עבודת הגשה 2 – מבני נתונים

(4.1

i) בכל איטרציה נעשה i צעדי חיפוש, ו i-1 צעדי backtrack

ננסה לחסום את מספר האיטרציות שנצטרך לעשות. כיוון שכאשר אנחנו עושים i-1 צעדים אחורה בכל איטרציה אז ההתקדמות היחסית שלנו היא 1 בכל איטרציה. בנוסף, באיטרציה הi אנחנו עושים i צעדי חיפוש, כלומר באיטרציה ה n/2 נצטרך לעשות n/2 ובכך נסיים את החיפוש.

ננסה לחסום את מספר האיטרציות מלמעלה ע"י n וננסה לחסום אותו מלמטה ע"י $\left\lfloor\frac{n}{2}\right\rfloor$ ונראה שבשני המקרים מדובר בסדר גודל של n^2

$$\sum_{i=1}^{n} i+(i-1)=\frac{n\cdot(n-1)+(n-1)\cdot(n-2)}{2}=n^2-2n+1\ \in O(n2)$$

$$\sum_{i=1}^{\left\lfloor\frac{n}{2}\right\rfloor} i+(i-1)\geq\quad \sum_{i=1}^{\frac{n}{2}-1} i+(i-1)\quad =\frac{(\frac{n}{2}-1)\cdot\left(\frac{n}{2}-2\right)+\left(\frac{n}{2}-2\right)\cdot\left(\frac{n}{2}-3\right)}{2}=n^2-8n+16\ \in \Omega(n2)$$

לכן זמן הריצה הוא $\theta(n^2)$

ii) בכל איטרציה התכנית תעשה i-2 פעמים חיפוש ולאחר מכן חזרה לאחור, כאשר בפעם הראשונה נעשה

i-2 צעדי חיפוש ואז i-2 צעדי חזור עד שנגיע לצעד חיפוש אחד וצעד חזור אחד.

ניתן לרשום את סכום כל הפעולות שנעשה

$$\sum_{i=1}^{n}(1+2\sum_{j=1}^{i-2}j)=\ \sum_{i=1}^{n}1+2\cdot\frac{(i-2)\cdot(i-3)}{2}=n+\sum_{i=1}^{n}i^2-5i+6=7n+(\sum_{i=1}^{n}i^2)-5\cdot\sum_{i=1}^{n}i\ =$$

$$=7n+\frac{n(n+1)(2n+1)}{6}-5\cdot\left(\frac{n(n-1)}{2}\right)=\frac{1}{3}\cdot n^3-2n^2+9n\in\boldsymbol{\theta}(\boldsymbol{n^3})$$

זמן הריצה של התכנית הוא $\theta(n^3)$

iii) בכל איטרציה נעשה (n_1+n_2) צעדים. כעת נשים לב כי מספר האיטרציות תלוי בהפרש (n1-n2), ככל שההפרש יהיה גדול יותר ככה נעשה קפיצות גדולות יותר ונעשה פחות איטרציות

ובנוסף, נשים לב שכאשר נגיע לאינדקס $(n-n_2)$אנו נגיע לאינדקס בו n_1 צעדי החיפוש יחרגו מהמערך והתכנית תסתיים.

ניתן לכתוב את פונקציית זמן הריצה כתלות בשלושת המשתנים:

$$T(n,n_1,n_2)=\frac{(n_1+n_2)(n\ -n_2)}{(n_1-n_2)}=n\cdot\frac{n_1+n_2}{n_1-n_2}-n2\cdot\frac{n_1+n_2}{n_1-n_2}$$

נשים לב כי כאן n יותר דומיננטי מn_2 לכן נתמקד בחלק הראשון של הביטוי , זאת הפונקציה שתתאר זמן ריצה של התכנית כתלות בשלושת המשתנים, **לכן זמן הריצה של הפונקציה הוא** $\boldsymbol{\theta(n \cdot \frac{n1+n_2}{n_1-n_2})}$

**

ניתן גם לנתח את זמן הריצה עבור הפרמטרים n1, n2 , מכיוון שהם חסומים בין 0 ל n-1.

הביטוי $n \cdot \frac{n_1+n_2}{n_1-n_2}$ יהיה בערכו המקסימלי כאשר המכנה יהיה המינימלי האפשרי והמונה המקסימלי, מכאן נסיק שצריך להתקיים $n_1 - n_2 = 1$ וזה יקרה כאשר $n_2 = n_1 - 1$

וכאשר n1 יהיה בערכו המקסימלי שהוא n-1.

נציב *n1 = n-1, n2 = n1-1* ונקבל:

$$T(n, n-1, n-2) == n \cdot \frac{(n-1+n-2)}{1} - (n-2) \cdot \frac{(n-1+n-2)}{1}$$

$$T(n) = 2n^2 - 3n + 2n^2 - 3n - 4n + 6 = 4n^2 - 10n + 6 \in \boldsymbol{O(n^2)}$$

$$T(n, n-1, n-2) = 4n^2 - 10n + 6 \in \boldsymbol{O(n^2)}$$

הביטוי $n \cdot \frac{n_1+n_2}{n_1-n_2}$ יהיה בערכו המינימלי כאשר המכנה יהיה המקסימלי האפשרי והמונה המינימלי, מכאן נסיק שצריך להתקיים $n_2 = 0$

ובנוסף צריך להתקיים $n_1 = 1$

$$T(n, 1, 0) = n \cdot \frac{1+0}{1-0} - 0 \cdot (\dots) = n$$

$$T(n, 1, 0) = n \in \Omega(n)$$

**

.4.2

א. <u>מערך לא ממוין:</u>

Backtrack insert – O (1)
Backtrack delete– O (1)

כאשר הפעולה שנרצה לעשות לה backtrack היא insert - נצטרך למחוק את האיבר האחרון שהכנסנו, שמיקומו כבר ידוע, לכן נבצע רק פעולה אחת.
כאשר הפעולה שנרצה לעשות לה backtrack היא delete - נצטרך להעביר את האיבר שנמצא במיקום של האיבר שמחקנו, לסוף המערך, לכן נבצע פעולה אחת.
ולאחר מכן נצטרך להוסיף את האיבר האחרון שמחקנו, למיקום ששוב ידוע – לכן נבצע פעולה אחת.

ב. <u>מערך ממוין:</u>

Backtrack insert – O (n)
Backtrack delete – O (n)

כאשר הפעולה שנרצה לעשות לה backtrack היא insert , במקרה הגרוע ביותר, נצטרך למחוק את האיבר הראשון והכי קטן במערך. במקרה זה נצטרך להזיז את כל איברי המערך מקום אחד שמאלה, ונבצע n פעולות.
כאשר הפעולה שנרצה לעשות לה backtrack היא delete , המקרה הגרוע ביותר הוא הוספת איבר מינימלי למקום הראשון, בו נצטרך להזיז את כל איברי המערך מקום אחד ימינה ונבצע n פעולות.

ג. <u>עץ חיפוש בינארי:</u>

Backtrack insert – O (1)
Backtrack delete – O (1)

כאשר הפעולה שנרצה לעשות לה backtrack היא insert , נמחק את הקודקוד האחרון שהוספנו. ניגש להורה שלו ונשנה את שדה הבן הימני/שמאלי לnull, ונשנה את שדה ההורה של הצומת לnull, מדובר במספר פעולות קבוע לכן זמן הריצה O(1).
כאשר הפעולה שנרצה לעשות לה backtrack היא delete, נוסיף את הקודקוד האחרון שמחקנו, בהתאם ל3 מקרי מחיקה שונים : עבור קודקוד עם 0 , 1 או 2 בנים.

- אם מחקנו קודקוד עם 0 בנים, נוכל לשלוף מהמחסנית בנוסף גם את האב הקודם שלו, כעת נצטרך להחזיר אותו ע"י השמת מצביע לקודקוד דרך אביו –אשר נוכל לגשת אליו בO(1). לכן עבור מחיקת קודקוד עם 0 בנים זמן הריצה הוא O(1)
- אם מחקנו קודקוד עם 1 בנים,נוכל לשלוף מהמחסנית בנוסף גם את האב הקודם שלו, כעת נצטרך להחזיר אותו ע"י השמת מבציע לקודקוד דרך אביו – אשר נוכל לגשת אליו בO(1), ואת הבן הקודם של האב, נצטרך לבחור אם לשים מימין או משמאל לקודקוד שהחזרנו, ובכך הכנסנו קודקוד בין 2 קודקודים במס' פעולות קבוע, לכן זמן הריצה גם כאן הוא O(1).
- אם מחקנו קודקוד עם 2 בנים, נוכל לשלוף מהמחסנית בנוסף גם את האב הקודם שלו, וגם את האב של הsuccessor שלו. ולאחר מכן נוכל להוסיף קודקוד חדש מתחת לאב של הsuccesor בO(1), ורק להחליף את הkey של הקודקוד שמחקנו. לכן בסה"כ זמן הריצה הוא O(1)

נוכל לבצע backtrack בO(1) הודות לשמירת מספר קבוע של מצביעים לקודקודים ששונו במהלך המחיקה/הכנסה.

ד. <u>עץ AVL:</u>

Backtrack delete – O (log(n))
Backtrack insert – O (1)

כאשר הפעולה שנרצה לעשות לה backtrack היא insert , במקרה הגרוע, נצטרך לאזן קודקוד אחד, שאיזונו הופר במהלך הכנסה. לכן נצטרך לבצע פעולה הפוכה לפעולת האיזון של הקודקוד, זמן הריצה הוא O(1) מכוון שנצטרך לשנות מצביעים של 3 קודקודים.
כאשר הפעולה שנרצה לעשות לה backtrack היא delete, במקרה הגרוע ביותר, עשינו במהלך המחיקה $log\ n$ איזונים של העץ – במקרה ומחקנו קודקוד כאשר מספר הקודקודים עבור הגובה היה מינימלי. נצטרך לבצע שינוים על $log(n)$ קודקודים שאיזונם הופר, ולכן זמן הריצה הוא $O(log(n))$