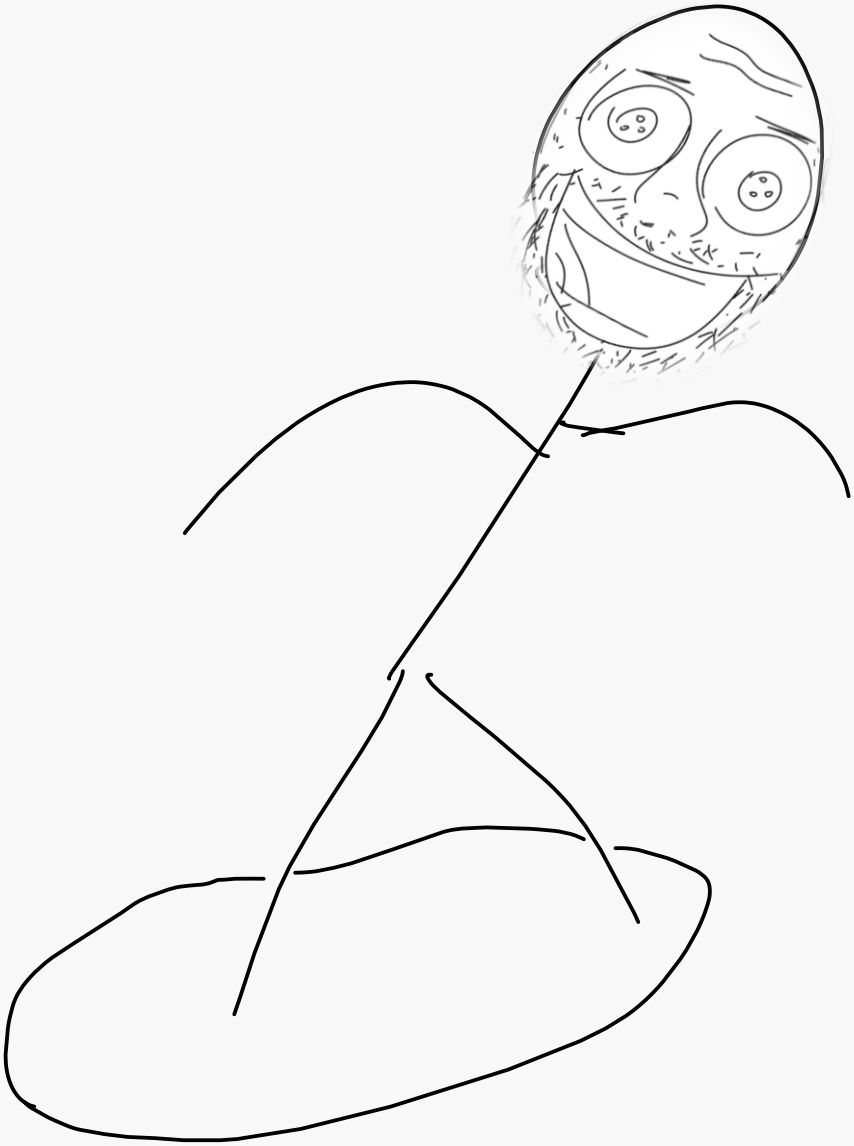
סיכומי הרצאות בקורס מבני נתונים

מרצה: פרופ' גדי לנדאו
סוכם על ידי בר וייסמן, שנה"ל תשפ"ב

"הקורס שלי זה כמו בישול של סבתא"



תוכן העניינים

תלחצו על נושא לבחירתכם, זה עושה מגניב כזה

בעיות ראשונות

<u>בעיה פשוטה:</u>

נתונה מטריצה $n \times n$. מצא את המקסימום בכל שורה בה.

ברור כי החסם התחתון לבעיה הוא n^2.

סימון: $M(i)$ הוא האינדקס המקסימלי של השורה ה- i.

נוסיף נתון – $M(i) \leq M(i+1)$ לכל i בתחום.

נמצא את המקסימום בשורה האמצעית - $M\left(\frac{n}{2}\right)$ בזמן $O(n)$.

כעת נחפש בחצאי השורות $\frac{n}{4}, \frac{3n}{4}$ – סה"כ בזמן $O(n)$.

כעת ב- $\frac{n}{8}, \frac{3n}{8}, \frac{5n}{8}, \frac{7n}{8}$. כל שלב עולה $O(n)$ פעולות, ויש $\log n$ שלבים.
לכן, עלות הפתרון: $O(n \cdot \log n)$.

<u>עוד אחת:</u>

מציאת מקסימום מבין n מספרים בעזרת <u>מספר בלתי מוגבל</u> (נסמן ב- t) של מעבדים.

נבנה מערך עזר B. תחילה, נקדיש n מעבדים ש"ינקו" את B – $O(1)$

נציב $O(n^2)$ מעבדים. כל מעבד ישווה בין כל זוג אפשרי של שני מספרים מהמערך.

אם בהשוואה כלשהי $A[i]$ יהיה קטן יותר מ- $A[j]$, נשים ב- $B[i]$ 1.

לאחר כל ההשוואות התא היחיד שנשאר עם ערך 0 הוא המקסימום שלנו.

לבסוף, נקדיש n מעבדים שיכריזו על המנצח, כל אחד משלם זמן $O(1)$.

<u>בסך הכל</u>, כל מעבד יבצע פעולות בעלות של $O(1)$.

איך מודדים אלגוריתם?

בהינתן פרמטר n, $T(n)$ מסמן את סיבוכיות הזמן של האלגוריתם.

בדרך כלל, n יהיה ביטוי לגודל הקלט.

נבדוק את מספר הפעולות של האלגוריתם על קלט בגודל n.

נחלק את האלגוריתם לקבוצות – סדרי גודל, שני אלגוריתמים יעילים באותה צורה אם יש להם אותו סדר גודל.

נמצא <u>חסמים עליונים</u> לסיבוכיות הזמן של האלגוריתמים.

נסמן באות O (מלשון Order) חסם עליון.

$$T(n) = O\big(g(n)\big)$$

הפונקציה g מהווה חסם עליון של T אם:

$$\exists n_0 \in \mathbb{N}, c \in \mathbb{R} \mid \forall n > n_0 : \ T(n) \leq c \cdot g(n)$$

חסם עליון למציאת מקסימום – $O(n)$.

גרפים ועצים

<u>עץ חופשי:</u>

1. גרף לא מכוון
2. ללא מעגלים
3. קשיר
4. $|V| = |E| + 1$

בעץ חופשי אין שורש. <u>עץ מושרש</u> הוא עץ חופשי, בו צומת אחת נבחרת להיות השורש.

לכל צומת בעץ יש הורה אחד, מלבד השורש – לו אין הורה.

צאצאים יכולים להיות הרבה, כאוות נפשו של כל צומת.

<u>גובה צומת:</u> אורך המסלול הארוך ביותר מהצומת לאחד מהעלים.

הגובה של עלי העץ הוא 0.

גובה העץ == גובה השורש.

<u>עומק צומת:</u> אורך המסלול מהשורש לצומת.

עומק השורש – 0.

<u>רמה בעץ:</u> הצמתים הנמצאים באותו עומק בעץ.

<u>תת עץ:</u> צומת וכל צאצאיה בעץ.

מקבלים עץ באמצעות מצביע. <u>עץ ריק</u> כאשר המצביע עליו הוא null.

עץ בינארי

- עץ בו לכל צומת יש לכל היותר 2 בנים.

עץ בינארי מלא: עץ בו כל הרמות מלאות מלבד הרמה התחתונה, והיא מלאה משמאל לימין.

עץ מלא טורבו[1]: עץ בו כל הרמות מלאות.

בעץ מלא טורבו מספר הצמתים יהיה $2^k - 1, k \in \mathbb{N}$.

עץ מאוזן: עץ שגובהו $O(\log n)$.

כל עץ מלא (ובפרט כל עץ מלא טורבו) הוא עץ מאוזן.

עץ כללי: מספר הבנים בכל צומת אינו מוגבל

עץ מסודר: יש יחס בין הבנים בכל צומת

עץ מיקום: יש יחס בין הבנים, המיקום של הסדר הראשוני נשמר גם לאחר מחיקות

איך מיישמים עצים?

עץ בינארי מימשנו באמצעות struct עם 3 תכונות: ערך, ומצביעים לשני הבנים שלו.

לא נוכל לממש כך עץ כללי, כי כמות השדות לא מוגבלת.

גם עבור כמות מוגבלת אבל גדולה – למשל לכל צומת יש לכל היותר מיליארד בנים. לא יעיל!

"סתם בונה סטראקטים מפונפנים, בשביל כלום!"

עץ מלא עדיף לממש בעזרת מערך, אחרת בעייתי – יש הרבה חורים.

אם העץ מלא טורבו – בכלל מושלם מערך. אם העץ רק מלא יהיו חורים בקצה הימני של המערך. חוסך מלא מקום של מצביעים (מצביעים זה טונאז'!). נעבוד בשיטה זאת עם ערימה בינארית.

בן שמאלי, אח ימני: מצביע לבן השמאלי ביותר, וכל בן מצביע לבן שמימינו (מין רשימה מקושרת של הבנים עם עצמם). אופציונלי: מצביע לאבא.

לא הוספנו אף צומת, וגם מספר המצביעים = $O(n)$, מספר הצמתים. בשיטה הדומה לצורה בה מימשנו עץ בינארי, מספר המצביעים הוא $O(n^2)$.

חיסרון של השיטה: זמן גישה לילדים הוא $O(n)$, לעומת 2 השיטות הקודמות $O(1)$.

[1] מקורי של גדי

אלגוריתמים בסיסיים בעצים

נרצה לחשב לכל צומת את העומק שלה (=המרחק שלה מהשורש):

עבור השורש – נאתחל ב-0.

בכל פעם שנרד ברמה, נוסיף 1, בכל פעם שנעלה ברמה נוריד 1.

סריקת Pre-Order.

נראה כי נהג מונית, שעובר על כל העץ וכותב את ערכי עומקי העצים נוסע ב- $O(n)$?

מה למדנו? $|E| = O(n)$. עוברים על כל קשת בדיוק פעמיים, מספר הקשתות פעמיים הוא $O(n)$ ולכן עלות כל הנסיעה היא $O(n)$.

הצבעי החביב יורד מהמונית וצובע את הצומת ב- $O(n)$.

לטיול / סריקה שעשינו על העץ קוראים Pre-Order. בפעם הראשונה שנבקר בצומת כלשהי נכתוב בה, לפני הביקור בבנים!

לעומת זאת, חישוב גודל כל תת עץ בעץ – נעשה עם Post Order.

ניתוח לשיעורין

<u>מרוץ מגדל אשכול</u>, המטרה: להשיג כמה שיותר דולרים

1. מתחילים בקומה 0
2. יש n צעדים
3. בכל צעד יש 2 אפשרויות:
   a. עלייה של קומה אחת
   b. ירידה של כמה קומות שרוצים
4. בכל מעבר של קומה מקבלים 1 דולר (על כל קומה דולר אחד)

נעלה $n-1$ פעמים ונרד בצעד האחרון את כל הקומות.

בעיה – אם ניקח את הצעד הכי יקר (עולה $n-1$) ונכפיל במספר הפעולות, נקבל שייקח לנו $O(n^2)$ זמן.

נסכום בנפרד את העליות ואת הירידות (~שיטת <u>הספירה</u>):

נשלם $O(n)$ דולרים על עליות. אי אפשר לרדת מתחת לקומה 0, ולכן מספר העליות חסום מלמעלה על ידי מספר העליות וגם מספר הירידות הוא $O(n)$. סה"כ: $O(n)$.

שיטה נוספת – שיטת <u>הקופונים</u>:

קופון מבצע $O(1)$ עבודה. גדי מחלק לכל שחקן 2 קופונים בכל עלייה לקומה חדשה (קופון עלייה וקופון ירידה), בסוף יישארו (אולי) קופוני ירידה, ולכל הפחות ינוצלו $n/2$ קופוני עלייה ובסה"כ $O(n)$ קופונים.

מונה בינארי:

1. מתחיל ב0.
2. מוסיפים למונה n פעמים 1.
3. עלות החלפת ביט מ-0 ל-1 או מ-0 ל-1 היא $O(1)$.

כמה ביטים יתחלפו?

- בחֵצי מהפעמים מתחלף ביט 1 (כאשר $lsb = 0$) – $\frac{1}{2}n \cdot 1$
- ברבע מהפעמים מתחלפים 2 ביטים (כאשר $lsb = 01$) – $\frac{1}{4}n \cdot 2$
- בשמינית מהפעמים מתחלפים 3 ביטים (כאשר $lsb = 011$) – $\frac{1}{8}n \cdot 3$

$$n \cdot \sum_{i=1}^{\log n} \frac{i}{2^i} \leq 2n = O(n)$$

בשיטת הקופונים: בכל צעד נביא 2 קופונים: אחד שיהפוך $0 \to 1$, והשני – כאשר מאחוריו יש שורת 1ים (וגם הוא 1) להפוך אותו בחזרה ל- 0.

יש n צעדים ובכל צעד 2 קופונים, ולכן $2n$ קופונים כלומר מספר ההחלפות הוא $O(n)$.

פונקציית פוטנציאל

אינטואיציה: דמי כיס. נניח שאנחנו דמי כיס קבועים בכל שבוע, ובכל שבוע אנחנו מוציאים כמה כסף שאנחנו רוצים (כמובן שלא ניתן לחרוג מכמות הכסף שיש לנו). בחלק מהשבועות מוציאים יותר כסף, ובחלק פחות. המטרה – חישוב הוצאה שנתית.

בסוף השנה יש לפחות 0 כסף (התחלנו ב- 0 ולא ניתן לבזבז יותר ממה שקיבלנו).

כמובן שסך הכסף: $52 \times$ דמי כיס לשבוע (לפי ההנחה שדמי הכיס קבועים), ואם נרצה לדייק נוכל לחסר מהמכפלה את כמות הכסף שיש כרגע בארנק.

פונקציית הפוטנציאל מנסה למצוא את דמי הכיס השבועיים של אלגוריתם (= עלות ממוצעת של פעולה), באופן פשוט וקל לחישוב.

בעיית המונה הבינארי בעזרת פונקציית פוטנציאל:

סימונים הקשורים בפונקציית הפוטנציאל:

לא תמיד $\Phi_0 = 0$!

$$c_i = \text{cost of the i'th move}$$
$$\sum_{i=1}^{n} c_i = \text{total cost}$$
$$\Phi_i = \text{value of } \phi \text{ after the i'th step}, \Phi_i \geq 0$$

דמי הכיס לאחר השבוע ה- i – $\hat{c}_i = c_i + \Phi_i - \Phi_{i-1}$

עבור הבעיה שלנו:

$$\hat{c}_1 = 1 + [1 - 0] = 2$$
$$\hat{c}_2 = 2 + [1 - 1] = 2$$
$$\hat{c}_3 = 1 + [2 - 1] = 2$$

נסכום את כל דמי הכיס – טור טלסקופי:

$$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} c_i + \Phi_n - \Phi_0$$

נחלץ את סכום ה- c_iים: (לרוב $\Phi_0 = 0$)

$$\sum_{i=1}^{n} c_i = \sum_{i=1}^{n} \hat{c}_i - \Phi_n + \Phi_0 = \sum_{i=1}^{n} \hat{c}_i - \Phi_n$$

$$\Rightarrow \sum_{i=1}^{n} \hat{c}_i \geq \sum_{i=1}^{n} c_i$$

נגדיר t_i להיות מספר ה1ים הרצופים מימין בצעד ה- i.

$$c_i = t_{i-1} + 1$$

כי הופכים t_{i-1} 1ים לאפסים ו-0 אחד ל-1.

$$\Phi_i - \Phi_{i-1} = 1 - t_{i-1}$$

כי כל האחדים הרצופים הפכו ל-0, ו-0 אחד הפך ל-1.

ולכן: $\hat{c}_i = c_i + \Phi_i - \Phi_{i-1} = 2$, לכל i.

משחק <u>מגדל אשכול</u> – נפתור בעזרת פונקציית פוטנציאל:

נגדיר Φ_i להיות הקומה בה אנו נמצאים אחרי הצעד ה- i.

$$\Phi_0 = 0$$
$$\hat{c}_1 = 1 + [1 - 0] = 2, \text{must go up}$$
$$\hat{c}_{2_{up}} = 1 + [2 - 1] = 2$$
$$\hat{c}_{3_{down\ 2\to 0}} = 2 + [0 - 2] = 0$$
$$\hat{c}_{3_{down\ 2\to 1}} = 1 + [1 - 2] = 0$$

נשים לב:

$$\hat{c}_{i_{down}} = c_i + \Phi_i - \Phi_{i-1} = 0$$
$$\hat{c}_{i_{up}} = c_i + \Phi_i - \Phi_{i-1} = 2$$

$$\Rightarrow \sum_{i=1}^{n} c_i \leq \sum_{i=1}^{n} \hat{c}_i \leq \sum_{i=1}^{n} \hat{c}_{i_{down}} = 2n = O(n)$$

ניתוח לשיעורין – תרגול

- $\text{amort} = \hat{c}$
- $\text{Time}(i) = c(i)$

תהי $Op_1, \dots, Op_m$ סדרה של m פעולות (הזמן לביצוע – מירבי).

לכל $1 \le i \le m$ נסמן ב- $\text{Time}(i)$ את הזמן לביצוע הפעולה Op_i.

הזמן הכולל נסמן ב- $\text{total} = \sum_{i=1}^{m} \text{Time}(i)$

הגדרה: נגיד שהזמן <u>המשוערך</u> לפעולה ה- i הוא $\text{amort}(i)$ אמ"מ:

$$\text{total} \le \sum_{i=1}^{m} \text{amort}(i)$$

ישנן 3 שיטות לבצע את פעולת ה- amort:

1. <u>שיטת הצבירה</u>

בשיטה זו נחשב את $\sum \text{Time}$ באופן ישיר, ונקרא לו total'.

$$\text{amort}(i) = \frac{\text{total}'}{m} \Rightarrow m \cdot \text{amort}(i) = \text{total}'$$
$$\Rightarrow \sum_{i=1}^{m} \text{amort}(i) = \text{total}' \ge \text{total} = \sum_{i=1}^{m} \text{Time}(i)$$

2. <u>שיטת הקופונים</u>

בשיטה זו נקצה מראש לכל פעולה כמות מסוימת של קופונים, כאשר קופון אחד ישלם על יחידת זמן אחת. חלק מהפעולות יקבלו מספר גדול של קופונים מאחרות במטרה "לעזור" בעתיד. אחרי חלוקת הקופונים צריך להוכיח שכמות הקופונים מספיקה לשלם על סיבוכיות.

$$\hat{c}(i) = \#\text{coupons}$$

3. <u>שיטת הפוטנציאל</u>

בשיטה זו נבחר פונקציה Φ הממפה את מצב מבנה הנתונים ל"פוטנציאל" של המבנה ברגע מסוים בזמן ביצוע סדרת הפעולות.

$\Phi(i)$ – הפוטנציאל אחרי ביצוע הפעולה ה- i

$\Phi(0)$ – הפוטנציאל כאשר המבנה ריק

שתי דרישות על פונקציית הפוטנציאל: $\Phi(i) \ge 0$, $\Phi(0) = 0$

העלות המשוערכת $\hat{c}$ תהיה: $\hat{c}(i) = c(i) + \Phi(i) - \Phi(i-1)$.

$$\sum_{i=1}^{m} \hat{c}(i) = \sum_{i=1}^{m} c(i) + \Phi(m) - \Phi(0) = \text{total} + \Phi(m) - \Phi(0) = \text{total} + \Phi(m) \ge \text{total}$$

דוגמא – מערך דינמי

ראינו כי הכנסה למערך דינמי עולה $O(1)$ משוערך.

כעת – עבור מערך דינמי שיכול גם לקטון

נרצה לתמוך גם בשאילתא של מחיקה ולשמור על $O(1)$ זמן משוערך, וגם לחסוך במקום.

- טעות טרגית – כאשר אנו משתמשים בפחות מחצי מהמקום נקטין אותו בחצי. כך אם נהיה עם בדיוק חצי מהאיברים, נוכל למחוק ולהוסיף ולמחוק איברים לסירוגין וכך אחרי m פעולות נבצע $O(m \cdot n)$ עבודה – כלומר $\mathbf{O(n)}$ זמן משוערך.

הגדרה: מקדם עומס – α: מספר האיברים בפועל במערך מחולק בגודל המערך.

$$\alpha = \frac{n}{m}$$

- נרחיב כאשר $\alpha = 1$, ונכווץ כאשר $\alpha = \frac{1}{4}$.

נגדיר את פונקציית הפוטנציאל להיות:

$$\Phi(T) = \begin{cases} 2n - m & \alpha(T) \geq \frac{1}{2} \\ \frac{m}{2} - n & \alpha(T) < \frac{1}{2} \end{cases}$$

פעולת הכנסה:

נפריד ל-3 מקרים.

1. התבצעה הרחבה וגם $\alpha \geq 1/2$.

$$\hat{c}(i) = \frac{m}{2} + 1 - \frac{m}{2} + 2 = 3 \Leftarrow \begin{cases} c(i) = n + 1 = \frac{m}{2} + 1 \\ \Delta\Phi = (2n - m) - \left(2(n-1) - m/2\right) = -\frac{m}{2} + 2 \end{cases}$$

2. לא התבצעה הרחבה וגם $\alpha \geq 1/2$.

$$\hat{c}(i) = 1 + 2 = 3 \Leftarrow \begin{cases} c(i) = 1 \\ \Delta\Phi = (2n - m) - (2(n-1) - m) = 2 \end{cases}$$

3. $\alpha < 1/2$ (ברור כי במקרה זה לא תתבצע הרחבה)

$$\hat{c}(i) = 1 - 1 = 0 \Leftarrow \begin{cases} c(i) = 1 \\ \Delta\Phi = \left(\frac{m}{2} - n\right) - \left(\frac{m}{2} - (n-1)\right) = -1 \end{cases}$$

כלומר בכל מקרה העלות המשוערכת להכנסה היא $O(1)$.

פעולת <u>הוצאה</u>:

נפריד ל-3 מקרים.

1. התבצע כיווץ וגם $\alpha < {}^{1}/_{2}$.

$$\hat{c}(i) = -\frac{m}{2} + 1 + \frac{m}{2} + 1 = 2 \Leftarrow \begin{cases} c(i) = n + 1 = \frac{m}{2} + 1 \\ \Delta\Phi = \left({}^{m}/_{2} - n\right) - \left(m - (n+1)\right) = -\frac{m}{2} + 1 \end{cases}$$

2. לא התבצע כיווץ וגם $\alpha < {}^{1}/_{2}$.

$$\hat{c}(i) = 1 + 1 = 2 \Leftarrow \begin{cases} c(i) = 1 \\ \Delta\Phi = \left({}^{m}/_{2} - n\right) - \left({}^{m}/_{2} - (n+1)\right) = 1 \end{cases}$$

3. $\alpha \geq {}^{1}/_{2}$ (ברור כי במקרה זה לא יתבצע הכיווץ)

$$\hat{c}(i) = 1 - 2 = -1 \Leftarrow \begin{cases} c(i) = 1 \\ \Delta\Phi = (2n - m) - (2(n+1) - m) = -2 \end{cases}$$

כלומר בכל מקרה העלות המשוערכת ל<u>הוצאה</u> היא $O(1)$.

$$\forall i : \text{amort}(i) = O(1) \Longrightarrow \sum_{i=1}^{m} \text{amort}(i) = O(m)$$
$$\Longrightarrow \text{total} = O(m)$$

<u>מונה בינארי</u> התומך גם בפעולת <u>הפחתה</u>:

המונה יכול לייצג ערכים אי שליליים בלבד, נניח כי $n/2 = 2^k - 1$.

נראה כי סדרה של n פעולות תעלה $O(\log n)$ משוערך – ניקח את המקרה הגרוע:

ב- $n/2$ הצעדים הראשונים נוסיף אחד $(++)$, וכך נקבל שורת 1ים רצופים באורך $O(\log n)$.

כל $n/2$ הצעדים הראשונים עלו $O(n)$ זמן.

כעת ב- $n/2$ הצעדים הנותרים נבצע את הפעולות $(--,++)$ לסירוגין – עלות של $O(n \cdot \log n)$.

ולכן בסך הכל:

$$\text{total}' = O(n \cdot \log n) \Longrightarrow \text{amort}(i) = O(\log n)$$

<u>תרגיל טוב:</u>

בצעו שינוי ב<u>מבנה</u> המונה הבינארי כך שיתמוך גם ב<u>הגדלה</u> וגם ב<u>הפחתה</u> בזמן $O(1)$ משוערך.

דוגמא נוספת לניתוח לשיעורין: <u>מערך דינמי</u>

יש לנו מערך, ואנחנו רוצים למלא אותו.

כשהמערך מתמלא אנחנו יוצרים מערך חדש, מעבירים את האיברים מהישן לחדש וממשיכים למלא. כאשר החדש מתמלא בונים חדש חדש וכך הלאה...

המערך צריך להיות לפחות חצי מלא, נניח שכל פעם מכפילים את גודל המערך.

כמה עבודה מבצעים על $n = 2^i$ איברים? (מתחילים ממערך בגודל 1)

<u>נפריד לשני מקרים:</u>

1. קומדיה – הוספה רגילה – סה"כ עבודה $O(n)$
2. טרגדיה – הוספה שדורשת הכפלת גודל המערך

יש $\log n$ טרגדיות, ולכן העבודה היא:

$$\sum_{i=1}^{\log n} 2^i = O(n)$$

<u>שיטת הפוטנציאל:</u>

נגדיר $\phi_i = 2n_i - s_i$.

עבור צעד <u>קומדיה</u>:

$$\hat{c}_i = 1 + (2n_i - s_i) - (2n_{i-1} - s_{i-1}) = 3$$

עבור צעד <u>טרגדיה</u>:

$$\hat{c}_i = c_i + \phi_i - \phi_{i-1} = n_i + \big(2n_i - 2(n_i - 1)\big) - \big(2(n_i - 1) - (n_i - 1)\big) = 3$$

ולכן עלות n הפעולות היא $O(3n) = O(n)$.

מבנה נתונים

מוגדר על פי הפעולות שהוא תומך בהן ← עונה על שאילתא (Query)

- כיצד לממש אותן?
- באיזו סיבוכיות (לרוב זמן) אנו עונים עליהן

שני מבני נתונים בסיסיים ראשוניים – מחסנית ותור.

איבר ייוצג על ידי ערכו והמפתח שלו: $[data, key]$

מחסנית

נסמן ב- x את המפתח לאיבר.

1. $push(x)$, מכניסים את האיבר x למחסנית
2. $pop()$, הוצא והחזר את האיבר האחרון שהוכנס – LIFO

מימוש מחסנית בעזרת מערך:

נסמן ב- n את כמות האיברים במבנה שאו מנתחים, top יסמן את ראש המחסנית.

```
pop(S):
    value = S[top]
    S[top] = NULL
    --top
```

מימוש מחסנית בעזרת רשימה מקושרת:

ניתן לדחוף את האיברים לרשימה בסדר המקורי שלהם, אבל יהיה הרבה יותר יעיל לדחוף בכל פעם לתחילת הרשימה, וכך פעולות הכנסה והוצאה יעלו $O(1)$, לעומת ההכנסה הרגילה – $O(n)$.

במימוש זה top הוא מצביע לתחילת הרשימה (כלומר האיבר האחרון שנכנס)

```
pop(S):
    if top == NULL
        return ERROR
    node = top
    top = node.prev
    x = node.value
    delete node
```

```
push(S, x):
    node = new Node()
    node.value = x
    node.prev = top
    top = node
```

תור

נסמן ב- x את המפתח לאיבר.

1. enqueue(x), מכניסים את האיבר x לתור
2. dequeue(), הוצא והחזר את האיבר הראשון שהוכנס – FIFO

מימוש תור באמצעות מערך מעגלי:

נאחסן שני מצביעים – להתחלה הנוכחית ולסוף הנוכחי, בנוסף גודל. אתחול:

$$\begin{cases} \text{rear} = 0 & \text{first empty space} \\ \text{front} = 0 & \text{index of the first in the queue} \\ \text{size} = 0 & \text{how many elements} \end{cases}$$

```
enqueue(q, x):
    if size == n
        return ERROR
    q[rear] = x
    rear = (rear + 1) % n
    ++size
```

```
dequeue(q):
    if size == 0
        return ERROR
    x = q[front]
    front = (front + 1)% n
    --size
```

מימוש ע"י רשימה מקושרת:

מצביעים להתחלת הרשימה לסופה – כך נוכל להכניס ולהוציא מהתור ב- $O(1)$.

בסיבוכיות מקום נוספת $O(1)$ – מימוש תור בעזרת שתי מחסניות:

נקדיש מחסנית אחת להכנסה (S_1) ומחסנית שנייה להוצאה (S_2).

הכנסה: בכל פעם שנכניס פשוט נכניס ל- S_1.

הוצאה: אם S_2 לא ריקה נוציא ממנה – $pop(S_2)$.

אחרת - נרוקן את S_1 ל- S_2, ואז נעשה $pop(S_2)$.

נתונות שתי רשימות מקושרות, וידוע כי החל ממקום מסוים הן שוות (זנבות חופפים). הצע אלגוריתם יעיל למציאת אורך הזנב החופף.

נרוקן את הרשימות לתוך 2 מחסניות, ונאתחל מונה ב- 0. כל עוד ראשי המחסניות שווים נגדיל מונה ונעשה pop() לשתי המחסניות. ערכו הסופי של המונה הוא אורך הזנב החופף.

תור עדיפויות

"זה הנושא הראשון שהוא טיפה מעניין"

לא מבקרים בית חולים אם המנהל לא מחכה לך בפתח[2]

מעתה והלאה נעסוק ברשומות, בכל רשומה יש שדות ומפתח. n רשומות, מתחילים ממבנה נתונים ריק / מקבלים ישר מבנה נתונים עם רשומות.

גבוה / נמוך – תיעדוף על פי הכי גבוה / הכי נמוך.

פעולות:

1. מצא "חביב" – ראשון בסדר העדיפויות
2. הוסף רשומה
3. הוצא "חביב"
4. בנה את מבנה הנתונים ל- n רשומות
5. לאחד 2 מבנים
6. שיפור חביבות

סיפור עם מוסר השכל:

אמא אחת ילדה ילד באמצע חורף באזורים האלו מנותקים פעם פעם פעם והיא הייתה צריכה להחליט באביב ללכת לעיר ולרשום את הילד ואז היא אמרה "אם אני ארשום שהוא נולד באביב הוא ילך לצבא יותר מאוחר, אם ארשום בסתיו הוא ילך לבית הספר יותר מוקדם" שאלו אותה – למה לא שתרשמי את התאריך הנכון?

מסקנה: את העדיפות לא אנחנו קובעים.

מערך ממוין (מעגלי):

מצא חביב – $\mathrm{O}(1)$

הוסף רשומה – $\mathrm{O}(\mathrm{n})$

הוצא חביב – $\mathrm{O}(1)$

בנה מבנה, שקול למיון

איחוד – $\mathrm{O}(\mathrm{n})$

שיפור חביבות – $\mathrm{O}(1)$

רשימה מקושרת (לא ממוינת!):

מצא חביב – $O(n)$

הוסף רשומה – $O(1)$

הוצא חביב – $O(n)$

בנה ל-n רשומות – $O(n)$

איחוד – $O(1)$

שיפור חביבות – $O(1)$

---

[2] לנדאו, 2022

ערימה בינארית

- עץ בינארי מלא מיושם במערך.
- בכל צומת יש רשומה
- תכונת הערימה:
  - המפתח של הרשומה בצומת V "חביב" יותר (או שווה) מכל המפתחות בתת העץ שלו.

למשל בערימת מקסימום – כל מפתח של רשומה גדול יותר מכל המפתחות בתת העץ שלו.

<u>פעולות:</u>

1. מצא חביב – נמצא בשורש הערימה, $O(1)$
2. הוסף רשומה – נדחוף למקום הפנוי הבא ופעולה בסיסית עד עצירה, $O(\log n)$
3. הוצא חביב – $O(\log n)$
4. בנה את מבנה הנתונים ל- יפתח רשומות – $O\left(\text{יפתח}\right)$
5. לאחד 2 מבנים – $O(m \cdot \log n)$ אם $m \ll n$, אחרת $O(m+n)$ (הוספה לערימה הגדולה או בניית ערימה חדשה לגמרי)
6. שיפור חביבות – נשפר ונפעיל פעולה בסיסית עד עצירה, $O(\log n)$

<u>פעולה בסיסית (Heapify)</u>

בהינתן 3 צמתים – x, y, z כך ש- x הוא האבא של y, z:

א. דו קרב בין y ו- z
ב. המנצח נגד ההורה. אם הילד ניצח, מחליפים בינו ובין ההורה.

ב<u>הוספת רשומה</u> נדחוף אותה למקום הפנוי הבא ואז נפעיל את הפעולה הבסיסית עד שהרשומה החדשה היא השורש, או שהפסידה להורה / אח.

<u>הוצא חביב</u>

מוציאים את האיבר הראשון ושמים במקומו את האחרון – פתרנו את בעיית המבנה.

כעת נפעיל את הפעולה הבסיסית על השורש ושני בניו, עד שהאבא ניצח או שהאחרון שוב עלה.

בניית מבנה הנתונים ל- n רשומות

נניח שאנו עובדים עם ערימה מלאה טורבו.

נדחוף את הכל למערך – $O(n)$ עבודה.

עבור כל העלים, עבודה $\frac{n}{2} \times 0$

שכבה אחת מעליהם – עבודה $\frac{n}{4} \times 1$. עולה שוב – $\frac{n}{8} \times 2$.

בסך הכל:

$$\sum_{i=1}^{\log n} \frac{n}{2^i} \times (i-1) \leq n \times \sum_{i}^{\infty} \frac{i-1}{2^i}$$

שיטת הקופונים:

לכל צומת נותנים שני קופונים. בקופון אחד הוא משתמש, את הקופון השני הוא מעביר לבנק הקופונים של ההורים שלו. ככה לכולם יש מספיק קופונים כדי להגיע עד למעלה, ולכן הסיבוכיות היא $O(n)$.

ערימה בינומית

<u>עצים בינומיים:</u>

הגדרה רקורסיבית – B_0 = עץ עם צומת אחת.

$$B_i = \text{Union}(B_{i-1}, B_{i-1})$$

החביב יותר מבין שורשי B_{i-1} יהיה השורש של B_i.

<u>תכונות:</u>

- גובהו של B_i הוא i, נוכיח באינדוקציה:

$$h(B_0) = 0$$

נניח עבור B_k ונוכיח עבור B_{k+1}:

ניקח את עץ B_k השמאלי – הוא בגובה k ולשורש שלו יש אבא (שהוא השורש של עץ B_{k+1}), כלומר גובה כל העץ הוא $k+1$.

- מספר העצים ב- B_i הוא 2^i, נוכיח באינדוקציה:

בסיס האינדוקציה – ברור. נניח שב- B_k יש 2^k, אזי בעץ B_{k+1} יש $2^k + 2^k = 2^{k+1}$ צמתים.

- מספר הילדים של השורש B_i הוא i, בכל צעד נוסף בן אחד, ברור.

<u>תרגיל:</u> הוכיחו כי ישנם בדיוק $\binom{k}{i}$ צמתים בעומק i. (גם באינדוקציה!)

החיסרון של עצים בינומיים – מספר האיברים חייב להיות חזקה של 2. <u>הפתרון</u>: יער מקושר

ערימה בינומית היא רשימה מקושרת של עצים בינומיים, ובה אין שני עצים מאותו סדר.
הייצוג הבינארי של מספר האיברים בערימה יראה את העצים בהם נשתמש.

למשל עבור $n = 20 = 2^4 + 2^2$ נשתמש ב- B_2, B_4.

ניתן לכל מספר n את העצים הדרושים לו, ותכונת הערימה תתקיים <u>לכל עץ בנפרד</u>.

- רשימה מקושרת של שורשי העצים ביער. (לכל היותר $\log n$ עצים)

פעולות על המבנה:

1. מצא חביב – מוצאים את החביב מבין שורשי העצים, לכן $O(\log n)$.
2. הוצא חביב – נוציא את החביב ואז נאחד את הצאצאים עם הערימה, לכן $O(\log n)$.
3. הוסף רשומה – כמו איחוד ערימות כאשר הערימה השנייה היא B_0, לכן $O(\log n)$.
4. בנה ל- n רשומות – כמו להוסיף n פעמים 1 למונה בינארי, לכן $O(n)$ משוערך.
5. אחד 2 ערמות – כמו חיבור מספרים בינאריים עם שארית, ולכן $O(\log n)$.
6. שפר חביבות – עולים למעלה בעץ עד שמגיעים לשורש או נעצרים, לכן $O(\log n)$.

ערימת פיבונאצ'י

"זה מהיותר קשים"

- יער של עצים, בו כל עץ מקיים את תכונת החביבות.
- שורשי העצים משורשרים.
- שני סוגי צמתים, אפורים ושחורים – רגילים וצבועים.
- מצביע min לשורש המינימלי.
- ניתן לנתק לצומת לכל היותר בן אחד, ברגע שמנתקים מצומת בן היא משחירה.

סימונים:

$$\begin{cases} Degree[x] = degree\ of\ node\ x - number\ of\ children \\ Mark[x] = x's\ mark - grey\ or\ black \\ t(H) = number\ of\ trees \\ m(H) = number\ of\ marked\ nodes \end{cases}$$

פונקציית הפוטנציאל של המבנה:

$$\Phi(H) = t(H) + 2 \cdot m(H)$$

<u>הוסף רשומה:</u>

1. צור עץ חדש בעל חולייה אחת
2. הוסף משמאל ל- min
3. עדכן את min

הפעולה עולה $O(1)$ משוערך ובפועל.

בפוטנציאל גדל רק מספר העצים ב-1, ולכן $\Delta\Phi = 1$.

$$\hat{c}_i = c_i + \Delta\Phi = O(1) + 1 = O(1)$$

<u>איחוד ערימות:</u>

שרשר את שתי הערימות (מקשרים בין הרשימות המקושרות), עדכן מינימום.

הפעולה עולה $O(1)$ משוערך וגם בפועל.

הפוטנציאל לא השתנה – הפוטנציאל של האיחוד הוא סכום הפוטנציאלים.

$$\hat{c}_i = c_i + \Delta\Phi = O(1) + 0 = O(1)$$

<u>הוצא חביב:</u>

1. הוצא את החביב ושרשר את ילדיו לרשימת השורשים.
2. צמצם את רשימת השורשים כך שלא יישארו שני עצים בעלי אותו מספר ילדים.

תהליך הצמצום יתבצע על ידי מערך מונים (המונה את מספר הבנים) של רשימת העצים – בכל פעם שיהיו לנו שני עצים עם אותו מספר הבנים נאחד אותם ונוסיף את עץ האיחוד לתא הבא במערך, עד שלא יהיו שני עצים עם אותו מספר בנים.

אנחנו נפגוש בכל עץ בפעם הראשונה רק פעם אחת, ובכל איחוד מספר העצים יקטן ב-1.

ניתוח סיבוכיות הזמן של פעולת הוצא חביב:

נסמן ב- $D(n)$ את מספר הילדים המקסימלי לעץ כאשר בערימה n רשומות.

סיבוכיות הזמן של הוצא חביב היא $O\big(D(n)+t(H)\big)$

סדר גודל של מספר העצים החדש בערמה (נוספים לנו $D(n)$ עצים).

סיבוכיות הזמן המשוערכת של האלגוריתם:

$$c_i = t_i + D(n)$$

לאחר הפעולה אין שני עצים עם אותה דרגה, ולכן: $t_i \le D(n)+1$.

$$\Phi_i - \Phi_{i-1} = t_i + 2m_i - (t_{i-1} + 2m_{i-1}) = t_i - t_{i-1}$$

$$\Rightarrow \hat{c}_i \le t_{i-1} + D(n) + D(n) + 1 - t_{i-1} \Rightarrow \hat{c}_i = O\big(D(n)\big)$$

לכן הפעולה עולה $O\big(D(n)\big)$ משוערך.

<u>שפר חביבות:</u>

- שפר את החביבות.

אם האבא לא מסומן:

1. נתק את הצומת x מאביה
2. סמן את האבא בשחור
3. הוסף את העץ ששורשו x לרשימת העצים, עדכן את min

אם האבא מסומן:

1. נתק את x מ- $p[x]$ והוסף את העץ לרשימת העצים
2. נתק את $p[x]$ מ- $p\big[p[x]\big]$, צבע לאפור והוסף אותו לרשימת העצים:
   a. אם $p\big[p[x]\big]$ אפור, סמן אותו וסיימנו
   b. אחרת, חזור לשלב 2 עבור $p\big[p[x]\big]$

נניח שיש רצף של α אבות שחורים החל מאביו של x.

נעשה $O(\alpha)$ פעולות.

עלות משוערכת $O(1)$:

$$c_i = \alpha + O(1)$$
$$t_i = t_{i-1} + \alpha + 1$$
$$m_i \le m_{i-1} - \alpha + 1$$
$$\Rightarrow \Delta\Phi \le \alpha + 2\cdot(-\alpha+2) = 4-\alpha$$
$$\Rightarrow \hat{c}_i = \big(\alpha + O(1)\big) + 4 - \alpha = O(1)$$

מחיקת איבר:

- שפר את החביבות של x למקסימום (בערמת מינימום $x \leftarrow -\infty$)
- הוצא חביב

סיבוכיות זמן משוערכת: $O\big(D(n)\big)$:

1. שיפור חביבות ב- $O(1)$
2. הוצא חביב ב- $O\big(D(n)\big)$

מהו מספר הילדים המקסימלי של עץ בערמה? (נמצא חסם עליון ל- $D(n)$)

נגדיר: העץ עם הכי מעט צמתים שלשורשו k ילדים $= S_k$.

כאשר כל הערימה מורכבת מעץ אחד, $S_{k_{max}}$ יהיה העץ שלנו, והמספר שלנו הוא k_{max}.

$$S_0 = 1$$
$$S_1 = 2$$

לבן ה- i מימין יש לפחות $i - 2$ ילדים (כשנכנס חייב היה להיות עם לפחות i ילדים, ואולי לקחו לו אחד והוא הפך לשחור), ולכל הפחות S_{i-2} צמתים בתת העץ שלו.

לכן מספר הצמתים בעץ S_k לכל הפחות:

$$\underbrace{1}_{root} + \underbrace{1}_{left\ child} + \sum_{i=2}^{k} S_{i-2} = 2 + \sum_{i=0}^{k-2} S_i$$

תכונות חיוניות של מספרי פיבונאצ'י (הוכחות פשוטות באינדוקציה):

1. $F_k \geq \phi^k$
2. $k \geq 2 \Longrightarrow F_k = 2 + \sum_{i=0}^{k-2} F_i$

$$\Longrightarrow n \geq S_k \geq F_k \geq \phi^k$$

$$\Longrightarrow k \leq \log_{\phi} n$$

$$\Longrightarrow D(n) = O(\log n)$$

- הפעולות מצא חביב ובנה ל-n רשומות טריוויאליות: גישה ל-min ו-n הוספות.

ערימת min – max

תור עדיפויות דו כיווני: מינימום ומקסימום כאחד.

הפעולות בהן תומך המבנה:

1. הוסף רשומה, $O(\log n)$
2. הוצא מינימלי, $O(\log n)$
3. הוצא מקסימלי, $O(\log n)$
4. מצא מינימלי, $O(1)$
5. מצא מקסימלי, $O(1)$

כדי לענות על השאילתות האלו יכלנו פשוט לבנות ערימת מינימום וערימת מקסימום לרשומות שלנו, אך בערימת $min - max$ נתחזק 2 ערימות תוך כדי שימוש בחצי מהזיכרון שהיינו משתמשים בפתרון הטריוויאלי.

ערימת max – min: עץ בינארי שלם (או שלם טורבו) המיוצג במערך.

תכונת הערימה – לכל צומת x במבנה:

1. אם x ברמה זוגית – הוא המינימום בתת העץ המושרש בו.
2. אם x ברמה אי זוגית – הוא המקסימום בתת העץ המושרש בו.
   - כמובן שהרמה של השורש היא 0.

בעצם הערימה מורכבת מ-2 ערימות – ערימת מינימום ומקסימום משורשות אחת עם השנייה, כל אחת עם $n/2$ איברים.

הכנסה:

באופן דומה להכנסה לערימה בינארית רגילה – נכניס את x למקום הפנוי הראשון במערך.

כעת נשאר לטפל רק בתוכן הערימה, טיפלנו במבנה.

נניח בה"כ שהצומת נכנסה לרמה זוגית, נסמן $y = parent(x)$.

- אבחנה: במסלול בין x לשורש כל הצמתים ברמות האי זוגיות גדולים מכל הצמתים ברמות הזוגיות.

1. עבור $x < y$:

במקרה כזה x חייב להישאר ברמה זוגית (כדי שהתכונה תישמר). ננסה לפעפע את x כלפי מעלה <u>רק ברמות הזוגיות</u> (כלומר ננסה להחליף את x עם $parent(y)$ וכך רקורסיבית עד עצירה או ש- x הפך לשורש).

2. עבור $x > y$:

במקרה זה כעת לא מתקיימת תכונת הערימה – צומת ברמה זוגית הגדולה מצומת ברמה אי זוגית. לכן נחליף בין x ו- y ונפעפע את x כלפי מעלה <u>רק ברמות האי זוגיות</u>.

מיונים

<u>קלט:</u> n רשומות, לכל רשומה מפתח.

<u>פלט:</u> n רשומות ממוינות: $a_{i_1}, a_{i_2}, \dots, a_{i_n}$, כך ש- $a_{i_j} \le a_{i_{j+1}}$.

במקרה הכללי (ללא הגבלות על המפתחות), החסם התחתון למיון n רשימות הוא $n \cdot \log n$.

מיונים פשוטים

מיונים אינטואיטיביים ויקרים.

<u>מיון מקסימום Max Sort</u>
בסיבוב ה- i נמצא את האיבר המקסימלי ה- i ונשים אותו במקום המתאים.

<u>סיבוכיות זמן:</u> $O(n^2)$

<u>מיון בועות Bubble Sort</u>
נעבור על המערך לכל היותר n פעמים. בכל מעבר כאשר נראה זוג איברים שאינם מסודרים נחליף ביניהם, וכך לאט לאט בעלי המפתחות הגדולים יבעבעו קדימה והקטנים אחורה.

במעבר ה- i נגיע עד האיבר ה-$n - i + 1$, כי בכל מעבר המקסימלי בסיבוב זה יגיע למקומו הסופי. לכן, <u>סיבוכיות הזמן</u> היא $O(n^2)$.

למיון זה יש יתרון על $Max\ Sort$ – תוך כדי הבאת המקסימום למקומו אנו מקרבים את שאר האיברים למקומם הממוין, ולכן ניתן לפעמים לעצור אחרי פחות מ- n איטרציות.

לעומת זאת ב- $Max\ Sort$ אנחנו עושים $O(n)$ החלפות, בעוד שב- $Bubble\ Sort$ נעשה $O(n^2)$ החלפות – פעולה יחסית יקרה.

<u>מיון הכנסה Insertion Sort</u>
במעבר ה-i על המבנה $i - 1$ האיברים הראשונים. ניקח איבר ונעבור על המקומות, נדחוף אותו במקום שבו האיברים יישארו ממוינים וכך בשלב הבא יהיו i איברים ראשונים ממוינים.

<u>סיבוכיות זמן:</u> $O(n^2)$, במקרה הכי גרוע נקבל את האיברים ממוינים בסדר הפוך וכך נבצע במעבר ה- i $O(i)$ פעולות כלומר בסך הכל $O(n^2)$.

מיון זה מעולה כאשר הרשומות מגיעות להן <u>טיפין טיפין</u>, ולא כולן בידינו ברגע המיון.

מיונים אלו קלים יותר למימוש, והדבר מהווה יתרון לעומת מיונים אחרים. בנוסף, כאשר מספר הרשומות הוא קטן, או שאנו מבצעים את המיון רק פעם אחת לא באמת אכפת לנו מסיבוכיות הזמן. עבור מקרים אחרים מאוד לא רצוי להשתמש במיונים אלה.

מיונים מתוסבכים

מיונים פחות קלים ויותר זולים.

<u>מיון ערמה – Heap Sort</u>

א. נכניס את האיברים לערמת מינימום.
ב. נוציא $n-1$ פעמים את החביב, נחליף אותו עם המקום האחרון, ונחזיר לערמה את תכונת החביבות.

אחרי האיטרציה ה- i כל i האיברים הימניים במערך יהיו ממוינים, והחלק השמאלי של המערך יהווה את הערמה החדשה.

<u>סיבוכיות זמן:</u> $O(n \cdot \log n)$. בניית הערמה עולה $O(n)$, נשלם n פעמים $O(\log n)$, ולכן בסך הכל – $O(n \cdot \log n)$.

<u>מיון מיזוג – Merge Sort</u>

4 קבצי מחשב: שני קבצי קלט (I_1, I_2) ושני קבצי פלט (O_1, O_2).

בקבצי הקלט יש בלוקים בגודל 2^i ממוינים, בקבצי הפלט נרצה בלוקים ממוינים בגודל 2^{i+1}.

מיזוג 2 בלוקים ממוינים: עם שני מצביעים לתחילת כל בלוק. נשים את המינימום ונקדם את המצביע שלו, עד שהגענו לסוף של שני הבלוקים. בשיטת הקופונים – נחלק קופון לכל איבר ונשלם רק כאשר ניצחנו (נכנסו לפלט). כל איבר יכול לנצח רק פעם אחת ובכל השוואה איבר אחד בדיוק מנצח. בסוף התהליך שני הסרטים מלאים, כל סרט באורך n ולכן עלות של איטרציה אחת היא – $O(n)$.

לאחר כל איטרציה כזאת הקבצים מחליפים תפקיד, קבצי הפלט הקודמים יהיו הקלט החדש של קבצי הקלט הקודמים. גם i גדל ב-1.

בסיבוב הבא נשלם $O(n)$ על המיזוג. יהיו $\log n$ איטרציות (סיימנו כאשר $2^i = n$ לכן $i = \log n$). בסך הכל – עלות $O(n \cdot \log n)$.

מיון מיזוג – טוב במיוחד כאשר יש לנו זיכרון איטי וזיכרון מהיר.

- האיטי זול, המהיר יקר.

כאשר בתוכנית שלנו יש מעבר מידע בין הזיכרונות (בבאסים שמעבירים בלוקים)

גורם הבעיות במיון ערמה הוא ההחזרה של תכונת החביבות. כאשר המערך מאוד גדול המערך שלנו מתפרש על המון בלוקים, ובמיון הערמה אנחנו קופצים בין הבלוקים האלו בלי הפסקה, למרות שמשתמשים רק באחוז קטן מהבלוק. <u>בזבוז של זמן!</u>

לעומת זאת, במיון מיזוג נסרוק <u>איברים רציפים</u> – נשתמש בכל האינפורמציה של הבלוק.

במה שתיארנו קודם לכן – ההעברה בין הקבצים – יש משהו <u>מטומטם</u>.

<u>הרכיב המטומטם:</u> מלא נסיעות הלוך חזור מהזיכרון המהיר לאיטי.

כעת, יש לנו m מוישה קבצי קלט ו- m מוישה קבצי פלט.

פלט	קלט
m	1
m^2	m
$\vdots$	$\vdots$
m^i	m^{i-1}

נעצור כאשר $i = \log_m n \Leftarrow n = m^i$. איך נבצע את ההשוואות עבור ה- m.

נבנה ערמת מינימום ל- m הראשונים וכל פעם נוציא חביב ונכניס את האיבר הבא מהקובץ שלו.

עלות שלב: $n \cdot \log m$, בסך הכל:

$$\log_m n \times n \cdot \log n = \frac{\log n}{\log m} \times n \cdot \log m = n \cdot \log n$$

חסכנו במספר ההעברות מהזיכרון הראשי למהיר.

<u>מיון מהיר – Quick Sort</u>
הקונספט: יש לנו מערך בגודל x. נבחר מספר p (מלשון $pivot$ – חציון) וניצור מצב כך שכל מי שנמצא משמאל אליו הוא קטן ממנו, וכל מי שמימינו גדול / שווה לו. לאחר מכן רקורסיבית נמיין כל חלק בנפרד.

האלו שמשאל ל- p ושמימין לו לא קשורים, ולכן נמיין <u>בנפרד</u> כל חלק.

קטנים מ- p_0		גדולים / שווים מ- p_0	
קטנים מ- p_1	גדולים / שווים מ- p_1	קטנים מ- p_2	גדולים / שווים מ- p_2

1. <u>חלוקת האיברים לשני החלקים:</u>

באנגליה רוב האצטדיון הוא לאוהדים הביתיים, יש יציע מוזנח לאוהדי החוץ. יש אוהד מקומי חוליגן שנכנס לאוהדי החוץ לתוך היציע שלהם ומתחיל לעשות מהומות. המשטרה שולפת אותו מהיציע וכל הקהל מסתכל עליו כשהמשטרה לוקחת אותו לכיוון ההפוך והוא צורח כמו <u>תרנגול שחוט</u>.

יש לנו שני מצביעים (R, L) לקצוות המערך (R לקצה הימני ו- L לשמאלי) שמתקדמים לכיוון אמצע המערך. כאשר R מצביע על מישהו שקטן מ- p ו- L מצביע על מישהו שגדול / שווה מחליפים ביניהם וממשיכים לקדם. מתי עוצרים? כאשר R יותר שמאלי מ- L.

עלות לכל חלק בנפרד: $O(x)$.

עלות לכל שלב: $O(n)$ סכום כל תתי המערכים הוא n.

כל המשחק הוא בחירת ה- $pivot$ – הרבה בלאגן של הסתברות.

2. <u>בחירת $pivot$:</u>

לא טוב לקחת את הקצוות – העסק לא יתחלק לנו כמו שצריך. נרצה להגיע למצב שבו לכל איבר יש סיכוי שווה להיבחר. טריק פשוט שלא כזה טוב אבל בסדר:

נתקדם משמאל לימין עד שנקבל x, y איברים כך ש- $x \neq y$.

נבחר את הגדול להיות ה- $pivot$. כך בכל קבוצה יש לפחות איבר אחד. אם היינו לוקחים את הקטן יכול היה להיות שהוא המינימלי וכל שאר האיברים ייכנסו לתת מערך אחד:

"אני יכול לטחון ככה מים פוראבר"

נבחר שני מספרים באופן אקראי ונחליף אותם עם שני האיברים הראשונים, וכך בהסתברות גבוהה הם שונים ובחרנו $pivot$ המחלק באופן הגון את המערך.

אם נבחר את ה- $pivot$ בצורה נכונה יהיה מספיק בשר בשני תתי המערכים והמיון מהיר.

סיבוכיות הזמן:

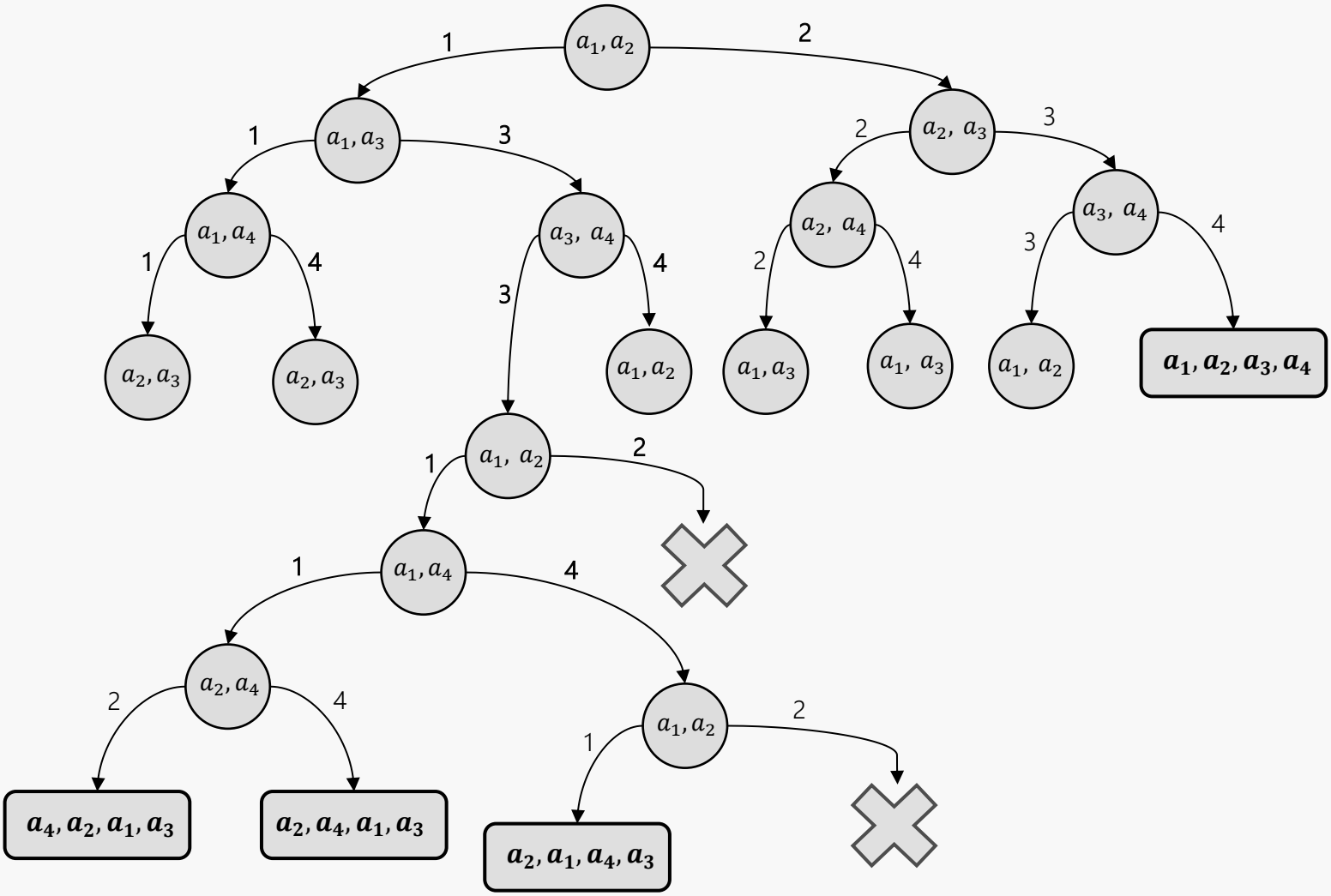
- במקרה הגרוע – $O(n^2)$
- בממוצע – $O(n \cdot \log n)$

חסם תחתון למיונים במקרה הכללי

(כאשר המפתחות הם מספרים ממשיים, או באופן כללי אין הגבלה עליהם)

עץ החלטה – עץ המתאר את הפעולות של האלגוריתם.

עץ החלטה ל-Bubble Sort: עבור 4 איברים במערך – $a_1, \dots, a_4$.



אם היינו ממלאים את העץ הוא היה מתאר את כל הפעולות האפשריות במיון. סיבוכיות הזמן של האלגוריתם היא גובה העץ. נניח שתום מחזיק את המיון הכי טוב בעולם.

הוא בונה עץ החלטות בינארי, בו יש $n!$ פרמוטציות וכל פרמוטציה מסתיימת בעלה אחר, כלומר – יש $n!$ עלים. גובה עץ ההחלטות הוא לפחות $\log n!$ במקרה הטוב (כשהעץ הוא מלא טורבו), אבל:

$$\log n! = O(n \cdot \log n)$$

ולכן – סיבוכיות הזמן הכי טובה שאפשר לקבל במיון במקרה הכללי הוא $n \cdot \log n$ (אפילו יותר גרוע אם עץ ההחלטה לא מאוזן).

מיונים לינאריים

מיונים שנעשים במקרים מיוחדים – נתונים נוספים על המפתחות.

<u>מיון ספירה – Count Sort</u>

כאשר מספר המפתחות הוא $O(n)$. נקצה 3 מערכי עזר בגודל $O(n)$ (בהתחלה מאופסים).

בסך הכל 4 מערכים: מערך הקלט, עזר 1-3 (נסמן B_1, B_2, B_3).

מערך הקלט

1	2	3	4	5	6	7	⋯
17	28	2	17	4	4	17	⋯

מערך מונים B_1

1	2	3	4	⋯	17	⋯	28	⋯
0	1	0	2	⋯	3	⋯	1	⋯

מערך מיקום B_2

1	2	3	4	5	6	7	⋯
1	1	1	2	1	2	3	⋯

מערך סכומים של B_1 B_3

1	2	3	4	5	⋯	17	18	⋯	28	29
0	0	1	1	3	⋯	3	6	⋯	6	7

מערך הפלט S

1	2	3	4	5	6	7	⋯
2	4	4	17	17	17	28	⋯

- מילוי B_1: מערך מונים סטנדרטי.
- מילוי B_2: 17 מופיע בפעם הראשונה באינדקס 1, בפעם השנייה ב-4 ובפעם השלישית ב-6, ובהתאם לכך הערכים ב-B_2 באופן דומה 28 שמופיע פעם אחת באינדקס 2.
  - מילוי B_1, B_2 עולה $O(n)$.
- מילוי B_3:

$$B_3[i] = \sum_{k=1}^{i-1} B_1[i]$$

או לחילופין:

$$B_3[i] = \begin{cases} 0 & i = 1 \\ B_3[i-1] + B_1[i-1] & i > 1 \end{cases}$$

  - כך גם מילוי B_3 עולה $O(n)$.

כלומר, מילוי מערכי העזר עולה $O(n)$. כעת אשכרה נמיין את המערך, למערך פלט S.

$$S\left[B_2[i] + B_3\left[A[i]\right]\right] \leftarrow A[i]$$

<u>סיבוכיות האלגוריתם:</u> $O(n+m)$ זמן ומקום, כך ש- n הוא גודל הקלט ו-m מספר המפתחות.

מיון בסיס – Radix Sort

נניח כי מספר המפתחות n, תחום המפתחות $[0, n^2 - 1]$.

נרצה למיין את המספרים בעזרת $count\ sort$. אם פשוט נשתמש במיון ספירה הסיבוכיות תהיה $O(n + n^2) = O(n^2)$. הטריק: נמיין את המספרים לפי ספרת האחדות, ואז לפי ספרת העשרות, ... , עד לספרה האחרונה. כל מיון יתבצע עם $count\ sort$ ובסוף כל האיטרציות נקבל את הרשומות ממוינות.

מערך הקלט	99	00	65	16	07	18	33	24	02	23
ממוין אחדות	00	02	33	23	24	65	16	07	18	99
ממוין עשרות	00	02	07	16	18	23	24	33	65	99

במקרה זה – $n = 10, m = 100$.

הוכחת נכונות המיון – קצרה ומטומטמת.

נסתכל על שני איברים i, j במערך כך ש- $i \leq j$.

$$\begin{cases} i = a \cdot 10 + b \\ j = c \cdot 10 + d \end{cases}, 0 \leq a, b, c, d \leq 9$$

נפריד למקרים:

i. עבור $a < c$ – במיון של ספרת העשרות i יצא לפני j וסיימנו.
ii. עבור $a = c$:
   a. עבור $b < d$ במיון ספרת העשרות i יצא לפני j ובמיון ספירה הסדר הפנימי בין איברים שווים נשמר, ולכן בסוף המיון i יצא לפני j.
   b. עבור $b = d$ אין חשיבות לסדר ביניהם (כי $i = j$).
   c. עבור $b < d$ – לא אפשרי (כי $i \leq j$).
iii. עבור $a > c$ – לא אפשרי (כי $i \leq j$).

כלומר – המיון מצליח למיין!

מיון 1000 רשומות כאשר המפתח הוא תעודות הזהות: $n = 10^3, m = 10^9$

הפעם נריץ 3 סיבובים של מיון ספירה, בכל פעם נמיין לפי 3 ספרות.

סיבוכיות הזמן של האלגוריתם: עבור k סיבובים ו-n רשומות – $O(n \cdot k)$

כאשר תחום המפתחות הוא $[0, n^k]$ הטריק הזה טוב כאשר $k = O(\log n)$, אחרת הסיבוכיות היא $O(n \cdot \log n)$ ויש לנו טריקים יותר טובים בסיבוכיות כזאת.

מיון לפי תאריך לידה – 3 סיבובים:

1. לפי היום (גודל 31).
2. לפי החודש (גודל 12).
3. לפי השנה.

לסיכום – ממיינים מהחלק הפחות חשוב אל החלק החשוב.

עץ חיפוש

"אסור להגיד פה פפרוני"

יש לנו n רשומות, לכל רשומה מפתח.

<u>פעולות:</u>

1. מצא רשומה
2. הוסף רשומה
3. הוצא רשומה

לצורך נוחות נניח שכל המפתחות שלנו שונים.

- נלמד כ-5 מבני נתונים בנושא.

<u>שני המבנים הקלאסיים:</u>

<u>רשימה מקושרת לא ממוינת:</u>

1. מצא רשומה – $O(n)$.
2. הוסף רשומה – אם צריך לבדוק שאין כפילות מפתחות $O(n)$, אחרת $O(1)$.
3. הוצא רשומה – $O(n)$.

<u>מערך ממוין:</u>

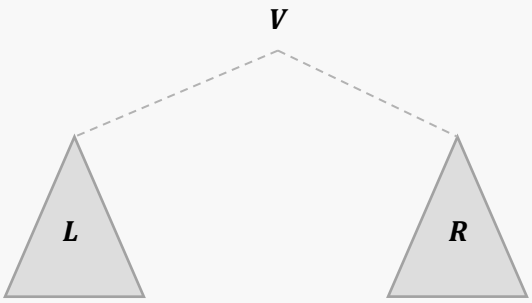
1. מצא רשומה – $O(\log n)$.
2. הוסף רשומה – $O(n)$.
3. הוצא רשומה – $O(n)$.

עצי חיפוש <u>בינאריים / כלליים</u>, <u>מאוזנים / לא מאוזנים</u>.

- רוב העצים שנעסוק בהם כלליים ומאוזנים.
- ברוב המקרים – בכל צומת יש רשומה.

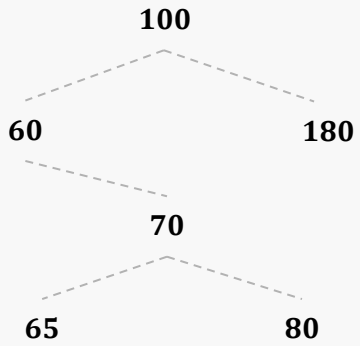
עץ חיפוש בינארי

<u>תכונת עץ החיפוש הבינארי:</u> תהי V צומת בעץ חיפוש בינארי. המפתח של V גדול מכל המפתחות בתת העץ השמאלי שלו (L), וקטן מכל המפתחות בתת העץ הימני שלו (R).



תכונה נחמדה של עץ חיפוש בינארי: סריקת $In - Order$ תיתן את איברי העץ באופן ממוין.

נשים לב: נובע מכך שהחסם התחתון לבניית העץ הוא $\Omega(n \cdot \log n)$, אחרת היינו יכולים לבנות עץ חיפוש ולסרוק אותו ב-$O(n)$, וכך מצאנו מיון שעולה פחות מ-$n \cdot \log n$ $\Leftarrow$ סתירה.



- נסמן ב- $h = O(n)$ את גובה העץ.

<u>פעולות:</u>

1. מצא רשומה – יכול להיות ספגטי ואני בקצה, $O(h)$.
2. הוסף רשומה – מציאת מקום מתאים והוספה, $O(h)$.
3. הוצא רשומה – $O(h)$.

<u>מצא רשומה:</u>

1. אם המפתחות שווים – מצאנו.
2. הצומת גדולה מהמפתח – פנה שמאלה. אחרת – פנה ימינה.
3. הגענו לעלה והמפתחות לא שווים – עוצרים.

הוסף רשומה:

מצא רשומה, $O(h)$.

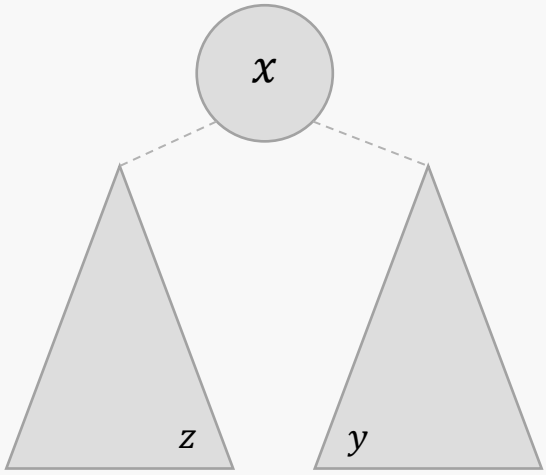
1. אם מצאנו – ללכת ללקוח, $O(1)$.
2. אם לא מצאנו – נתקענו באבא המיועד! נוסיף אותו במקום המתאים, $O(1)$.

הוצא רשומה:

מצא רשומה, $O(h)$.

1. לא מצאנו – מעולה! $O(1)$.
2. מצאנו עלה – הורד עלה וסיימנו, $O(1)$.
3. לרשומה בן יחיד – חבר את הנכד לסבא באותו חיבור כמו ההורה, $O(1)$.
4. צומת פנימית עם 2 ילדים:

המועמדים להחליף את השורש (x): הגדול ביותר בתת העץ השמאלי (הקודם – z) והקטן ביותר בתת העץ הימני (העוקב – y). ל- y, z יש לכל היותר ילד אחד – בכיוון ההפוך.



נבחר ב- z למשל.

i. מצא x, z, $O(h)$.
ii. בטל את x והכנס במקומו את z.

מבחינת מבנה – תתבטל z, מבחינת תוכן – תתבטל x.

שתי דרכים להעביר מ-z ל-x:

1. לוקחים את כל הערכים של z ושותלים אותם אצל x.
2. חותכים וקושרים את z, הלב החדש, במקום x, הלב המקולקל.

בסך הכל – עבור צומת פנימית עם 2 ילדים, סיבוכיות של $O(1)$ נוסף למציאה, סה"כ $O(h)$.

כל הפעולות על העץ עולות $O(h)$, לכן בהמשך נרצה לאזן את העץ ולהקטין את גובהו.

מציאת מינימום בעץ: פנה שמאלה עד שלא ניתן לפנות שמאלה. (עלות: $O(h)$)

מציאת מקסימום בעץ: פנה ימינה עד שלא ניתן לפנות שמאלה. (עלות: $O(h)$)

מציאת עוקב ל-x: (עלות: $O(h)$)

1. מצא את הצומת.
2. אם ל-x בן ימני, מצא מינימום בתת העץ הימני.
3. אחרת טפס מ-x מעלה כל עוד הצומת הוא בן ימני. הצומת אליה פנינו מבן שמאלי הוא העוקב.

- עוקב: הקטן ביותר שגדול ממנו.

מציאת קודם ל-x: (עלות: $O(h)$)

1. מצא את הצומת.
2. אם ל-x בן שמאלי, מצא מקסימום בתת העץ השמאלי.
3. אחרת טפס מ-x מעלה כל עוד הצומת הוא בן שמאלי. הצומת אליה פנינו מבן ימני הוא הקודם.

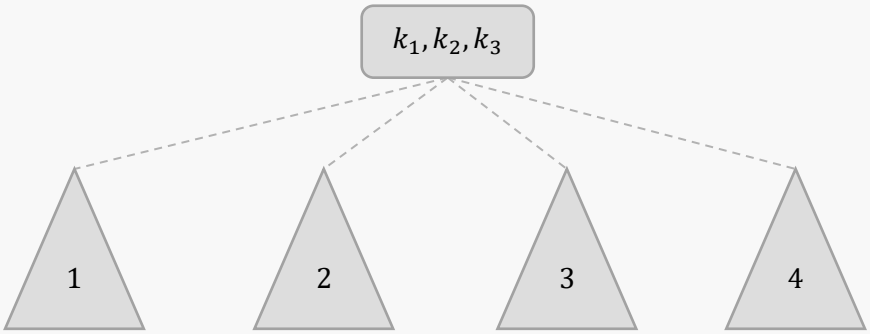
- קודם: הגדול ביותר שקטן ממנו.

עץ 2-3-4

עץ חיפוש (כללי) מאוזן.

לכל צומת 2, 3 או 4 ילדים, ובצומת יש 1-3 רשומות.

מבנה של צומת פנימית:



<u>עלה:</u> בעל 1-3 רשומות, תתי עצים $null$.

<u>תכונת עץ החיפוש:</u> [כל תת עץ 1] $< k_1 <$ [כל תת עץ 2] $< k_2 < \cdots <$ [כל תת עץ 4].

<u>תכונת העץ:</u> כל העלים באותה רמה.

גובהו של עץ 2-3-4 הוא $\Theta(\log n)$: לכל היותר $\log_2 n$ ולכל הפחות $\log_4 n$.

<u>הפעולה הבסיסית על העץ:</u> פיצול.

נניח שצומת N רוצה להיכנס לרשומה (המכילה A, B, C) ואין מקום.

1. נעלה את B להורה שלו
   a. ניצור 2 צמתים – A, C:
      1. אם $N < B$ נשים את N ב- A.
      2. אם $N > B$ נשים את N ב- C.
   b. נוסיף את B להוריו רקורסיבית.

<u>עלות הפעולה:</u> $O(h) = O(\log n)$.

פעולות:

1. מצא רשומה – $O(\log n)$.
2. הוסף רשומה – $O(\log n)$.
3. הוצא רשומה – $O(\log n)$.

מצא רשומה:

1. אם אנחנו נמצאים באחת הרשומות בצומת – סיימנו.
2. חפש בתת העץ המתאים.
3. אם הגענו לעלה – סיים.

הוסף רשומה:

מצא את הרשומה – $O(\log n)$:

1. מצאנו – ללכת ללקוח.
2. לא מצאנו – הגענו לעלה:
   a. אם יש מקום בעלה (לכל היותר 2 רשומות בעלה) הוסף לעלה, $O(1)$.
   b. אין מקום. הפעל פעולה בסיסית – פיצול, $O(\log n)$.

הוצא רשומה:

מצא את הרשומה – $O(\log n)$:

a. לא מצאנו – יופי.
b. מצאנו את הרשומה:

1) הרשומה היא עלה:
   a) יש רשומות נוספות – נמחק וגמרנו, $O(1)$.
   b) יש רשומה בודדת – לב הבעיה.
2) הרשומה היא צומת פנימי:
   a) נחליף את הצומת בעוקב או בקודם שלה (כמו בבינארי).
   b) נוציא את העלה שהחלפנו כמו מקרה הוצא עלה, $O(1)$ נוסף.

לב הבעיה – פתרון:

נוציא את הרשומה וכך הלכה לה צומת. נטפל בתתי העצים שלה באחת מהדרכים הבאות:

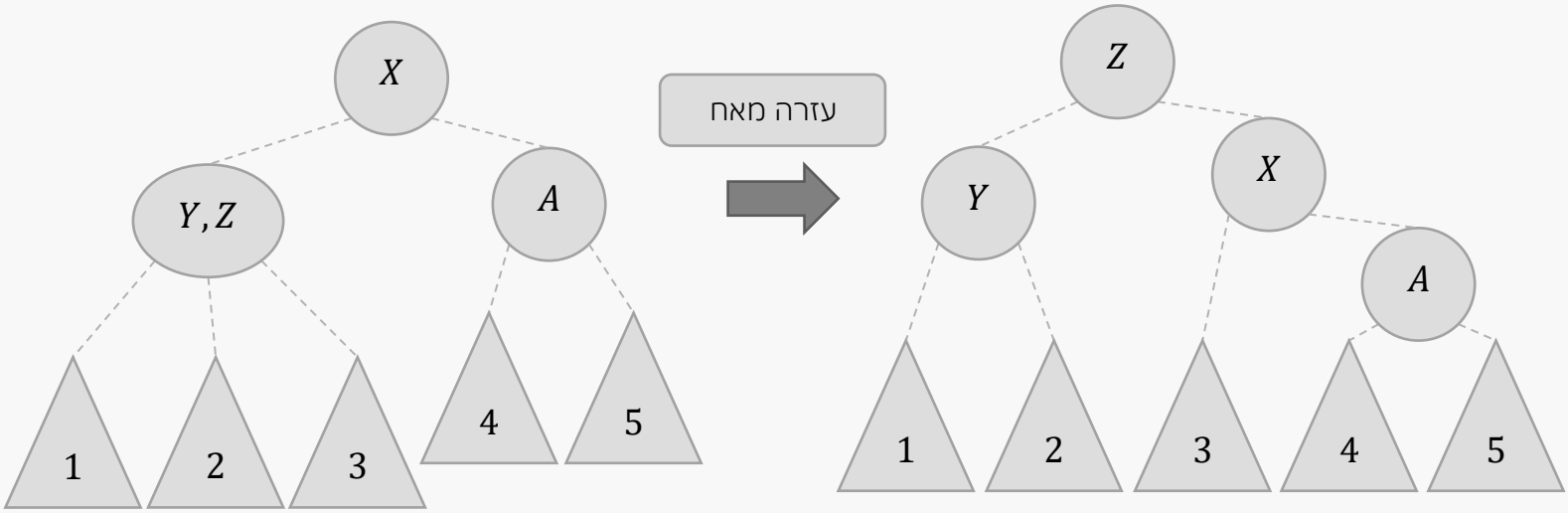
א. עזרה מאח – רק אם לאח יותר מרשומה אחת.
ב. עזרה מהורה – רק אם להורה יותר מרשומה אחת.
ג. הגדלת הבעיה – אם אח והורה לא יכולים לעזור.

ההסבר על הדרכים בעמוד הבא מתייחס לאיזון כללי, בהוצאת רשומה נקשר בין תתי העצים של הרשומה שהוצאנו לאביה וכך נוציא את הרשומה באופן מאוזן.

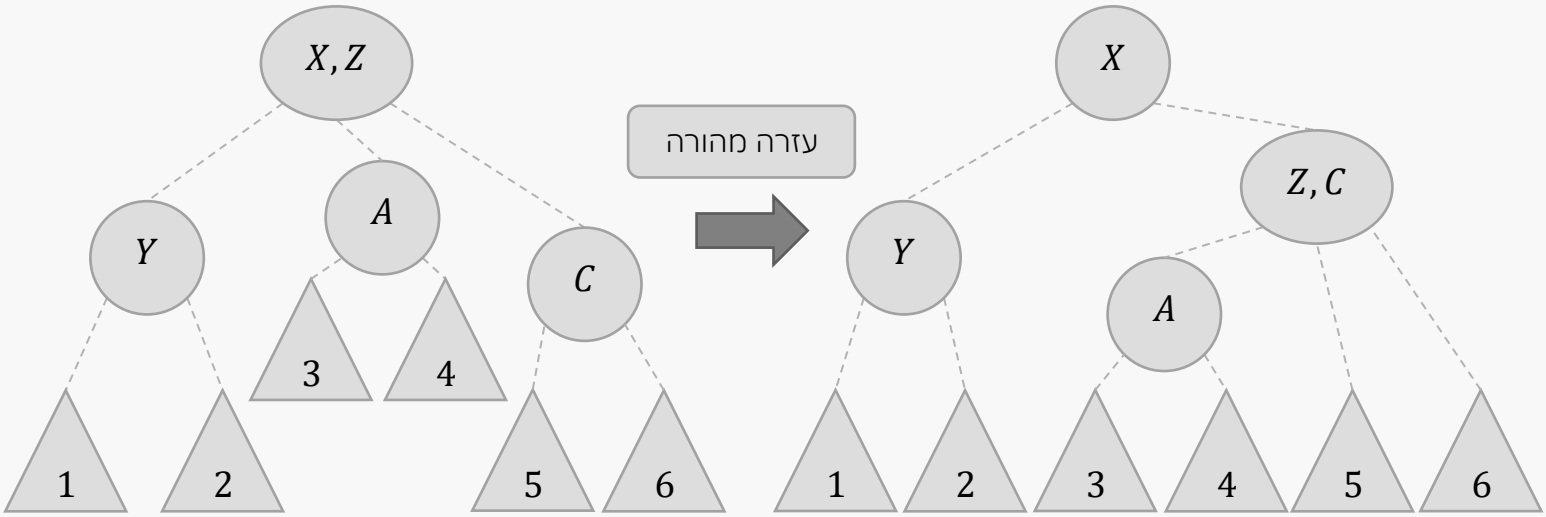
איזון עלי העץ:

בשרטוטים הבאים A ותת העץ שלו לא מאוזנים עם שאר העץ – נרצה להעמיקם.

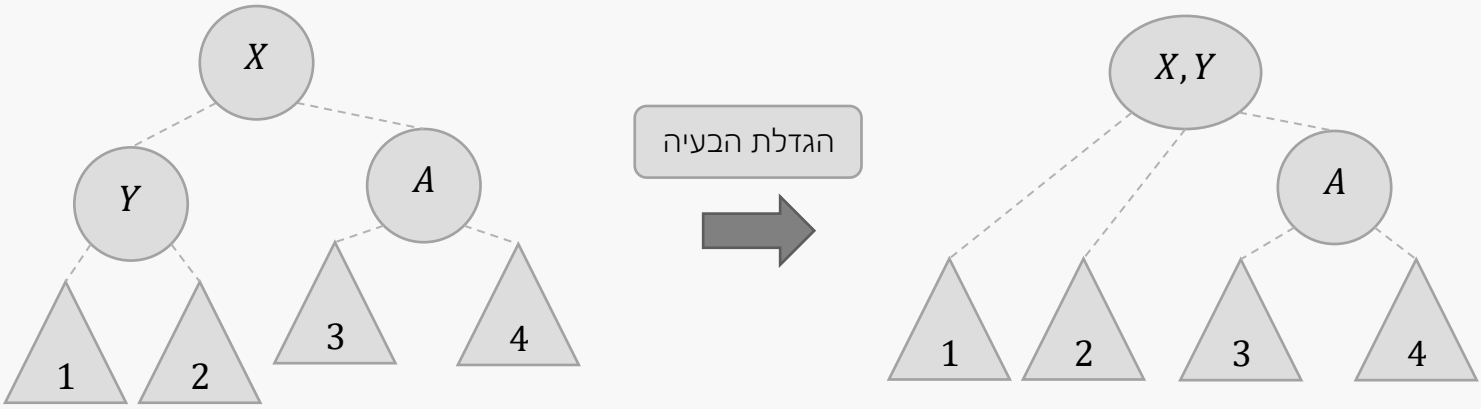
עזרה מאח – רק כאשר לאח יש יותר מרשומה אחת.



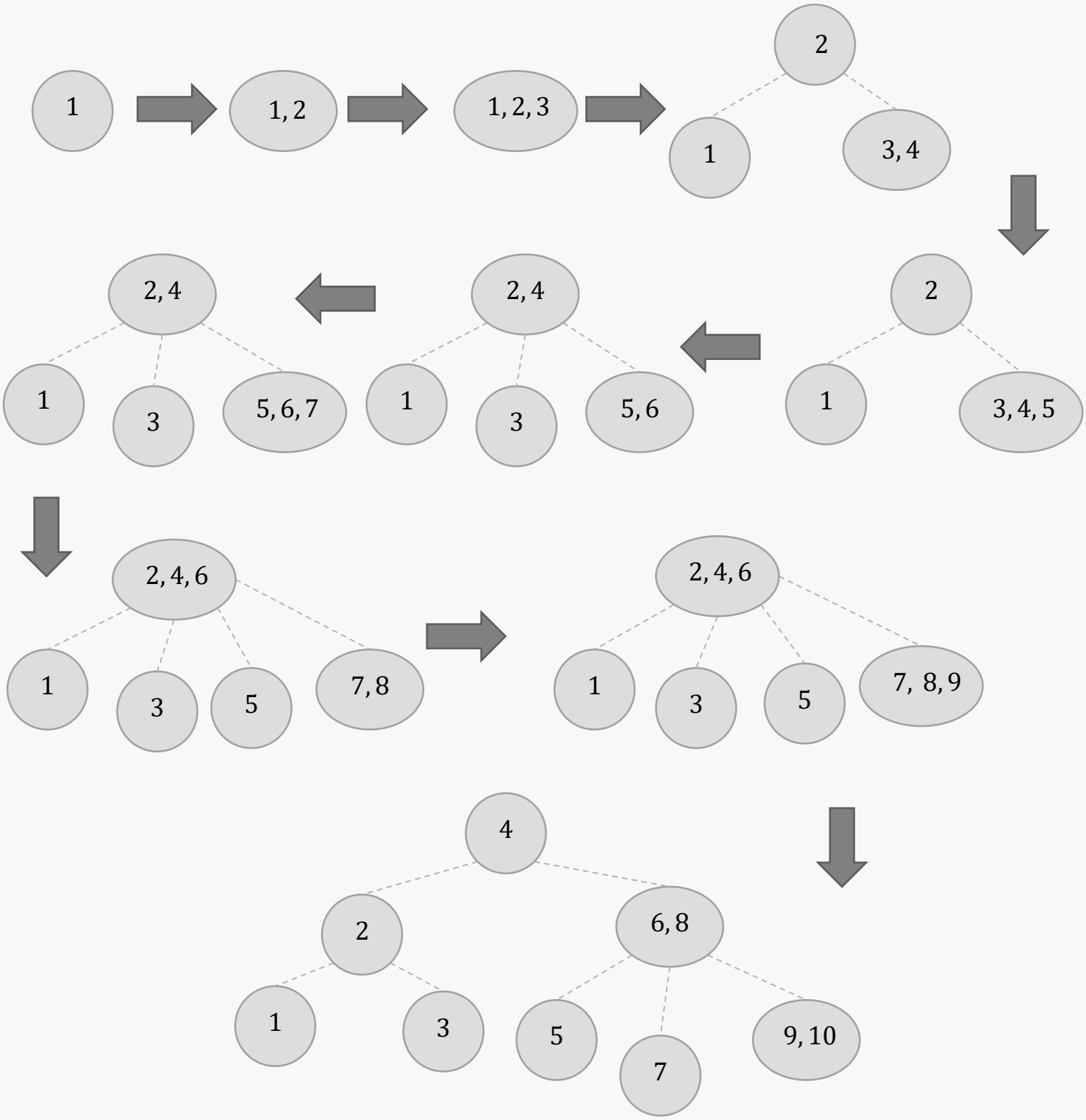
עזרה מהורה – רק כאשר האחים לא יכולים לעזור וגם להורה יותר מרשומה אחת.



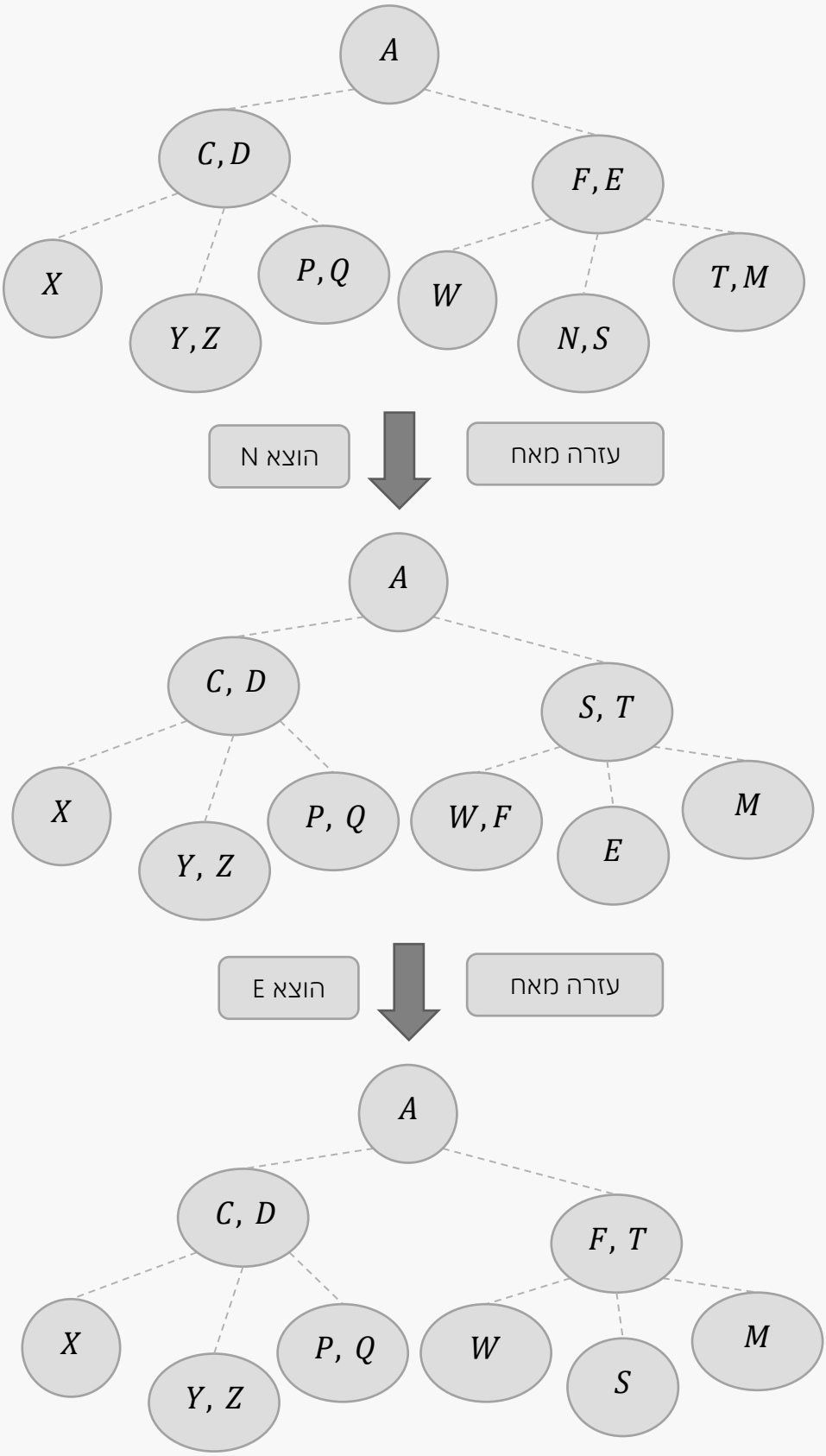
הגדלת הבעיה – אם לא הצלחנו להיעזר באח / הורה, ניקח את האח ונצרפו להורה (הגבהה).

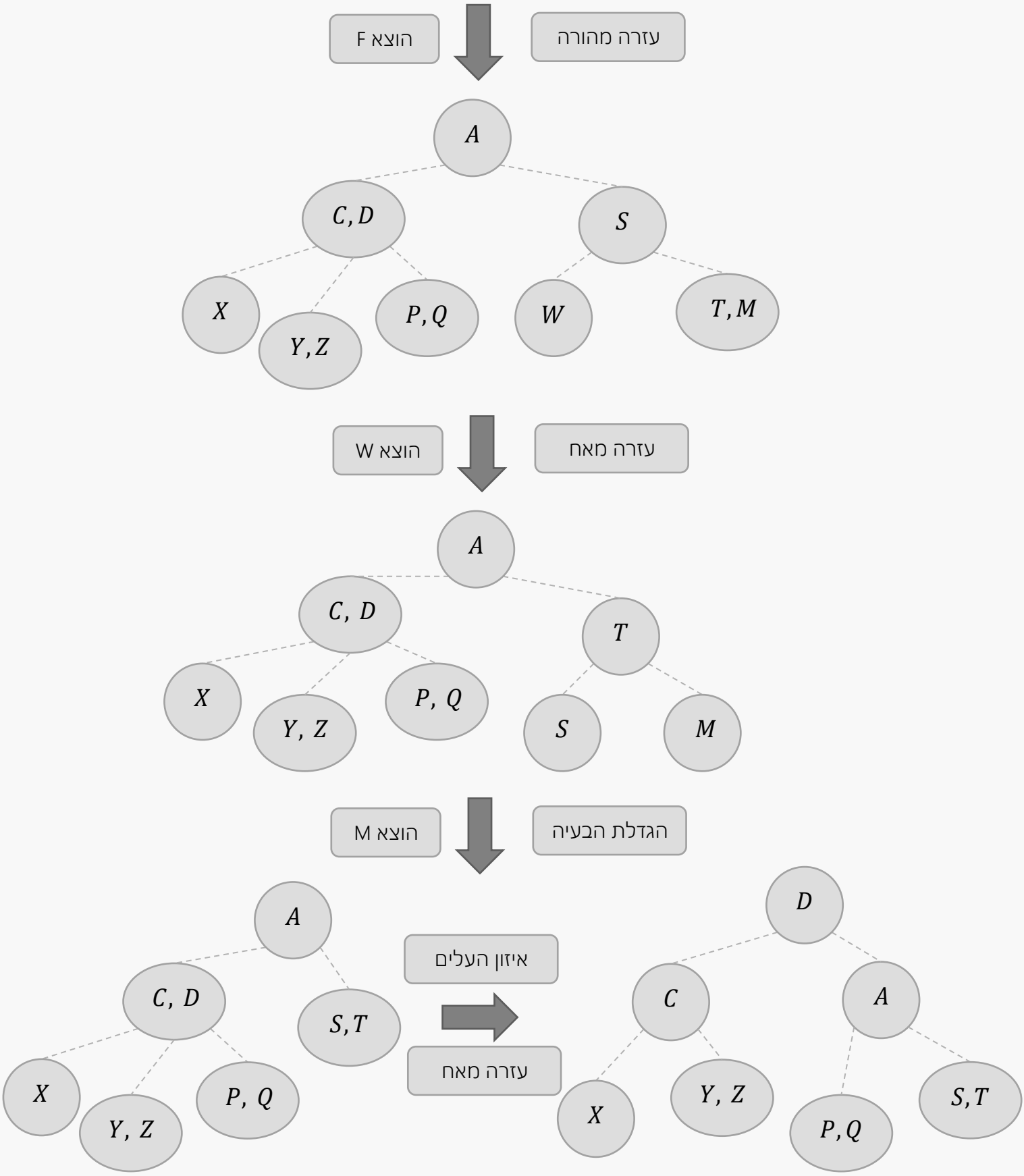


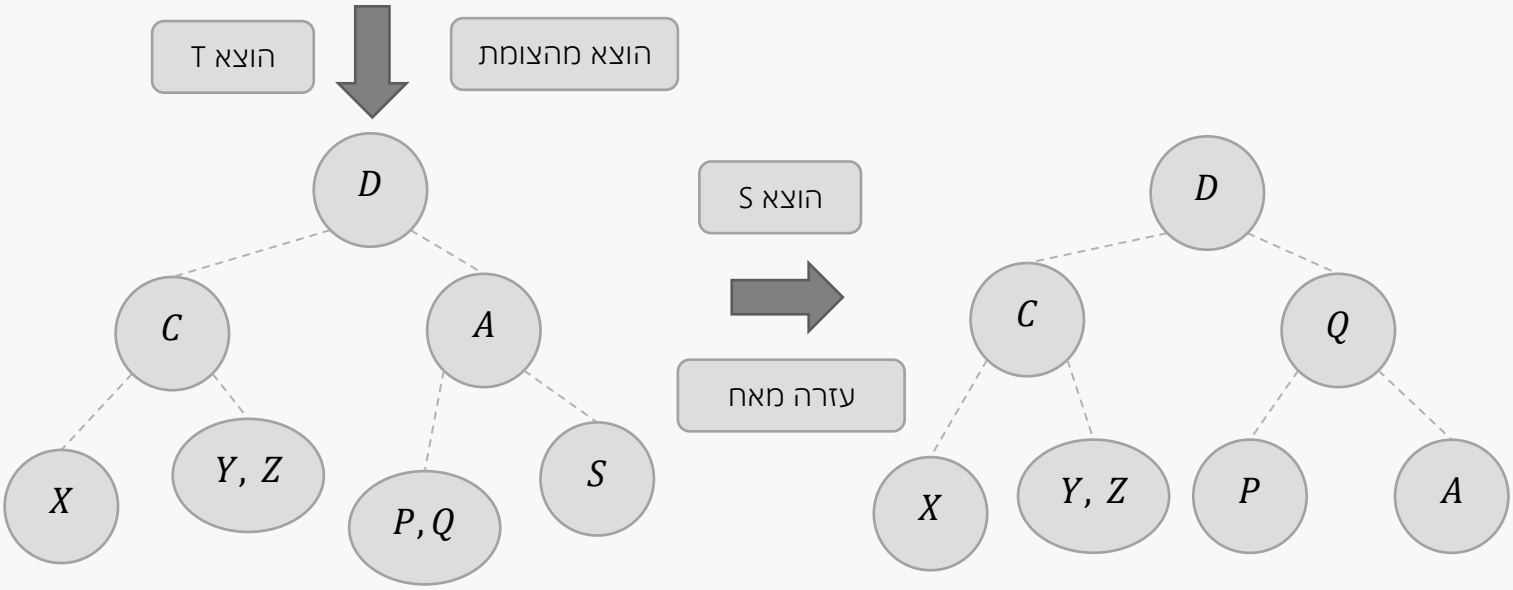
<u>דוגמא להוספה:</u> נכניס את המספרים **10 – 1** לפי הסדר לעץ 4-3-2.

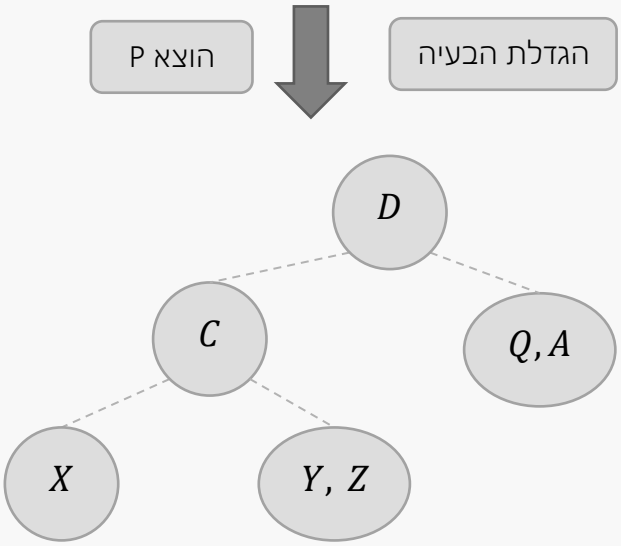


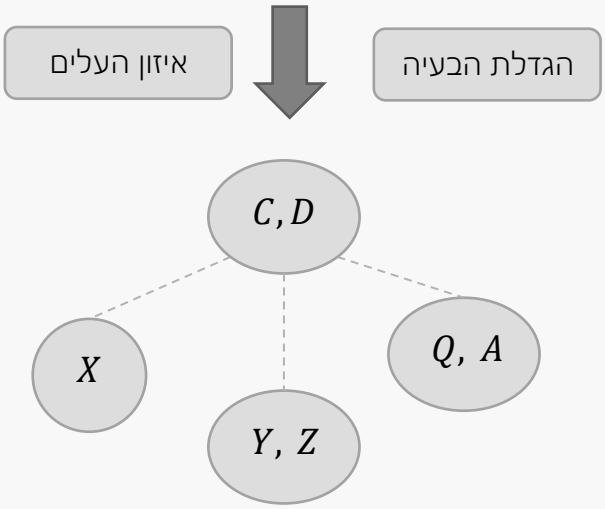
דוגמא להוצאה: נוציא מספר איברים מעץ 2-3-4 הבא.




עזרה מהורה
הוצא F
A
C,D
S
X
Y,Z
P,Q
W
T,M
עזרה מאח
הוצא W
A
C, D
T
X
Y, Z
P, Q
S
M
הגדלת הבעיה
הוצא M
A
C, D
S,T
X
Y, Z
P, Q
איזון העלים
עזרה מאח
D
C
A
X
Y, Z
P,Q
S,T


הוצא T
הוצא מהצומת
D
C
A
X
Y, Z
P, Q
S
הוצא S
עזרה מאח
D
C
Q
X
Y, Z
P
A


הוצא P
הגדלת הבעיה
D
C
Q, A
X
Y, Z

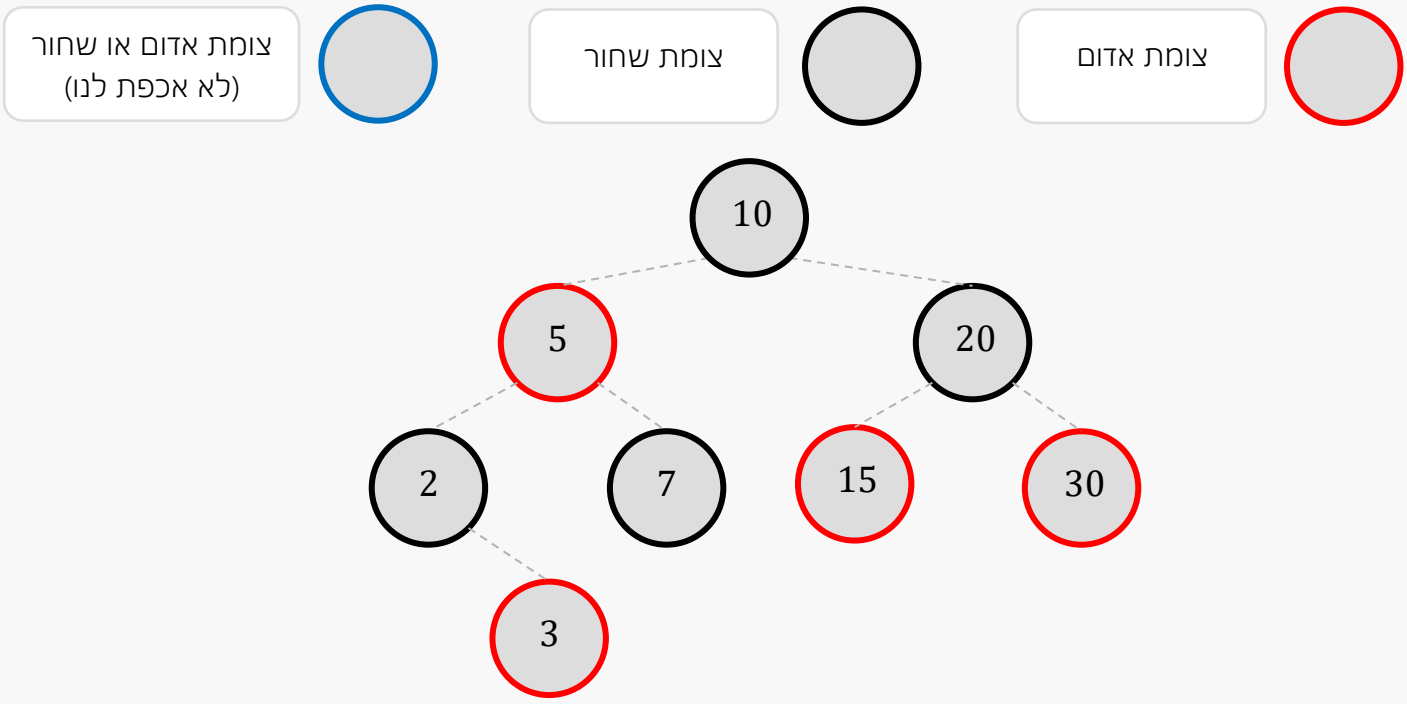

איזון העלים
הגדלת הבעיה
C, D
X
Y, Z
Q, A

עץ אדום שחור

"האדום שחור ואינטואיציה זה כמו שני צדי המתרס"

- עץ חיפוש בינארי מאוזן
- לכל צומת צבע – אדום / שחור
- השורש שחור, $NULL$ שחורים
- לצומת אדום ילדים שחורים
- לכל צומת פנימית המסלולים ממנו לעליו מכילים אותו מספר של צמתים שחורים.

<u>סימונים:</u>



<u>פעולות:</u>

1. מצא רשומה – $O(\log n)$.
2. הוסף רשומה – $O(\log n)$.
3. הוצא רשומה – $O(\log n)$.

מצא רשומה – זהה לפעולה על עץ חיפוש בינארי, כאן העץ מאוזן ולכן $O(\log n)$.

הוסף רשומה, הוצא רשומה – גם זהה, לאחר מכן נצטרך לאזן את העץ.

תכונת הצמתים השחורים בכל מסלול שקולה לתכונה שכל העלים באותה הרמה ב- 2-3-4.

תרגום עץ אדום שחור לעץ 2-3-4:

אם נתרגם עץ 2-3-4 לעץ אדום שחור, גובה העץ האדום השחור יהיה לכל היותר פעמיים גובהו של עץ ה2-3-4. ולכן גובהו של עץ אדום שחור הוא $O(\log n)$.

הוסף רשומה:

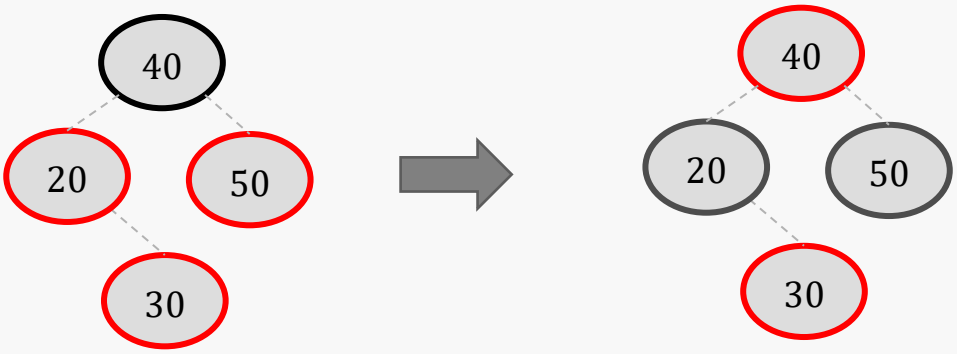
מצא את הרשומה.

(1) מצאנו – ללכת ללקוח
(2) לא מצאנו.
(a) הוסף רשומה איפה שנתקענו, צבע באדום.
(i) הורה שחור – גמרנו
(ii) הורה אדום – לטפל

טיפול בהורה אדום:

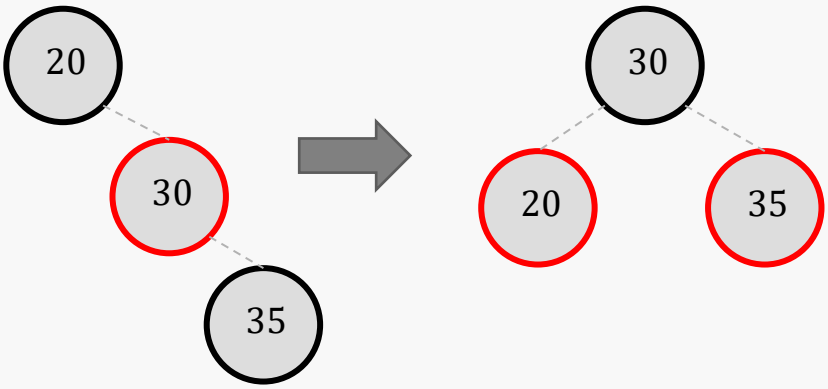
נפעיל סוג של פיצול על העץ.

אם גם הדוד אדום: בצע פיצול – צבע מחדש (אחר כך בדוק ש-40 בסדר)

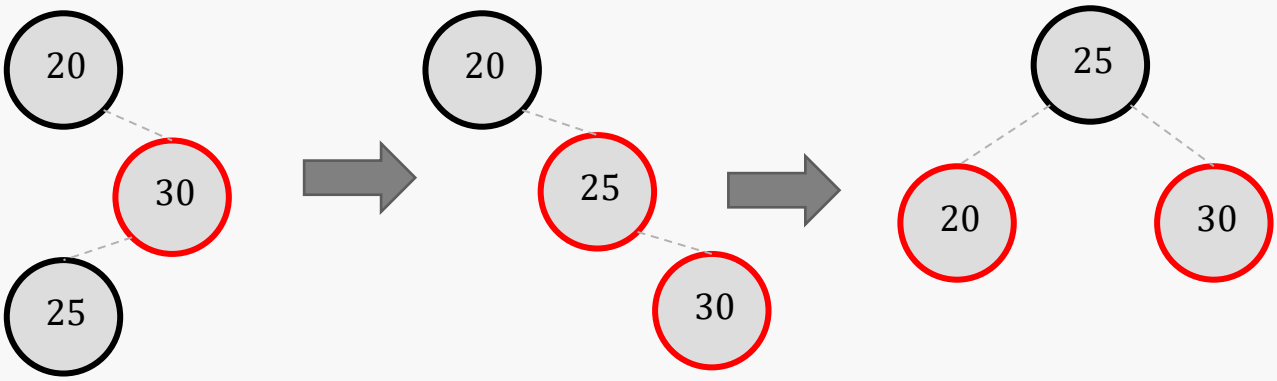


אם הדוד שחור:

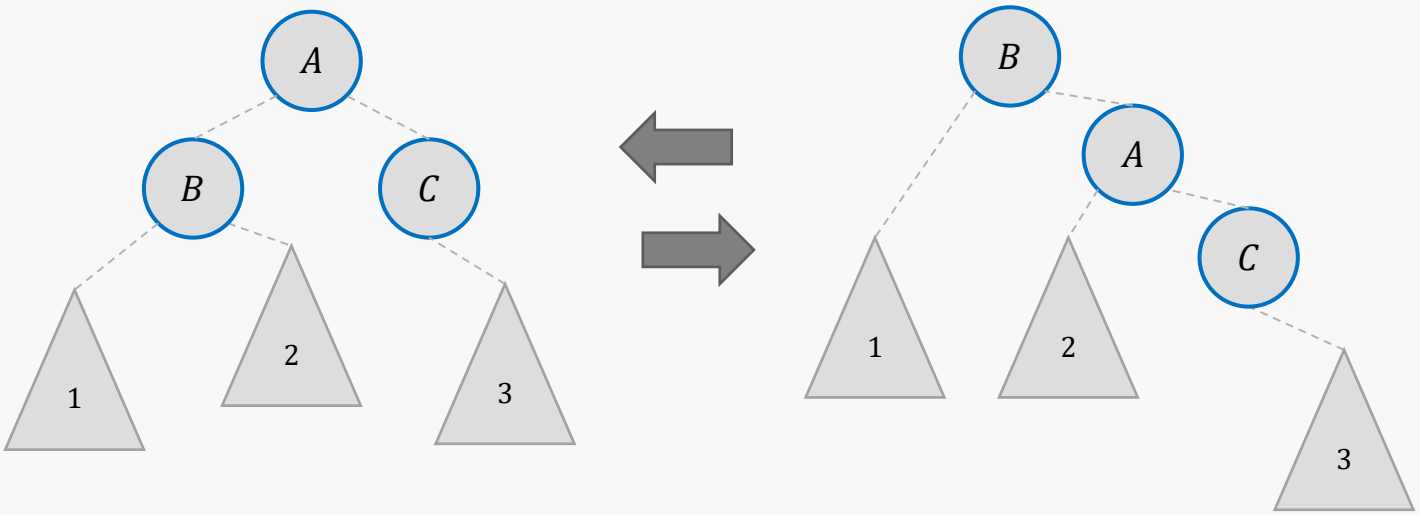
1. מקרה ראשון – באותו קו עם סבא:



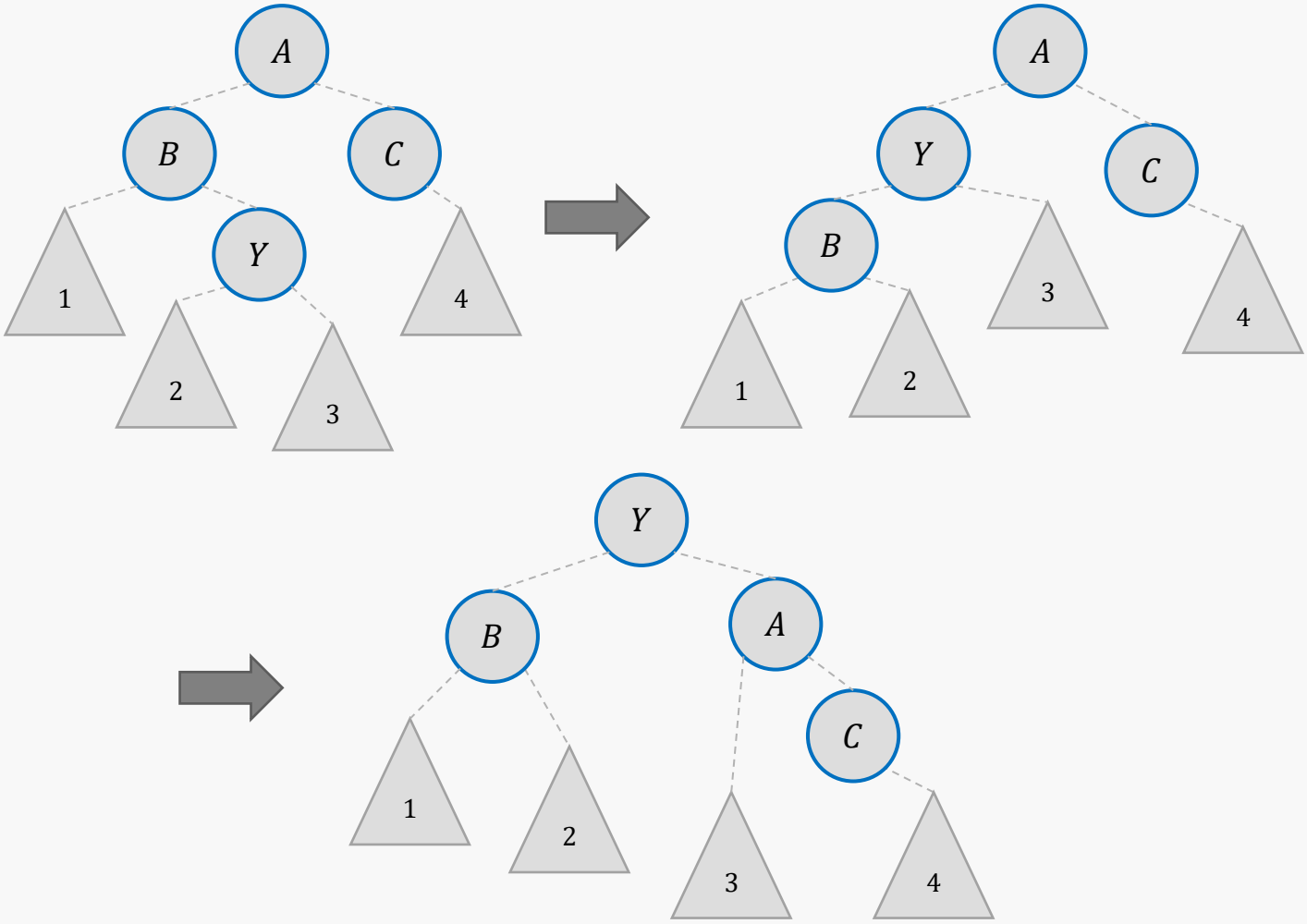
2. מקרה שני – לא באותו קו עם סבא:



הפעולה הבסיסית בעץ אדום שחור: רוטציה



תת עץ 2 לא השתנה ברוטציה – כדי לטפל בו נצטרך 2 רוטציות:



הוצא רשומה:

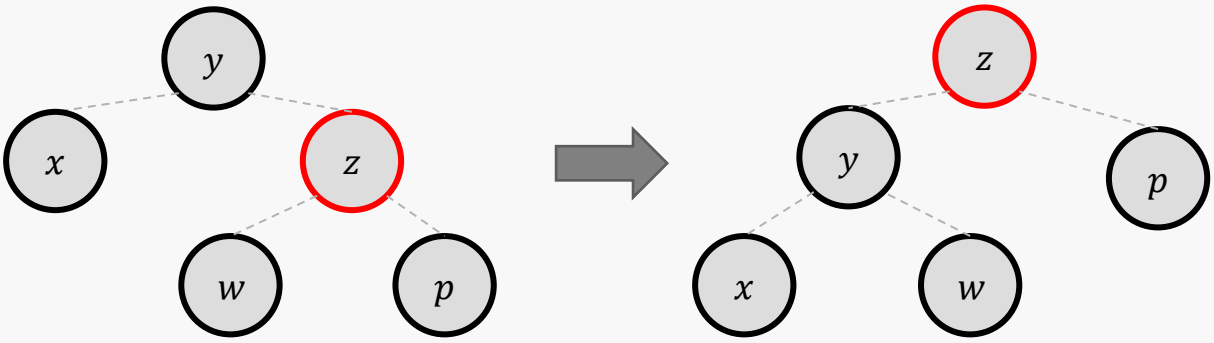
מצא רשומה.

1. לא מצאנו – יופי
2. מצאנו.
   a. עלה
      i. עלה אדום – קל
      ii. עלה שחור – כל הסיפור (מספר השחורים במסלולים ישתנה)
   b. צומת פנימית – החלף עם עוקב / קודם. אם החלפנו בעלה חזור ל-a, אם יש לו בן יחיד חזור ל-b. בטוח הוצאנו שחור ובני היחיד אדום, לכן נשחיר את הבן.

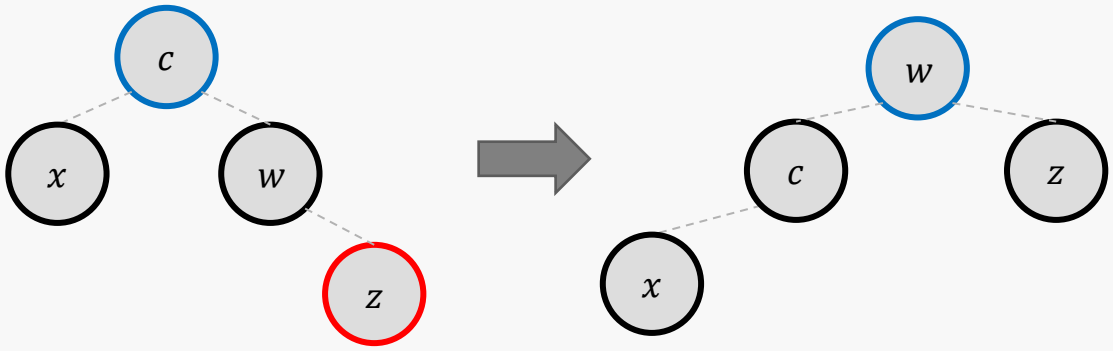
כל הסיפור – הוצאת עלה שחור

בתת העץ ששורשו x לאחר ההסרה חסר שחור למסלולים של תת עץ אחד, כלומר בעץ 2-3-4 שלו הוא יהיה גבוה ב-1.

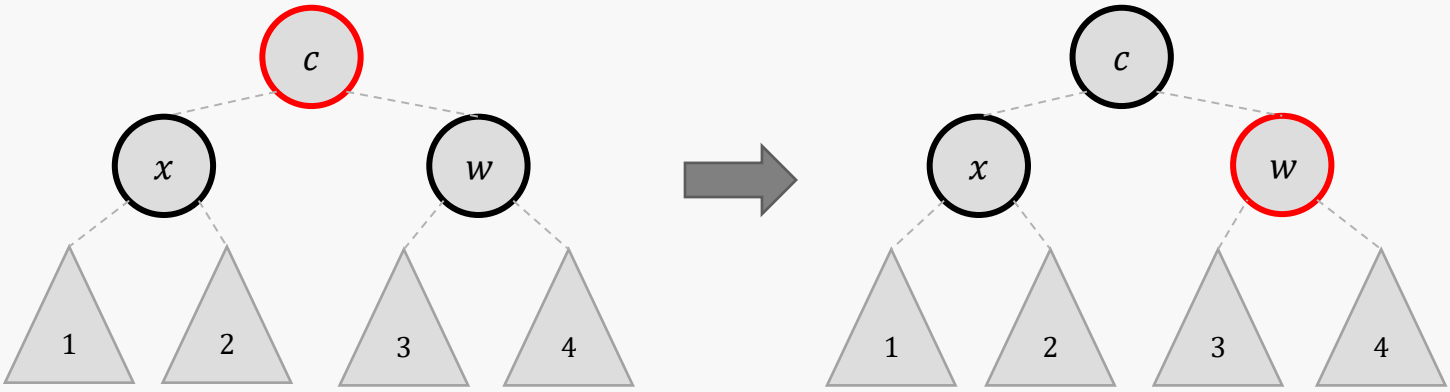
1. אם x אדום נצבע בשחור ופתרנו את הבעיה.
2. אם x שחור ולו אח אדום: נפעיל רוטציה.



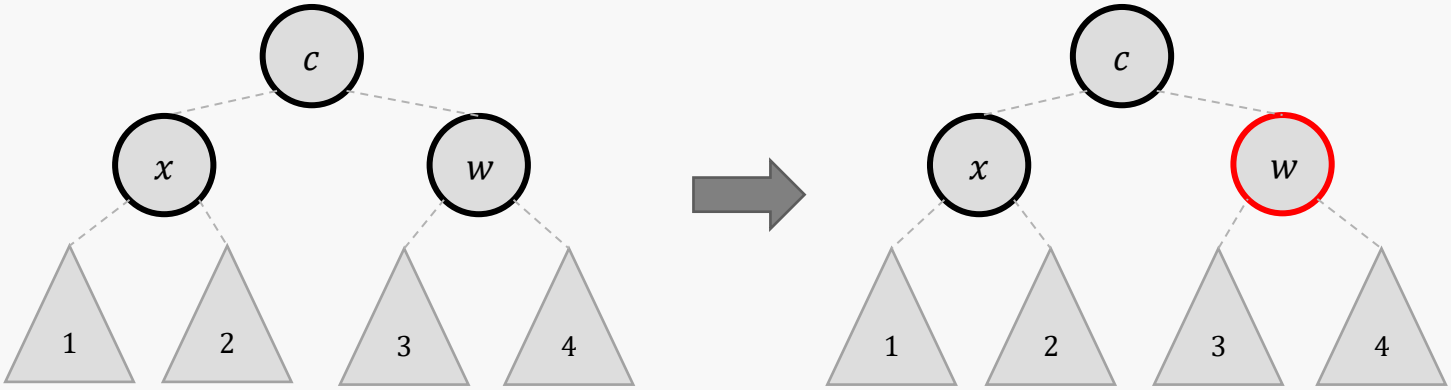
3. אחרת: ניעזר באח / ניעזר בהורה / נגדיל את הבעיה.
   a. עזרה מאח: אם לאח יש בן אדום. אם באותו קו עם סבא רוטציה אחת, אחרת שתי רוטציות וסיימנו.



b. עזרה מהורה: אם לא ניתן להיעזר באח וההורה אדום. צבע הורה לשחור וצבע אח לאדום.



c. הגדלת הבעיה: אם לא ניתן להיעזר באח / הורה. נצבע אח באדום ונגדיל את הבעיה – כעת גם לאח יש פחות שחורים.



נסמן ב- $bh(V) = b(V)$ (bh – black height) את מספר הצמתים השחורים בכל מסלוליו.

- לכל v מספר הצמתים הפנימיים בתת העץ שלו הוא לפחות $2^{b(v)} - 1$.

הוכחה באינדוקציה על גובה העץ:

<u>בסיס:</u> גובה 0, v $null$. $2^{b(v)} - 1 = 0$ ולכן הטענה נכונה.

<u>צעד:</u> נניח שהטענה נכונה עבור עץ בגובה h, נוכיח עבור $h + 1$.

יהי v צומת בגובה $h + 1$. יהי u בן של v. אם u שחור אז $b(u) = b(v) - 1$.

לכן גודל תת העץ הימני הוא: $2^{b(v)-1} - 1$. אחרת $b(u) = b(v)$ ויש יותר צמתים. הדבר סימטרי עבור הבן השני, ולכן בשני תתי העצים יחד יש לפחות $2 \cdot \left(2^{b(v)-1} - 1\right) - 1$ צמתים. נוסיף את v עצמו, ונקבל שמספר הצמתים בתת העץ הוא לפחות $2^{b(v)} - 1$, והוכחנו את צעד האינדוקציה.

יהי r השורש של עץ אדום שחור שגובהו h. אזי: $b(r) \geq \frac{h}{2}$.

<u>הוכחה:</u> אם מצאנו צומת אדום, בטוח ראינו עוד אחד שחור. ולכן בכל מסלול מספר השחורים גדול ממספר האדומים ו- $b(r) \geq \frac{h}{2}$.

<u>מסקנה:</u> יהי h גובהו של עץ אדום שחור ובו n צמתים. אזי:

$$h \leq 2 \cdot \log(n+1) = O(\log n)$$

כלומר עץ אדום שחור הוא עץ מאוזן.

פעולות שונות על עץ החיפוש, כמו מציאת מינימום / מקסימום / עוקב / קודם מתבצעות באופן זהה לשאר עצי החיפוש, בפרט עץ חיפוש בינארי.

שתי פעולות נוספות: split, join

- Join(T1, x, T2)

T1, T2 הם עצים אדומים שחורים ו-x הוא מפתח כך ש- $T_1 < x < T_2$. נרצה לבנות עץ אדום שחור שהוא האיחוד של שני העצים וצומת שמפתחה x.

נסמן $B(v) = \begin{cases} b(v)+1 & v.color = black \\ b(v) & v.color = red \end{cases}$ ($b(v)$ כולל v עצמו).

- מצא את $B(T_1), B(T_2)$ (ב- $O(\log n)$)
  - אם $B(T_1) = B(T_2)$, הצומת של x יהיה הצומת, T_1 הבן השמאלי ו-T_2 הבן הימני.
  - אם $B(T_1) > B(T_2)$

יורדים על השדרה הימנית (הולכים רק ימינה) של T_1 עד לצומת v המקיימת $B(v) = B(T_2)$.

חבר את x ל- $p(v)$ בתור בן ימני (אדום), את T_2 כבן ימני של x, ו-v ואת תת העץ שלו לבן השמאלי של x.

  - אם $B(T_1) < B(T_2)$

יורדים על השדרה הימנית של T_2 עד לצומת v המקיימת $B(v) = B(T_1)$.

חבר את x ל- $p(v)$ בתור בן שמאלי (אדום), את T_1 כבן שמאלי של x, ו-v ואת תת העץ שלו לבן הימני של x.

- Split(T, x)

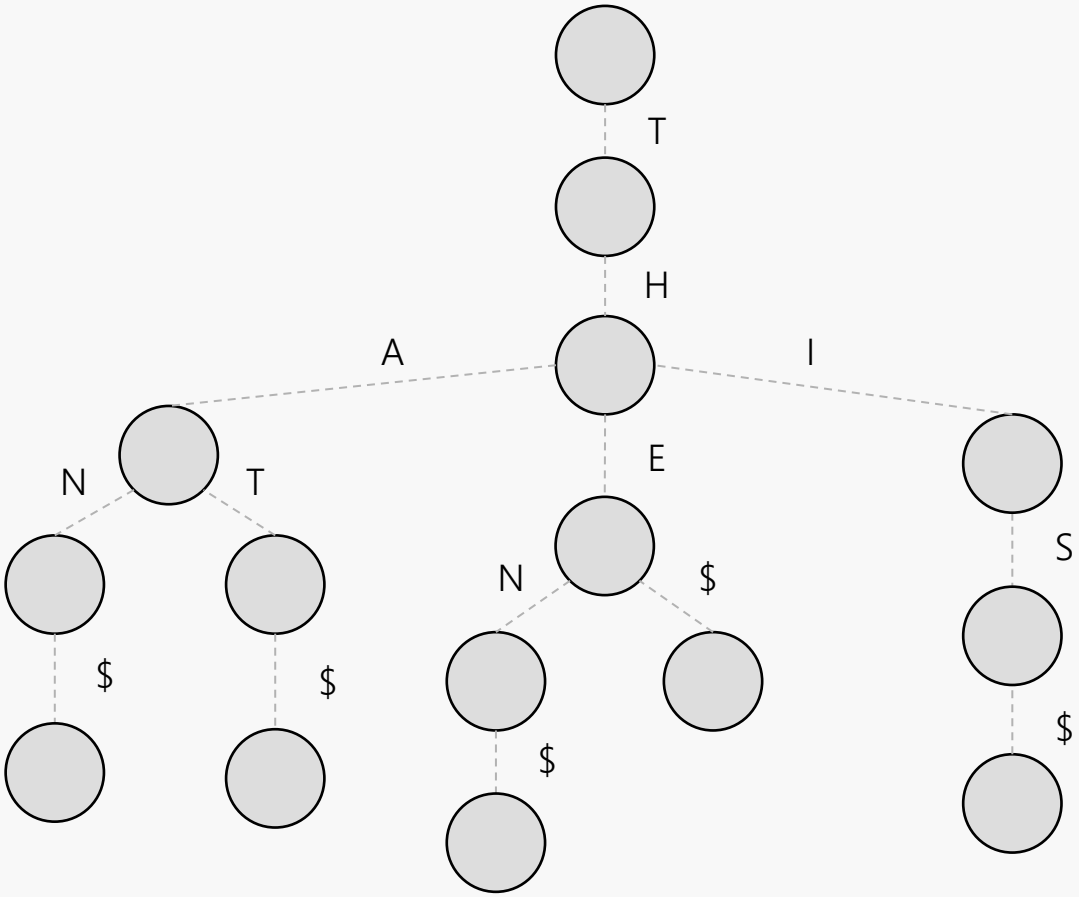
נתון עץ אדום שחור T ומפתח x, נרצה להחזיר T1, T2 כך ש- $T_1 < x < T_2$.

נסתכל על המסלול שעברנו בחיפוש x בעץ. בכל פנייה נוסיף ל-T_1 את תתי העצים השמאליים ול-T_2 את תתי העצים הימניים (עם join). סיבוכיות הזמן היא $O(\log^2 n)$, בפועל סיבוכיות הזמן של מציאת הצומת המתאים ב-join היא $O\big(B(T_1) - B(T_2)\big)$, כאשר נסכום נקבל טור טלסקופי ובסך הכל $O(\log n)$.

עץ מילים – TRIE

"מה שאנחנו נלמד יהיה די מפגר"

נניח שיש לנו את המילים {THAN, THEN, THE, THIS, THAT}, נבנה עץ חיפוש:



פעולות על המבנה:

<u>מצא רשומה:</u>

קל: מתגלגלים ומתגלשים עד שנתקעים בדולר.

<u>הוסף רשומה:</u>

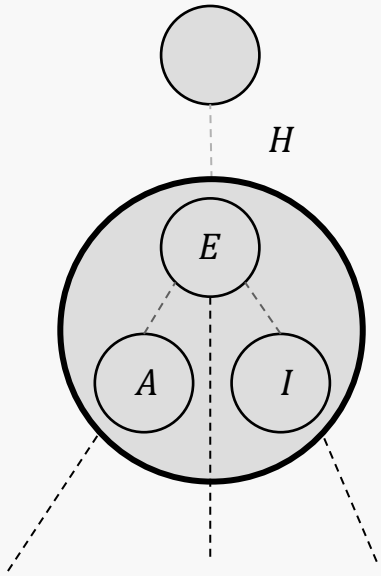
מתגלגלים ומתגלשים עד שנתקעים ואז מוסיף אותיות ודולר בסוף.

<u>בטל רשומה:</u>

מתגלגלים ומתגלשים עד שנתקעים ואז מלמטה למעלה נתחיל לבטל צמתים עד שנגיע לצומת עם בן אחר, ושם נפסיק לבטל.

איך בונים את זה?

- נאיבי: בכל צומת להחזיק מערך של האלפבית. לא טוב לאלפבית עם הרבה אותיות, מעולה עבור DNA או בינארי (אחרת יהיה מאוד דליל).
- יותר טוב (עבור אלפבית גדול): בכל צומת לתחזק עץ חיפוש של הילדים.



עלות הבנייה: $O(n^2)$ מקום וזמן. ניתן לבנות בזמן ומקום $O(n)$ – לא נלמד.

שימוש: נניח שיש לנו את המחרוזת $abab\$$ ואנחנו רוצים לראות האם יש תת מחרוזת שלה, נבנה עץ מילים של הסיפות שלה (תת מחרוזת שהיא המחרוזת החל ממקום מסוים) ונחפש שם. למשל כאן נבנה עץ חיפוש מהמילים:

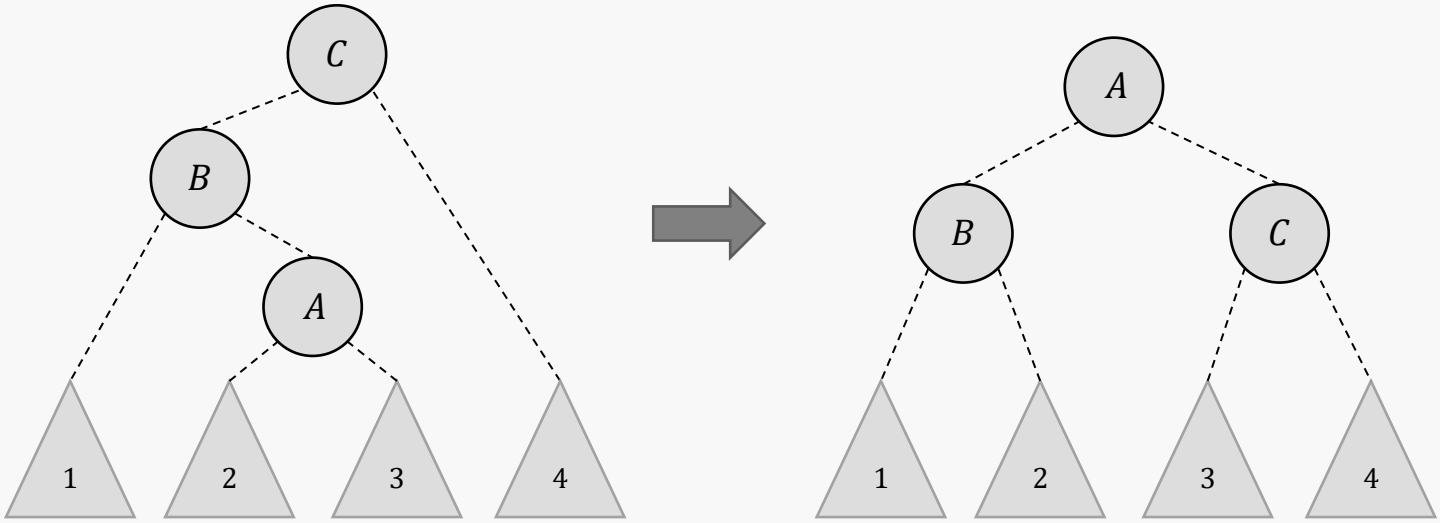
$$abab\$, bab\$, ab\$, b\$, \$$$

עץ Splay

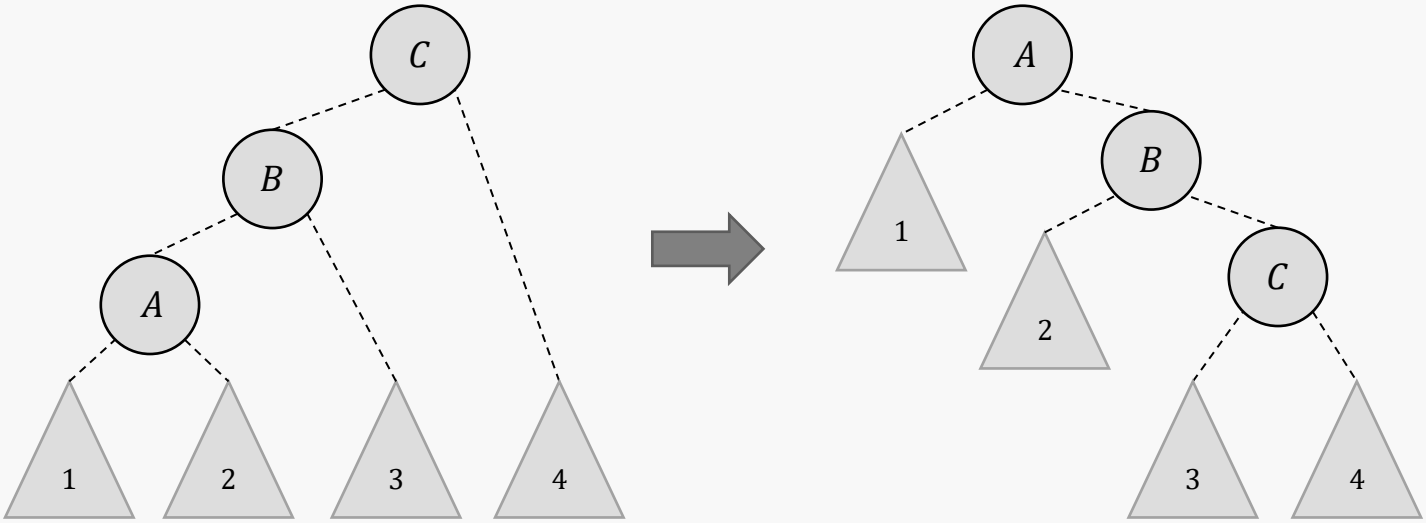
"קוראים לו Splay Tree, והוא אחלה"

עץ חיפוש בינארי לא (בהכרח) מאוזן. יש שתי פעולות בסיסיות המורכבות מ-2 רוטציות.

1. רוטציית zig zag



2. רוטציית zig zig



כאשר נחפש בעץ צומת נהפוך אותה להיות שורש העץ על ידי הרוטציות zig zig ו-zig zag (ואולי עוד אחת- zig או zag בסוף).

נסמן ב- $s(x)$ את גודל תת העץ ששורשו x, ובנוסף $r(x) = \log s(x)$.

פונקציית הפוטנציאל של המבנה היא:

$$\Phi_i = \sum_{v \in V_i} r(v)$$

ננתח לשיעורין את עלות רוטציית zig zig:

נבצע 2 פעולות: $c_i = 2$. נסמן ב-$r'(x)$ את $r(x)$ לאחר הרוטציה. לאחר הרוטציה A החליף את C: $r'(A) = r(C)$. בנוסף, $r(B) \geq r(A), r'(A) \geq r'(B)$. כעת נחשב את $\Delta\Phi$:

$$\begin{aligned}\Delta\Phi &= \big(r'(A) + r'(B) + r'(C)\big) - \big(r(A) + r(B) + r(C)\big)\\ &= r'(B) + r'(C) - r(A) - r(B) \leq r'(A) + r'(C) - 2 \cdot r(A)\\ &= \log\big(s'(A)\big) + \log\big(s'(C)\big) - 2 \cdot \log\big(s(A)\big)\end{aligned}$$

פונקציית ה-log היא קעורה, ולכן:

$$\frac{\log\big(s(A)\big) + \log\big(s'(C)\big)}{2} \leq \log\left(\frac{s(A) + s'(C)}{2}\right) \leq \log\left(\frac{s(A)}{2}\right) = \log\left(s'(A) \cdot \frac{1}{2}\right) = r'(A) - 1$$

כלומר:

$$\begin{aligned}\hat{c}_i = c_i + \Delta\Phi &= 2 + \mathrm{r}'(\mathrm{A}) + r'(C) - 2 \cdot r(A)\\ &= 2 + \mathrm{r}'(\mathrm{A}) + r'(C) - 2 \cdot r(A) + r(A) - r(A)\\ &\leq 3 \cdot r'(A) - 2 \cdot r(A) - r(A) = 3\big(r'(A) - r(A)\big)\end{aligned}$$

נסכום את עלות כל ההעלאה של צומת לשורש (כל הרוטציות):

$$\sum_{z-z} 3\big(r'(A) - r(A)\big) = 3 \cdot (\log s(root) - \log s(x)) \leq 3 \cdot \log n = O(\log n)$$

פעולות נוספות על המבנה:

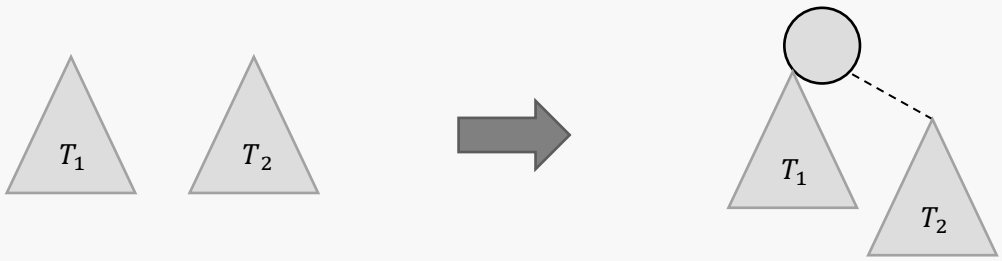
- LookUp(x) (חיפוש עם העלאה)

חפש את הרשומה.

  - אם נמצא: הבא לשורש ע"י רוטציות.
  - אחרת הבא לשורש את הצומת האחרונה שבה נתקלנו (איפה שנתקענו).
- Join(T1, T2) (סוג של איחוד)

נרצה לבצע פעולת join לעצים T_1, T_2 כך שכל המפתחות ב-T_1 קטנים מכל המפתחות ב-T_2.

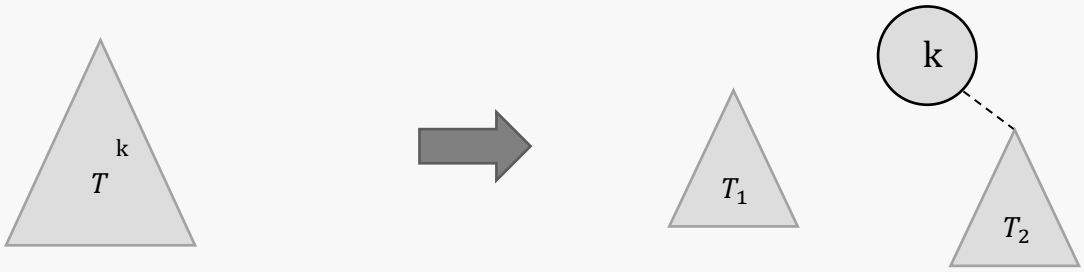
  - המקסימלי מ-T_1 יעלה לשורש המשותף.
  - T_2 ישורשר מימין לשורש, שאר T_1 משמאל.



- Split(T, k)

פצל עץ T לפי מפתח k.

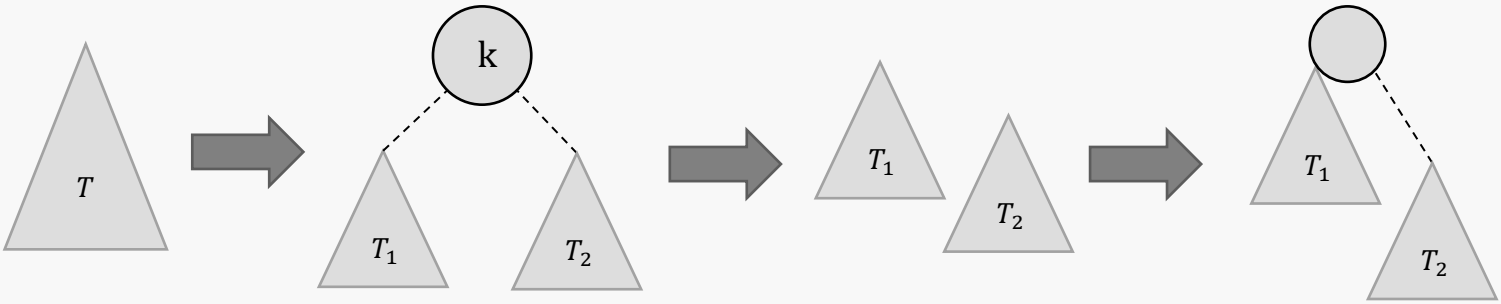
- הבא את הקודם של k לשורש.
- נתק בין השורש לבנו השמאלי.



- Delete(T, k)

בטל את הרשומה שמפתחה k מעץ T.

- הבא את k לשורש.
- מחק את השורש, נותרנו עם שני תתי העצים של בניו T1, T2.
- צרף את שני תתי העצים, join(T1, T2).



עץ דרגות

הרחבה של עצי חיפוש: עץ דרגות. נרצה לענות על השאילתות הבאות (בנוסף לשאילתות הרגילות של עץ חיפוש)

1. Select(i) – תחזיר את האיבר ה-i הקטן ביותר במבנה.
2. Rank(x) – תחזיר את הדירוג של x מבין איברי המבנה.

בעץ לכל צומת יש שדה s(v) שהוא גודל תת העץ המושרש ב-v.

- Select(i)

נסמן $k = s(root.l) + 1$.

- אם $i = k$ החזר את r.
- אם $i < k$ חפש בתת העץ השמאלי את i.
- אם $i > k$ חפש בתת העץ הימני את $i - k$.

סיבוכיות הזמן: $O(\log n)$.

- Rank(x)

$$
\begin{aligned}
&r \leftarrow s(x.l) + 1\\
&y \leftarrow x\\
&while\ y \neq root[T]\\
&\qquad if\ y = p(y).r\\
&\qquad\quad k = k + s(p(y).l) + 1\\
&\qquad y = p(y)\\
&return\ k
\end{aligned}
$$

סיבוכיות הזמן: $O(\log n)$.

<u>תחזוקת שדה ה-$size$ בהכנסה:</u>

- יורדים מהשורש לכיוון עלה
- עלה מהצומת החדשה לשורש
  - בכל שורש שעברנו דרכו בירידה – נוסיף 1 לשדה ה-$size$.
  - תיקון $s(v)$ בעת רוטציות אם צריך.
- f שדה מרחיב לעץ חיפוש בינארי עם n צמתים. אם שינוי במידע של v משפיע רק על f של אבותיו של v ניתן לתחזק את f ב-$O(\log n)$.

טבלת גיבוב – Hash Table

"זו שיטה על הכיף כיפאק"

תעודת זהות – 9 ספרות: $x_1x_2x_3\ y_1y_2y_3\ z_1z_2z_3$.

נסתכל על המספר z_1y_2. מבנה נתונים מאוד פשוט: נבנה מערך בגודל 100 (כי המספר הוא דו ספרתי 0-99) ובו רשימות מקושרות של מפתחות זהים.

מספר ההשוואות יהיה קטן מאוד! איך נעשה את זה?

יש לנו פונקציה שמכווצת את התחום המפתחות. במקרה זה תחום של 10^9 קטן ל-10^2. בנוסף נרצה שהפונקציה תפרוש את האיברים באופן מאוזן.

2 שיטות גיבוב: <u>פתוחה וסגורה</u>. גדי די בטוח שזאת הפתוחה.

<u>שיטה נוספת</u>: לבנות מערך שהרשומות נכנסות לתוכו. המערך יהיה בגודל $O(n)$ אבל יותר גדול מ-n. אם פתאום באים שני חברים שרוצים להיכנס לאותו המקום ננסה להכניס אותו למקום הפנוי הבא.

בעיה בהוצאה: למשל היו יותר מדי 47 ושניים נכנסו למקומות 48, 49. אם נוציא את 47 ולאחר מכן נחפש עבור 47 שוב ולא נמצא, למרות שיש! נפתור בעזרת סימון: יד ראשונה או יד שנייה. יד ראשונה: אף אחד עוד לא היה שם, יד שנייה: אנחנו לא יודעים. לאחר שנוציא את 47 נסמן אותו יד שנייה. כשנחפש אותו נשים לב שהוא יד שנייה וניקח מחברים שלו יד ראשונה.

שיטה נוספת לפתרון: <u>סדרת פונקציות</u>, נפעיל את הפונקציות לפי הסדר על המפתח עד שנמצא מקום ריק יד ראשונה. ההסתברות שייווצרו פקקים קטנה ככל שהפונקציות שונות אחת מהשנייה.

מדי פעם צריך לרענן את המערך – להחזיר ליד ראשונה. "פריש מיש".

תכנות דינמי

"שלב ב' לא בוכים"

הטוב מ-7: המטרה היא לנצח 4 משחקים.

ילד רוצה לחשב את ההסתברות שהקבוצה שלו תנצח. $P(I,J)$

I – מספר הניצחונות של הטובים. J – מספר הניצחונות של הרעים.

$$0 \le I,J \le 4; I+J < 8$$

בתום המשחק $I=4$ או $J=4$. נניח שהמשחק הוגן:

$$P(I,J) = \frac{P(I+1,J)+P(I,J+1)}{2}$$

פונקציה רקורסיבית! הנוסח המלא:

$$P(I,J) = \begin{cases} 1 & I=4 \\ 0 & J=4 \\ \frac{P(I+1,J)+P(I,J+1)}{2} & else \end{cases}$$

מה סיבוכיות הזמן של הפונקציה אם התחום היה $1,\dots,n$?

$$P(2,1) = \frac{P(3,1)+P(2,2)}{2} = \frac{\frac{P(4,1)+P(3,2)}{2}+\frac{P(3,2)+P(2,3)}{2}}{2} = \cdots$$

מספר קריאות אקספוננציאלי, הרבה חזרות על חישובים. נשמור כל חישוב שעשינו, ולפני כל חישוב חדש נבדוק האם כבר ביצענו את אותו החישוב. במידה ולא עשינו – נעשה. אחרת – סיימנו ללא חישוב נוסף.

רעים / טובים	0	1	2	3	4
0	0.5	0.34375	0.1875	0.0625	0
1	0.65625	0.5	0.3125	0.125	0
2	0.8125	0.6875	0.5	0.25	0
3	0.9375	0.875	0.75	0.5	0
4	1	1	1	1	X

סיבוכיות זמן עם זיכרון: n^2 במקום 2^n.

<u>שלבים פתרון בעיה עם תכנות דינמי:</u>

א. מקבלים בעיה.
ב. (לא בוכים) נותנים לבעיה פתרון רקורסיבי אקספוננציאלי (לא בוכים).
ג. מחליפים את הפתרון הרקורסיבי האקספוננציאלי בשימוש בטבלה.
ד. הטבלה מדברת בעד עצמה.

Longest Common Subsequence – LCS

- תת מחרוזת של A היא תת סדרה של x_n, למשל $x_2\ x_3\ x_5$.

קלט: שתי מחרוזות A, B.

פלט: אורך תת המחרוזת המשותפת הארוכה ביותר.

<u>דוגמא:</u>

$$ABBCDBA, ABDC$$
$$LCS = 3\ (ABD \times 2, ABC \times 2)$$

פתרון רקורסיבי אקספוננציאלי:

$$LCS(A,B) = LCS(a_1, \dots, a_n\ ; b_1, \dots, b_m) = LCS(n,m)$$
$$LCS(i,j) = \begin{cases} LCS(i-1, j-1) + 1 & a_i = b_j \\ \max\{LCS(i-1,j), LCS(i,j-1)\} & a_i \neq b_j \\ 0 & i = 0 \parallel b = 0 \end{cases}$$

נמלא את הטבלה עבור:

$$A = ABBCDBA$$
$$B = ABDC$$

j \ i	ε	A	B	B	C	D	B	A
ε	0	0	0	0	0	0	0	0
A	0	1	1	1	1	1	1	1
B	0	1	2	2	2	2	2	2
D	0	1	2	2	2	3	3	3
C	0	1	2	2	3	3	3	3

איך נקבל מחרוזת? נעקוב אחרי המסלול של קבלת הערכים (מאיפה נוסף ערך לכל משבצת) ונקבל מסלול בו הצעדים האלכסוניים מהווים את האינדקסים של המחרוזת.

מילוי הטבלה עולה $O(n \cdot m)$, וגם עולה $O(m \cdot n)$ מקום (אפשר גם $n + m$ מקום – בכל פעם אנחנו צריכים רק את השורה שלי ואת השורה שמעליי).

הטבלה מדברת בעד עצמה!

Lowest Common Ancestor – LCA

קלט: עץ (כללי) T.

נכין את העץ לשאילתות.

<u>קלט לשאילתה:</u> שני מצביעים לצמתים.

<u>פלט לשאילתה:</u> האב הקדמון הנמוך ביותר של הצמתים. (הצומת הכי עמוקה ששני הצמתים צאצאים שלו)

נבצע הכנה ב- $O(n)$, מענה על השאילתא ב- $O(1)$.

<u>הכנה:</u>

1. לתת לכל צומת את העומק שלו.
2. בטיול כל העץ (preorder) נהג המונית יצעק בכל מפגש את הצומת והעומק שלו.
3. כל צומת בטבלה שולח מצביע לצומת שלו בעץ.
4. כל צומת בעץ שולח מצביע ל<u>אחד</u> מהמופעים שלו בטבלה.

<u>מענה על השאילתא:</u>

נלך למערך שלנו ובין האצבעות של הצמתים בעץ נמצא את האחד בעל העומק המינימלי.
איך נמצא את המינימלי ב- $O(1)$? RMQ!

Range Minima Query – RMQ

<u>קלט:</u> מערך מספרים.

<u>קלט לשאילתה:</u> מערך המספרים, שני אינדקסים.

<u>פלט לשאילתה:</u> אינדקס המינימום ביניהם.

נבצע הכנה ב- $O(n)$, מענה על השאילתא ב- $O(1)$.

<u>שתי שיטות:</u> נשלב אותן כדי לפתור את הבעיה.

א. טבלה. <u>קלט:</u> מערך באורך x. נכין טבלה בגודל x^2 ובה התשובה לכל צמד.

<u>דוגמא:</u> $1, 2, 1, 2, 3$.

מתחת לאלכסון הראשי אין צורך למלא – הטבלה סימטרית.

ימין / שמאל	1	2	3	4	5
1	1	1	3	3	3
2		2	3	3	3
3			3	3	3
4				4	4
5					5

מענה על שאילתא ב- $O(1)$, אבל הכנה ב- $O(x^2)$.

ב. מערך בגודל n, נחלק אותו לחתיכות בגודל $\frac{\log n}{2}$. בהינתן שאילתא יש שתי אפשרויות: שני האינדקסים באותו בלוק, או שלא.

<u>שתי אפשרויות, ארבע שאילתות</u>

4 – 2 האינדקסים באותו בלוק.

3 – אחרת, נצטרך לענות על 3 שאילתות (1, 2, 3). מינימום לכל בלוק (1, 2), ולכל הבלוקים ביניהם (שאילתא 3).

שאילתות 1, 2, 4 משתמשות בטריק הטבלה (הבלוק בגודל $\frac{\log n}{2}$ ולכן אין בעיה).

<u>טיפול בשאילתא 3:</u> נרצה למצוא את המינימום מקבוצת החתיכות הרציפה.

א. בנה מערך עזר B ובה מכל חתיכה המינימום שלה (בגודל $x = \frac{n}{\log n}$). מותר לנו להשתמש בהכנות לא ליניאריות כאשר הגודל הוא x.

ב. "בדידים". נפזר על המערך בגודל x "מקלונים" בגודל 2^i. נמצא את המינימום לכל מקל. כמה מקלות אנחנו צריכים? שני מקלות. ניקח את החזקה הקטנה ביותר שמכיל המספר וניקח 2 כאלו.

נמלא טבלה של המקלות:

$x = \frac{n}{\log n}$	2	5	1	7	1	2	3	12	6
2^0	2	5	1	7	1	2	3	12	6
2^1	2	1	1	1	1	2	3	6	
2^2	1	1	1	1	1	2			
2^3	1	1							

מספר האיברים בטבלה הוא $O(x \cdot \log x)$, עלות החישוב היא גם $O(x \cdot \log x)$.

זמן החישוב:

$$\frac{n}{\log n} \cdot \log \frac{n}{\log n} = O(n)$$

טבלת האינדקסים של סדרות רבות יכולות לחזור על עצמן, בזבוז. במקום זאת, בכל סדרה נגדיר את ההתחלה בתור 0, מכיוון שבכל פעם ההפרש הוא לכל היותר 1 נגדיר את הסדרה באופן בינארי: 1 עלייה, 0 ירידה.

$$17, 18, 17, 18, 19 \Longrightarrow 0, 1, 0, 1, 1$$

יש $2^{\frac{\log n}{2}} = \sqrt{n}$ ייצוגים בינאריים שונים ל- $\frac{\log n}{2}$ ביטים, ולכן $\sqrt{n}$ טבלאות שונות. נכין $\sqrt{n}$ טבלאות, כל אחת בגודל $\frac{\log^2 n}{4}$. בסך הכל עלות ההכנה: $O(n)$.

<u>הכנה:</u>

יש לנו חתיכה בגודל $\frac{\log n}{2}$.

1. מצא ייצוג בינארי.
2. בנה עץ בינארי בגובה $\frac{\log n}{2} - 1$, הטבלאות רק בעלים.
3. כל חתיכה תעבור על העץ (0-שמאלה, 1-ימינה) ותחליט קיים / לא קיים.

- אם קיים, הצבע לטבלה המתאימה.
- אחרת, בנה את הטבלה.

<u>מענה על השאילתא:</u> קל, $O(1)$.

פתרנו את בעיית ה- RMQ כאשר ההפרש בין שני איברים סמוכים הוא 1 או 1- (וכך גם פתרנו את בעיית ה- LCA). מה קורה במקרה הכללי?

עץ קרטזי

"זה עץ מאוד חביב"

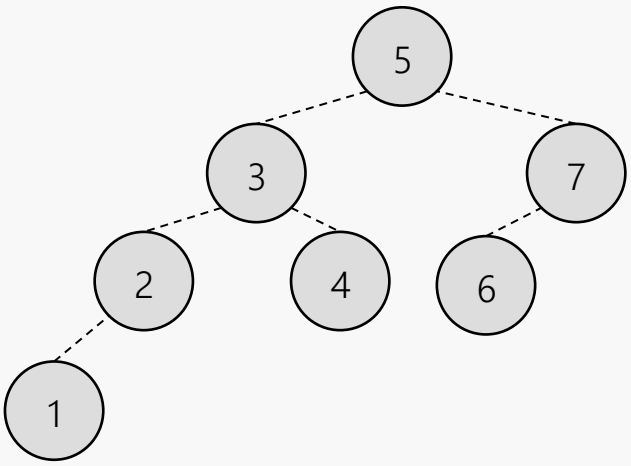
<u>קלט:</u> מערך מספרים.

<u>פלט:</u> עץ קרטזי.

שורש העץ - האינדקס של המינימום במערך. רקורסיבית, בונים את תתי העצים כך שהבן השמאלי הוא שורש של העץ הנובע מתת המערך השמאלי, והימני לימין.

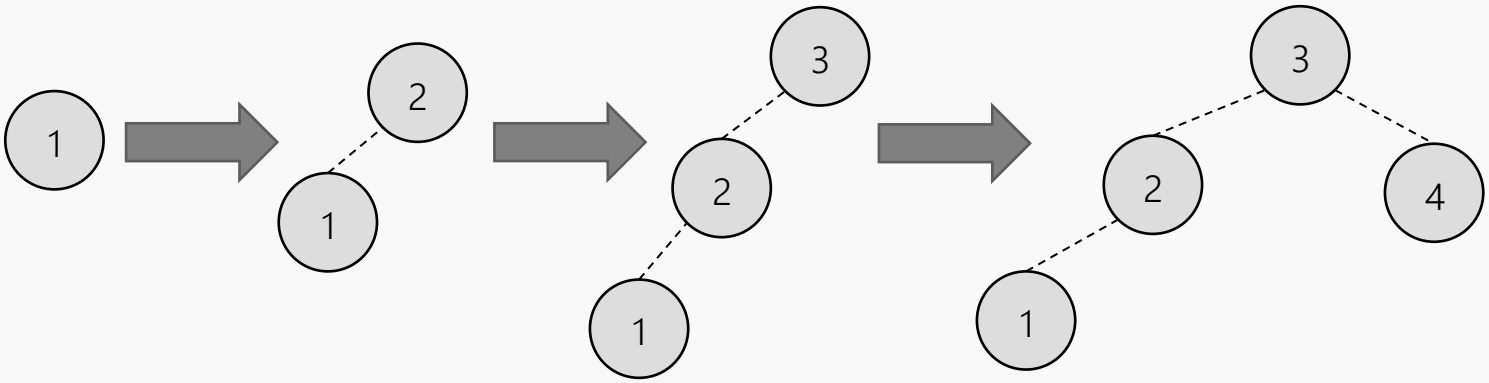
<u>דוגמא:</u>

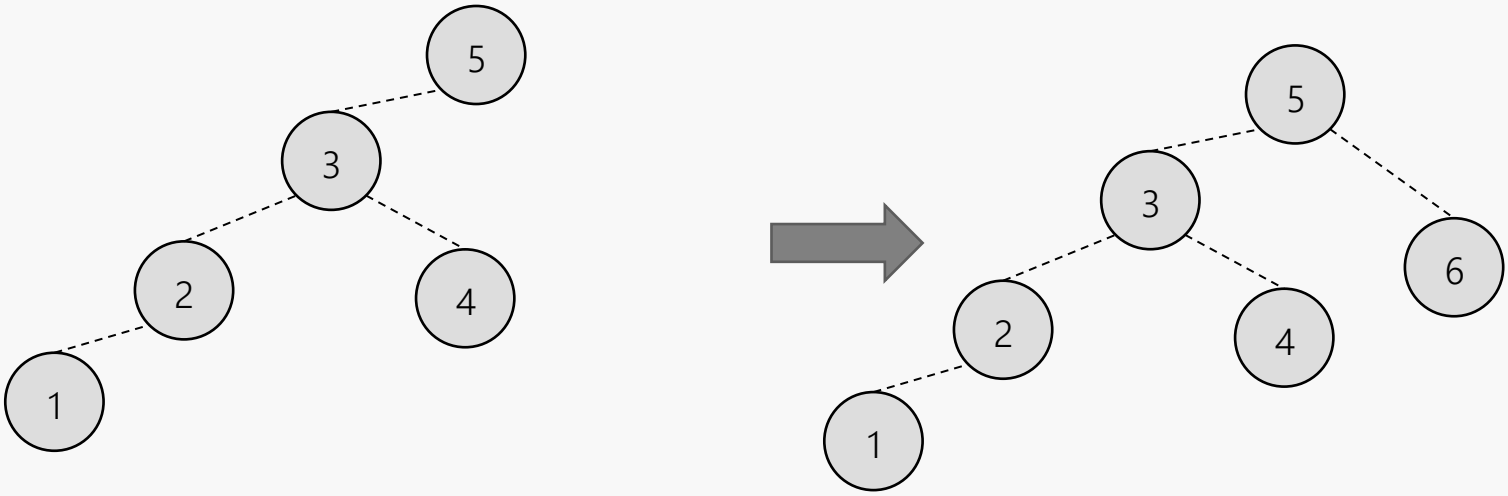
8	2	1	5	−6	12	3	*value*
1	2	3	4	5	6	7	*index*

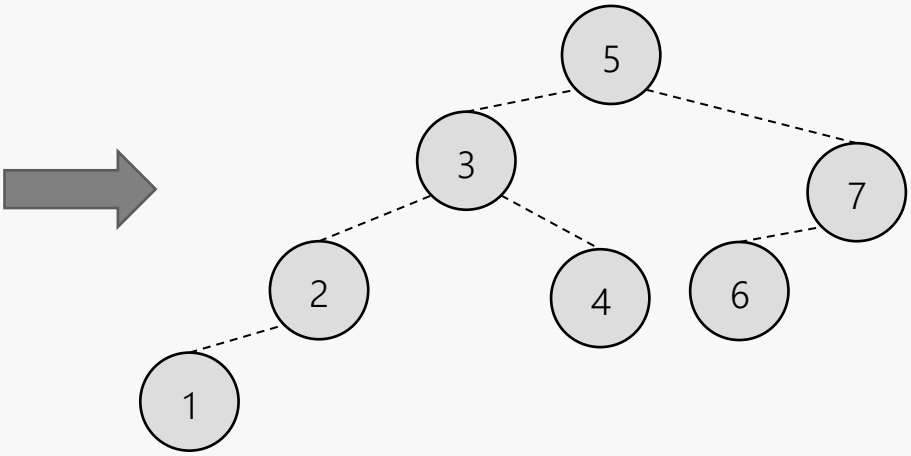


הוספה לעץ קרטזי:

הולכים משמאל לימין ובכל פעם נוסיף איבר.







מספר הפעולות להוספה הוא כגודל הדופן הימנית של העץ. האחרון במסלול הימני הוא תמיד האחרון שהכנסנו. סיבוכיות הזמן לבניית עץ היא משוערך $O(n)$.

<u>פונקציית הפוטנציאל</u> של המבנה היא אורך הדופן הימנית.

נסמן t_i בתור מספר הצמתים מהעלה בדופן הימנית שעקפתי.

$$c_i = t_i + 1$$
$$\Delta\Phi = 1 - t_i$$
$$\Rightarrow \hat{c}_i = 2$$

$$\sum \hat{c}_i = \sum c_i + \Phi_n - \Phi_0$$
$$\Rightarrow \sum c_i \leq \sum \hat{c}_i = 2n$$

העץ הקרטזי פותר את בעיית ה-RMQ במקרה הכללי.

1) בנה עץ קרטזי מהמערך.
2) הכן LCA ב-$O(n)$ לעץ.
3) בהינתן שני אינדקסים i,j, $RMQ(i,j)$ הוא ה-LCA של הצמתים של האינדקסים בעץ.

העברנו את בעיית ה-RMQ הכללית לבעיה המצומצמת של הפרש $1,-1$.

עוד יתרון של העץ הקרטזי בהקשר של RMQ: לשני מערכים יכול להיות אותו עץ קרטזי, ולשני מערכים עם אותו עץ קרטזי תהיה אותה הטבלה ואותן תשובות לשאילתות RMQ. השאלה היא – כמה עצים קרטזיים ניתן לבנות מ-x רשומות? בדיוק $Catalan(x)$.

בנוסף, $Catalan(x) \leq 4^x$. עבור $x = \frac{\log n}{4}$:

$$4^{\frac{\log n}{4}} = \sqrt{n}$$

נרצה להכין $\sqrt{n}$ טבלאות (ב-$O(n)$ עבודה) בעזרת עצים קרטזיים.

א. בנה עץ קרטזי לחתיכה בגודל $\frac{\log n}{4}$.
ב. ייצר סדרה בינארית שתשמש מסלול בעץ הטבלאות.
ג. נייצר מילים של 0ים ו1ים שהיא 1ים ככמות הצמתים שעקפנו בהוספת איבר לעץ ו-0.

(למשל עבור העץ בדוגמא קודם לכן: **010100110010**)

ד. גודל המילה לכל בלוק הוא לכל היותר $2 \cdot \frac{\log n}{4} = \frac{\log n}{2}$.
ה. התאם לכל חתיכה את הטבלה שלה.

ולכן עבור גודל בלוק $\frac{\log n}{4}$ ונוכל לבצע הכנה ל-RMQ ב-$O(n)$ עבודה.

מצא אחד – Union-Find

קלט: n איברים, לכל איבר שם הקבוצה (מצביע לראש הקבוצה). נרצה להכין מבנה נתונים העונה על השאילתות:

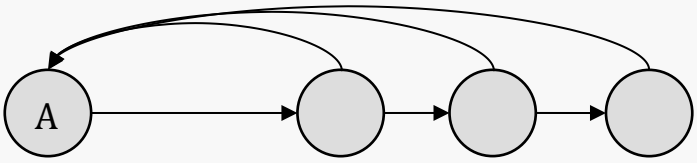
1. <u>מצא</u>
   קלט: מצביע לאיבר.
   פלט: שם הקבוצה של האיבר.
2. <u>אחד</u>
   קלט: שני מצביעים.
   פלט: שתי הקבוצות מאוחדות.

<u>פתרון ראשון</u>

- לכל איבר נוסיף את שם הקבוצה, לכן מצא ב-$O(1)$.
- כאשר מאחדים נשנה את שם הקבוצה לאיברי הקבוצה הקטנה.

אם נתחיל מקבוצות של איברים בודדים ונגיע למצב של קבוצה אחת גדולה, יכול להיות שהאיחודים יעלו <u>הרבה</u>. נוסיף לכל ראש קבוצה שדה של גודל הקבוצה.

כל קבוצה תיראה כך:



עבור סדרה של n פעולות סיבוכיות הזמן תהיה $O(n \cdot \log n)$. פעולת מצא תעלה $O(1)$, בכל פעם שאיבר מפסיד באיחוד המשפחה שלו גדלה בלפחות פי 2 (אם הן שוות – אחרת אפילו יותר). לכן ניתן להפסיד לכל היותר $\log n$ פעמים. בסך הכל עבור n פעולות $O(n \cdot \log n)$.

<u>פתרון שני</u>

כל קבוצה תהיה עץ, ראש הקבוצה בשורש.

באיחוד: נבדוק האם השורשים שונים, אם כן נאחד את העצים. איך מאחדים?

א. נאחד את העץ הנמוך לגבוה.

גובה	מספר צמתים מינימלי
0	1
1	2
2	4
3	8

כלומר העצים יהיו מאוזנים, והשאילתות יעלו $O(\log n)$.

ב. נאחד את העץ הקטן (פחות צמתים) לגדול.

באותה הצורה נקבל שגובה העץ יהיה $O(\log n)$ וכך גם עלות הפעולות.

בסך הכל קיבלנו כי עלות הפעולות היא $O(\log n)$. הרווחנו באחד (כעת האיחוד הוא $O(\log n)$ נקי, קודם לכן לשיעורין), והפסדנו במצא ($O(\log n)$ במקום $O(1)$).

טריק

המטרה של כל איבר להיות קרוב ככל הניתן לשורש. נניח שיש לנו עץ ומסלול בו מעלה עד לשורש. כאשר אנחנו עולים מהעלה לשורש (במטרה למצוא את שם הקבוצה) את כל הצמתים ותתי העצים שלהם שנתקלנו בהם במהלך העלייה נקשר בתור בנים לשורש.

כך נקטין את גובה העץ משמעותית ל- $\log^* n$, וזו גם עלות הפעולות.

דברים נחמדים

מטריצה דלילה

נתון מערך $n \times n$, נתחיל ריק. נרצה לתמוך בפעולות:

א. הוסף איבר.
ב. הדפס איבר.

המערך מתחיל ריק, ויש לאתחל אותו ב-$O(n^2)$. ידוע שיש $O(n)$ פעולות, כיצד נענה על השאילתות ללא ההכנה היקרה?

כאשר אנחנו מוצאים איבר אנחנו יכולים להיתקל בזבל, בהנחה שאין לנו חזיר בר חיפאי אין דרך לדעת האם הערך אמיתי או זבל.

נתחזק מערך בגודל $O(n)$, ונדע בכל רגע נתון כמה ניצלנו (איפה אין זבל). מהתאים המתאימים במערך הדו ממדי ומערך העזר יהיו מצביעים דו כיווניים.

בהוספת איבר נוסיף מצביעים מתאימים ובהדפסה נוודא שאין זבל (המערכים מצביעים אחד לשני בהתאמה) ולאחר מכן נדפיס.

עלות n פעולות היא $O(n)$.

תור משופר

אנחנו רוצים לתחזק תור, ולהוסיף לו את פעולת מצא מינימלי. נשתמש בעץ קרטזי, ובפרט הדופן הימנית שלו.

האיבר המינימלי הוא השורש ונמצא אותו ב-$O(1)$. בנוסף, פעולת הוצאה תעלה $O(1)$ והוכחנו כי פעולת הוספה עולה $O(1)$ משוערך.

מה אם נוכל להוסיף משני הצדדים? נילחם עם הדופן הימנית עד מציאת מקום מתאים.

אם ניתן להיכנס ולצאת משני הכיוונים? לכו לטרז'ן.

שחזור מחרוזת ב-LCS במקום לינארי

לא בחומר.

הרעיון:

נכין לשאילתת LCS ב-$O(n^2)$ זמן ו-$O(n)$ מקום.

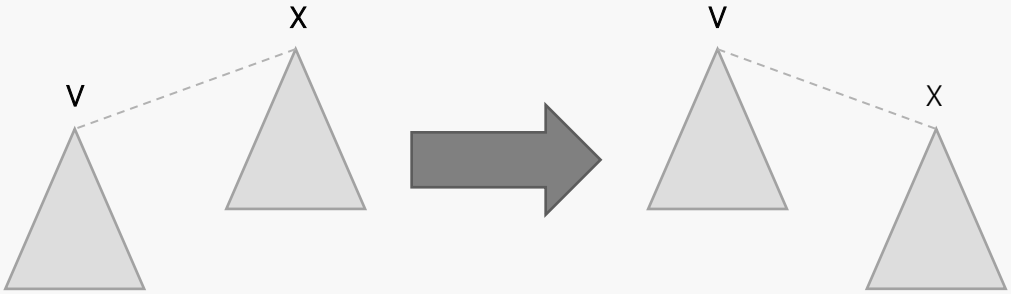
נניח שקיבלנו כי LCS=7. נעצור במקום כלשהו במחרוזת B. נניח שלפני העצירה LCS=4 ואחריה 3. נפעיל רקורסיבית LCS על הרישא של B (עד העצירה) עם A, על הסיפא (אחרי העצירה) עם A באופן הפוך כלומר במקום משמאל לימין מימין לשמאל. שלב זה יעלה $O\left(\frac{n^2}{2}\right)$. לאחר מכן נחפש את המקומות היחידים בהם מתקיים שסכום שני ה-LCSים יצאו 7 באלכסון. לאחר מכן נמשיך לקרוא רקורסיבית עד לעצירה, וכך נמצא את המסלול המתאים. עלות כל האלגוריתם הוא $O(n^2)$, וסיבוכיות המקום היא $O(n)$.

שאלות לתרגול

1. תכנן אלגוריתם המקבל כקלט ערימה בינומית ומצביע לאחד הצמתים בערימה V. האלגוריתם יכפיל את ערכי הצמתים בתת העץ ששורשו V ויתקן את הערימה. נתחו את סיבוכיות הזמן כפונקציה של n, k כך ש- n מספר הצמתים בערימה ו- k מספר הצמתים בתת העץ של V.
2. נתון מערך עם n איברים, ולהם $\log n$ מפתחות שונים. הדפס אותו באופן ממוין.
3. נתונות 2 ערימות מקסימום בינאריות H_1, H_2, הממומשות על ידי מצביעים ולא על ידי מערך. ידוע כי לשתיהן אותו גובה h, בנוסף H_1 היא עץ מלא טורבו. הראו כיצד ניתן לאחד את הערימות לערימת מקסימום אחת.
4. נתון מערך A המיוצג על ידי <u>ערימת מינימום</u>. מצאו את האיבר ה-k הכי גדול ב-A בסיבוכיות זמן $O(n \cdot \log k)$ (תוך כדי שימוש בערימות מינימום בלבד).
5. נתונים n מספרים שלמים וכאשר הם ממוינים ההפרש בין שני איברים עוקבים הוא לכל היותר 5. הצע אלגוריתם מהיר למיון המספרים.
6. יש n רשומות, $\log n$ מפתחות שונים, מספרים ממשיים. נתון שלחצי מהרשומות מפתח זהה, לרבע מפתח זהה אחר וכך הלאה. מיין ביעילות.
7. יש <u>בחירות</u> לראשות העיר בצרפת. n בוחרים (נניח אי זוגי) ו-m מועמדים, לכל מועמד מפתח. נרצה לענות על השאלה: האם יש מועמד שקיבל יותר מ-50% מהקולות ב-$O(1)$ זיכרון נוסף חוץ מהקלט ואין לשנות את הקלט.
8. הצע אלגוריתם לאיחוד שני עצי Splay מעורבבים (לא כמו Join).
9. חלק עץ חיפוש ל-k איברים הכי קטנים ו-$n-k$ השאר. רמז: להוסיף שדה.
10. בנה עץ חיפוש בינארי מאוזן ממערך ממוין, בשתי דרכים.
11. נקודות בריבוע היחידה במישור נתונות על ידי (x, y). יש להציע מבנה נתונים התומך בפעולות הבאות:
    a. $insert\big((x,y)\big)$, $O(\log n)$.
    b. $delete\big((x,y)\big)$ $O(\log n)$.
    c. $line(m)$ הדפסת על הנקודות במבנה הנמצאות על הישר $x_i + y_i = m$, בצע ב- $O(k + \log n)$ כך ש-k מספר הנקודות בפלט.
12. הצע תמיכה בשאילתא $less(x)$ בעץ חיפוש המחזירה את סכום האיברים שקטנים או שווים ל-x בעץ.

שאלות לתרגול – פתרונות

1. נניח בה"כ כי הערימה היא ערימת מקסימום.
   a. הכפל את ערכי המפתחות של תת העץ – $O(k)$ (הסדר בתוך תת העץ נשמר).
   b. נסתכל על $X = parent[V]$:
      i. אם $key[V] \leq key[X]$ סיימנו.
      ii. אחרת, נחליף בין V ותת העץ שלו ו- X ותת העץ שלו וחזור לתחילת שלב 2. (עד שהגענו לשורש או נעצרנו ב- a).

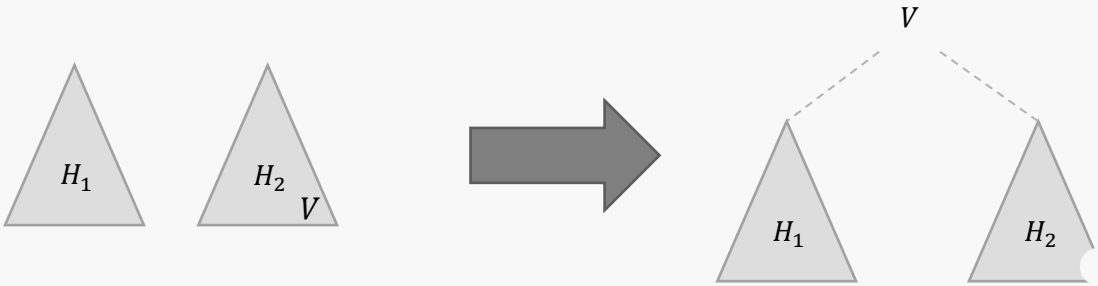


<u>סיבוכיות הזמן:</u> גובה העץ ש- V נמצא בו הוא $O(\log n)$, לכן בסך הכל $O(k + \log n)$.

2. נפתור בעזרת <u>מיון הכנסה</u>, נחפש את האיבר עם חיפוש בינארי במערך:
   i. אם מצאנו איבר שכזה בחיפוש הבינארי נשרשר את החדש אליו – עולה $\log\log n$ (חיפוש בינארי על $\log n$ איברים), ויש $n - \log n = O(n)$ רשומות כאלה.
   ii. אחרת הוסף חדש למערך – עולה $\log n$ (הזזה במיון ההכנסה $O(\log n)$ רשומות), ויש $\log n$ כאלה (איברים "חדשים").

   בסך הכל – סיבוכיות הזמן היא $O(\log^2 n + n \cdot \log\log n)$.

   - ניתן לפתור גם עם עץ חיפוש.
3. ניקח את העלה הימני ביותר מ-H_2 ונעלה למעלה, הוא יהיה האבא של שני שורשי הערימות. כל H_1 בנפרד מקיימת את תכונת הערימה וגם כל H_2, נשאר רק לדאוג לכך שהשורש יהיה החביב – נבצע את הפעולה הבסיסית על השורש עד שנגיע לאיזון מחדש – $O(\log n)$.



4. ניעזר בערימת מינימום בגודל $k+1$.
   a. הכנס k איברים מהערימה לערימת העזר
   b. עבור כל אחד מ- $n-k$ האיברים (איטרטיבית):
      i. הכנס לערימת העזר, $O(\log k)$
      ii. הוצא מערימת העזר חביב, $O(\log k)$

אחרי כל האיטרציות ערימת העזר תכיל את k המספרים הגדולים ביותר, והמספר שלנו בראשה. <u>סיבוכיות הזמן</u>: $O(k \cdot \log k + n \cdot \log k) = O(n \cdot \log k)$.

5. נמצא את הערך המינימלי במערך m, ונחסר אותו מכל איברי המערך – $O(n)$. כעת כל איברי המערך נמצאים בתחום $[0, 5n]$ (המקסימלי הוא לכל היותר $m+5n$), נפעיל עליהם מיון מניה ב- $O(n)$ ונוסיף בחזרה לכל איברי המערך את m. בסך הכל – סיבוכיות הזמן היא $O(n)$.
6. שני פתרונות:
   a. נבנה רשימה מקושרת דו כיוונית, שאורכה $O(\log n)$ ובכל איבר בה יש אחד המפתחות ומספר ההופעות. הרשימה תהיה ממוינת לפי מספר ההופעות בסדר יורד. ברגע שנוסיף למונה של איבר הוא (אולי) יבעבע שמאלה. כמה פעמים הלכנו והחלפנו: נניח שיש 1000 רשומות. נסתכל על האיבר שסופר יש 50% מהרשומות: לפחות 250 מהאחרונות שנספרו מגיעים באופן מיידי לתא. לפחות 125 מהרשומות הבאות עברו לכל היותר שני איברים, וכן הלאה. בסך הכל: $O(n)$ (שוב מופיע הסכום $\sum \frac{i}{2^i}$). לאחר מכן נעבור ונמיין את הרשימה לפי המפתחות (ולא מקודם לפי מספר המופעים) בפחות מ-$O(n)$ ונמיין את הרשומות לפי הרשימה שקיבלנו ב- $O(n)$. בסה"כ: $O(n)$ עבודה.
   b. כמו שאלת הבחירות ולמחוק כל פעם את המנצח. כך נקטלג את האיברים לפי הגודל ב- $\frac{n}{2}+\frac{n}{4}+\cdots+1=O(n)$ עבודה ונמיין לאחר מכן ב- $O(n)$.
7. נחזיק שני משתנים: שם וכמות. נוסיף נחסיר מונה ב<u>שיעור ספרות</u> לכתוב פורמלי. להוסיף כשמופיע זהה ולהוריד כשמופיע שונה. בסוף נישאר עם שם אחד. אם לשם יש יותר מ-50% הוא המנצח, אחרת <u>אין מנצח</u>.
8.
9. נרחיב לעץ דרגות ונפצל (split) עם k.
10. שני פתרונות:
   a. רקורסיבית, הכנס חציון לשורש ורקורסיבית בנה את תת העץ השמאלי הוא תת המערך השמאלי והימני עם ימין.
   b. נבנה שלד של עץ מלא (לאו דווקא טורבו) בגודל n ונכניס לתוכו את המערך בסריקת $in-order$ "הפוך" (מבחינת רקורסיה) לפי המערך משמאל לימין.

11. נבנה עץ חיפוש של עצי חיפוש (שהמפתח בכל עץ חיפוש הוא x) כך שהמפתח הוא $m = x + y$.
   a. <u>הכנסה:</u> נחפש את הרשומה לפי המפתח m, אם קיים הכנס לתוך עץ החיפוש בתוך הצומת. אחרת, בנה חדש. $O(\log n)$.
   b. <u>הוצאה:</u> נמצא את עץ החיפוש הקטן המתאים, אם לא קיים סיימנו. הוצא את הרשומה מהעץ הקטן, ומחק את העץ הקטן מהגדול במידה ולא נותרו רשומות. $O(\log n)$.
   c. <u>Line:</u> מצא את העץ שמפתחו m והדפס את העץ בסריקה. $O(\log n + k)$.
12.