תרגיל 2- מבני נתונים 67109:

שם: דן קורנפלד

חלק 1:

שאלה 1:

סעיף א:

1. מכיוון שנתון 2 רשימות ממוינות (בסדר עולה) arr1,arr2 (לא בהכרח בעלות אותו אורך) נרצה לחבר בניהן לרשימה אחת ארוכה וממוינת אשר מכילה את 2 הרשימות הממוינות.
באלגוריתם merge sort אנו משתמשים בתהליך דומה מאוד.
אלגוריתם:
   (a) נגדיר 2 מצביעים רצים על הרשימות: i,j=0 כאשר i הוא המצביע של arr1 ו-j של arr2.
   (b) נרוץ בלולאה ונבדוק אם 2 הרשימות לא הגיעו לסופן.
      i. נשווה בין האיבר במקום ה-i של arr1 והאיבר במקום ה-j של arr2.
      ii. נכניס לרשימה החדשה את האיבר הקטן מבניהם. (לצורך המחשה והדיוק, נגדיר שאם יש איברים שווים, האיבר של arr1 יכנס לרשימה קודם).
      iii. האינדקס של הרשימה שהוסיפה את האיבר לרשימה המלאה יעלה ב-1
   (c) נחזור על הפעולה הזו שוב ושוב כך שבכל איטרציה של הלולאה יכנס בדיוק איבר אחד לרשימה החדשה, עד שאחד הרשימות תגיע לסופה.
   (d) במצב בו אחד הרשימות הסתיימו, נכניס בזה אחר זה את האיברים שלא הוכנסו של הרשימה השניה (שלא הוכנסו עד הסוף) לרשימה המלאה. ★ כמובן שנתון ש-2 הרשימות ממוינות בסדר עולה, ולכן הכנסה של הרשימה שלא הסתיימה תכניס איברים ממוינים גם כן בסדר עולה, לרשימה הגדולה שכל איבריה קטנים או שווים לאיברים שנכנסים לאחר הלולאה.
2. נכונות האלגוריתם: באלגוריתם זה יש 2 רשימות ממוינות ובכל פעם נכנס לרשימה הגדולה האיבר הקטן ביותר מבין 2 הרשימות, ולכן בהכרח שבכל הכנסה לרשימה הגדולה של איבר כלשהו, האיבר הנכנס יהיה גדול או שווה מכל האיברים שקדמו לו.
3. סיבוכיות זמן ריצה: באלגוריתם זה בכל איטרציה של הלולאה נבצע פעולות אריתמטיות (כמו השווה של איברים ממערכים) בסיבוכיות זמן של $O(1)$, ויתבצע לכל היותר $length(arr1) + length(arr2)$ איטרציות, מכיוון שבכל איטרציה יכנס איבר אחד לרשימה החדשה, כך שבסוף $length(arr1) + length(arr2)$ לכל היותר, יהיה מעבר על 2 הרשימות וכל איברי הרשימות הקטנות יכנסו לרשימה החדשה. על כן זמן הריצה יהיה $O(length(arr1) + length(arr2))$.
4. סיבוכיות מקום: מלבד משתנים ספציפית כמו האינדקסים i,j שהגדרנו לעיל (שמשתנים אלו לא תלוים בגודל הקלט) המקום שיוקצה לרשימה החדשה ויכיל את כל האיברים ברשימה הראשונה + את כל האיברים ברשימה השניה, ולכן סיבוכיות המקום תהיה באופן לינארי כגודל הרשימות arr1,arr2 המתקבלות. מסיבה זו סיבוכיות המקום תהיה $O(length(arr1) + length(arr2))$.

סעיף ב:

1. בלי הגבלת הכלליות $q < p$.

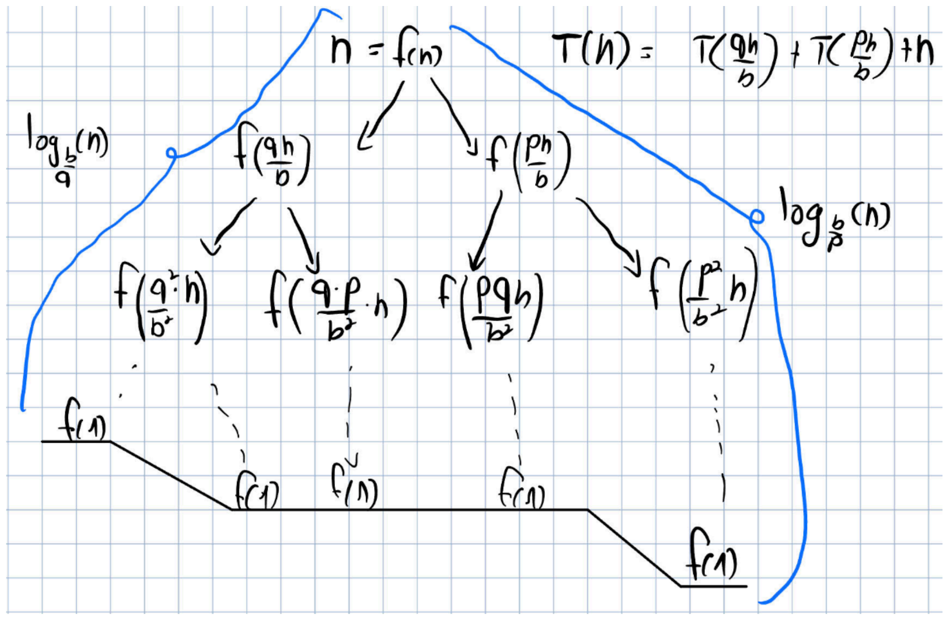

Figure 1: העברתי מהאייפד

2. ניתוח זמן ריצה: $T\left(\frac{qn}{b}\right) + T\left(\frac{pn}{b}\right) + cn$

(a) בדומה לתרגול, מכיוון שאנו מחלקים את המערך בצורה לא שווה, החלוקה של המקדם הקטן יותר $\frac{q}{b}$ היה מגיע ל-$f(1)$ "מהר יותר" (מחלוקת שברים).

(b) כמו כן, בכל רמה של העץ, נבצע n עבודה, מכיוון שאם נחבר את ביצוע כל הפעולות בכל רמה של כל ה-$f\left(\frac{x \cdot n}{b}\right)$ נקבל $f(n)$, כלומר נבצע "c · n עבודה" (ראינו בתרגול הסבר לכך).

(c) השוני בעיקר בגובה העץ, שכן בהנחה ש $p \neq q$, גובה העץ בין הענפים לא בהכרח יהיה שווה.

(d) אם נסתכל על המקרה הטוב ביותר, הענף השמאלי בתרשים: אם כל הענפים היו "במתנהגים" באותו אופן נקבל: $T(n) = \Omega(n \cdot log_{\frac{b}{q}}(n))$, כאמור: בכל שלב יש n עבודה, כפול הגובה של העץ: $log_{\frac{b}{q}}(n)$.

(e) כמו כן, אם נסתכל על המקרה הגרוע ביותר, הענף הימני בתרשים: אם כל הענפים היו "מתנהגים" באותו אופן נקבל: $T(n) = O(n \cdot log_{\frac{b}{p}}(n))$ כאמור: בכל שלב יש n עבודה, כפול הגובה של העץ: $log_{\frac{b}{p}}(n)$.

(f) מכיוון ש $log_{\frac{b}{p}}(n)$ ו $log_{\frac{b}{q}}(n)$ תלוים בקבוע כלשהו, מהגדרה מתקיים ש: $T(n) = \theta(nlog(n))$

שאלה 2:

סעיף א:

נתון $T:\mathbb{N}\to\mathbb{R}^+$ המוגדרת להיות: $T(n)=2T\left(\frac{n}{2}\right)+\frac{n}{log(n)}$, $T(1)=1$.

ראשית, מכיוון ש $\frac{n}{log(n)}$ לא מהצורה n^c, לא ניתן להשתמש במשפט האב הפשוט.

כמו כן, עבור משפט האב המורחב: נבדוק אם קיים $\varepsilon>0$ כך ש $\frac{n}{log(n)}=O\left(n^{(log_b(a))-\varepsilon}\right)$: אניח בשלילה שהתנאי מתקיים, כלומר מהגדרת O מתקיים: $\exists N\in\mathbb{N},\ \exists c>0\ \forall n\geqslant N\ f(n)\leqslant cg(n)$ כלומר $\frac{n}{log(n)}<c\cdot n^{(log_b(a))-\varepsilon}=c\cdot\frac{n^{(log_b(a))}}{n^{\varepsilon}}=c\cdot\frac{n}{n^{\varepsilon}}$. מכיוון ש $n\to\infty$, אזי החל ממקום מסוים $n\neq 0$ ויתקיים: $\frac{1}{n^{\varepsilon}}<c\cdot\frac{1}{log(n)}\Leftarrow c>\frac{n^{\varepsilon}}{log(n)}$. מכלל לופיטל:

$\lim_{n\to\infty}\frac{n^{\varepsilon}}{log(n)}=\frac{\varepsilon n^{\varepsilon-1}}{\frac{1}{n}}=\frac{\varepsilon\frac{n^{\varepsilon}}{n}}{\frac{1}{n}}=\varepsilon n^{\varepsilon}\to\infty$, כלומר $\frac{n^{\varepsilon}}{log(n)}$ שואף ל-∞ בסתירה להנחה שהחל ממקום מסוים הביטוי קטן מ-c כלשהו, ולכן התנאי הראשון של משפט האב לא מתקיים.

עבור התנאי השני של משפט האב: $\lim_{n\to\infty}\frac{1}{log(n)}=0=\lim_{n\to\infty}\frac{\frac{n}{log(n)}}{n}$ מהתרגול הראשון לא ייתכן ש $\frac{n}{log(n)}=\Omega(n)$, כלומר $\frac{n}{log(n)}\neq\theta(n)$, ולכן גם התנאי השני של משפט האב המורחב לא מתקיים.

בתנאי השלישי: נבדוק אם מתקיים: $\frac{n}{log(n)}=\Omega\left(n^{1+\varepsilon}\right)$, גם תנאי זה לא מתקיים, מכיוון שהוכחנו ש לכל ε מתקיים $log(n)<n^{\varepsilon}$.

לאור כל הסיבות לעיל לא ניתן להכריע באמצעות משפט האב את סיבוכיות זמן הריצה של הפונקציה הנ"ל.

סעיף ב:

מהגדרת ה-log, $log_b(a)=x\Leftarrow b^x=a$, כלומר:

1. $log_b(a)<c$ מכיוון שחזקה מונו עולה למספרים חיוביים, אעלה את 2 האגפים כחזקה באותו בסיס קרי: $b^{log_b(a)}<b^c$, כלומר $a<b^c\overset{b^c>0}{\Leftarrow}\frac{a}{b^c}<1$. מכיוון שכל הפעולות היו ניתנות לביצוע ($\Longleftrightarrow$) התנאי שקול, ולכן יתקיים אם $\frac{a}{b^c}<1$ אז $T(n)=\theta\left(n^c\right)$
2. $log_b(a)=c$ באותו אופן יתקיים $b^{log_b(a)}=b^c$ כלומר $a=b^c\overset{b^c>0}{\Leftarrow}\frac{a}{b^c}=1$ מכיוון שכל הפעולות היו ניתנות לביצוע ($\Longleftrightarrow$) התנאי שקול, ולכן יתקיים אם $\frac{a}{b^c}=1$ אז $T(n)=\theta\left(n^c log_b(n)\right)$

3. $log_b(a) > c$ מכיוון שחזקה מונו עולה למספרים חיוביים, אעלה את 2 האגפים כחזקה באותו בסיס קרי: $b^{log_b(a)} > b^c$, כלומר $a > b^c \overset{b^c>0}{\Leftarrow} 1 < \frac{a}{b^c}$. מכיוון שכל הפעולות היו ניתנות לביצוע ($\Longleftrightarrow$) התנאי שקול, ולכן יתקיים אם $\frac{a}{b^c} > 1$ אז $T(n) = \theta\left(n^{log_b(a)}\right)$

ולכן, התנאים מתקיימים בגרסה הפשוטה של משפט האב ■

תרגיל 2- מבני נתונים 67109:

שם: דן קורנפלד

חלק 2:

שאלה 3:

סופקו בכתב

תרגיל 2- מבני נתונים 67109:

שם: דן קורנפלד

חלק 3:

שאלה 4:

1. אלגוריתם:
   (a) ניצור מערך חדש בגודל המערך הנתון.
   (b) נגדיר 2 אינדקסים: $i = 0, j = A.length - 1$
   (c) נעבור על כל תא במערך הנתון ונבדוק: אם האיבר בתא שווה ל-0, נשים במערך החדש במקום ה-i 0, ונעלה את i ב-1 ואם האיבר בתא שווה ל-2, נשים ב-j במערך החדש 2, ונוריד את j ב-1.
   (d) הלולאה תעצור עד שנעבור על כל המערך הנתון. ★ לא יתכן שהשמת 0\2 תפגע במספר השני, שכן סה"כ המופעים הם לכל היותר גודל המערך
   (e) לאחר מכן, נשים את כל ה"1" באינדקסים בין i ל-j.

   ★ למען הסדר הטוב, מכיוון שהתבקשנו למיין את המערך ולא להחזיר מערך אחר, בסוף השמת המערך החדש נעבור בלולאה על המערך החדש ועל המערך הישן ולכל אינדקס נכניס את הערך במערך החדש למערך הישן, במקום המערך החדש. סיבוכיות המקום עדיין תהיה O(n) כמבוקש
2. נכונות האלגוריתם: מכיוון שהתבקשנו ליצור מערך ממוין עם 3 איברים, ניתן להסתכל על 3\2 האיברים הללו ולהכניס את כל אחד מהם לקצה אחר של המערך, ולאחר מכן להכניס את האיבר האמצעי בין האיברים שכבר נכנסו למערך. למעשה, סכום המופעים של 0,1,2 הוא A.length, ולכן אם נכניס 2\0 לתא במערך החדש בכל פעם שיופיע במערך הנתון את אותו איבר, עדיין נכניס את אותו כמות מופעים, בצורה ממויינת.
3. סיבוכיות זמן ריצה: אנו עוברים על כל המערך הנתון פעם אחת, כך שבכל איטרציה אנו מבצעים פעולות שאינן תלויות באורך המערך ((O(1)), ולכן המעבר על המערך הראשון בסיבוכיות זמן ריצה של $O(n)$, תלוי באופן לינארי באורך הקלט. כמו כן, השמת האיברים 2\0 מתבצעת תוך כדי המעבר על המערך הראשון, אך במקרה הגרוע ביותר לא יהיו מופעים של 2/0, ונצטרך לעבור על כל המערך החדש ולהכניס לכל תא ותא את הערך 1. במצב זה גם כן סיבוכיות זמן הריצה תהיה $O(n)$, מכיוון שהאלגוריתם תלוי באורך הקלט. לכן, $T(n) = O(n) + O(n) = O(n)$ כמבוקש.
4. Pseudo-Code:

   (a) before the Pseudo-Code
   A= the given array,
   B= the new array (the same length of A)
   i= 0 the first pointer for the new array, for insert 0
   j=A.length-1 the second pointer for the new array, for insert 2

   (b) for k←0 to A.length-1 do
   - i. if A[k]=0 do
     - A. B[i]=0
     - B. i←i+1
   - ii. if A[k]=2 do
     - A. B[j]=2

B. j←j-1

(c) for k←i to j do

i. B[k]=1

(d) for k←0 to A.length do

i. A[k]←B[k]

משפט האב:

1. אם:

(a) $a \geqslant 1, b \geqslant 1, c \geqslant 0$ קבועים

i. $T(n) = aT\left(\frac{n}{b}\right) + n^{c}$

ii. ובדרך כלל $T(1) = \theta(1)$

(b) אז:

i. $T(n) = \theta\left(n^{c}\right)$ $when$ $log_b(a) < c$

ii. $T(n) = \theta\left(n^{c} log_b(n)\right)$ $when$ $log_b(a) = c$

iii. $T(n) = \theta\left(n^{log_b(a)}\right)$ $when$ $log_b(a) > c$

משפט האב המורחב: $T(n) = \begin{cases} \theta(1) & n = 1 \\ aT\left(\frac{n}{b}\right) + f(n) & n > 1 \end{cases}$ אינטואיציה נוחה בהקלטה באזור 32-37

תרגול מס 2

1. אם קיים $\varepsilon > 0$ כך ש $f(n) = O\left(n^{(log_b(a))-\varepsilon}\right) \Longleftarrow T(n) = \theta\left(n^{log_b(a)}\right)$ (אינטואיציה: גם אחרי שמורידים ε מהעלים, אם הם עדיין יותר גדולים אז למעשה זה יהיה הסיבוכיות זמן ריצה)

(a) אם $f(n) = \theta\left(n^{log_b(a)}\right) \Longleftarrow T(n) = \theta\left(n^{log_b(a)} log_b(n)\right)$ (אינטואיציה: אם סדר גודל של ההתחלה הוא כמו של הסוף, הסיבוכיות זמן ריצה תהיה רמת הסיבוכיות של כל גובה, כפול גובה העץ)

(b) אם קיים $\varepsilon > 0$ כך ש $f(n) = \Omega\left(n^{log_b(a)+\varepsilon}\right)$ וקיים קיימים $N \in \mathbb{N}$ $0 < c < 1$ כך שלכל $n \geqslant N$: $af\left(\frac{n}{b}\right) \leqslant c \cdot f(n) \Longleftarrow T(n) = \theta(f(n))$ (אינטואיציה: $f(n)$ חוסם מלמעלה את $n^{log_b(a)+\varepsilon}$, לכן $f(n)$ "מנצח" את $n^{log_b(a)}$ וגם אם נוסיף ε כלשהו, $f(n)$ עדיין ינצח את $n^{log_b(a)+\varepsilon}$, כלומר $f(n)$ "הרבה יותר גדול". בנוסף יש תנאי נוסף כדי לוודא שיש דעיכה בין הרמות, ולכן יש את התנאי הנוסף)

שאלה 5 (שאלת בונוס 5 נקודות)

תנאי הרגולריות במקרה 3 של משפט האב הוא: עבור $n_0 \in \mathbb{N}, c < 1$ ולכל $n > n_0$ מתקיים $a \cdot f\left(\frac{n}{b}\right) \leq c \cdot f(n)$. הביאו דוגמה לקבועים $a \geq 1$, $b > 1$ ופונקציה $f(n)$ העונה על כל התנאים שבמקרה 3 של משפט האב, פרט לתנאי הרגולריות.

שאלה 5 - תשובה

נבחר $a = 1$, $b = 2$, ואת הפונקציה הבאה עבור $k \in \mathbb{N}$

$$f(n) = \begin{cases} 4n^2 & n = 2k-1 \\ n & n = 2k \end{cases}$$

מתקיים כי $\log_b(a) = 0$ אז $f(n) = \Omega\left(n^{\log_b(a)+\varepsilon}\right)$ מתקיים לכל $0 < \varepsilon < 1$. אבל תנאי הרגולריות לא מתקיים:

$$\frac{a \cdot f\left(\frac{n}{b}\right)}{f(n)} = n \geq 1$$

לכל $n = 2m$, ו-m אי זוגי.