# סדר קרס מבני נתונים:

## נושא 1: סיבוכיות זמן ריצה:

**סימון:**

O חסם עליון - מתאר את הסדר הגדול ביותר. הפונקציה לא חורגת יותר מ $O(f(n))$ פעולות. למשל: חיפוש איבר במערך ממוין. המערך הגדול ביותר, הכי הרבה לא שם אבוא עסוף המערך וזה דורש לחפש על כל הפרמטר. אז, גם שסיבוכיות החיפוש הוא $O(n)$.

$\Omega$ חסם תחתון - מתאר את הסדר הכי טוב ביותר, אז החסם הנמוך ביותר שמאפשר לאלגוריתם לפעול. למשל, כדי לאכול את המעבר המאפשר לא אמון, אתה חייב לפחות פעם אחת. לכן, הפעולה הטובה ביותר $\Omega(n)$.

$\Theta$ חסם הדוק - כאשר החסם העליון והחסם התחתון שווים.

**הגדרות פורמליות:**

$f(n)$ - מספר הפעולות עם האלגוריתם שלנו. $g(n)$ - פונק' שאינה עולה. $f(n), g(n) \geq 0$.

חסם עליון $O$: $f(n) = O(g(n))$ אם קיימים קבועים $c > 0$ ו $n_0 \geq 0$ כך שלכל $n \geq n_0$ מתקיים: $f(n) \leq c \cdot g(n)$.

(הסל מלא מעל $n_0$, הפונק' $g(n)$ כפול קבוע גדול פחות מסמן את עמו הרבה שלנו עלמלה).

חסם תחתון $\Omega$: $f(n) = \Omega(g(n))$ אם קיימים קבועים $c > 0$ ו $n_0 \geq 0$ כך שלכל $n \geq n_0$ מתקיים: $f(n) \geq c \cdot g(n)$.

(הפונק' תמיד תגדל יותר מה פעמיים n (לא כפול בפחות מ $n_0$ מעל).

חסם הדוק $\Theta$: $f(n) = \Theta(g(n))$ אם קיימים קבועים $0 < c_1, c_2$ ו $n_0 \geq 0$ כך שלכל $n \geq n_0$ מתקיים $c_1 g(n) \leq f(n) \leq c_2 g(n)$.

(הפונק' חסומה מלמעלה ומלמטה ע"י אותה משפחת פונק' ארבע. בפועל).

$\infty = T$: אפשר לבדוק אם $f(n)$ גדול יותר מ $g(n)$ ולכן $f(n) = \Omega(g(n))$.

$0 = T$: הפכה לגמרי אם $f(n)$ גדול פחות ולכן $f(n) = O(g(n))$.

$\infty > T > 0$: $f(n) = \Theta(g(n))$, אפשר לבדוק בקלות.

$$T = \lim_{n \to \infty} \frac{f(n)}{g(n)}$$

מספר הנפלאות לאלגוריתם הקבועה:

## נושא 2: מחסנית (Stack) ותור (Queue):

מחסנית: לקחת: נכנס אחרון, יוצא ראשון.

פעולות מרכזיות: push - הכנסה לראש המחסנית. pop - הוצאה מהראש המחסנית. שניהם מתבצעים ב $O(1)$.

תור: לקחת: נכנס ראשון, יוצא ראשון.

פעולות מרכזיות: Enqueue - הכנסה לסוף התור. Dequeue - הוצאה מהראש התור. שניהם מתבצעים ב $O(1)$.

תרגיל: יש לנו 2 מחסניות $S_1, S_2$ - יש לבצע 1 ב $S_2$ לבוא:

1. Enqueue (10)
2. Enqueue (20)
3. Enqueue (30)
4. Dequeue()
5. Enqueue (40)
6. Dequeue()

נושא 3: הפרד ומשול:

הגישה אומרת: אם תוכל לפתור את כל הבעיה כמה חלק אתה לבעיה קטנה יותר, אמור יותר ואז לחבר את התוצאות.

הגישה מורכבת מ3 שלבים: 1) הפרד - פצל את הבעיה המקורית לכמה תת בעיות.

2) משול - פתור כל תת בעיה בנפרד (נניח בחיפוש בינארי). ואם הבעיה מספיק קטנה, הפתרון שלה מיד.

3) כבש - צרף את התוצאות של תת הבעיות (נניח בחיפוש בינארי, הפתרון לבעיה המקורית.

## מיזוג מערכים:

נתונים משוואת נסיגה ושאלים 3 שאלות: 1) מה הגודל המקורי?

2) כמה ילדים יש לו?

3) מה גודל הקלט של הילד?

1. גודל מקורי - $n$. גודל הקלט בשורש.
2. כמה ילדים? 2.
3. גודל כל ילד? $\frac{n}{2}$

$$T(n) = 2T\left(\frac{n}{2}\right) + n$$




$n$

$\frac{n}{2} \cdot 2 = n$

$\frac{n}{4} \cdot 4 = n$

$$T(n) = 4T\left(\frac{n}{2}\right) + n$$




$n$

$\frac{n}{2} \cdot 4 = 2n$

$\frac{n}{4} \cdot 16 = 4n$

$\frac{n}{8} \cdot 32$

כלל

$n + 2n + 4n + 8n + \dots$

אבל הקלט הכי מפסל כי כבר כמה מחלקים בב.

כרגע 1 1 גמרים, ב: 4 גמרים: ג: 16 גמרים.

ברמה ה: $4^k$ גמרים. הרמה האחרונה הכי מפסל ולכן

ברמה האחרונה: $4^{\log_2 n}$ גמרים

$$2^n = 4^{\log_2 n} = n^{\log_2 4}$$

$$2^n + \dots + 8n + 4n + n \in \Theta(2^n)$$

## מיון המזגים:

אם סורים? פותחים את הקוסמה שום שורה עד שמגיעים תוצאה (תקינה) ושאלים: מתי הגיע לתוך הפירה?

יפרק את הבעיה בגודל $n$.

$$T(n) = 2T(\text{גודל}) + n$$

פירוק תוצאות — הזמן שלוקח — הזמן שלוקח לסרוק סתם

$$T(n) = T(n-1) + n$$

$$T(n-1) = T(n-2) + n-1$$

$$T(n) = T(n-2) + n + n-1$$

$$T(n-2) = T(n-3) + (n-2)$$

$$T(n) = T(n-3) + (n-2) + n + n-1$$

$$T(n) = T(n-k) + \underbrace{n + (n-1) + (n-2) + \dots}$$

המספרים הולכים מ n-k ל n

1. מחלקים את מסמך הטנר ומקבלים את הקבוצה הבאה כמו.
2. מגדירים את k ומאתחלים מערך של מספרים (מערך חשב לינ' הקדם).
3. מעדכנים את ההפשים של L ל R כדי לעצור את k.
4. מעלים את k.
5. מחלקים 3 פעמים לינאר חוקיות.

הערות:

תנאי עצירה: נניח $T(1)=1$ : $T(n-k)=T(1) \leftarrow n-k=1$ $\quad k=n-1$

$$T(n)=\underset{\underset{T(n-(n-1))=1}{\parallel}}{T(n-k)}+\sum_{i=n-k}^{n} n$$

$k=n-1$

$n-(n-1)=1$

$$\sum_{i=1}^{n} n = n+\dots+2+1 = \frac{n(n+1)}{2} = \frac{n^2+n}{2}$$

סכום חשבוני

$$T(n)=1+\frac{n^2+n}{2}$$

$$n^2+n+2 \le n^2+n^2+2 = \frac{1}{2}n^2+2n^2+1 = 100n^2$$

בי"ס $c=100$ ו $n_0=1$ כך שלכל $n$ מתקיים $T(n)\le 100n^2$ ולכן

$T(n)=O(100n^2)=O(f(n))$.

בי"ס $c=\frac{1}{4}$ $n_0=1$ כך שלכל $n$ מתקיים $T(n)\ge \frac{1}{4}n^2$ ולכן $T(n)=\Omega(f(n))=\Omega(n^2)$.

סה"כ $\Theta(n^2)$.

$$T(n)=T\left(\frac{n}{2}\right)+1 \qquad i=1 \quad T\left(\frac{n}{2^i}\right)+i$$

$$=T\left(\frac{n}{4}\right)+1+1 \qquad i=2 \quad T\left(\frac{n}{4}\right)+2$$

$$=T\left(\frac{n}{8}\right)+1+1+1 \qquad i=3 \quad T\left(\frac{n}{8}\right)+3$$

$$=T\left(\frac{n}{10}\right)+1+1+1+1$$

$$T(n)=T\left(\frac{n}{2^i}\right)+i$$

עצירה כאשר $\frac{n}{2^i}=1$ :

$n=2^i$

$\log_2 n=\log_2 2^i$

$i=\log_2(n)$

$T(n)=1+\log_2(n)$

$c=100$ עבור $n_0=5$

$\log_2 n \le 1+\log_2(n) \le 100\log_2(n)$

לכל $c=1$, $n_0=1$.

## משפט האב:

$$T(n)=a\cdot T\left(\frac{n}{b}\right)+f(n)$$

מספר הבנים: $n^{\log_b a}$.

מקרה 1: מספר הבנים גדול מ $f(n)$ פי רב הפולינום ב $\varepsilon$ והסיבוכיות היא: $\Theta(n^{\log_b a})$.

מקרה 2: מספר הבנים שווה ל $f(n)$, והקבלה מתחלקת בשווה בכל שלבי העץ. והסיבוכיות היא: $\Theta(n^{\log_b a}\cdot \log(n))$.

מקרה 3: מספר הבנים קטן מ $f(n)$ פי רב הפולינום והסיבוכיות היא $O(f(n))$.

מקרה 4: כאשר יש אבן $f(n)$ יקומבינטורית. אשר $f(n)=O(n^{\log_b a}\log^k n)$. אי אפשר את הקבלה הגיאומטרית בד. הסיבוכיות היא $\Theta(n^{\log_b a}\log^{k+1} n)$.

**החלפת משתנים:** נניח שקיבלנו $T(n)=2T(\sqrt{n})+\log(n)$.

הנוסחה לא יכולה לטפל אי הסדר הזה.

הטריק - החלפת משתנים: נגדיר $n=2^m$ ולכן $m=\log_2 n$.

נציב את המשתנה החדש: $T(2^m)=2T(2^{\frac{m}{2}})+m$

נסמן אותו: $S(m)=T(2^m)$

$$S(m)=2\cdot S\left(\frac{m}{2}\right)+m$$

לפי מאסטר:

$m=2^{\log_2 m}$ $\qquad n=2^{\log_2 n}$

עבור ב: $S(m)=\Theta(m\log(m))$.

מהחלפה ב: $m=\log_2 n$: $\quad T(2^{\log_2 n})=\Theta(\log n\log\log n)$

$$T(n)=\Theta(\log(n)\log(\log(n)))$$

תמיד לחשוב גמר מסלול מול עמן ביבה ודאון!

## נושא 4: ניתוח אלגוריתמים:

ניתוח אלגוריתם אומר: בואו נסתכל על סדרה משמה שם n פעולות במערך הנתון. נחשב את האות שם טמא פיחוד ונחלק בכל. הקבוצה הזו האות "הממוצע" (אלגוריתם) שם פעולה בגדלא.

3 שיטות לחישוב:

שיטה 1: שיטת הפיבונאצ': מחשבים את סך הפעולות לפי הפעולות ומחלקים בכל.

חוקי המערך הבינארי: מתחילים עם מערך בגודל 1.

כשמגיעים איבר, אם יש מקום, עולה 1.

אם אין מקום (מלא) מכפילים מערך חדש בגודל כפול. מעתיקים אליו את כל האיברים הישנים (עולה x פעולות לx איברים), ואז מוסיפים את החדש (+1).

אם נכפל n הפעמים: יש n פעולות וזיכרון ההעתקה לתא הגיע ומלא n פעולות.

מה מתמקים? כשהגדל הוא חזקה של 2 (איבר הבא, 4, 8, 16...). סכום הפעולות: $n + \dots + 4 + 2 + 1$. סכום סדרה הנדסית, סכום שקול ל $2n$.

סה"כ: שמונה פחות מ $2n$ הפעולות - חלק בn פעולות ונקבל $\underline{8}$. אבל מגיע לשמרין הוא $O(1)$.

שיטה הבנק: במקום לסכום הכל בסוף, אנחנו מחייבים מטבעות חדש שמשמשים מראש על כל פעולה, כמארה "חסוך" לפעולות הקרות.

מבוא המערך הבינארי, אנחנו קובעים שכל פעולה תקבל 3 מטבעות: כמחיר מיידי: 1. מטבע אחד משמש עבור ההכנסה עצמה שם האיבר החדש נכנס.

2. מטבע שני נשמר בבנק ומחכה לאיבר החדש (כדי לשלם על העתקה בל איבר שם האיבר הישן מגיע).

3. מטבע שלישי נשמר בבנק ומחכה לאחר מהפרים הישן (כדי לשלם על העתקה בל איבר שם הישן במעבר הבא).

כשהמערך מתמלא ונגיע להכפיל אנו משתמשים בכסף מטבעות בבנק כדי לשלם על העתקה של כל האיברים מבלי שבבנק מכניס יותר.







* המערך בגודל 2 ומלא, כדי להכפיל צריך 2.
* להעתיק את איבר מס' 2 הישן עולה מטבע אחד. נבח מהקופה ולא נגע בקופה 2 מטבעות.
* איבר מס' 2 נכנס לתא הנתון ומשלם סה"כ 3: 1 לכניסה שלו ו2 מטבעות קדמו בקופה. כתא נשמרו 3 מטבעות בקופה.
* המערך מלא ומגיע להכפיל מקום לגודל 4.
* מכפילים את איבר 3, משלם מטבע על מלאו ו2 בקופה. סה"כ 3 מטבעות.
* מעתיקים לתא איבר 1 ומשלם מטבע בקופה. נותר מטבע בקופה.

בואי נסתכל על המצב רגע לפני יום הדין – אנחנו עכשיו במערך של 4 מקומות מלאים (איברים 1, 2, 3, 4), וצריכים להכפיל אותו ל-8.

בואי נבדוק כמה מטבעות נשארו לכל איבר באופן אישי (לפי חוק: כל אחד מביא 2 מטבעות לעצמו):

- **איבר 1:** הגיע עם 2 מטבעות. הוא הועתק כבר **פעמיים** בחיים שלו! (פעם אחת כשהכפלנו מ-1 ל-2, ופעם שנייה כשהכפלנו מ-2 ל-4). לכן הוא בזבז 2 מטבעות.
  - **יתרה אישית: 0.** (הוא באמת התרוקן לגמרי).
- **איבר 2:** הגיע עם 2 מטבעות. הוא הועתק **רק פעם אחת!** (כשהכפלנו מ-2 ל-4). לכן הוא בזבז רק מטבע 1.
  - **יתרה אישית: 1.** (הוא מסודר, יש לו מטבע להעתקה הקרובה!).
- **איבר 3:** הגיע לא מזמן עם 2 מטבעות. עדיין לא הועתק מעולם.
  - **יתרה אישית: 2.** (יש לו עודף!).
- **איבר 4:** הגיע הרגע עם 2 מטבעות. עדיין לא הועתק מעולם.
  - **יתרה אישית: 2.** (יש לו עודף!).

**סך הכל בקופה הכללית:** 0 + 1 + 2 + 2 = **5 מטבעות.** (בדיוק המספר שהגענו אליו קודם!).

**אז מי צריך תרומה ומי תורם?**

עכשיו אנחנו צריכים להעתיק 4 איברים, וזה עולה 4 מטבעות.

- איבר 2 מסודר – הוא משלם על עצמו מהיתרה שלו.
- איבר 3 מסודר – יש לו 2, הוא משלם 1 על עצמו.
- איבר 4 מסודר – יש לו 2, הוא משלם 1 על עצמו.
- **אבל מי משלם על איבר 1 המסכן שנתקע בלי כסף?**

הנה הקסם: לאיבר 3 (ולאיבר 4) יש מטבע עודף! איבר 3 פשוט לוקח את המטבע המיותר שלו ומכסה את החוב של איבר 1.

**המסקנה ממה שגילית:**

לא כולם עניים! בכל פעם שהמערך מוכפל, רק האיברים הכי-הכי ותיקים (החצי הראשון של המערך) מגיעים למצב שהם מבזבזים את כל המטבעות שלהם וצריכים עזרה. והאיברים הכי חדשים (החצי השני של המערך) בדיוק מביאים איתם את ה"מטבע השלישי" העודף הזה כדי לממן אותם. ככה הבנק לעולם לא נכנס למינוס.

**שיטת הפוטנציאל:** מגדירים פונקציה שנקראת פונק' הפוטנציאל וסימנה: $\Phi(D_i)$. הפונ' ממפה כל מצב $D_i$ למספר ממשי $\Phi(D_i)$.

$D_i$ - הפוטנציאל של מבנה הנתונים לאחר הפעולה ה $i$ סדרה $\langle a_1, a_2, \dots, a_n \rangle$. מצב התחלתי הוא $D_0$.

הנוסחה המפצלת: $\hat{C}_i = C_i + \Phi(D_i) - \Phi(D_{i-1})$

| | | |
|---|---|---|
| השינוי בפוטנציאל, מסומן $\Delta\Phi$. | הדרוש האמיתי שמודד הרצ (1) בעסה כלפי או n להפעלה למשל) | העלות לשלמדן - "העבודה היעילה" שאנחנו רוצים לחקור. |

הפוטנציאל הוא שני.

איך זה עובד בפועל? כאשר $C_i$ גדול, הפעולה יקרה, אנו מגדירים את הפונק' כך שהפוטנציאל יגדל. השינוי חיובי. ולכן העלות לשלמדן בפועל מוגבלת מהעלות האמיתית. ("סופגים" את העודף).

כאשר $C_i$ קטן, הפעולה זולה, אנו מגדירים את הפונק' כך שהפוטנציאל יקטן תמיד ב $-1$. השינוי הנטו שלו שלילי (למשל $-1$) ויוצר מרווח כדי לכסות את העלות האמיתית הגבוהה, כך העלות לשלמדן נשארת נמוכה ויציבה.

דוגמה: מערך דינמי

הגדלה (כפילות) פס המערך כאשר = size

הקטנה: חצי הכיווצים כאשר המערך מלא =num

הרעיון לשלמדן לסדרת פעולות הוא:

$$\sum_{i=1}^{n} \hat{C}_i = \sum_{i=1}^{n} C_i + \Phi D_n - \Phi D_0$$

* $\Phi D_0 = 0$.
* $\Phi D_i \geq 0$ לכל $i \geq 1$.

דוגמה: מערך הקטן כהכפלה בחצי. מתחילים בגודל 1 ושל גדר הערך מכפיל מערך חדש בגודל מי.

כאשר הקבוצה מעתיקים את כל הכיווצים מהמערך הישן לחדש

שיטה א': הגדלה בד. בפעם ה $i$ מעתיקים $i$ איברים למערך חדש. סה"כ סדרה ב $n$ פעולות.

$$\sum_{i=1}^{n} C_i = \sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

כביר

סדרה חשבונית $C_i = i$ כמה שעולה בפועל

להעתיק $i$ איברים עבור $i$

$$\frac{\sum C_i}{n} = \frac{\frac{n(n+1)}{2}}{n} = \frac{n+1}{2} = O(n)$$

$\hat{C}$ - עלות לשלמדן

שיטה ב': מתחילים עם מערך בגודל 2 וכשגדר המקום נגמר גדל את גודל המערך.

רעיון: כל פעולה לא תגרום גדל 2 עלות 1 כי הוספה ושמירה במערך.

כל הגדלה תגרום גדל $2^i$, תעלה $2^i$ בגלל העתקת כל הכיוונים

$$\sum_{i=1}^{n} C_i = \sum_{i=1}^{n} 1 + \sum_{j=0}^{\log_2(n-1)} 2^j = n + (2n-1) = 3n-1$$

עלות הכפלה

מעתיקים הכפלה

מעתיקים עלות הכפלה

הגדר ה משמעות לחלק את

מערך ערוך

מעתיקים סוף n איברים.

$2^j \leq n-1$

$j \leq \log_2(n-1)$

$$\hat{C} = \frac{3n-1}{n} \in O(1)$$

מערך דינאמי - פונקציה:

$$\Phi D_i = 2(n_i - \frac{l_i}{2}) = 2n_i - l_i$$

אורך

גודל

כאשר $l_i = n_i$ "נפרק את הפונקציה".

כאשר $n_i < l_i$ אז הגבלנו את מבנה הנתונים.

הכנסה למערך דינאמי: $n_i = n_{i-1} + 1$ עם איברים

$l_{i-1} = C_i$ עלות בפועל

גודל המערך

העתקה ביניים לה בלבד: $l_{i-1} = n_{i-1}$ ← אפס. הפעולה בגלל המערך

שווה למס איברים אין. אין מקום לעוד איבר. $l_i = 2l_{i-1}$ מכפילים ב 2.

אחרי ההוספה מס איברים עלה ב 1: $n_i = n_{i-1} + 1$

$$\Phi D_i - \Phi D_{i-1} = 0$$

$$\Phi D_{i-1} = 2(n_{i-1}) - l_{i-1} = 2l_{i-1} - l_{i-1} = l_{i-1}$$

$n_{i-1} = l_{i-1}$

$$\Phi D_i = 2n_i - l_i = 2n_{i-1} + 2 - 2l_{i-1} =$$

מס איברים סאחר הכפלה

גודל

$$2 = 2n_{i-1} + 2 - 2n_{i-1}$$

$$\hat{C}_i = C_i + \Phi D_i - \Phi D_{i-1}$$

עלות לשלמדן

$$= l_{i-1} + 1 + 2 - l_{i-1} = 3 \in O(1)$$

דוגמות נוספות שלא הפוטנציאל:

1. מחסנית עם פעולת מחיקה מרובה : Multipop :

push(x) מכניס איבר אחד. $c_i = 1$.

pop() מוציא איבר אחד. $c_i = 1$.

Multipop(k) שולף k איברים ברצף $c_i = k$.

אם נתקלה סדרה של פעולות ה - multipop אז לשווה O(n) ולכן הכל.

נניח בעבור פוטנציאל שהגדלת המודלת לא פחות הוא O(1).

ראשית, נתבונן אצל פעולה יחידה מה יקרה? איפה הסקה הוא רשל גדים איברים במחסנית ולכן ה multipop יכול למחוק מה איברים ולמלא מה.

אם, פונק הפוטנציאל הוא מספר האיברים שנמצא במחסנית: $\Phi_i = \text{size of stack}$. (הפוטנציאל מתחיל מס במחסנית ריקה).

נגזור push: $c_i = 1$ הגדלנו קודם. $\Delta\Phi_i$ : כושר מקסימלי איבר הפוטנציאל עלה בד: $\hat{c}_i = \underbrace{c_i}_{1} + \underbrace{\Phi_i - \Phi_{i-1}}_{1} = 2$.

נגזור Multipop: $c_i = k$ הגדלנו קודם. $\Delta\Phi_i$ : כושר מוציאים k איברים, לפי הנחת פונק הפוטנציאל, גם הפוטנציאל ירד ב k. $\Delta\Phi = -k$. $\hat{c}_i = k + (-k) = 0$

כיוון בפעולה של הmultipop, הוא "פיצה" את הפוטנציאל שהיצרנו במחסנית.

2. מונה בינארי: מונה בינארי מתחיל בכל 0000. מקבלים כל פעם בו.

1000 ← 0000 שינוי ב-1 אחד.

0100 ← 1000 שינוי 2 ביטים.

1100 ← 0100 שינוי ב-1 אחד.

0010 ← 1100 שינוי 3 ביטים.

במערכה הכי גרוע (נעבור מ 0000 לשרה 4 ביטים) נשנה $O(\log(n))$ ביטים.

אנחנו נניח שפעולה לשחרור הוא O(1).

שלב 1: מאיפה מגיעה הסקה? שורה לינה עד 1 שהופכים לה. ביטים שהם 0 ולא מסכנים.

אל פונק' הפוטנציאל שלנו מוגדר: מספר ה'1'ים במערכה $\Phi_i = \text{\#1 פעמים}$

שלב 2: נגזור מעמטי: נניח שיש t אחדות 0 אחד, 1...110, ברגע שלפני 1 נכפה שה מחליפה התהפכו (t פעולות) 0 הנהפכים (פעולה אחת).

סה"כ $c_i = t+1$

הפוטנציאל: לפי נוסחת הפוטנציאל אנחנו מכפיס לרשות גם האחרונה.

הוא t אחדות וכדי לאמר להפוך אפסים t אחדות (השלב: t-) ובסלבי '1' מ הפיכה הם (שלב: 1+)

לכן $\Delta\Phi = 1-t$

$$\hat{c}_i = c_i + \Delta\Phi_i$$

$$\hat{c}_i = t+1 + (1-t) = 2$$

$$\hat{c}_i = 2 \in O(1)$$

אמה $O(\log(n))$ ? נשאל את עצמנו - כמה ספרות בינאריות (ב.ע.) צריכות כדי לייצג את המספר n ?

נניח הגדל למספר 0000 = n. כמה ספרות תדרשות לכמותו? 4.

המשתנה שלנו את מדד המספר הוא $\log_{10}$.

נשתמש בבסיס בינארי (2,1), הפונק' שקובעת את מספר הביטים הוא $1 + \log_2 n$ שלא $O(\log_2 n)$

## נושא ב: עצים:

**מושגים:**

שורש: הצומת הפנימי הבודד, אין לו אבות.

עלה: צומת שאין לו בנים בכלל. (מצוי בדרך הנתיבון על הפה)

דרגה: מספר הבנים של הצומת.

אבות קדמונים: כל מי שנמצא במסלול מהצומת עד השורש.

דאבות: כל מי שנמצא למטה מאתנו עד עומק מסוים.

עומק של צומת: המרחק (מספר הקשתות) מהשורש עד אותו צומת.

גובה של עץ: האפלון הכולל ביותר מהשורש עד לעלה הכי רחוק למטה.

רמה: קבוצת קודקודים בעומק מסוים.

* אלו עצים מתפסים כמעגלים כיוונים, כפי שלמן אפשר יש לפחות הנמר 2 בנים.



עץ סדור = עץ שבו כל מסלולים בסדר הזכים

עץ חיפוש בינארי: עץ בינארי שבו לכל צומת בעץ, כל הקודקודים שנמצאים בתת העץ השמאלי קטנים ממנו! ומי שנמצאים בעץ הימני גדולים ממנו.

**אלגוריתמים בעץ חיפוש בינארי:**

1. **חיפוש:** (אלגוריתם: מתחילים בשורש. אם המספר שמחפשים קטן מהצומת הנוכחי הולכים לשמאלה. אם גדול - הולכים ימינה. ממשיכים עד שמוצאים אותו שקבלים כ null.

סיבוכיות: בגרוע ביותר מה שמחפשים הוא עלה (לא גבוה העץ) $O(h)$.

2. **הכנסה:** (אלגוריתם: כמו בחיפוש - מגיעים למקום המתאים ומוסיפים. הבעיה שיוצרים מחדש לחלוטין את העץ (הנעלם הרבה שהוא מוחק אין מחפשים אותו כמו עלה).

סיבוכיות: יהיו את גובה העץ ולכן $O(h)$.

3. **מחיקה:** נשים לב- אם נמחק צומת באמצע העץ אנו מקלקלים את כל הקשרים שלו. ולכן יש מקרים:

1) מחיקת עלה: אין לו בנים לכן מוחקים אותו ועכשיו.

2) לצומת יש בן אחד: הילד של הצומת שמוחקים נמצא עולה ותופס את מקומו ומוחקים את הצומת.

3) לצומת יש 2 בנים: מוצאים מחליף לצומת שאותו רוצים למחוק. הפונה המחליף יהיה הימנימום בתת עץ ימני.

למה? הוא הצומת הגדול ביותר, הכי קרוב לערך הצומת המוחק. מיסלה יהיה גדול מהשמאלי ומ הצומת שרוצים למחוק.

מעבירים ליהי- כלומר (לא שווה) ערך שאר הנתונים כמות עץ ימני.

התהליך: ? מוצאים מחליף. הולכים מעל ימינה עד הכי שמאלה עד לצומת שאין לו בנים. אזור הימנימום בתת עץ ימני.

2. הפתקה: מעדכנים את הערך של המחליף לערכו הקודם לפני.

3. מחיקה פיזית: מוחקים את הצומת של המחליף (מחיבה קלה כי אין לו בנים, עלה).

סיבוכיות: הסעיף הכי יותר סבוכים ולכן בעל $O(h)$ (פעולת הסתכלות ומחיבה עד עלה בגרוע ביותר עה $O(h)$ ).

**עץ מלא:** עץ בו כל צומת בעץ יש בדיוק 0 בנים או 2 בנים. כלומר, לא יכול צומת עם בן אחד.

דוגמה: מלא:



לא מלא:



**עץ שלם:** עץ בו "חסר" אם הבנים בשמאל מלא.





* מספר הנעלים בעץ הוא בדיוק 1 יותר מהמספר הקודקודים הפנימיים בעץ. מוכיח עלה בינאריקדיה על כמות הקודקודים. בדפד: מגיעים לנעלים עם עה קודקודים. עדיין שקיים עצ.

לצמוח בתוך מעגל פנימי.

אם הסגל היה כמו: לא שאול. ה"חוט "כעת", "בינלאומי" עלולים לחרוג מעל הגבולות.

**סיכום:** רק שרשרת כבדה, השמאלית, נכנסים אז מתכנים גיבור הפלי, אנחנו וגדים מהשמאל נכנס. מצב עמן ריבה שווה לגובה הפלי.

**במקרה הכי גרוע:** אם נכניס לפי את המספרים 5,4,3,2,1 נקבל את:



הפלי הזה בדיוק רשימה מקושרת וגובהו שלו שווה כמעט כמו כלומר $O(n)$.

**חוקים:**

חוק 1: שמור הגובה ההיסטורי $O(h)$ בכל מצומת מיסוף, הוא יכול הכנסה.

חוק 2: הגובה של h הוא $O(\log n)$ עבור n.

**מסקנה:** עמן ריבה בפלי מיסוף נצא בהרבה גולף בסדר $O(\log n)$ לעומת $O(n)$.

**חוק 1:** $h = 1 + \max(h_{left}, h_{right})$

**חוק 2:** $h \le 1.44 \log_2(n+2) - 0.328$

**הפשר לפלי AVL:** אי אפשר להסתמך על מצב המספרים שייכנסו בסדר גבוה שמרומם הפלי יחסית קבוע. אם יכנסו קלנים מעניים המערכת תהפוך לרשימה $O(n)$.

**שלבים סריקה:**

**DFS:** חיפוש עמוק, מבקר רמה אחר רמה. משתמשים בתור (queue) לניהול סדר ביקור הקודקודים.

**BFS:** 1. Pre order: ביקור בשורש, ביקור בתת פלי שמאלי, תת פלי ימני.

2. In order: ביקור בתת פלי שמאלי, שורש, ימין.

3. Post Order: ביקור בתת פלי שמאלי, ימין, שורש.

**חשוב לזכור:** לא נוצר לשחק מבקר פלי רק מ In order אבל נוצר לשחק מ pre order!

**נושא 6: עץ AVL:**

**הרעיון של AVL:** עץ חיפוש בינארי רגיל עם "חישוב" שמסתכלים ומונע מהעץ להסיק ל"מקלישר" ויצרור $O(n)$.

**הגדרה:** לכל צומת כמה יש נתון שנקרא גורם איזון (BF). הקומה בנו: מבנה תת פלי ימני פחות גובה תת עץ שמאלי. (גם הפיכה בסדר, אבל עקבי).

**מבני בקרה:** גורם איזון מוכר: $-1, 0, 1$. כל אחר שרשלי מצטבר אי נמצא מאזן.

**גורם איזון לא חוקי:** $-2, 2$. תת עץ אחד גבוה מהשני, וחייב לתקן.

**הגדרה:** כרגע שמאלי כאשר ב BF שווה ל $-2$ עוברים הכל ולפני שמאלי ת $-2$ לתקן.

כי. לדעת איך לתקן נסתכל על המצבים שיש את הפליה: מרקמת הגבוה לצלה טוב שלפנו כי. לבנים את הבית הרחוק שקבר את הילדה?

4 מקרים אפשריים:

1. LL (שמאל-שמאל), הכל הישר שמאלה: הוסיפו לבנות תת עץ **השמאלי**. בלב הפלי **השמאלי**. הפלי השמאלי כבר מלא, ריאה כמו קו ישר ולכן שמאלה.

הפתרון: בצע מקה - זוכרים את הבן השמאלי ומשנים אותו לאמצע להיות השורש הנחש. הארבע שלו "נופל" מצד והופך לבנו הימני שלו.




2. RR (ימין-ימין) הכל הישר הימני: אותו דבר כמו השמאלי, תת עץ ימני כבר מלא. מראה של קו ישר ולכן מצוה.

הפתרון: בצע שמאלה, ההפך מ LL. בונים את הבן הימני ומעלים בשורש, והארבע נופל שמאלה.







**כל שמאלה** → רק מצד. שניהם מבחן.

**שלי (לא מפר את הלוגן).** מצד **ימין** → רק שמאלה.

**צריך להחליט את מה הקודקוד**

**אם חדים ממחוק בן עם 2 בנים**

3. RT שמאל-ימין, אינדוקציה: הוספנו בעץ הימני. לכן הוספנו בשמאל (כלומר שמאל זה ימין). כלומר שנקבל עץ מושלם.

הסתכלו: נתון רמה: נצטרך שמאלה על מנת כדי לשמר (מערכת TT) ונצטרך ימינה על השורש.




4. RT ימין שמאל: הוסיפנו בצד השמאלי של הצד הימני. הלכו ימינה ואז שמאלה.

הסתכלו: נתון רמה: קודם נתון ימינה על הצד ואז נתון שמאלה על השורש.



## נושא 7: עץ B:

עץ B הוא עץ "שווה ונאמן" אם נשווה בעומק החסון, אם יש לו מלא אבא אצלם הרבה בשביל לדסס כדי שיבנה מ מיו.

**מבני נתונים:** 1. נשתמש בפרמטר $t \geq 2$ אינני ( ...1,2,3 ).

2. ברגע קודקוד פנים פרט לשורש הוא לפחות $\lceil \frac{m}{2} \rceil$ ילדים כלומר m. לעומת בעלי B עם פרמטר $t=m$, לכל מפתח פנים: לפחות $t = \frac{m}{2}$ ילדים (ברבע לפחות t) ולא 5.

3. בדיוק השורש לפחות 2 ומקסימום m. אז השורש הוא עלה. (אין מצומות 1 ילדים).

4. בכל קודקוד פרט לשורש יש מינימום $t-1$ מפתחות ומקסימום $m-1$ מפתחות. לעומת בעלי B שהפרמטר $t=m$, לכל קודקוד לא שורש יש לפחות 4 מפתחות ומקסימום $\underset{\overset{\shortparallel}{3}}{B} = t-1$ מפתחות.

5. בכל קודקוד פנים מס' הבנים הוא כמס' המפתחות פחות 1. (לחלופין אל לא כמו מוגדרא :



6. כל הנתיבים באורך דומה.

* מושג: עץ 2-3 : הוא עץ B שבו $t=3$.

**ניתוח גובה ומורכבות:**

גובה העץ הוא $\log_m n$.

למה? אנחנו רוצים לסמן את n הגבירים שלנו, כשהתחלנו בבית הוא שהעלה הכי גבוה!

כמה שיותר הכי גבוה מייצרים שיהיה כמה שיותר "רחב". איך נעשה זאת? נעלה כל קודקוד המינימום פנים אפשרי שהחוקים מתקיים.

זאת אומר $\lceil \frac{m}{2} \rceil = t$ לגלגל.

בכל קודקוד פנים לפחות t ילדים ולפחות t-1 מפתחות (חוץ שהלכנו כמו מחוצה




העלה הכי "גבוה": כמה o: הכי רחב אפשר עם עומק 1, ולכן יש לו 2 ילדים.

רמה 1: יש לנו בקודקוד שכל אחד מהם יש לו ילדים מינימום. מינימום מפתחות בכל אחד: $t \cdot 1$ לפי החוקים.

כמה 2: יש לנו $t$ קודקודים. בכל אחד $t-1$ מפתחות (מינימום).

סה"כ $t(t-1)$ מפתחות. מכל קודקוד יוצאים t ילדים ולכן מרמה 2 יוצאים $t \cdot t = t^2$ ילדים.

רמה h: יש לנו $t^{h-1}$ מפתחות. בכל אחד $t-1$ ילדים (מינימום).

כמה ב: יש לנו $t^2$ ילדים ובכל אחד $t-1$ מפתחות. סה"כ $t^2(t-1)$ מפתחות.

סה"כ $t^2(t-1)$ מפתחות. מכל קודקוד יוצאים t ילדים ולכן מרמה 2 יוצאים $t \cdot t^2 = t^3$ ילדים.

$$\text{סה"כ מפתחות בכל העלים: } (t-1)\,t^{h-1} + \dots + (t-1)t^2 + (t-1)t + (t-1)$$

$$\text{רמה } h \text{ (עלים): יש } (t-1)\,t^{h-1} \text{ מפתחות.}$$

$$\text{כמה ב: יש } t^2 \text{ ילדים ובכל אחד } t-1 \text{ מפתחות. סה"כ } (t-1)t^2 \text{ מפתחות.}$$

$$\text{סה"כ } (t-1)t^2 \text{ מפתחות. מכל קודקוד יוצאים t ילדים ולכן מרמה 2 יוצאים } t \cdot t^2 = t^3 \text{ ילדים.}$$

$$\sum t^{i-1} \text{ סכום סדרה הנדסית שסופית } \frac{t^h-1}{t-1}$$

$$n \geq \underset{\text{מספר המפתחות}}{\underset{\shortparallel}{(t-1)}} \sum_{i=1}^{h} t^{i-1} + 1$$

$$\text{לפי}$$

$$n \geq (t-1)\sum_{i=1}^{h} t^{i-1} + 1$$

$$\text{מצמצם תרגיל}$$

$$n \geq \cancel{(t-1)}\,2 \cdot \frac{t^h-1}{\cancel{t-1}} + 1$$

$$1 + 2(t^h - 1) \le n$$
$$1 + 2t^h - 2 \le n$$
$$2t^h - 1 \le n$$
$$t^h \le \frac{1+n}{2}$$
$$\log_t t^h = \log_t \frac{1+n}{2}$$

log

$$h \le \log_t \frac{1+n}{2} = 1 + h\log_t \frac{1+n}{2}$$ ← עדכון מקבלים
1+ הגדלים מ

הגבה חסום מלמעלה ולכן סיבוכ $O(\log_m n)$ .

אלגוריתם הכפר לפלש ד: 1. חיפוש: יורדים במערך הפלש עד לתחילה לאלה המתאים להקסה.

2. הכסה לעלה: אם יש מקום - מכניסים ומסמנים. אם לא, מבצעים מפה overflow.

3. פיצול: מפצלים ל-2 את הצומת. הפרש התכונה: עלה לעלה.

4. פיצוי: התכונה חוצה רקורסיבית כלפי מעלה בעזרת הפרוק. רק פעם שורש תגדיל את גובה העץ.

אלגוריתם מחיקה מעץ ד: מתלבטים לגבי:

1. מחיקת עלה: אם נשאר מקום (מספר ילדים $\ge \lceil \frac{m}{2} \rceil - 1$) אין סיבוך.

אם נגיע חוסר underflow הצומת נהפך "גרעין מדי" ומקבלים לטיפול בחוסר.

2. מחיקה מרובדת פנימי: מוצאים מחליף (הילד בסדר ו הימיומים בתת עץ ימני).

מחליפים ביניהם.

מוחקים פיזית את המחליף מהמקום המעוי. המעלה הזו תמיד עלה אין תלות לגובה ד.

איך מטפלים בחוסר? א. אם לאחי השכנים (אם מימין/משמאל) יש "עודף" ילדים (יותר מהמינימום): 1. לוקחים ממנו איבר.

2. בוחרים אבר לדם הורה ו לא אבר ישירות אלינו.

עולה לאבא ורק לא יורד אלינו.

הגובה לא משתנה והעץ נשאר מאזן.

ב. מצב: אם לשניהם אין אברים מותרים:

מבצעים איחוד - לצומת בנות בלב + שם שלו + אבר מהבאו מאבריו בינהם ומאחדים את כולם לצומת אחד בגדול.

תוצאה: לאבא יש אבר אחד פחות. אם גם לו חוסר אבל הוא אבא התהליך מתפשט למעלה עד לשורש.

סיבוכיות הזמנים: בכל מקרה לאורך גובה העץ לכן: $O(\log_m n)$ .

כלומר שורש העץ מתרוקן גובה העץ קטן. כלומר בשורש אבר אחד רב הילדים התמזגו לצומת אחת. השורש הישן נעלם ומצומת המאוחד הוא השורש החדש.

| | מערך | מערך ממוין | רשימה | TAB |
|---|---|---|---|---|
| הכסה | $O(n)$ | $O(n)$ | $O(1)$ | $O(\log_m n)$ |
| הוצאה | $O(n)$ | $O(n)$ | $O(1)$ * * מצביע אחר | $O(\log_m n)$ |
| חיפוש | $O(n)$ | $O(\log n)$ | $O(n)$ | $O(\log_m n)$ |

## הוספה לעץ AVL:

כפי שלמדנו בקבוצת הפיבונאצ'י ומבאנו את הגובה הוא מאוזן ראשון ובכל צעד אחר (בעל יחיד או כפול). כל הפעולות נעשות בכך.

בעת הוספה, הגיעו לא מצא אחד עלולה להפיר את הגובה וכל הבנים מעל ל בעלי ל עורכים פה השורש, עם עצים מקסימלי. $O(\log n)$.

**עדכון גובה של AVL:** $O(n\log n)$. למה? עוברים n איברים ומבצעים אותם עדכון אחרי עה אחרי לעלי AVL.

אבל כפי אבל בלב אנחנו נחסך הרבה לרגעים - מעלון ל AVL מאוזן ומסה $O(\log n)$ אם הגלגל מעלה ייקח לפחות $O(\log n)$.

הבעיה במציאת הגובה הוא עם קבוע לוגן $O(n)$.

סה"כ $O(n\log n)$.

**עבור סריקת עץ (inorder):** $O(n)$. למה? הסבר כ inorder-in: תת עץ שמאל ← שורש ← תת עץ ימין.

נקבל כל אחד מ n הבנים (בפועל מקבלים על כל n הבנים) ומשים עליו פעולה של $O(1)$, לכן הכפלה.

סה"כ $O(n)$.

סה"כ עבור ריצה על אלגוריתם המלון: עדכון + סריקה: $O(n) + O(n\log n) = O(n\log n)$.

**שדה size:** size.x מייצג את מס הבנים הכולל מתחת הצומת שלו הוא x, כולל x עצמו.

$$x.size = x.left.size + x.right.size + 1$$

עדכון

עבור בלבד ריקן אנו: $size = 0$.

איך שומרים על עץ מאוזן? – בהכנסה | מחיקה מעדכנים את ה size של הצמתים **במסלול**. הבדלון לוקח $O(1)$ לכל צומת, והעומק הכי גרוע $\log n$ צמתים $(=h)$ ולכן $O(\log n)$.

– סיבוב לימין – עלה משנה את הסביבה **מקומי** של תת עץ. לא בהכרח משפיע מתחת עבור כל הדרך, אלא מעדכנים עדכון ב 2 הצמתים ששינו את מקומם בסיבוב.

עה $O(1)$ כי מדובר אותו חבש 1-2 צמתים עה סבים בודקים שר תת עצים הקיים (לא שינוי שם כבר) + 1 עבור האבא.

**אלגוריתם select:**

הפעולה מקבלת שורש עץ תת עץ x ומספר k ומחזירה את הבדבר שבמקום שלו נמצא במערך הוא בגודלו.

$Select(x, k)$

שורש תת עץ

גודל הוא בגודלו

נשתמש מלמעלה x ונדרש לבדוק את הגדלים הסמוכים שלו x בתנאים אלו:

נסמן ב r את המיקום של x: $r = x.left.size + 1$

עבור עבור עם האיברים שמאלה.

3 מקרים אפשריים: 1) $r == k$ הגענו אם שווים, זה האיבר שאנחנו מחפשים.

נחזיר את x.

2) $r > k$: האיבר שאנחנו מחפשים קטן יותר. לכן נמצא בצד שמאל של x.

נמשיך בריקורסיה $select(k, x.left)$.

3) $r < k$: האיבר שאנחנו מחפשים גדול יותר. נמצא בצד ימין של x.

נמשיך בריקורסיה $Select(x.right, k)$.

סיבוכיות זמן: המקרה הכי גרוע מתבטא כמה (גם לא מגמה-בלבד): $O(\log n)$

ירדנו מעל פעמים בעץ.

**אלגוריתם Rank** – מה המיקום הסדור של צומת בעץ?

מקבלת צומת סבסבית x כמערך T ומחזירה מיקום סדור – כמה איברים קטנים יותר שווים אליו. $Rank(T, x)$

1. כמה איברים קטנים מ x בתת עץ שלו? כל אלו שמשמאל. $x.left.size$ = rank.

2. בסיפוס דומה: עולים מ x לאבא שלו וסוכמים כל פעם שעולים מצומת כלשהי current ל אבא שלו - p, יש ב מקרים:

1) עלינו מצד שמאל לבן – הבא וגם את כל אלו שמשמאלו בנים מ x ולא חוקים, לכן נוסיף לסיים ל rank.

2) עלינו מצד ימין לאבא - p הבא וגם את כל שמאל קטנים מ x ונוסיף לסכומה: $rank = rank + p.left.size + 1$ לאבא.

3) נשמיטים נשמרים מסימים את הלולאה.

סיבוכיות זמן: כמו ב select שמגבים מ x ומעלה לשורש לאורך עמוד יחיד, אורך העמודים מתחלף הגבה וכן: $O(\log n)$.

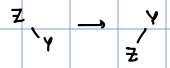
**תירגול מבוא למדעי המחשב בשפת JAVA :**

1. רוטציה יחידה:

כאשר הגלי "כבד" מדי בצד החיצוני. תיקון זה דומה לשני מקרים שונים (1,0).

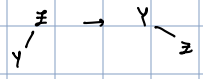
א. מערה RR- איך מציל? נתק את הבן שלו z ונפה בצורה מדויקת. הבן הימני Y אותו אין לו מקום מעניין (-1 | 0). הגלי נוצר בשורה חדשה מעניין. הקשר - רוטציה שמאלית.

Y עולה למעלה, Z יורד למטה והופך לבן שמאלי של Y.



ב. מערה LL- איך מציל? נתק את השמאלי של Z ונפה בצורה מדויקת. הבן השמאלי Y מאוזן אי נוצר שמאלה (0 | 1).

התיקון- רוטציה מעניין. מושכים את הבן השמאלי Y למעלה בזמן הבן הימני של Y.



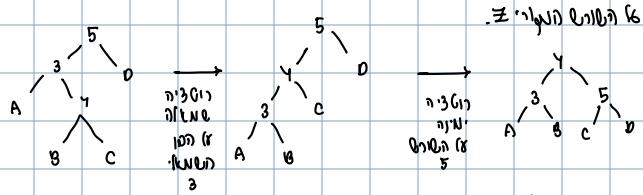
2. מקרים שנראים שונים רוטציה כפולה:

א. מערה RL- הנתק אלי השמאלי של Z בגובה מדי (+2) והבן השמאלי Y נוצר מעניין (-1).

יש לנו צ'ינצ- הלכו שמאלה ל Y ואז מעניין.

איך מטפלים? רוטציה שמאלה על הבן השמאלי Y . זה מישר את הצינצ ויצר מערה (LL) של LL.

רוטציה מעניין על השורש הנעלם Z.

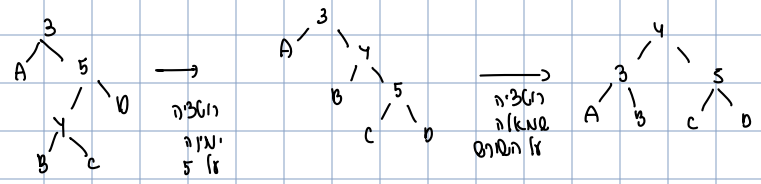



ב. מערה LR : איך מציל? נתק את הבן שלו z בגובה מדי (-2) והבן הימני שלו Y נוצר שמאלה (+1).

שוב צ'ינצ- הלכו מעניין ואז שמאלה.

התיקון - 1. מבצעים רוטציה יחידה על הבן הימני Y. מיישר לגרסה של RR.

2. תיקון של השורש - רוטציה שמאלה.



## נושא 8: ערימה בינארית ( Binary Heap ) וחור אדם:

**חוק אדם הראשון:** אל איבד ש "אינדקסים" שאינם מספר.

כשרבה זקוק איך, נקרוק את הוכרחים החשבות הבלהה ב-מע.

**יחס אב לבן בעץ בינארי:** התוכנה לאב מתחת) אינדקס $i$. אם בנות מס-ם נמצא במערך באינדקס $i$: הבן השמאלי שלו באינדקס $2i$, הבן הימני באינדקס $2i+1$.

**גובה בעץ בינארי כמעט מלא:** אלא שבכל הרמות מלאות, פרט אולי רמה התחתונה בה ס"ם כדלי ברמה התחתונה משמאל לימין.

נגדיר מ אלא בינארי כמעט שלם: $h=\log_2 n$.

**תכונות ערמה:** ערמת מקסימום: לכל בנות, האב של הבן גדול יותר שווה לערך שלו. האיבר המקסימלי בשורש.

מינימום מינימום: לכל בנות, האב קטן מהבנים שלו. האיבר המינימלי בשורש.

⊛ כמה בעת הרבה ביקרו אל מינים בעלי?

הפרטים אחרות מינימום מקסימום:

1. אלא כמעט מלא.

2. יחס ערכים בין אבא לבן.




**תחלבת יתפאל מינה:** כאשר שלנו הפרה בעקבות אחת ספציפית טעה ואנחנו רוצים לתקן אותו.

ראלו לעומת ספציפית $i$. תני מצביע הם מערכת תקינה לגמרי, הבעיה- הגמאדמנו $i$ מפר את הכלל - הוא מגדמו בקטן ממנו ( מקסימום מעמים, הוא גדול יותר מהבן).

**הפתרון הלוגי:** 1. השוואה- נשווה בין $i$ לבנים המתאים: אבא- $i$, בן שמאלי $2i$ ובן ימני $2i+1$ ובמצא מ-המעמים.

2. החלפה- אם האבא הוא גדול - הנשא תקין ולא נוגעים עוד.

אם אחד הבנים גדול, נחליף בינו לבין השמאלי- מעלים את הבנים המתאים. הבן הבנים עלה לתחתון האבא, והאבא הנמוך "נפל" בחזרה לתוך עמדתו.

הבעיה- ה- המצב הקלן כמו מקומה מאבא, אבל כל לחלק שכבר בנו מהבנים המתאים.

הפתרון- בדוגו לבדוק רקורסיבי אל אינדקס חדש ומשרך לרשת המלא את הדרך לאבא מה שבדוק מהבן או מעל לתחתית הנשא עלה.

עד כמה Heapify . מתי נשתמש בו? $i$ נניח מה מס. מערך סתמי ומבולבל שם לפסוק מקסימום חוקית. מה מקרה: $O(n)$. למה? בגלל

$$\text{Total work} = \sum \text{מספר הפעמים} \times \text{הגובה המקסימלי}$$

2. נשמתחבים את המקסימום, כלומר את השורש, ומפשר מחליפים מקומו.



כי שאר הבנים חייבים בעמדתם נקבע לגבי הבלבל ונפסם אל טלון.

נלכה בעקבות הנגדה- הבלבל הוא בגרף מ) אינדקס (מתחילת הערמה) ושבר בעלה מקסימום.

$$\text{מספר הפעמים} \le n\left(\frac{1}{1}+\frac{3}{8}+\frac{2}{4}+\frac{1}{2}+\dots\right) = O(n)$$

$$\frac{1}{2}\cdot 1+\frac{2}{4}\cdot 2+\frac{3}{8}\cdot 3+\dots$$




**סיבוכיות:** $O(\log(n))$ - כל שלב משווה עם הבן ומחליף אם צריך - הגובה הוא קבוע $O(1)$. בסך הכל יבוצע לכל היותר $\log(n)$ פעמים - לכן $O(\log(n))$.

## רשת 9: מבוא למיון ומיון הכסה:

מיון הכסה: האלגוריתם פה קלה: רוצים להכניס כלי אחד לעמדת כלים ממוינת.

עוברים על המערך ומעדכנים את העץ המתאים.

על מערך (בלתי מוגבל) מבנה כך:

1) מחלקים את המערך לב כלים והחלק שוא ממוין.

2) בוחרים הישר (מינימום) לפי. הלולאה הישנה כלה $i=1$ עד $n$.

3) רשומה כן הראבר קיבת (key) לא מוחבים במערך הממוין כי. להקים לעמדה קבן.

4) כי. עמדה לו מעלה הלוגריתם מצא (height) מהבלוגים הבנים מהעמדה מעמדת מיזה ומעמד אר ה עמד.

רעיון סיבוכיות עמ:

בעמדה הטוב: בעמדת ממוין בסדר עולה, לא נצטרך יותר מ לבד, לכל עדכון מה הראבים עדכון תלוי בהפוך עם הבן הראבים. סך הכל: $O(n)$.

בעמדה הגרוע: מעדכן ממוין בסדר יורד, לכל עדכון נצטרך לבד מה הראבים עדכון תלוי בהפוך עם הראבים. סך הכל: $O(n)$.




בעמדה הבינוני: מעדכן ממוין

(הפוך בסדר)

לאיבר בעמדה t ראשה $O(1)$ בממוין.

לאיבר בעמדה r ראשה $O(n)$ בהשוואה בעל-סדרה אחת.

לאיבר בעמדה בה לכל הכי יצא 2 בסדרים.

סכום חסכון: מ 1 עד $n-1$.

$= 1+2+\dots+n-1$

כי בעמדה בת יצא $n-1$ חסמנים.

$\underline{\underline{O(n^2)}}$

הערה הממוצע: מעדכן אבלי. ומבולגן. בממוצע נצטרך להזיז $\frac{n}{2}$ איברים שרשרת לפי יצא.

אבל $\Omega(n^2)$ רשמי $O(n^2)$ זה ממוצע הסיבוכיות.

## רשת 9: הוכחות בעזרת:

מבוא סמ: כמה כיתות מלא עם n מלים, כל-מים בדיוק $n-1$ בודקות שנמצא.

אופיון באינדוקציה. נניח $n$ על מס הלים:

$n=0$: ברור $n=1$ מלא הכפלה ב 0 שנה, אין כל בודקות שנמצא.

צעד: נניח נכון עבור עד $n-1$ עבור נראה מתי לא מלא.

יהי שלבים $n$ מלא. נסתכל על עלה בעומק הכי נמוך. מכוון שמלא, אין לו אחיו.

מכוון שנמצא מלא עמוק, כל שכניו לפי אבא ב בנים. אם מלא. אם לא מלא. אם נסתכל על עלה בעומק הכי נמוך. מכוון שמלא, אין לו אחיו.

נבפס $n-1 = 1+2-n$. זה. הנתון האינדוקטיבי, מסוים מלא לא $n-1$ מלים יש ב $n$ בודקות שנמצא.




★ רשת 9 יצר הגדים בהתרה, וכמה יש $n=1+2+1-n$ מלים ו $n-1=1+2-n$ $n-1$ בודקות שנמצא שבדרך.

נושא 10: בניית עץ AVL ממערך ממוין ב $O(n)$:

אם נרצה לבנות עץ AVL מ־n איברים שלא היה קיים מקודם, כל הכנסה תקח $O(\log n)$ ולסך n איברים $O(n \log n)$. ובדיוק גם המון מוצלחת לאלגן.

שנרד ל $O(n)$ כלפי, $\boxed{\text{Divide and Conquer אלגוריתם}}$:

המערך ממוין ואפשר ליצור שמאזן בזמן לינארי שמאזן בזמן חיפוש לכל השורשים כמו שרצינו לה גם נוסף.

האלגוריתם: ① לוקחים את האמצע כשורש והוא הרשתי.

② את חצי השמאלי של המערך שולחים בהנחה רקורסיבית כמו עץ שמאלי:

③ את חצי הימני של המערך שולחים בהנחה רקורסיבית כמו עץ ימני.

למה $O(n)$? אפשר לבקרים בכל איבר פעם אחת בלבד.

- מחלקים הזמנה $O(1)$ - נשים לו לבקר הזמנה.
- מחלקים פונים חדשה $O(1)$.
- שמרים בתוך מערך $\frac{n}{2}$.

$$T(n) = 2T(\frac{n}{2}) + O(1)$$

שיטת המסטר: $n = n^{\log_2 2}$

$O(1) < n$ מנרה 1 הסוגים מקרים.

ולכן $O(n)$

- ההבדל מאלגו הוא שלפני מאמצים היה לוקח $O(n)$ ולכן:

$$T(n) = 2T(\frac{n}{2}) + n$$
$$n = n^{\log_2 2}$$

ולכן $O(n\log(n))$.

מיזוג 2 עצי חיפוש בינרי:

נתונים 2 עצי חיפוש $T_1, T_2$ כך שכל האיברים ב $T_1$ קטנים מאלו בעומקים ב $T_2$.

נתון איבר נוסף $m$ כך ש $T_1 < m < T_2$. נרצה לעשות AVL אחד.

1) הערה הקל: $h(T_3) = h(T_1)$ אם נבדלים ב 1, אז יוצרים פונה חדשה $O(1)$ עם $m$ והסכים לא יוצר כתיב אלא שמאלי. סה"כ $O(1)$.

2) הערה מורכב: לעשות $T_3$ גבוה בהרבה מ $T_2$. אז לכן לתבר אם כי נקבל אלא לא שלם.

ספרלו: א. נתן) משוים $T_1$ ערך רק מאיזה סה גדולים כך ש $h(T_1) \le h(T_2) + 2$. $O(\log(n))$

ב. ננתק את תת העץ הימני שהיה מאיבר ריק ונחבר שורש $m$. $O(1)$

ג. לפי השמאלי של $m$ נחבר את מה שנמצא מצד. $O(1)$.

ד. לפי הימני של $m$ נחבר אם $T_2$. $O(1)$.

ה. בחזרה לשורש הבנים מתבנים בהיה ומבצעים תיקונים עם צריך. (מקסימום לעלה, ריטרן $O(1)$ בכל גיסום, סה"כ $O(\log(n))$.

$$O(\log(n)) + O(1) + O(1) + O(\log(n)) = O(\log(n))$$

1. המלכודת: למה לא צריך $O(\log n)$ רוטציות בהכנסה (Insert)?

כשאת עולה מהעלה חזרה למעלה לכיוון השורש, את באמת מעדכנת את הגבהים ובודקת את פקטורי האיזון (BF) של הצמתים לאורך המסלול (שזה לוקח $O(\log n)$ צעדים).

אבל, ברגע שאת פוגשת את הצומת הראשון **שיוצא מאיזון** (כלומר, ה-BF שלו הופך ל-2 או 2−), את מבצעת שם תיקון על ידי רוטציה (בודדת או כפולה).

הקסם של פעולת ההכנסה בעצי AVL הוא **שהרוטציה הזו מחזירה את תת-העץ בדיוק לגובה שהיה לו לפני שהכנסנו את האיבר החדש!** מכיוון שגובה תת-העץ חזר לקדמותו, אף אבא קדמון שלו (שנמצא גבוה יותר במסלול לכיוון השורש) לא יגביש שום שינוי בגובה של הבנים שלו. לכן, פקטורי האיזון של כל האבות למעלה יישארו תקינים לחלוטין!

**המסקנה:** בפעולת הכנסה (Insert), **נדרשת לכל היותר רוטציה אחת בלבד** (בודדת או כפולה) כדי לאזן את העץ כולו.

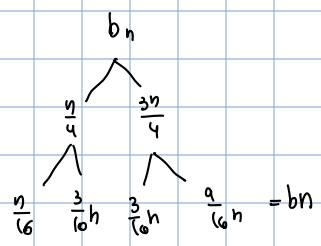
2. מתי הניתוח שלך כן נכון? בפעולת מחיקה (Delete)!

הסיבה שחשבת על $O(\log n)$ רוטציות היא הגיונית מאוד, כי זה בדיוק מה שקורה במחיקה.

במחיקת צומת, אם ביצעת רוטציה כדי לתקן צומת שיצא מאיזון, הרוטציה עלולה להקטין את גובה תת-העץ. ההקטנה הזו יכולה לגרום לאבא שלו לצאת מאיזון, ואז נצטרך רוטציה נוספת, וכן הלאה לאורך כל המסלול עד השורש. לכן במחיקה במקרה הגרוע נבצע $O(\log n)$ רוטציות.

סיכום התשובה הפורמלית למבחן עבור הכנסה (Insert):

- זמן ריצה במקרה הגרוע: $O(\log n)$ (חיפוש המקום למטה, ועדכון גבהים לאורך המסלול למעלה).
- מספר רוטציות מקסימלי: 1 (רוטציה בודדת או RR/LL או רוטציה כפולה RL/LR בהחלט, ולאחר מכן העץ כולו מאוזן).
- סיבוכיות זמן הריצה הכוללת: $O(\log n) + O(1) = O(\log n)$.


$bn$
$\frac{n}{4}$
$\frac{3n}{4}$
$\frac{n}{16}$
$\frac{3}{16}n$
$\frac{3}{16}n$
$\frac{9}{16}n$
$= bn$