I assert that the work I submitted is entirely my own.
I have not received any part from any other student in the class, nor did I give parts of it for others to use.
I realize that if my work is found to contain code that is not originally my own, a formal case will be opened against me with the BGU disciplinary committee.

Gal Levy 206055527

Noam Deutsch 207651837

שאלה 4

1.

i. ננתח את זמן הריצה של הפונקציה. נשים לב כי במקרה הכי גרוע, האיבר X שאנו מחפשים נמצא באינדקס האחרון במערך הלא ממוין. בכל איטרציה נתקדם i צעדים קדימה ו i-1 צעדים אחורה, כלומר בסוף כל איטרציה נתקדם צעד אחד, כך שבסוף האיטרציה הi נהיה באינדקס הi במערך.

$\frac{n}{2}$ איטרציות, $i=\frac{n}{2}$, לכן באיטרציה הבאה נבצע $\frac{n}{2}$ צעדים ונגיע לסוף המערך, כלומר בהכרח נמצא את X או נסיים את הריצה אם X לא איבר במערך. לכן במקרה הכי גרוע זמן הריצה יהיה:

$$\sum_{i=1}^{\frac{n}{2}} i+\sum_{i=1}^{\frac{n}{2}-1} i$$

כאשר בצד שמאל נסכמים הצעדים קדימה ובצד ימין הצעדים אחורה. נשים לב כי באיטרציה ה$\frac{n}{2}$ כאשר הגענו לסוף המערך אין צורך לחזור לאחור מכיוון שסיימנו את הריצה, לכן מספר האיטרציות בצעדים לאחור קטן באחד.

ננתח את זמן הריצה:

$$\sum_{i=1}^{\frac{n}{2}} i+\sum_{i=1}^{\frac{n}{2}-1} i=\frac{\frac{n}{2}\cdot(1+\frac{n}{2})}{2}+\frac{\frac{n}{2}-1\cdot(1+\frac{n}{2}-1)}{2}=\frac{n+\frac{n^2}{2}}{4}+\frac{\frac{n^2}{4}-\frac{n}{2}}{2}=\frac{\frac{2n+n^2}{2}}{4}+\frac{\frac{n^2}{4}-\frac{2n}{4}}{2}$$

$$=\frac{2n+n^2}{8}+\frac{n^2-2n}{8}=\frac{2n^2}{8}=\frac{n^2}{4}=\theta n^2$$

כלומר בסה״כ זמן הריצה הוא θn^2.

ii. ננתח את זמן הריצה של הפונקציה:

נשים לב כי במקרה הכי גרוע, האיבר X שאנו מחפשים נמצא באינדקס האחרון במערך הלא ממוין.
בכל איטרציה אנו מתקדמים צעד אחד "חדש" קדימה, כלומר מגיעים לאינדקס שלא הגענו אליו באיטרציות הקודמות, וכל שאר הפעולות זה צעדים אחורה וקדימה באיברים בהם ביקרנו באיטרציות קודמות, כלומר לאיבר האחרון במערך מגודל n, נגיע רק באיטרציה הn-1.
נשים לב כי בכל איטרציה אנו מתקדמים צעד אחד קדימה בתחילת האיטרציה, ולאחר מכן מבצעים צעדים נוספים אחורה וקדימה בהתאם לו, אך מספר הצעדים אחורה זהה למספר הצעדים קדימה.

לכן במקרה הכי גרוע זמן הריצה יהיה:

$$\sum_{i=1}^{n-1}(1+2\cdot\sum_{j=1}^{i-2}j)$$

כאשר הסיגמה השמאלית מייצגת את מספר האיטרציות שנבצע עד שנגיע לאיבר האחרון (n-1), והסיגמה הפנימית מייצגת את מספר הצעדים קדימה ואחורה, בתוספת צעד אחד קדימה.

ננתח את זמן הריצה:

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

$$=(3+(n-1)^2-3(n-1)+3)\cdot\frac{n}{2}=(3+n^2-2n+1-3n+3+3)\cdot\frac{n}{2}$$
$$=(n^2-5n+10)\cdot\frac{n}{2}=\frac{n^3-5n^2+10n}{2}=\theta n^3$$

כלומר בסה"כ זמן הריצה הוא θn^3.

iii.

- ננתח את זמן הריצה במקרה "הטוב" ובמקרה "הגרוע" ובכך נמצא חסם תחתון ועליון.

בכל מקרה נרצה לנתח את זמן הריצה במקרה בו האינדקס שאנחנו מחפשים נמצא בתא האחרון במערך בגודל n.

- נשים לב כי קצב ההתקדמות בכל איטרציה הוא $(n1-n2)$. בכדי לחשב את מספר האיטרציות שנצטרך לבצע בכל מקרה נסמן את מספר האיטרציות ב-k, ונפתור את המשוואה הבאה:

$$k\cdot(n1-n2)+n1=n$$

כלומר נרצה לבדוק כמה איטרציות נצטרך לבצע כאשר קצב ההתקדמות בכל איטרציה הוא $(n1-n2)$, בתוספת *n1* צעדים שנבצע בסוף הריצה ונגיע לסוף המערך (n), ונקבל:

$$k=\frac{n-n1}{n1-n2}$$

- **חסם תחתון(מקרה טוב):** במקרה זה נרצה לעבור על המערך כמה שפחות פעמים עד שנגיע לסופו (ונמצא את האיבר או נצא מהמערך) . מאחר ו $0\le n2\le n1<n$, נבחר:

$$n1=n-1$$

$$n2=0$$

נשים לב כי במקרה זה נצטרך שתי איטרציות בלבד עד שנגיע לסוף המערך, באיטרציה הראשונה נתקדם n-1 צעדים במערך ובאיטרציה הבאה נתקדם עוד צעד 1 ונגיע לסוף המערך. כלומר בסך הכל נתקדם n צעדים וזמן הריצה הוא $\Omega(n)$.

- **חסם עליון(מקרה גרוע):** במקרה זה בכל איטרציה נתקדם צעד 1 קדימה ונחזור 0 צעדים אחורה, כלומר:

$$n1=1$$

$$n2=0$$

בכדי לחשב את מספר האיטרציות שנצטרך לבצע נציב את הערכים בנוסחה:

$$k=\frac{n-n1}{n1-n2}=n-1$$

בכל איטרציה אנו מתקדמים בסה"כ צעד אחד במערך, לכן זמן הריצה יהיה:

$$\sum_{i=1}^{n-1}1=n-1=O(n)$$

- בסה"כ קיבלנו כי זמן הריצה במקרה הכי טוב הוא$\Omega(n)$ ובמקרה הכי גרוע הוא $O(n)$, לכן סה"כ זמן הריצה הוא $\theta(n)$.

4. 2.

i. Unsorted Array Backtrack:

במימוש היעיל ביותר יעשה שימוש במחסנית. המחסנית תשמור אובייקט בכל פעם שמבוצעת מחיקה / הכנסה. באובייקט עצמו נשמור את כל הפרטים הרלוונטיים לפעולה.

- יצירת מחסנית (מחוץ לפונקציית ה-Backtrack) יעלה $\theta(1)$ זיכרון ולא ישפיע על זמן הריצה.

Backtrack func

- הוצאת איבר מהמחסנית $\theta(1)$
- בדיקה איזו פעולה נרצה לבטל $\theta(1)$
  - ביטול הכנסה: $\theta(1)$
    - נמחק את האיבר שנוסף בסוף המערך $\theta(1)$
    - נוריד את ערך המצביע ב-1 $\theta(1)$
  - ביטול מחיקה: $\theta(1)$
    - נחזיר את האיבר שתופס את מקומו של המחוק לסוף המערך $\theta(1)$
    - נחזיר את האיבר שנמחק למקומו באינדקס המקורי $\theta(1)$
    - נגדיל את ערך המצביע ב-1 $\theta(1)$

כלל הפעולות הן בסדר גודל של קבוע, לכן זמן הריצה יהיה $\theta(1)$.

$$T(n) = \theta(1) + \theta(1) + \theta(1) + \theta(1) + \theta(1) + \theta(1) = \theta(1)$$

ii. Sorted Array Backtrack:

במימוש היעיל ביותר יעשה שימוש במחסנית. המחסנית תשמור אובייקט בכל פעם שמבוצעת מחיקה / הכנסה. באובייקט עצמו נשמור את כל הפרטים הרלוונטיים לפעולה.

- יצירת מחסנית (מחוץ לפונקציית ה-Backtrack) יעלה $\theta(1)$ זיכרון ולא ישפיע על זמן הריצה.

Backtrack func

- הוצאת איבר מהמחסנית $\theta(1)$
- בדיקה איזו פעולה נרצה לבטל $\theta(1)$
- המקרה הגרוע ביותר הוא כשמכניסים או מוחקים איבר שקטן מכל איברי המערך (כלומר מכניסים איבר לאינדקס הראשון או מוחקים איבר מהאינדקס הראשון). הסיבה לכך היא שלאחר המחיקה או ההכנסה נצטרך לבצע הזזה של ככל האיברים המערך קדימה או אחורה בהתאמה לפעולה.

  כלומר נצטרך להזיז n-1 איברים במערך (במקרה הקיצון שבו כל המערך מלא) בעלות $\theta(n)$.

  לכן זמן הריצה יהיה:

$$T(n) = \theta(1) + \theta(1) + \theta(1) \ldots + (n-1) = \theta(n)$$

iii. Backtracking BST

נבצע backtrack להכנסה או מחיקה, נחלק למקרים:

- מקרה 1: Backtracking insertion

  בהנחה והמימוש נעשה עם שימוש במחסנית, נוציא מראש המחסנית את האובייקט שמייצג את האיבר שהכנסנו ונרצה לבטל את ההכנסה שלו.

  נשתמש בשדה Parent של האיבר ונשנה את המצביע לילד הרלוונטי ל-.null לכן, backtrack של פעולת הכנסה תמיד יהיה $\theta(1)$ ללא תלות בגובה העץ.

- מקרה 2: Backtracking Deletion
  - מקרה 1: ביטול מחיקה של <u>עלה</u>

נוציא מהמחסנית את האובייקט שנמחק, ניגש לשדה Parent ונשנה את המצביע לילד לעלה שנמחק. בכך נחזיר אותו לעץ.
העלות תהיה $\theta(1)$ ללא תלות בגובה העץ.

- מקרה 2: ביטול מחיקה של קודקוד עם בן אחד
נוציא מהמחסנית את האובייקט שנמחק (עדיין מצביע לילדים "המקוריים" שלו), ניגש לשדה Parent ונגדיר את המצביע לבן שלו להיות הקודקוד שנמחק. בכך נחזיר את הקודקוד לעץ עם הבן שלו.
העלות תהיה $\theta(1)$ ללא תלות בגובה העץ.
- מקרה 3: ביטול מחיקה של קודקוד עם שני בנים
נוציא מהמחסנית את האובייקט שנמחק, המכיל שדה ששומר את הקודקוד של העוקב שלו. נשנה את המצביע של Parent אל הקודקוד המקורי, ואת המצביע של האבא של העוקב, אל העוקב עצמו.
העלות תהיה $\theta(1)$ ללא תלות בגובה העץ.

בכל קריאה לפונקציית ה- Backtrack ימומש אחד מהמקרים המפורטים לעיל, ולכן זמן הריצה תמיד תהיה $T(n) = \theta(1)$.

iv. Backtracking AVL
נבצע backtrack להכנסה או מחיקה, נחלק למקרים:

- מקרה 1: Backtracking insertion
כשנבצע הכנסה בעץ AVL, נכניס קודקוד לעץ ונבצע לכל היותר איזון אחד ("סיבוב" מספר קודקודים בעץ כדי לשמור על עץ מאוזן) בעלות של $\theta(1)$. לכן, בביטול הכנסה נשנה את המצביעים של האיבר שנרצה למחוק (בדומה ל-BST) ונאזן את העץ בחזרה (רוטציה הפוכה לזו שנעשתה בהכנסה) בעלות של $\theta(1)$.

- מקרה 2: Backtracking deletion
כשמבצעים מחיקה בעץ AVL, מוחקים קודקוד באופן "רגיל" בדומה לפעולת המחיקה ב- BST, ולאחר מכן מאזנים את העץ.
במקרה הגרוע ביותר, לאחר איזון אחד של העץ קודקוד אחר יהפוך ללא מאוזן, ולכן נצטרך להמשיך ולבצע בדיקות איזון מהקודקוד שמחקנו ועד שורש העץ. כלומר במקרה הגרוע ביותר נבצע בדיקות איזון כגובה העץ - $\theta(\log n)$.
כשנבצע ביטול מחיקה, נאזן את העץ $logn$ פעמים כדי להשיבו למצב שלפני המחיקה ונכניס את האיבר שנמחק לעץ בעלות של $\theta(1)$ (כפי שמתואר בסעיף הקודם).

לכן זמן הריצה במקרה הגרוע ביותר הוא $\theta(logn)$.