# אינדקסים

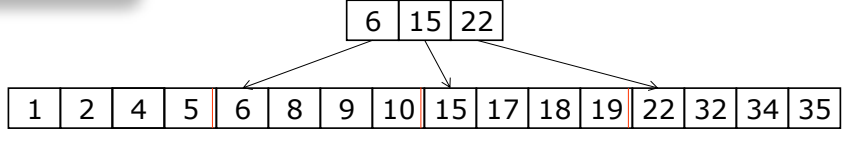


## חיפוש רשומה

נתון קובץ ממוין בעל N רשומות ורוצים למצוא את הרשומה בעלת מפתח K

- פתרון נאיבי: סריקה סדרתית
  - ידרוש בממוצע קריאת N/2 רשומות, כלומר הבאת N/2b בלוקים
- חלוקת הקובץ לשני חלקים ושמירת הערכים המפרידים:
  - ידרוש בממוצע קריאת N/4 רשומות, כלומר N/4b בלוקים
- חלוקת הקובץ ל-m חלקים ושמירת הערכים המפרידים:
  - ידרוש בממוצע קריאת N/2m רשומות, כלומר N/2mb בלוקים

החלוקה הזו היא אינדקס לקובץ



# חיפוש רשומה

נתון קובץ ממוין בעל N רשומות ורוצים למצוא את הרשומה בעלת מפתח K

- חלוקת הקובץ ל-m חלקים ושמירת הערכים המפרידים:
  - ידרוש בממוצע קריאת N/2m רשומות, כלומר הבאת N/2mb בלוקים

- אבל אם m מאוד גדול אז הוא לא נכנס בזיכרון ויש למצוא את הרשומה בעלת מפתח K בקובץ ממוין בעל m רשומות
- **חזרנו לאותה בעיה בקנה מידה יותר קטן:**
  - **נוכל להמשיך רקורסיבית עד ש-m יהיה קטן מספיק ← ייתן מבנה של עץ**
  - **נוכל לבחור את m בחכמה (כמה קטן?)**

- מה נעשה עבור קובץ לא ממוין?



# אינדקס

אינדקס הוא מבנה חיפוש אשר לכל ערך מפתח, מצביע לרשומה (רשומות) עם מפתח זה

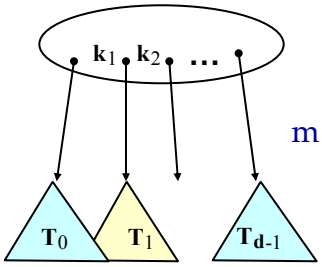
- מאפשר גישה ישירה לרשומות בקובץ מובנה (יחס) גדול
- מקל על חישוב פעולות כמו בחירה, צירוף, חיתוך ועוד

למשל, חישוב $R \cap S$, אם יש אינדקס על R:
לכל רשומה ב-S נבדוק באינדקס אם היא נמצאת ב-R
- לפחות |S| גישות לאינדקס של R ולרשומות ב R (לאו דווקא בבלוקים)
- אם S ממוין, אז ייתכן ששתי קריאות עוקבות תתייחסנה לאותו בלוק של R, ומספר הגישות יהיה קטן יותר

ישנם סוגים שונים של אינדקסים: עץ חיפוש (בינארי ו/או מאוזן), טבלת ערבול, ...

עבור קובץ גדול מאוד, גם האינדקס גדול מאוד וכולו או חלקו, **נשמר בדיסק**
צריך לקחת בחשבון את עלות הקריאה מהדיסק (ולעבוד בבלוקים)

# עצי B (עם פרמטרים $m \geq 3$, b)

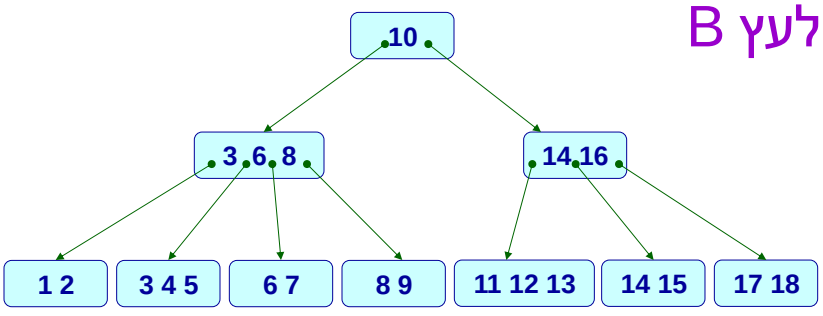



- ✓**לשורש** דרגה בין 2 ו-m.
- ✓**לצומת פנימי** (שאינו שורש) דרגה בין $\lceil m/2 \rceil$ ו- m
- ✓**כל העלים**: באותו מרחק מהשורש
- ✓**הרשומות** נמצאות רק בעלים,
  מספר הרשומות בעלה בין $\lceil b/2 \rceil$ ו- b
- ✓**מפתחות**: בצומת פנימי עם d ילדים יש d−1 מפתחות,
  כך שהמפתח ה-i מקיים: $\max\{x \in T_{i-1}\} < k_i \leq \min\{x \in T_i\}$

**גובה העץ**: מספר הרמות לא כולל רמת העלים



# דוגמה לעץ B

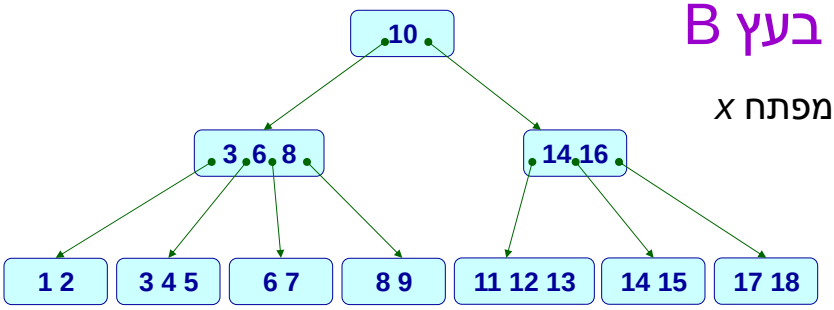



עם b=3 ,m=5

האם זהו עץ B עבור m = 3 ? m = 4 ? m = 7?

# חיפוש בעץ B

מחפשים מפתח $x$



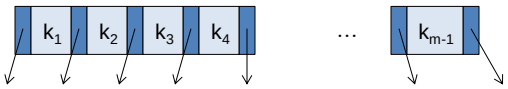

1. $v$ מצביע לצומת, מאותחל לשורש
2. כל עוד $v$ אינו מצביע לעלה,
   - מצא את $i$ שעבורו $v.k_i \le x < v.k_{i+1}$
   - $v := root\ of\ T_i$ (הילד ה-$i$ של $v$)
3. חפש את $x$ בעלה $v$



# גודל הצמתים הפנימיים והעלים

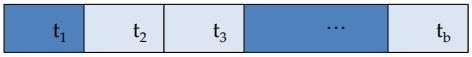
אם גודל מפתח הוא $k$ וגודל מצביע הוא $p$, אז גודל צומת $s_1$ הוא

$$m\,p + (m-1)\,k = s_1$$



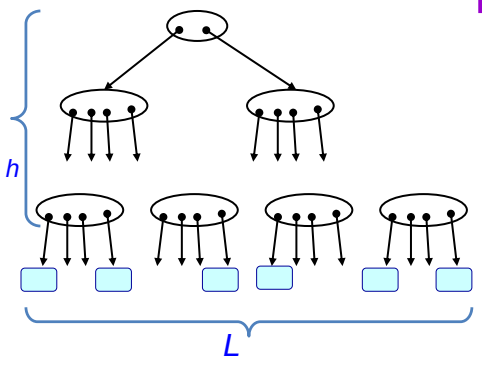

אם גודל רשומה הוא $t$, אז גודל עלה $s_2$ הוא

$$b\,t = s_2$$




הגדלת $b$, $m$ מגדילה את הצמתים / עלים ומקטינה את גובה העץ
(בדרך כלל, $b$, $m$ הם בין 100 ל-200)

# גובה מינימאלי של עץ B




| מקרא | |
|---|---|
| מספר עלים | $L$ |
| מספר עלים מינימלי | $L_{min}$ |
| מספר עלים מקסימלי | $L_{max}$ |
| גובה | $h$ |
| גובה מינימלי | $h_{min}$ |
| גובה מקסימלי | $h_{max}$ |

מספר העלים בעץ בגובה $h$

לפחות $L_{\min} = 2\lceil m/2 \rceil^{h-1}$

לכל היותר $L_{\max} = m^h$

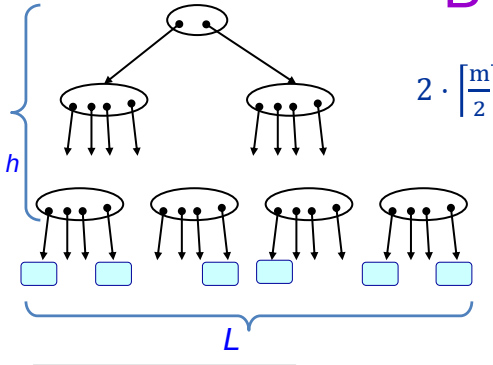
אם בעץ יש L עלים

$$m^{h_{\min}-1} < L \leq m^{h_{\min}}$$

ולכן $$h_{\min} = \lceil \log_m L \rceil = \left\lceil \frac{\log_2 L}{\log_2 m} \right\rceil$$



# גובה מקסימאלי של עץ B




| מקרא | |
|---|---|
| מספר עלים | $L$ |
| מספר עלים מינימלי | $L_{min}$ |
| מספר עלים מקסימלי | $L_{max}$ |
| גובה | $h$ |
| גובה מינימלי | $h_{min}$ |
| גובה מקסימלי | $h_{max}$ |

הגובה המקסימאלי מקיים: $2 \cdot \left\lceil \frac{m}{2} \right\rceil^{h_{max}-1} \leq L$

אם L ו- m גדולים

$$h_{max} \leq 1 + \frac{\log_2 L - 1}{\log_2 m - 1} \approx 1 + \frac{\log_2 L}{\log_2 m}$$

מאחר ו-$h_{max}$ מספר שלם:

$$h_{max} \leq 1 + \left\lceil \frac{\log_2 L}{\log_2 m} \right\rceil = 1 + h_{min}$$

$$h_{max} - h_{min} \leq 1 \Leftarrow$$

# גובה ממוצע

אם מתחילים בעץ ריק, ומבצעים רק הכנסות (בלי הוצאות)
וכל פרמוטציות הקלט שוות הסתברות
אפשר להוכיח שצפיפות המפתחות, בעלים ובצמתים הפנימיים:

E(number of keys / maximal number of keys) = ln 2 ≅ 0.69

עבור $N$ מפתחות, מספר העלים הצפוי הוא $E(L) = \frac{N}{0.69b}$

$$E(h) = \log_{0.69m} L = \log_{0.69m} \frac{N}{0.69b} = \frac{\log_2 N - \log_2 0.69b}{\log_2 0.69m}$$

$$= \frac{\log_2 N - \log_2 0.69 - \log_2 b}{\log_2 0.69 + \log_2 m} \cong \frac{0.54 + \log_2 N - \log_2 b}{-0.54 + \log_2 m}$$

אם $m$ גדול

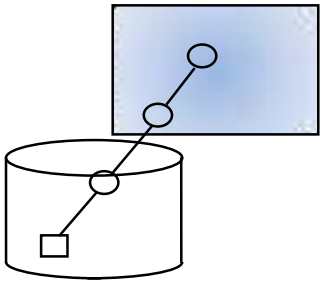
| **מקרא** | |
|---|---|
| מספר עלים | $L$ |
| גובה | $h$ |



# דוגמה מספרית

קובץ בגודל 10 GB, עלים בגודל 75 KB (כל אחד)
m=200
מספר העלים הצפוי הוא $L = 10\ GB / (75\ KB \times 0.69) = 201{,}650$
מספר הבנים הצפוי לכל צומת פנימי $m \times 0.69 = 138$
מספר הצמתים בכל רמה:

$v_1 = L / (m \times 0.69) = 1462$
$v_2 = v_1 / (m \times 0.69) = 10$
שורש העץ: $v_3 = 1$



זאת אומרת שגובה העץ הוא 3 וניתן לשמור את
השורש ואת ילדיו בזיכרון הראשי (באופן קבוע)

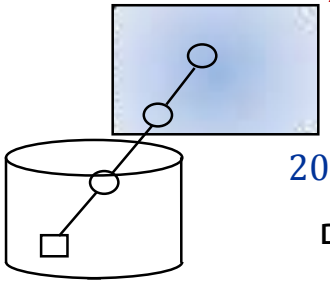
# המשך הדוגמה: זמן החיפוש בעץ-B

מעבר מהשורש אל העלה המתאים

- עץ בגובה 3: השורש וילדיו בזיכרון הראשי, העלים והוריהם על הדיסק
- דרגת העץ m = 200

גודל מסילה: 225 KB

גודל עלה: 75 KB $\Leftarrow$ במסילה יש $\frac{225KB}{75KB} = 3$ עלים



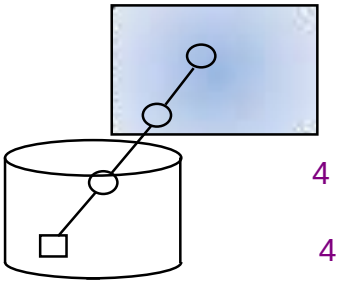
גודל מפתח 20B, גודל מצביע 10B

$\Leftarrow$ גודל צומת פנימי

$200 \times 10B + 199 \times 20B = 5980B \cong 6KB$

$\Leftarrow$ במסילה יש $\left\lfloor \frac{225KB}{6KB} \right\rfloor = 37$ צמתים פנימיים





# המשך הדוגמה: זמן החיפוש בעץ-B

| | |
|---|---|
| <u>זמן קריאת צומת פנימי</u> | |
| זמן העברת צומת פנימי | 4 ms/37 = 0.11 ms |
| זמן תזוזה (ממוצע) | 3.4 ms |
| השהיית סיבוב (ממוצעת) | 4 ms/2 = 2 ms |
| סה"כ (בערך) | 5.51 ms |
| <u>זמן קריאת עלה</u> | |
| זמן העברת עלה | 4 ms/3 = 1.33 ms |
| זמן תזוזה (ממוצע) | 3.4 ms |
| השהיית סיבוב (ממוצעת) | 2 ms |
| סה"כ (בערך) | 6.74 ms |
| <u>זמן קריאה כולל</u> | 5.51 ms + 6.74 ms = 12.25 ms |

# הכנסה לעץ B

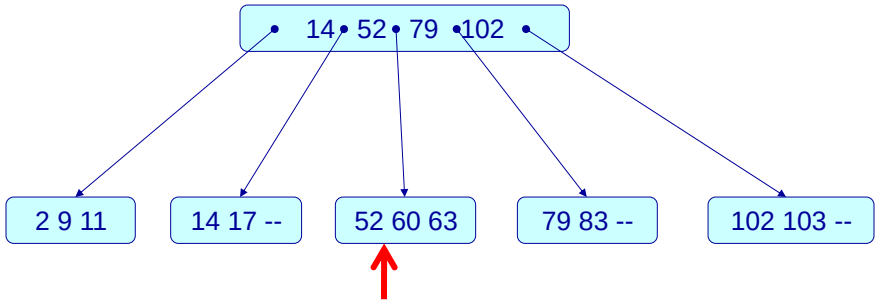
1. מצא את העלה אליו צריך להוסיף
2. אם יש בו פחות מ $b$ רשומות, הוסף את הרשומה
3. אחרת, יש לו b+1 רשומות, וצריך לפצל:
   - א. צור עלה חדש, והעבר אליו $1 + \lfloor b/2 \rfloor$ רשומות
   - ב. $\lceil b/2 \rceil$ רשומות נשארות בעלה הישן
   - ג. הוסף מפתח להורה
4. אם יש גלישה בצומת פנימי (יותר מ-m ילדים): פצל והוסף מפתח להורה
5. אם יש גלישה בשורש: פצל אותו לשנים וצור שורש חדש



# דוגמה להכנסה

הכנס רשומה עם מפתח 56

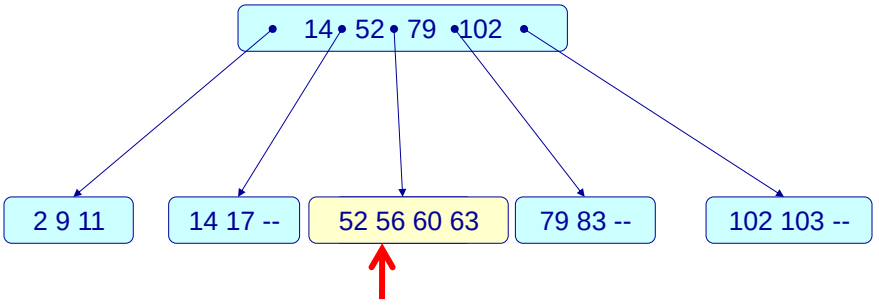
m = 5 , b = 3



# דוגמה להכנסה

הכנס רשומה עם מפתח 56
צריך לפצל

m = 5 , b = 3

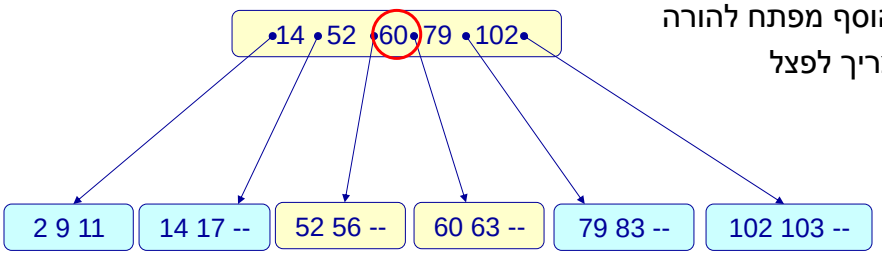





# דוגמה להכנסה

הכנס רשומה עם מפתח 56
צריך לפצל
הוסף מפתח להורה
צריך לפצל

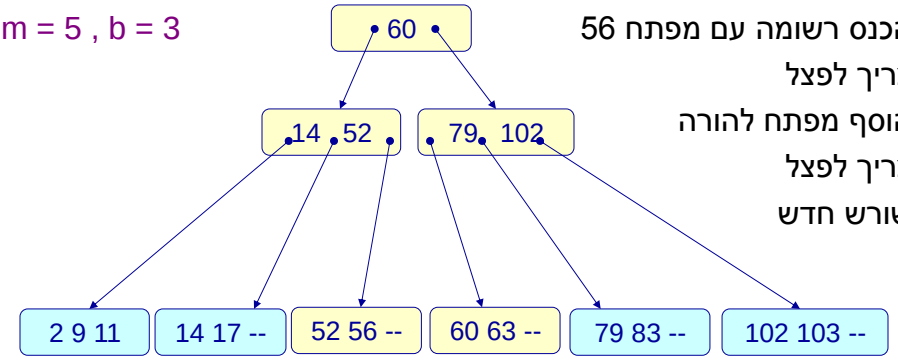
m = 5 , b = 3



# דוגמה להכנסה

m = 5 , b = 3

הכנס רשומה עם מפתח 56
צריך לפצל
הוסף מפתח להורה
צריך לפצל
שורש חדש






# זמן ההכנסה

אם אין פיצול, זמן ההכנסה הוא $T_{find}$+rotation = 12.25*ms*+4*ms* = 16.25*ms*

מה ההסתברות לפיצול בסדרה של הכנסות? אם נוסיף $N$ רשומות לעץ ריק

במקרה הגרוע: מספר העלים הוא $N/(b/2) = 2N/b$

⇦ מספר הפיצולים לכל היותר $2N/b$.

בממוצע: מספר העלים הצפוי הוא $L=N/(b \ln 2) = 1.44N/b$ וזה מספר הפיצולים הצפוי.

ההסתברות שבזמן הכנסה יקרה פיצול = $L/N = 1.44/b$
לכן, עבור $b = 12$ הסתברות פיצול = 12%
ועבור $b = 50$ הסתברות פיצול = 3%

כיוון שזמן הפיצול דומה לזמן הכתיבה, ניתן להתעלם מזמן הפיצול (אם $b$ גדול)

# הוצאת רשומה

1. חפש את הרשומה.
2. אם סילוקה לא יקטין את מספר הרשומות בעלה מתחת ל-$\lceil b/2 \rceil$ מחק את הרשומה וסיים.

   אחרת:
   - אם לעלה של הרשומה יש אח שכן עם $\geq \lceil b/2 \rceil + 1$ רשומות: העבר רשומות, עדכן את הצומת, את האח ואת המפתח המפריד אצל ההורה
   - אחרת: אַחֵד את שני האחים, וסלק את המפתח המפריד מההורה, תוך שימוש באלגוריתם להוצאת רשומה מעלה
3. אם לשורש נותר ילד יחיד: מחק את השורש, הילד הופך לשורש החדש.

כאשר מאחדים אחים: מספר הרשומות $\left\lceil \frac{b}{2} \right\rceil + \left( \left\lceil \frac{b}{2} \right\rceil - 1 \right) \leq \frac{b+1}{2} + \frac{b+1}{2} - 1 = b$

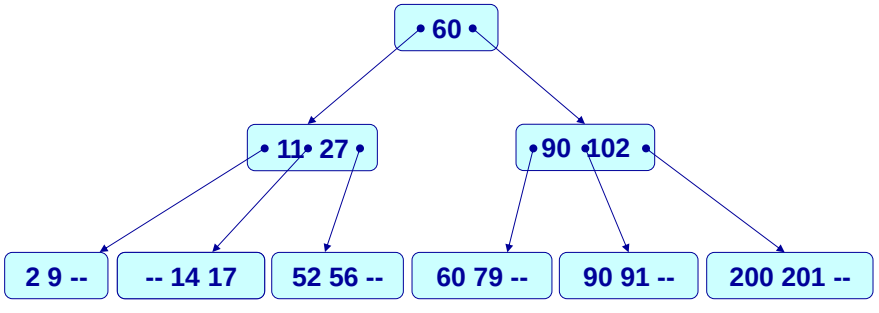
**זמן הוצאה:** דומה לזמן ההכנסה



# דוגמת הוצאה

הוצא את 11

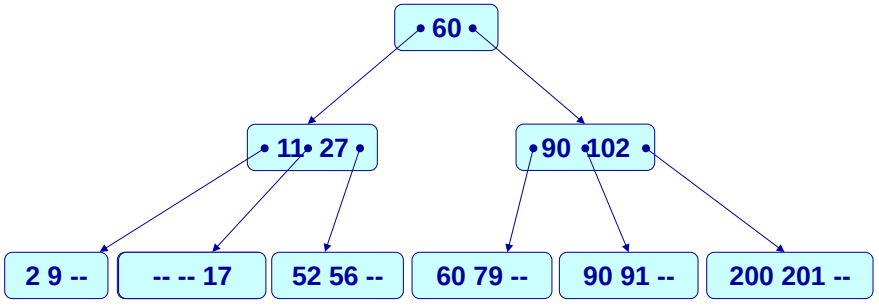
m = 5 , b = 3



# דוגמת הוצאה

הוצא את 14

m = 5 , b = 3

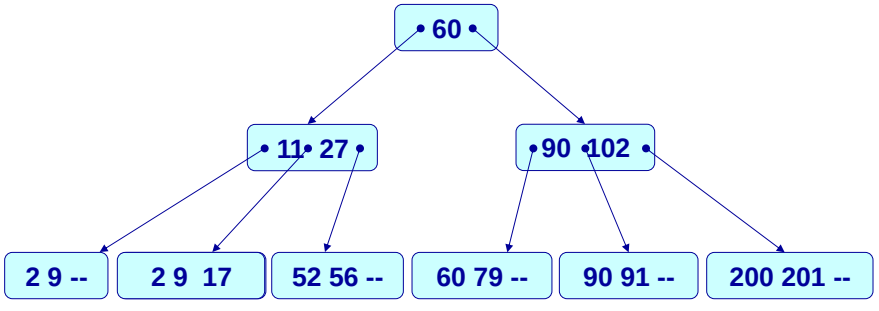





# דוגמת הוצאה

הוצא את 14: איחוד

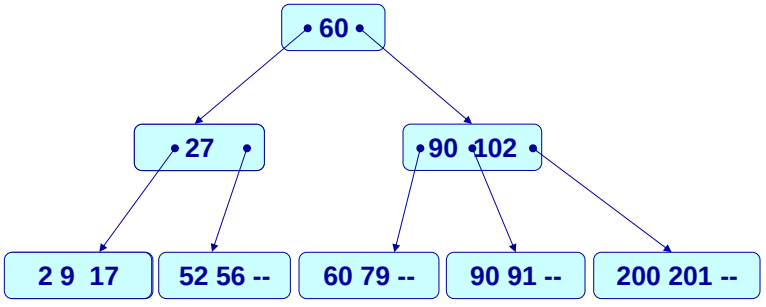
m = 5 , b = 3



# דוגמת הוצאה

הוצא את 14: הוצא מפתח ברמה מעל

m = 5 , b = 3

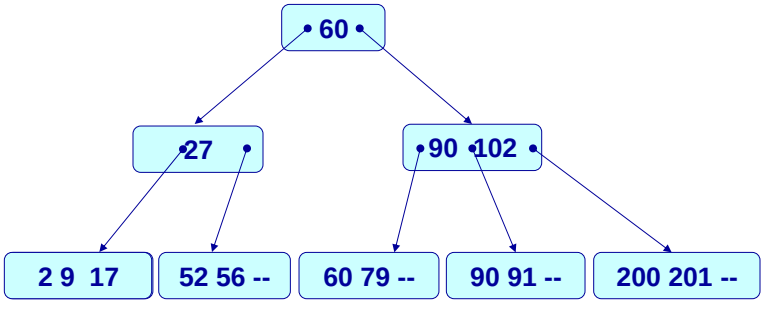





# דוגמת הוצאה

הוצא את 14: הסר את השורש

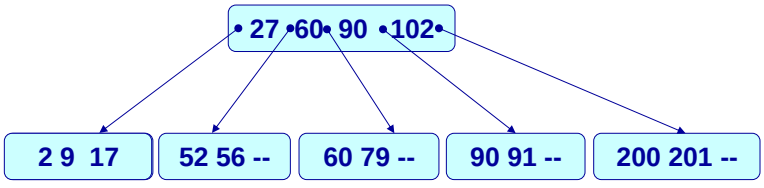
m = 5 , b = 3



# דוגמת הוצאה

הוצא את 14: הסר את השורש

m = 5 , b = 3

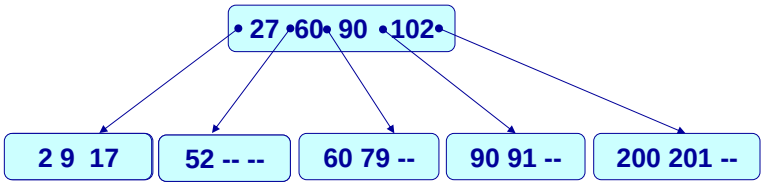





# דוגמת הוצאה

הוצא את 56

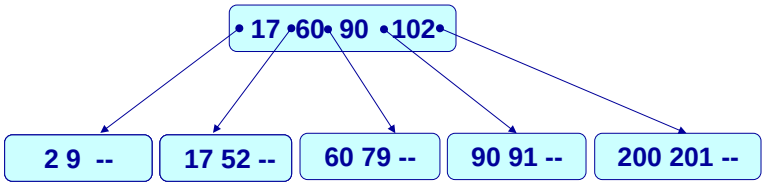
m = 5 , b = 3



# דוגמת הוצאה

הוצא את 56 ואזן עלים

m = 5 , b = 3






# מעבר סדרתי על עץ-B

נעריך את הזמן לקריאת כל העלים של קובץ בגודל 10 GB

בכל עלה 75 KB × ln 2 ≅ 52 KB

מספר העלים 10 GB / 52 KB = 201,650

זמן קריאת עלה $T_{leaf}$ = 6.74 ms

סה"כ: 6.74ms × 201,650 = 1359.121sec = 22min 39sec

- למה בחישוב זמן ההעברה של עלה הנחנו שהוא מלא, למרות שבממוצע הוא רק ln 2 מלא?
- מה היחס בין הזמן למעבר הסדרתי על עץ B לבין זמן העיבוד של קובץ סדרתי רציף?

# אם אין מקום לסבים בזיכרון ראשי?

אפשר להחזיק 3 רמות בזיכרון המשני

זמן חיפוש:

$T_1 = 2 \times T_{internal} + T_{leaf}$

בדוגמה שלנו

$T_1 = 2 \times 5.51 + 6.74 \text{ ms} = 17.76 \text{ ms}$



# אם אין מקום לסבים בזיכרון ראשי? פתרון 2

נגדיל את דרגת הצמתים הפנימיים: נגדיל צמתים פנימיים ל- 75 KB

אם גודל מפתח + מצביע הוא 30B

דרגת העץ תהיה: $m = 75 \text{ KB} / 30\text{B} = 2560$

מספר ילדים ממוצע: $(\ln 2)m \cong 0.69 \times 2560 = 1766$

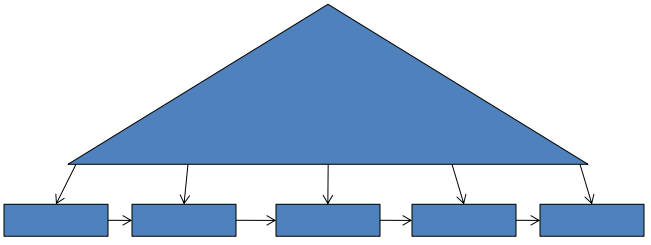
אם בקובץ 201,650 עלים, מספר האבות $201{,}650 / 1766 = 114.2 \cong 115$

כל האבות הם בנים של השורש, ואותו אפשר להחזיק בזיכרון הראשי!

זמן החיפוש: $T_2 = 2 \times T_{leaf} = 2 \times 6.74 \text{ ms} = 13.48 \text{ ms}$

# שרשור של העלים

לעיתים, מוסיפים מצביעים: מכל עלה לעלה הבא אחריו
מייעל קריאה של כל רשומות הקובץ באופן סדרתי על פי המפתח







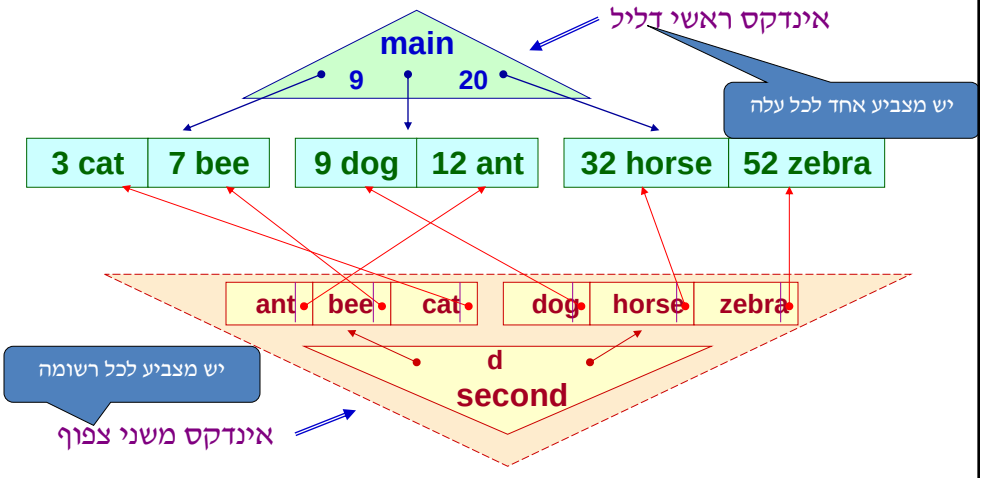
# רשומות עם כמה מפתחות

נרצה לחפש רשומה גם לפי מס' ת"ז וגם לפי שם
- מפתח ראשי (מס' ת"ז)
- מפתח משני (שם)

**אינדקס ראשי**: הרשומות יסודרו לפי המפתח הראשי, למשל בעץ B
**אינדקס משני**: מבנה חיפוש נוסף לפי המפתח המשני

- לכל רשומה מקורית נבנה מיני-רשומה, המכילה את המפתח המשני, והפניה אל הרשומה המקורית במבנה הראשי
- את המיני-רשומות נארגן בקובץ במבנה כלשהו, למשל עץ B

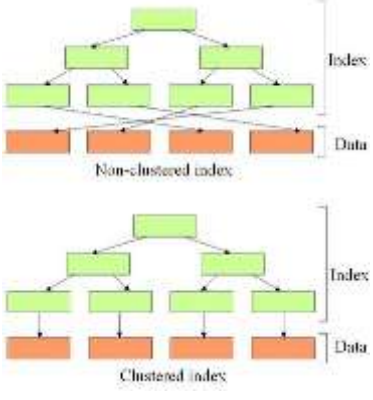
# רשומות עם כמה מפתחות: דוגמה






# אינדקס מקובץ ואינדקס לא מקובץ

- אינדקס מקובץ (clustered): סדר הרשומות בקובץ תואם את סדר המפתחות באינדקס
  - זוג מפתחות סמוכים באינדקס צפויים להימצא באותו בלוק או בבלוקים סמוכים בדיסק
  - לעיתים קרובות אינדקס ראשי הוא אינדקס מקובץ
  - טוב לחיפושי טווח אבל מגדיל את העלות על עדכונים
- אינדקס לא מקובץ (non-clustered): סדר הרשומות **לא** תואם את סדר המפתחות
  - זוג מפתחות סמוכים באינדקס **לא** צפויים להימצא באותו בלוק או בבלוקים סמוכים בדיסק
  - אינדקס משני לרוב יהיה לא מקובץ



# אינדקס משני נעוץ (pinned)

במיני-רשומה (באינדקס המשני) יש מצביע לעלה שבו הרשומה המקורית

חיפוש (לפי מפתח משני):

- חפש באינדקס המשני, קבל כתובת של עלה
- קרא את העלה וחפש את הרשומה בו

הכנסה:

- הכנס את הרשומה החדשה בעזרת האינדקס הראשי לעלה L
- הוסף לאינדקס המשני מיני-רשומה המכילה מפתח משני ומצביע ל-L
- אם העלה של האינדקס הראשי פוצל, עדכן את כל המפתחות באינדקס המשני שהצביעו לרשומות שנדדו לעלה אחר.

האם לשמור מצביע לכתובת (עלה ומספר רשומה בעלה) של רשומה?



# אינדקס משני בלתי-נעוץ (unpinned)

במיני-רשומה (באינדקס המשני) יש את המפתח הראשי של הרשומה המקורית (המפתח הראשי צריך להיות ייחודי)

חיפוש (לפי מפתח משני):

- חפש באינדקס המשני, קבל מפתח ראשי k
- חפש את k באינדקס הראשי

הכנסה:

- הכנס את הרשומה החדשה בעזרת האינדקס הראשי
- הוסף לאינדקס המשני מיני-רשומה המכילה מפתח משני ומפתח ראשי

# השוואה בין ארגון נעוץ ובלתי-נעוץ

**חסרון של אינדקס נעוץ:**

כשמעבירים רשומות ממקום למקום בקובץ
(למשל באיחוד או פיצול של עץ B)
יש לעדכן את המיני-רשומות הרלוונטיות בכל האינדקסים המשניים

- יתכן שיש הרבה אינדקסים משניים (חלקם אפילו זמניים)

**חסרון של אינדקס בלתי-נעוץ:**

זמן החיפוש כפול, כי מתוך האינדקס המשני מקבלים את המפתח הראשי
ולא את מיקום הרשומה



# אינדקס משני משולב נעוץ / לא-נעוץ

במיני-רשומה שבאינדקס המשני שומרים את המפתח הראשי של הרשומה
המקורית ומצביע למיקום (משוער) של הרשומה המקורית

חיפוש (לפי מפתח משני):

- חפש באינדקס המשני, קבל כתובת של עלה ומפתח ראשי k
- קרא את העלה וחפש את הרשומה בו
- אם לא נמצאה – חפש את k דרך האינדקס הראשי
  עדכן את כתובת העלה באינדקס המשני

הכנסה:

- הכנס את הרשומה החדשה בעזרת האינדקס הראשי לעלה L
- הוסף לאינדקס המשני מיני-רשומה המכילה מפתח משני, מפתח ראשי
  מצביע ל-L

# יתרונות הפתרון המשולב

- לא מבזבזים זמן מיותר לעדכן את כל האינדקסים המשניים, כאשר מזיזים רשומה
- בחיפוש רשומה, מחיר כפול (או קצת יותר) רק בפעם הראשונה שניגשים אליה דרך האינדקס המשני לאחר העברתה
- אם יש כמה רשומות עם אותו מפתח משני ניתן לעתים לזהות את הרשומה המבוקשת ע"פ המפתח הראשי מבלי לבדוק את כל הרשומות האפשריות



# דוגמה מספרית

נתון: קובץ עם $10^6$ רשומות
כל רשומה 150 B
גודל הקובץ 150 MB
גודל עלה 75 KB

| | |
|---|---|
| מספר מקסימאלי של רשומות בעלה | 75 KB / 150 B = 512 |
| מספר צפוי של רשומות בעלה | (ln 2) 512 = 355 |
| מספר צפוי של עלים | $10^6$/355 = 2817 |
| זמן קריאת עלה (חישבנו) | 6.74 ms |

מעבר סדרתי לפי אינדקס ראשי: הגורם הדומיננטי הוא זמן קריאת העלים
2817 × 6.74 = 18,987 ms ≈ 19 sec

# מעבר סדרתי לפי מפתח משני

ארגון לפי מצביעים לעלים:

לכל רשומה נבנה מיני-רשומה שבה מפתח 20 B + מצביע 10 B = 30 B

- מספר מקסימאלי של מיני-רשומות בעלה: 75KB/30 B = 2,560
- מספר צפוי של מיני-רשומות בעלה: (ln 2) 2,560 = 1,775
- מספר צפוי של עלי האינדקס המשני: $10^6$/1775 = 564

מספר זה זניח לעומת מספר הרשומות!

⇐ הזמן הדומיננטי הוא זמן קריאת הרשומות עצמן!



# זמן קריאת קובץ לפי אינדקס משני

- pinned (האינדקס המשני מכיל מצביעים לעלים):

  מעבר על כל הקובץ רשומה אחר רשומה (סדרת הקריאות בדיסק היא אקראית): קריאת עלה (חישבנו) 6.74 msec

  $10^6$ × 6.74 ms = 6,740 sec ≈ 1 hour 52 min 20 sec

- unpinned (האינדקס המשני מכיל מפתחות של האינדקס הראשי):

  יש צורך לבצע חיפוש לכל רשומה:

  חיפוש רשומה (חישבנו) 12.25 ms

  $10^6$ × 12.25 ms = 12,250 sec ≈ 3 hour 24 min 10 sec

**האם יש דרך יעילה יותר למעבר סדרתי לפי מפתח משני?**

# שאילתות טווח

**בהינתן** $x \le y$ , מצא את כל הרשומות בין x ל- y

דוגמה: עבור $\sigma_C$ כאשר C(x) הוא $a \le x \le b$
נקבל שאילתת טווח.

שאילתות טווח על אינדקס דליל יותר יעילות:

**נניח** k רשומות נמצאות בטווח, ובעלה יש (בממוצע) r רשומות

- אם השאילתה למפתח ראשי נעבור על k/r עלים
- אם השאילתה למפתח משני נעבור על k עלים



# צירוף $R(Y,Z) \bowtie S(X,Y)$ באמצעות אינדקס

יש אינדקס צפוף של S עבור Y
לכל רשומה $r \in R$
חפש את r.Y ב-S
עבור כל רשומה $s \in S$ שעבורה s.Y = r.Y
הוצא את $r \bowtie s$ לפלט

– קריאת R: B(R)

– בממוצע לכל ערך של r.Y יהיו T(S)/V(S,Y) רשומות של S
(כאשר V(S,Y) הוא מספר הערכים השונים של שדות Y ב-S)

– סה"כ: T(R) T(S)/V(S,Y)
(כי הגישות ל S אינן לפי בלוקים)