מבני נתונים 2020 סמסטר ב מועד ב

שאלה 1

ראשית נוכיח ש$D(n) = O(n)$. אכן $n = size_{left} + size_{right} + 1$ ולכן נקבל ש –

$$D(n) = size_{left} - size_{right} \leq size_{left} + size_{right} + 1 = n$$

נראה שיש עץ שבו $D(n) = \Omega(\mathrm{n})$. נגדיר עץ AVL עם n צמתים שיש לו שורש, והתת עץ השמאלי שלו הוא עץ מלא בגובה $h + 1$ והתת עץ הימני שלו הוא עץ מלא בגובה h (ואת שארית הצמתים אם קיימת נוסיף לתת העץ השמאלי פרט לצומת אחת בתת העץ הימני כדי שהפרש הגבהים עדיין יהיה 1). נקבל שמתקיים $n = 2^{h+1} + 2^h + 3 + k$ (כאשר k השארית) ונשים לב שזהו עץ AVL חוקי. עכשיו נשים לב ש –

$$D(n) = (2^{h+1} + 1 + k - 1) - (2^h + 2) = 2^h + k - 2 \overset{*}{\geq} \frac{n}{3} - 5 = \Omega(\mathrm{n})$$

*הסיבה היא רצף הפעולות האלגבריות הבא:

$$n = 2^{h+1} + 2^h + 3 + k \Leftrightarrow n = 3 \cdot 2^h + k + 3 \Leftrightarrow n - 5 - 2^{h+1} = 2^h + k - 2$$

$$n = 2^{h+1} + 2^h + 3 + k \Leftrightarrow n = 1.5 \cdot 2^{h+1} + k + 3 \Longrightarrow n \geq 1.5 \cdot 2^{h+1} \Longrightarrow n \geq 1.5 \cdot 2^{h+1} \Longrightarrow \frac{2n}{3} \geq 2^{h+1}$$

ולכן $2^h + k - 2 = n - 5 - 2^{h+1} \geq n - 5 - \frac{2n}{3} = \frac{n}{3} - 5$

שאלה 2

א. ניתן. נוכיח שמיון הכנסה עובד ב$O(n)$ במקרה זה. נסמן ב$N(j)$ את כמות ההיפוכים שעשינו עם $A[j]$ כאשר הכנסנו אותו למקום המתאים במערך הממוין $A[0,\dots,j-1]$. מיון הכנסה פועל ב$O(n)+O(\sum_{j=1}^{n-1}N(j))$ (כי צריך להשוות כל איבר לפחות פעם אחת לקודמו) אבל נשים לב שאם היינו צריכים לעשות היפוך $A[j] \leftrightarrow A[i]$ בעת הכנסת $A[j]$ למקום המתאים ($i<j$) אז בהכרח (i,j) היפוך ויש $2n$ כאלו לכן $\sum_{j=1}^{n-1}N(j)=2n$ כלומר האלגוריתם יעבוד ב$O(2n)+O(n)=O(n)$ זמן.

ב. לא ניתן. נניח בשלילה שניתן, ויהי מערך כלשהו עם n איברים. נבנה מערך חדש באורך $2n$ כאשר במקומות הזוגיים נשים $-\infty$ ובמקומות האי זוגיים נשים את אברי המערך המקורי. זה יעלה לנו $O(2n)=O(n)$. עכשיו מהנחת השלילה אפשר למיין את המערך הזה ב$O(2n)=O(n)$. כעת כל שנותר לעשות זה להעתיק את n האיברים האחרונים במערך החדש למערך המקורי לפי הסדר ב$O(n)$ וסה"כ מיינו את המערך המקורי ב$O(n)$ בסתירה לחסם התחתון על מיון השוואות של $\Omega(nlogn)$.

שאלה 3

א. נשתמש פשוט במערך שמאותחל בהתחלה לכל n המוצרים ואז נמיין אותו לפי מחיר ב$O(n)$ באמצעות מיון בסיס (כי יש רק 4 ספרות אז מקבלים $O(n) = O\big(4(n+10)\big)$).
עכשיו האתחול באמת ב$O(n)$ וכל הפעולות רצות ב$O(1)$ במקרה הגרוע שכן $select(k)$ זה לגשת לתא הk במערך ו$rank(x)$ זה לבדוק באיזה אינדקס אנחנו במערך (ע"י שמירה שלו או חישוב לפי הכתובת של התא הראשון והתא שלנו).

ב. נשתמש בעץ AVL ונקבל באופן טריוויאלי את כל הפעולות ב$O(logn)$ כדרוש. האתחול עדיין ב$O(n)$ כי במקרה שלנו אפשר למיין את המערך ב$O(n)$ וראינו שאפשר לבנות עץ AVL ממערך ממוין ב$O(n)$.

שאלה 4

Rank tree אחד שמכניסים אליו קואורדינטות x ועוד אחד עם קואורדינטות y. עכשיו מה שצריך לעשות זה באתחול פשוט לאתחל שני עצים ריקים ב$O(1)$. הכנסה היא הכנסה רגילה לשני העצים ב$O(logn)$. כדי לממש את $DomDiff(a,b)$ צריך לשים לב לציור הבא:

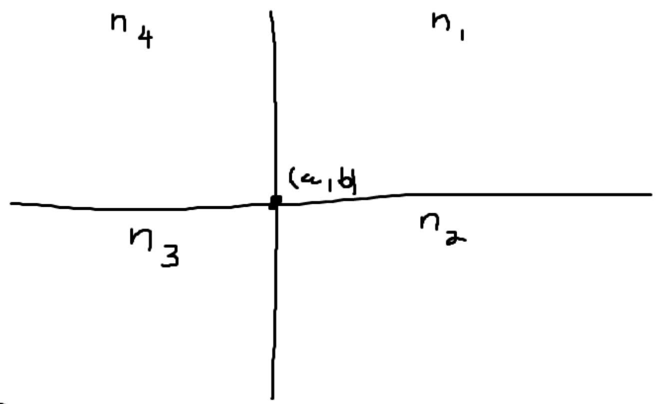
$$n = n_1 + n_2 + n_3 + n_4$$

$$Rank(T_x, a) = n_3 + n_4$$

$$Rank(T_y, b) = n_2 + n_3$$

$$\Downarrow$$

$$n - Rank(T_x, a) - Rank(T_y, b) = n_1 - n_3$$



לכן כל מה שצריך לעשות זה $size$ של אחד מהעצים ואז להחסיר ממנו $rank(T_x, a)$ וגם את $rank(T_y, b)$ לקבלת הדרוש.

שאלה 5

א. אפשר להגדיר $f(n) = logn$ ואז נשים לב שמתקיים ש –

$$n \overset{2\leq n}{\leq} nlogn \overset{\text{עולה } log}{\Longrightarrow} logn \overset{2\leq n}{\leq} \log(nlogn) \Longrightarrow logn = \mathrm{O}(\log(\mathrm{nlogn}))$$

$$nlogn \overset{2\leq n}{\leq} n^2 \overset{\text{עולה } log}{\Longrightarrow} \log(nlogn) \leq \log(n^2) \overset{\text{חוקי log}}{\Longrightarrow} \log(nlogn) \leq 2logn$$
$$\Longrightarrow \log(nlogn) = \mathrm{O}(logn)$$

ולכן $logn = \Theta(\log(nlogn))$ כדרוש.

ב. אפשר להגדיר $f(n) = log_*(n)$ ואז נשים לב שמתקיים ש –

$$log_*(n) \leq log_*(2^n) = log_*(n) + 1 \leq 2log_*(n)$$

ולכן מתקיים $log_*(n) = \Theta(log_*(2^n))$ כדרוש.

שאלה 6

א. נסמן בa_k את מספר העלים בעץ בינומי מדרגה $k > 0$. ע"י בדיקה ישירה $a_1 = 1$ ונשים לב שמההגדרה הרקורסיבית נובע ש$a_k = 2a_{k-1}$ (שכן כמות העלים של עץ מדרגה $k - 1$ הם a_{k-1} ועכשיו נוסף לו עוד בן שלו בעצמו a_{k-1} עלים והשורש לא היה עלה בעצמו ולכן לא נאבד עלה כי $k > 0$). לכן סה"כ $a_k = 2^{k-1}$ (בקלות מוכיחים באינדוקציה ממה שכבר נימקנו).

ב. ב0 או ב1. נוכיח ע"י הפרדה למקרים. אם לא התבצע $link$ בעת הכנסת האיבר אז פשוט הוספנו עץ עם צומת בודדת שהיא עלה בעצמה ולכן מס' העלים עלה ב1. אם התבצע $link$ אז היה כבר עץ בינומי מדרגה 0 ולכן שניהם הפכו לעץ בינומי מדרגה 1 שיש לו עלה אחד. עד כה מס' העלים לא עלה. יכול להיות שממשיכים לעשות $link$ים אבל מס' העלים לא ישתנה כי לשני עצים מדרגה $k - 1$ יש $2a_{k-1}$ עלים ומסעיף א' לעץ שייווצר מחיבור שלהם יהיו גם $a_k = 2a_{k-1}$ עלים.

ג. 1 שכן יש רצף פעולות שיוצר ערימת פיבונאצ'י שהיא שרשרת באורך n.

ד. n כי פשוט נכניס n איברים ואז הערימה שלנו תהיה רשימה של n עצים בעלי צומת אחת כלומר n עלים.

שאלה 7

ניצור טבלת האש בגודל $O(n^2)$ שתחזיק את כל הסכומים האפשריים של שני איברים מהמערך המקורי כמפתח וגם את שני המספרים עצמם (למשל ע"י שמירת מצביע). נשים לב שיש לכל היותר $O(n^2) = \binom{n}{2}$ כאלו ולכן אפשר לאכסן את כולם בטבלה. את זה אפשר לעשות ב$O(n^2)$ ע"י מעבר על כל מספר במערך וסכימה שלו עם כל מספר אחר והכנסה לטבלה ב$O(1)$ בתוחלת (נשתמש למשל בשיטת ה$chaining$ ואז מקדם העומס יהיה $\alpha = \frac{n}{m} \le \frac{\binom{n}{2}}{O(n^2)} = O(1)$ ולכן הפעולות באמת יהיו בזמן קבוע בתוחלת) עד כה $O(n^2)$ בתוחלת. כעת נעבור על הטבלה ועל כל מפתח k נחפש את $-k$ ואם נמצא אותו נחזיר את ארבעת המספרים המתאימים (שנמצאים עם $k, -k$) ואחרת נחזיר שאין 4 מספרים שכאלה. שלב זה הוא גם $O(n^2)$ בתוחלת ולכן סה"כ מקבלים אלגוריתם שעובד ב$O(n^2)$ בתוחלת.

סיבוכיות מקום נוסף תהיה $O(n^2)$ בשביל טבלת ההאש. סיבוכיות זמן במקרה הגרוע תהיה $O(n^4)$ כי סיבוכיות הזמן במקרה הגרוע של חיפוש בטבלת האש שיש בה n איברים היא $O(n)$ ואחרי שכבר יצרנו את הטבלה אכן יהיו בה $O(n^2)$ איברים ואנחנו עושים בה $O(n^2)$ חיפושים).