגלים של C בעומק k. נניח שהאלגוריתם מופעל על שורש של תת מעגל בעומק $k+1$ שהשורש שלו הוא שער AND או OR. הקריאה הרקורסיבית הנוכחית תסתיים בקריאות רקורסיביות לתתי מעגלים בעומק $k+1$ שהשורשים שלהם הם שערי NOT ולתתי מעגלים בעומק k, כאשר שורש המעגל אינו שער NOT יותר, ולכן לפי הנחת האינדוקציה האלגוריתם יעבור. נניח שהאלגוריתם מופעל עבור שורש של תת מעגל בעומק k+2 שהוא שער NOT. במקרה בו שער הNOT קולט משער NOT אחר, האלגוריתם ימחק את שער הNOT הנוכחי ויבצע קריאה רקורסיבית נוספת, ולכן לפי הנחת האינדוקציה הטענה נכונה. אחרת, הקריאה הרקורסיבית לצומת הקלט של שער הNOT תמחק את שער הNOT ותפעיל קריאות רקורסיביות לכל שערי הNOT שהיא תיצור שיהיו שורשי תתי מעגלים בעומק k+1, ולכל צמתי הקלט של השער ממנו שער הNOT קולט, שיהיו שורשי תתי מעגלים בעומק $k-1$, ולכן לפי הנחת האינדוקציה הטענה תהיה נכונה.

לכן מתקיים שכל שערי הNOT בC' מחוברים ישירות לקלטים.

הוכחת עומק: נוכיח ש$deph(C') \leq O(deph(C))$. נשים לב שהאלגוריתם נתקל בכל צומת *NOT,OR,AND* פעם אחת בדיוק. עבור כל צומת *NOT* שהוא ייתקל בו הוא לא יגדיל את עומק המעגל, ועבור כל צומת *AND,OR* שהוא ייתקל בו הוא גם לא יגדיל את עומק המעגל (העבר שערי *NOT* "למטה" לא מגדילה את העומק). לכן בסך הכל נקבל שמתקיים $deph(C') \leq deph(C)$.

שאלה 2

ננסה למנות את כל הפונקציות הבוליאניות לפי סדר. נגדיר $f_i:\{0,1\}^n \to \{0,1\}$ כך:

$$\left(f_i((2^n-1)_2)f_i((2^n-2)_2)\dots f_i(0_2)\right)_2 = i$$

כאשר k_2 הוא הייצוג של המספר k בבסיס 2. בעצם $f_i(m)$ הוא הספרה הm בייצוג הבינארי של *i*.

לכן, נשים לב ליחס הרקורסיבי הבא:

$$f_i(m) = \begin{cases} 1 \ \ i \in [2^m, 2^{m+1}-1] \\ 0 \ \ i < 2^m \\ f_{i-2^{\lfloor \log(i) \rfloor}}(m) \ else \end{cases}$$

נשתמש ביחס הרקורסיה הזה כדי לבנות מעגל כרצוי באופן רקורסיבי. נסמן $C^{(n)}(m)$ מעגל שמחשב את הפונקציות $f_0, \dots f_{2^m-1}$ עבור *n* ביטי הקלט שלנו.

בסיס: עבור הפלט f_0 נחזיר קבוע 0 (ראינו בשאלה 1 ששימוש בקבועים לא מגדיל את המעגל).

צעד: נניח שיש בידינו $C^{(n)}(m)$ עבור $m < 2^n - 1$ כלשהו. נבנה ממנו את $C^{(n)}(m+1)$ בעזרת יחס הרקורסיה שמצאנו. מתקיים לכל $i \in [2^m, 2^{m+1}-1]$:

$$f_i(k) = \begin{cases} 1 \ \ k = m \\ f_{i-2^m}(k) \ else \end{cases} = (k = m) \ or \ (f_{i-2^{m-1}}(k))$$

לכן נבנה את המעגל $C^{(n)}(m+1)$ כך: נתחיל מ$C^{(n)}(m)$. נוסיף פעולות *and* ו*not* בעומק 1 (הנחתי *fan in* בלתי מוגבל, *fan in* 2 לא משנה את התשובה) שיבדקו אם הקלט הנוכחי שווה ל*m+1*. לכל $i \in [2^m, 2^{m+1}+1]$ נבצע פעולת *OR* בין $f_{i-2^m}(input)$ (שנמצאת אצלנו בגלל שהתחלנו מ$C^{(n)}(m)$) לקלט הנוכחי) לשוויון הקלט הנוכחי ל*m+1* (שחישבנו), ונשלח את התוצאה לפלט f_i.

לפי הזהות הרקורסיבית שמצאנו, נקבל שהתוצאה שתתקבל עבור הפלט f_i נכונה בהכרח. לכן בעצם בנינו את $C^{(n)}(m+1)$ (הפלטים $f_0, \dots, f_{2^m-1}$ נכונים מהנחת האינדוקציה מאחר והם יינתנו על ידי המעגל $C^{(n)}(m)$).

נוכיח שגודל המעגל שבנינו חסום על ידי $O(2^{2^n})$ כרצוי:

בכל צעד בבניה, בעצם הוספנו למעגל $O(2^m+n)$ צמתים וקשתות. לכן בסך הכל נקבל נוסחה רקורסיבית לגודל המעגל:

$$S(m+1) = O(2^m+n) + T(n)$$

ולכן:

$$S(m) = S(0) + \Sigma_{k=0}^{m-1} O(2^m+n) = O(2^m+mn)$$

כדי לבנות את המעגל הסופי נבנה רקורסיבית את $C^{(n)}(2^n-1)$. בסך הכל נקבל:

$$SizeOf(C) = O(2^{2^n})$$

כרצוי.

שאלה 3

1. כיוון ->: תהיה $\{C_n\}_n$ משפחת מעגלים שקיימת עבורם מכונת טיורינג M שמקיימת את התנאים שהוגדרו בשאלה. נוכיח ש$\{C_n\}_n$ היא L-יוניפורמית.
נבנה מכונת טיורינג M' שבונה את המעגל C_n בהינתן 1^n. המכונה תעשה את הדבר הבא: בהינתן 1^n כקלט המכונה תמנה את האינדקסים i מ0 עד *n-1* ועבור כל אינדקס תריץ את M ותייצא את הייצוג המתקבל של הקודקוד ה*i* לפלט.
הוכחת נכונות: הפונקציה M מייצאת את הייצוג המלא של הקודקוד ה*i* (*i* מתקבל כקלט למכונה), והמכונה M' מונה את כל האינדקסים ומייצאת לפלט את הפלט של M על כולם ולכן בונה את המעגל C_n, כרצוי.
הוכחת זיכרון: המכונה M שומרת רק את האינדקס של הקודקוד הנוכחי (*logn* מקום) ואת מה שM' שומרת עבור הריצה שלה על האינדקס של הקודקוד הנוכחי בכל רגע נתון ולכן מכיוון שסיבוכיות הזיכרון של M היא גם $logn$ סיבוכיות הזיכרון הכוללת תהיה $O(\log(n))$.
כיוון <-: תהיה $\{C_n\}$ משפחת מעגלים L-יוניפורמית. נוכיח שקיימת מכונת טיורינג M כך שמתקיים:
- $M(1^n, i)$ מחזירה כפלט את השער ה*i* של C_n.
- M רצה ב*log-space*.
מהגדרה קיימת מכונת טיורינג M' כך ש$M'(1^n)$ מחזירה כפלט את המעגל C_n כך שM' רצה ב*log-space*. נגדיר מכונת טיורינג M כרצוי. המכונה $M(1^n, i)$ תריץ את $M'(1^n)$, אבל עבור כל צומת שהייצוג שלו נפלט, אם הצומת הוא הצומת ה*i* המכונה M תעצור ותחזיר את הייצוג של הצומת ה*i*, ואחרת נמחק את הייצוג (הפלט יישמר על סרט העבודה). המכונה M' תדע את מספר הצומת שנפלט בכל רגע נתון על ידי ספירת הצמתים שנפלטו עד כה.
הוכחת נכונות: בהכרח נגיע לצומת ה*i* כל עוד $i \leq n$ ולכן המכונה שהגדרנו בהכרח תעצור ותחזיר את הייצוג של הצומת ה*i*.
הוכחת זיכרון: המכונה רצה עם הזיכרון של M', הזיכרון הנדרש כדי לייצג את המונה שסופר את הקודקודים שנפלטו עד כה והייצוג של קודקוד אחד שנפלט בכל רגע נתון. הזיכרון שנדרש כדי לייצג קודקוד הוא הזיכרון שנדרש כדי לייצג את אינדקסי הצמתים מהם הוא מקבל קלט, שהוא $O(logn)$, והזיכרון הנדרש לייצוג המונה גם הוא $\log(n)$. לכן, מאחר וגם המכונה M' רצה בסיבוכיות זיכרון $O(\log(n))$, קיבלנו שבסך הכל M רצה בסיבוכיות זיכרון $O(\log(n))$, כלומר M רצה ב$log-space$.
2. נוכיח את שתי הטענות:
   a. $U_L - NC^k \subseteq U_L - AC^k \subseteq U_L - NC^{k+1}$:
   ראשית נוכיח ש$\boldsymbol{U_L - NC^k \subseteq U_L - AC^k}$. תהיה $L \in U_L - NC^k$. אזי שקיימת משפחת מעגלים $\{C_n\}_n$ בעלי $fan-in$ מוגבל ועומק $O(\log^k(n))$ ומכונת טיורינג שרצה ב*log-space* ומייצרת את C_n בהינתן קלט 1^n. נשים לב שהמעגלים $\{C_n\}_n$ ומכונת הטיורינג שמתאימה להם הם גם משפחת מעגלים $fan-in$ בעלי עומק $O(\log^k(n))$, וקיימת מכונת טיורינג שמייצרת אותם שרצה ב$log-space$. לכן קיבלנו בסך הכל ש-$L \in U_L - AC^k$. לכן $U_L - NC^k \subseteq U_L - AC^k$.

   עתה נוכיח ש$\boldsymbol{U_L - AC^k \subseteq U_L - NC^{k+1}}$. תהיה $L \in U_L - AC^k$ אזי שקיימת משפחת מעגלים $\{C_n\}_n$ עם $size(C_n) = poly(n)$ ו$deph(C_n) = O(\log^k(n))$ שמיוצרת על ידי מכונת טיורינג שרצה ב*log-space*, M, כך ש$\{C_n\}_n$, פותרת את השפה. נבנה משפחת מעגלים $\{C'_n\}_n$ שפותרת את השפה עם *fan-in* חסום שפותרת את השפה. נשים לב שניתן לבנות שער *OR* עם *fan-in* לא חסום בעזרת עץ של שערי *OR* עם *fan-in* 2, ובאופן זהה ניתן לבנות שער *AND* עם *fan-in* לא חסום בעזרת עץ של שערי *AND* עם *fan-in* 2. עבור כל מעגל C_n נבנה מעגל C'_n באופן הבא: את כל השערים ה"רגילים" (*OR*, *AND*, ו*NOT* בעלי *fan-in* של 2 ו1) נשאיר, ואת השערים בעלי *fan-in* גדול מ2 נחליף בעצים כמו שתיארנו. מכיוון שהעצים מבצעים פעולה זהה לצמתים המקוריים, משפחת המעגלים $\{C'_n\}_n$ פותרת

את L. מאחר ו$size(C_n) = poly(n)$ ולכן לכל צומת בC_n דרגת כניסה לכל היותר $poly(n)$ ועומק עץ כמו שתיארנו הוא $\log(fan-in)$, העומק של כל עץ שהוספנו הוא $\log(poly(n)) = O(\log(n))$. לכן עומק מסלול במעגל החדש C'_n הוא לכל היותר:

$$Length(Path') \leq O(\log(n)) * Length(Path) \leq O(\log(n)) * deph(C_n)$$
$$= O(\log(n)) * O(\log^k(n)) = O(\log^{k+1}(n))$$

ולכן $deph(C'_n) = O(\log^{k+1}(n))$. לכל היותר החלפנו כל צומת בC_n במספר פולינומיאלי של צמתים (בעץ) ולכן בC_n' יש לכל היותר $poly(n) * poly(n) = poly(n)$ צמתים. לכן $size(C'_n) = poly(n)$.

כדי לייצר את משפחת המעגלים C'_n בעזרת מכונת טיורינג, נבנה מכונה M' שבהינתן מעגל C_n מחזירה את C'_n ועובדת ב*log(n)* ואז נשתמש בהרכבת זיכרון כדי להוכיח שקיימת מכונה שמייצרת את C_n' ועובדת ב$log-space$.

נגדיר את M' כך: בהינתן מעגל C_n, נעבור על כל הצמתים שלו. עבור כל צומת נמנה את הבנים שלו. אם יש לו $fan-in$ 2, נעתיק אותו לסרט הפלט. אחרת, נחליף אותו בעץ כך: ראשית נשמור את מספר הבנים של הצומת. נתחיל משורש ואז נבנה באופן לא רקורסיבי את רמות העץ אחת אחרי השניה (נבנה כל רמה כרמה מלאה) (ניתן לעשות זאת בשיטה דומה לייצוג של ערימה בינארית במערך, כאשר האינדקס במערך יהיה אינדקס כל צומת בעץ), עד שכמות העלים ברמה הבאה תהיה גדולה מכמות צמתי הקלט של העץ. לאחר מכן נחבר את צמתי הקלט לעלים, ואם יש יותר קלט נדרש לרמה האחרונה בעץ מאשר צמתי קלט נחבר את צומת הקלט האחרון לכל העלים שנשארו. ניתן לעשות זאת בזיכרון $O(\log(n))$ מאחר וצריך לשמור רק את כמות צמתי הקלט, שדורשת זיכרון $\log(poly(n)) = O(\log(n))$, את כמות הצמתים ברמה הבאה בעץ שדורשת $\log(\#inputs) = \log(poly(n)) = \mathrm{O}(\log(n))$, ואת אינדקס הצומת הנוכחי בעץ שדורש $\log(poly(n)) = O(\log(n))$ זיכרון. בסך הכל נקבל שהמכונה M' שהגדרנו דורשת את הזיכרון שדורשת המכונה *M* וזיכרון $O(\log(n))$ נוסף ולכן בסך הכל היא עובדת בסיבוכיות $O(\log(n))$. לכן מהרכבת זיכרון קיימת מכונת טיורינג שמקבלת כקלט 1^n, רצה ב*log-space* ומחזירה כפלט את C'_n. לכן בסך הכל קיבלנו ש$L \in U_L - NC^{k+1}$. לכן בסך הכל קיבלנו ש$U_L - AC^k \subseteq U_L - NC^{k+1}$.

b. נוכיח ש$\boldsymbol{U_L - NC^k \subseteq DSPACE(\log^k(n))}$: תהיה $L \in U_L - NC^k$. מהגדרה קיימת משפחת מעגלים $\{C_n\}_n$ כך ש$size(C_n) = poly(n)$ ומתקיים $deph(C_n) = O(\log^k(n))$ ומכונת טיורינג *M* שבהינתן 1^n בונה את C_n ורצה ב*log-space*. נוכיח שבהינתן מעגל C_n קיימת מכונת טיורינג שמריצה אותו בסיבוכיות זיכרון $O(\log^k(n))$, ולאחר מכן הטענה שאנחנו רוצים להוכיח נובעת ישירות מהרכבת זיכרון. נגדיר את המכונה המבוקשת M' כך:

בהינתן קלט $< C_n, x >$ המכונה M' תחשב את $C_n(x)$ באופן רקורסיבי על ידי מעבר רקורסיבי על צמתי העץ (כמו ב*DFS* למשל) וחישוב הערך של הצומת הנוכחי באופן רקורסיבי (חישוב הערך של הבנים באופן רקורסיבי, ואז חישוב הערך של הצומת הנוכחי בעזרת ערכי הבנים). כמות הזיכרון הנדרשת לכך היא כמות הזיכרון הנדרשת לייצוג ערכי הבנים ($O(1)$ מכיוון שה*fan in* חסום), ועוד כמות הזיכרון שנדרשת לייצוג ערך הצומת הנוכחי (*O(1)*) ועוד כמות הזיכרון הנדרשת לזכור מהו המסלול המלא מהשורש על הצומת הנוכחי (כדי לחזור להורה של הצומת בסיום חישוב ערך הצומת הנוכחי). כדי לזכור את המסלול המלא מהשורש עד הצומת הנוכחי, ניתן פשוט לזכור את הדרך שעשינו עד לצומת הנוכחי – כלומר עבור כל צומת במסלול האם עברנו ממנו לצומת הקלט הראשון או השני (עבור שערי *NOT* יש רק צעד אחד אפשרי ואז זה לא משנה). הדבר דורש זיכרון $O(deph(C_n)) = O(\log^k(n))$. לכן בסך הכל קיבלנו שמכונת הטיורינג M' מחשבת את $C_n(x)$ בסיבוכיות זיכרון $O(\log^k(x))$ בהינתן קלט $< C_n, x >$. בסך

הכל מהרכבת זיכרון עם המכונה M שמחשבת את $< C_n >$ ב*log-space* בהינתן קלט 1^n, נקבל שקיימת מכונה M'' שמקבלת כקלט x (את המעגל המתאים לגודל של x ניתן להסיק מהאורך של x, בפרט ניתן ליצור מכונה שמייצרת את $C_{|x|}$ על ידי שימוש בM והתעלמות מאם האותיות מהן מורכב x הן 0 או 1), מקבלת אם ורק אם $x \in L$ ורצה בסיבוכיות זיכרון $O(\log^k(n))$. בסך הכל קיבלנו שמתקיים $U_L - NC^k \subseteq DSPACE(\log^k(x))$.

<u>שאלה 5</u>

נבנה מכונת טיורינג M כך ש$L(M) = FVAL$. המכונה תבצע את האלגוריתם הרקורסיבי הבא:
נסמן את ביצוע האלגוריתם עבור צומת p ופרמטר בוליאני x ב-$A(p,x)$. המשתנה x יכול לקבל אחד משלושה ערכים: $true, false, unknown$ שמסמנים את הערך של הצומת p. לשם פשטות נסמן מעתה $f(false) = AND, f(true) = OR$. נגדיר את $A(p,x)$: אם p הוא צומת קלט, נחזיר את ערך הקלט. אחרת: אם $x \in \{true, false\}$ והצומת האב (קיים כזה בהכרח כי $fan-out(p) = 1$) של x הוא מסוג $f(x)$, נמחק את נתוני הריצה הנוכחית מסרט העבודה ונריץ את $A(father(p),x)$.
אם $x \in \{true, false\}$ ו-p הוא הבן האחרון של הצומת האב שלו לפי הסדר בו הבנים נתונים בקלט, נמחק את נתוני הריצה הנוכחית מסרט העבודה ונריץ את $A(father(p),x)$. אם
$x \in \{true, false\}$, p אינו הבן האחרון של הצומת האב שלו לפי הסדר בקלט, ומתקיים $type(father(p)) \neq f(x)$, נמחק את נתוני הריצה הנוכחית ונריץ את A על הצומת הבן הבא בתור של הצומת האב של p עבור $x = unknown$. אחרת, $x = unknown$. במקרה זה, נמחק את נתוני הריצה הנוכחית ונריץ את A על הבן הראשון של p לפי הסדר בו הבנים של p מופיעים בקלט עבור $x = unknown$.

הוכחת נכונות:

נוכיח את הטענה הבאה באינדוקציה: $A(p, unknown)$ לבסוף מפעיל את $A(p, eval(p))$. (נסמן $eval(p)$ הוא הערך שהצומת p מקבל בהרצת הנוסחה הנתונה בקלט על הקלט הנתון שלה).

בסיס: עבור מעגל שמכיל רק קלט אחד שמחובר לפלט אחד ניתן לראות שהטענה מתקיימת כי יופעל $A(root, unknown) \to A(input, unknown) \to A(root, eval(input))$ ואז יוחזר הפלט הרצוי.
צעד: נניח שהטענה נכונה לכל המעגלים עד עומק *k*. נוכיח שהיא נכונה למעגלים בעומק *k+1*.

נוכיח באינדוקציה טענת עזר: עבור צמתים p שיש להם צומת אב שנמצאים בגובה (מרחק מקסימלי מעלה) עד *k-1*, הפעלת $A(p, unknown)$ על הצומת p תפעיל לבסוף את $A(father(p),x)$ כאשר x הוא ערך הפעלת השער שמייצג האב של p על כל הצמתים הבנים שלו מp והלאה לפי סדר הופעת הבנים בקלט כולל p.
בסיס: אם p הצומת הבן האחרון של האב שלו, לפי הנחת האינדוקציה $A(p, unknown)$ מפעיל את $A(p, eval(p))$, שיפעיל את $A(father(p),x)$ כרצוי (כלומר x יהיה כפי שדרשנו).
צעד: נניח שהטענה נכונה לכל צומת p' שהוא הצומת הבן הk האחרון של הצומת האב שלו לפי סדר הבנים בקלט. נוכיח שהטענה נכונה לכל צומת p שהוא הבן הk האחרון של צומת האב שלו.
לפי הנחת האינדוקציה (של האינדוקציה הראשית), הפעלת $A(p, unknown)$ תפעיל את $A(p, eval(p))$. אם סוג השער של האבא של p שווה ל$f(eval(p))$ אזי שנפעיל את $A(father(p), eval(p))$ שמהגדרת AND, OR (לצומת *NOT* יש רק בן אחד ולכן הוא לא רלוונטי) שווה ל-$A(father(p), eval(father(p)))$. אחרת, האלגוריתם יפעיל את
$A(next-son(p), unknown)$ ולכן מהנחת האינדוקציה יופעל לבסוף $A(father(p),x)$ כאשר x מתאים ל-*next-son(p)*. אבל, נשים לב שאם סוג השער של הצומת האב של p לא שווה ל$f(p)$, הערך x שמתאים לp זהה לערך שמתאים לבן הבא אחרי p, ולכן יופעל $A(father(p), x_p)$.
(סוף הוכחת טענת עזר).

האלגוריתם יקרא ל-*A* עבור הבן הראשון של p לפי סדר הופעת הבנים של p בקלט עבור *x=unknown*, ולכן לפי טענת העזר יופעל לבסוף $A(p, eval(p))$, כרצוי.

לפי הטענה שהוכחנו באינדוקציה, הפעלת $A(root, unknown)$ לבסוף תגרום להפעלת $A(root, eval(root)) = A(root, Circut(input))$ ולכן מכונת טיורינג שמריצה את $A(root, unknown)$, שהיא בעצם האלגוריתם שהצענו לפתרון הבעיה, תפתור את השפה $FVAL$.

הוכחת זיכרון: נרשום את הזיכרון שהמכונה שהגדרנו משתמשת בו: הזיכרון הנדרש לייצג את הצומת הנוכחי p (אינדקס כלומר $O(\log(n))$ זיכרון), במעבר לצומת הבא נדרש רגעית לייצג גם את

הצומת הבא (אינדקס, כלומר $O(\log(n))$ זיכרון), הזיכרון שנדרש לייצג את *x* (*O(1)*). בסך הכל קיבלנו שהמכונה שהגדרנו משתמשת בזיכרון $O(\log(n))$.

בסך הכל קיבלנו ש*FVAL* פתירה על ידי מכונת טיורינג שרצה בסיבוכיות זיכרון $O(\log(n))$ ולכן $FVAL \in Log = DSPACE(\log(n))$.

שאלה 6

נראה אלגוריתם רקורסיבי שמחשב את $BinMult_n$ בסיבוכיות זמן $O(n^{\log_2(3)})$.
יהיו $x, y \in \{0,1\}^n$ שני מספרים בעלי n ספרות בבסיס 2. נסמן $d^{(i)}(z)$ הספרה במקום הi של המספר z בייצוג בינארי. נסמן $n(z)$ מספר הספרות של x בייצוג בינארי.
עתה נסמן:

$$x_1 = \left(d^{\left(\frac{n(x)}{2}\right)}(x) \dots d^{(1)}(x) d^{(0)}(x)\right)_2$$

$$x_2 = \left(d^{(n(x))}(x) d^{(n-1)}(x) \dots d^{\left(\frac{n(x)}{2}+1\right)}(x)\right)_2$$

$$y_1 = \left(d^{\left(\frac{n(y)}{2}\right)}(y) \dots d^{(1)}(y) d^{(0)}(y)\right)_2$$

$$y_2 = \left(d^{(n(y))}(y) d^{(n-1)}(y) \dots d^{\left(\frac{n(y)}{2}+1\right)}(y)\right)_2$$

כאשר $n/2$ מעוגל למטה. נשים לב שבמקרה שלנו $n(x) = n(y) = n$. נגדיר $z|w$ המספר המתקבל מלשים את z ואת w (שני מספרים בינאריים) זה לצד זה. נקבל שמתקיים:

$$x \cdot y = \left(x_1 + x_2 \cdot 2^{\frac{n}{2}}\right)\left(y_1 + y_2 \cdot 2^{\frac{n}{2}}\right) = x_1 y_1 + (x_1 y_2 + x_2 y_1) \cdot 2^{\frac{n}{2}} + x_2 y_2 \cdot 2^n$$
$$= x_1 y_1 + \left((x_1 + x_2)(y_1 + y_2) - x_1 y_1 - x_2 y_2\right) \cdot 2^{\frac{n}{2}} + x_2 y_2 \cdot 2^n$$

כדי לחשב את $x \cdot y$, נשתמש באלגוריתם הרקורסיבי הבא:

ראשית נחשב רקורסיבית את $x_1y_1, x_2y_2, (x_1 + x_2)(y_1 + y_2)$, ואז נחשב מהם את $x \cdot y$ בעזרת הביטוי לעיל (מדובר בפעולות חיבור בלבד ולכן חישוב הביטוי לעיל בהינתן המכפלות ייקח $O(n)$ בלבד. אם שני המספרים באורך ביט 1 נכפיל לפי לוח הכפל עבור שני ביטים.

הוכחת נכונות: נוכיח באינדוקציה:

בסיס: עבור זוג מספרים באורך ביט אחד, האלגוריתם עובד מהגדרה.

צעד: נניח שהאלגוריתם עובד לכל $x, y \in \{0,1\}^m$ $such\ that$ $m \leq k$. נוכיח שהוא עובד לכל x, y באורך לכל היותר $k + 1$. לפי הנחת האינדוקציה החישוב הרקורסיבי יעבוד עבור $x_1y_1, x_2y_2, (x_1 + x_2)(y_1 + y_2)$, ולכן לפי הביטוי למעלה, נקבל תוצאה נכונה עבור $x \cdot y$.

לכן לכל $n \in \mathbb{N}$ לכל קלט $x, y \in \{0,1\}^n$, האלגוריתם יחזיר את $x \cdot y$.
הוכחת סיבוכיות: האלגוריתם מבצע 3 קריאות רקורסיביות עבור קלטים באורך $n/2$, פעולות חיבור (שמוסיפות סיבוכיות זמן $O(n)$) והכפלות ב2^n, שדורשות גם הן $O(n)$ זמן כי כדי להכפיל מספר בינארי בחזקה שלמה של 2, צריך רק להסיט אותו במאריך של החזקה שמאלה. בסך הכל נקבל נוסחת נסיגה:

$$T(n) = O(n) + 3T(\frac{n}{2})$$

ולכן לפי משפט המאסטר מתקיים $T(n) = O(n^{\log_2(3)})$. לכן האלגוריתם שהגדרנו רץ בסיבוכיות זמן $O(n^{\log_2(3)})$.

בסך הכל קיבלנו ש$BinMult_n \in DTIME(n^{\log_2(3)})$.

שאלה 7

ראשית נגדיר מעגל שמבצע חיבור בעומק חסום באופן רקורסיבי. נגדיר C_n מעגל שמחבר שני מספרים בני *n* ספרות בבסיס 2. נגדיר את C_n באופן רקורסיבי. ראשית עבור זוג מספרים בני ספרה אחת נגדיר את C_0: "אם 0,0 החזר 0, אם 0,1 או 1,0 החזר 1, אם 1,1 החזר 10".
עתה נוכיח טענת עזר:

יהיו $x, y \in \{0,1\}^n$ זוג מספרים בינאריים בני *n* ספרות. נסמן
$x = (x_{n-1}x_{n-2} \dots x_0), y = (y_{n-1}y_{n-2} \dots y_0)$. עתה נגדיר "חיבור יבש" של *x* ו-*y* באופן הבא:

$$dryAdd(x, y) = (OR(x_{n-1}, y_{n-1})OR(x_{n-2}, y_{n-2}) \dots OR(x_0, y_0))$$

נוכיח באינדוקציה את הטענה הבאה: בחיבור של *x* ו-*y*, יהיה נשא עבור הספרה הk אם ורק אם קיים $m \in \{0,1, \dots, k\}$ כך שלכל $m \le s < k$ מתקיים $dryAdd(x, y)_s = 1$ וגם מתקיים $x_m = y_m = 1$.

בסיס: עבור $k = 0$ לא יהיו נשאים ולכן הטענה נובעת באופן ריק.

צעד: נניח שהטענה נכונה עבור *k*. נוכיח שהטענה נכונה עבור *k+1*.
כיוון ראשון: נניח שיש נשא עבור הספרה ה*k+1* בחיבור של *x,y*. נחלק לשני מקרים:

$x_k = y_k = 1$: במקרה זה הטענה נכונה ומתקיים $m = k$.
$(x_k, y_k) = (1,0)$ או $(x_k, y_k) = (0,1)$: אזי שלפי היה נשא עבור הספרה הk בחיבור, ומתקיים הנחת האינדוקציה קיים *m* כך שלכל $m \le s < k$ מתקיים $dryAdd(x, y)_s = 1$ והיה נשא בחיבור של *m*. לפי הגדרת מקרה זה מתקיים $(x_k, y_k) = (1,0)$ או $(x_k, y_k) = (0,1)$ ולכן $dryAdd(x, y)_k = 1$. לכן לכל $m \le s < k + 1$ מתקיים $dryAdd(x, y)_s = 1$ ולכן הטענה מתקיימת.

נגדיר באופן רקורסיבי מעגל C_n שמבצע חיבור בין שני מספרים:
נתחיל מהמעגל C_{n-1}. עתה נוסיף את החיבור עבור הספרה ה-*n*. נסיר את הקשת שמתחברת לפלט שמייצג את הספרה ה*n* בחיבור. נוסיף צומת *OR* שמחשב את $dryAdd(x, y)_n$ ישירות מx_n, y_n.
מאחר והבנייה הרקורסיבית עבור כל C_n מוסיפה את החיבור היבש עבור הצומת ה-*n* (נגדיר מחדש את C_0, ונוסיף לו את חישוב החיבור היבש של x_0, y_0). מכיוון שהבנייה הרקורסיבית מתחילה מC_0, וכל צעד ברקורסיה מחשב את החיבור היבש עבור הספרה הבאה, כל C_n בעצם מחשב את החיבור היבש המלא של x, y, ולכן תוצאת החיבור היבש תהיה זמינה לנו בהמשך הבניה. לכל $k \in \{1, \dots n\}$ נוסיף למעגל צומת *AND* שמחובר לכל התוצאות
$dryAdd(x, y)_k, dryAdd(x, y)_{k+1}, \dots dryAdd(x, y)_{n-1}$, ונחבר את הפלט שלו לצומת *AND* שמחובר גם ל-x_k, y_k (נסמן צומת זה בz_k). לאחר מכן נוסיף צומת *OR* שמחובר לכל צמתי הz_k שהגדרנו בצעד הנוכחי של הבניה (לפי טענת העזר, צומת זה יכיל את התשובה לשאלה: "האם יש נשא עבור הספרה ה*n* בחיבור?"). נסמן צומת זה (ה*OR*) ב*Carry*. נחבר לצומת הפלט שמייצג את הספרה ה*n* בחיבור את המעגל המחשב את $XOR(Carry, x_n, y_n)$. נחבר לצומת הפלט שמייצג את הספרה ה*n+1* בחיבור את המעגל שמחשב את
$OR(AND(Carry, x_n), AND(Carry, y_n), AND(x_n, y_n))$.

הוכחת נכונות: נוכיח באינדוקציה:

בסיס: עבור $n = 1$ הבנייה נכונה מהגדרה.

צעד: נניח שהבנייה נכונה עבור *n*. אזי שנכונות הבניה עבור $n + 1$ נובעת באופן ישיר מטענת העזר שהוכחנו (השינויים שעשינו כדי להגיע מC_n לC_{n+1} לא משפיעים על התוצאות עבור כל הספרות מלבד *n+1*, *n+2*, ולכן תוצאות אלו יישארו נכונות כיוון שהוספת הספרה ה*n* לא משפיע על התוצאה עבור הספרות האלו).

הוכחת גודל: עבור כל צעד בבנייה הרקורסיבית, עבור כל k, הוספנו לכל היותר $O(n)$ קשתות. לכן בסך הכל בכל צעד בבנייה הוספנו לכל היותר $O(n^2)$ לגודל המעגל. נקבל נוסחת נסיגה $Size_{n+1} = Size_n + O(n^2)$ ולכן בסך הכל $Size_n = O(n^3)$.
הוכחת עומק: נשים לב שבכל צעד בבנייה, לא הוספנו לעומק המעגל (כי תוצאת הביט הבא בחיבור היבש מחושבת בעומק $O(1)$ בכל צעד בבנייה), כלומר עומק שני צעדים עוקבים בבנייה זהה. לכן עומק המעגל יהיה בסך הכל $O(1)$.

הוכחנו שקיימת משפחת מעגלים $\{C_n\}_n$ שמבצעת חיבור בעומק $O(1)$ ובגודל פולינומי. נסמן משפחה כזו ב-$\{Add_n\}_n$. באופן דומה ניתן לבצע חיסור ב*O(1)* למשל על ידי העברת הייצוג לשיטת המשלים ל-2, ביצוע פעולת החיסור בייצוג זה (בו היא נעשית כמו פעולת חיבור רגילה $(x - y = x + (-y)$) ואז העברה חזרה. ניתן לבצע חישוב $-x$ והמרה לשיטת המשלים ל2 ומשיטה זו בעומק *O(1)*, מאחר ופעולות אלו מורכבות מהפיכת הביטים של המספר והוספת 1.

עתה נבנה באופן רקורסיבי מעגל בעומק לוגריתמי (וגודל פולינומי) שמחשב את $BinMult_n$.
עבור $n = 1$ נבנה את המעגל לפי לוח הכפל בבסיס 2.

עבור n כללי, נבנה את המעגל עבור $BinMult_{n+1}$ באופן הבא:
נסמן את הקלטים ב$x, y \in \{0,1\}^{n+1}$. נשתמש בסימונים x_1, x_2, y_1, y_2 משאלה 6. ראינו בשאלה הקודמת שמתקיים $x \cdot y = x_1y_1 + \left((x_1 + x_2)(y_1 + y_2) - x_1y_1 - x_2y_2\right) \cdot 2^{\frac{n}{2}} + x_2y_2 \cdot 2^n$.
בנוסף הכפלה בחזקה שלמה של 2 ניתן לבצע בעומק $O(1)$ מאחר ומדובר פשוט בהזזה שמאלה של המספר בקבוע. כדי לחשב את $BinMult_{n+1}$ נשתמש במעגל הבא:

ראשית נחשב את $x_1y_1, x_2y_2, (x_1 + x_2)(y_1 + y_2)$ (את פעולות החיבור נבצע בעומק $O(1)$) באופן רקורסיבי (בעזרת המעגלים המתאימים $BinMult_{\frac{n}{2}}, BinMult_{\frac{n}{2}+1}$ כאשר חלוקה ב2 מעוגלת למטה).
לאחר מכן לפי מה שראינו (חיבור וחיסור בעומק $O(1)$ בגודל פולינומיאלי) ניתן לחשב את $x \cdot y$ בעזרת הביטוי מהשאלה הקודמת בעומק $O(1)$ כיוון שהוא מורכב מפעולות חיבור וחיסור בלבד.

הוכחת נכונות: נוכיח באינדוקציה.

בסיס: עבור קלטים בעלי ספרה אחת בבסיס 2 הטענה נכונה מהגדרת המעגל $BinMult_0$.

צעד: נניח שהטענה נכונה עבור כל גדלי הקלטים עד n. אזי שלפי הזהות $x \cdot y = x_1y_1 + \left((x_1 + x_2)(y_1 + y_2) - x_1y_1 - x_2y_2\right) \cdot 2^{\frac{n}{2}} + x_2y_2 \cdot 2^n$, אותה הוכחנו בשאלה הקודמת, הבניה עבור $BinMult_{n+1}$ תחזיר תוצאה נכונה כיוון ש$BinMult_{\frac{n}{2}}, BinMult_{\frac{n}{2}+1}$ יחזירו תוצאה נכונה לפי הנחת האינדוקציה.

הוכחת גודל: בכל צעד בבנייה הוספנו למעגל גודל $O(n^3)$ בגלל פעולות החיבור. לכן גודל המעגל הכולל יהיה לכל היותר $n \cdot O(n^3) = O(n^4) = poly(n)$.

הוכחת עומק: בכל צעד בבנייה הרקורסיבית הוספנו $O(1)$ עומק וקראנו למעגלים בעומק $Deph(BinMult_{\frac{n}{2}})$. לכן נקבל נוסחת נסיגה:

$$Deph(BinMult_n) = O(1) + Deph(BinMult_{\frac{n}{2}})$$

ולכן בסך הכל נקבל $Deph(BinMult_n) = O(\log(n))$.

לכן בסך הכל קיבלנו שקיימת משפחת מעגלים שמחשבת את $BinMult_n$ בגודל פולינומי ובעומק $O(\log(n))$, בעזרת שערי and, or, not בעלי *fan-in* לא חסום, ולכן מהגדרה מתקיים:

$$BinMult_n \in AC^1$$