

עצי 2-3 ועצי דרגות

חומר קריאה לשיעור זה

Chapter 19: B trees (381 - 397)

Chapter 15: Augmenting data structures (281 - 290)

עצים מאוזנים

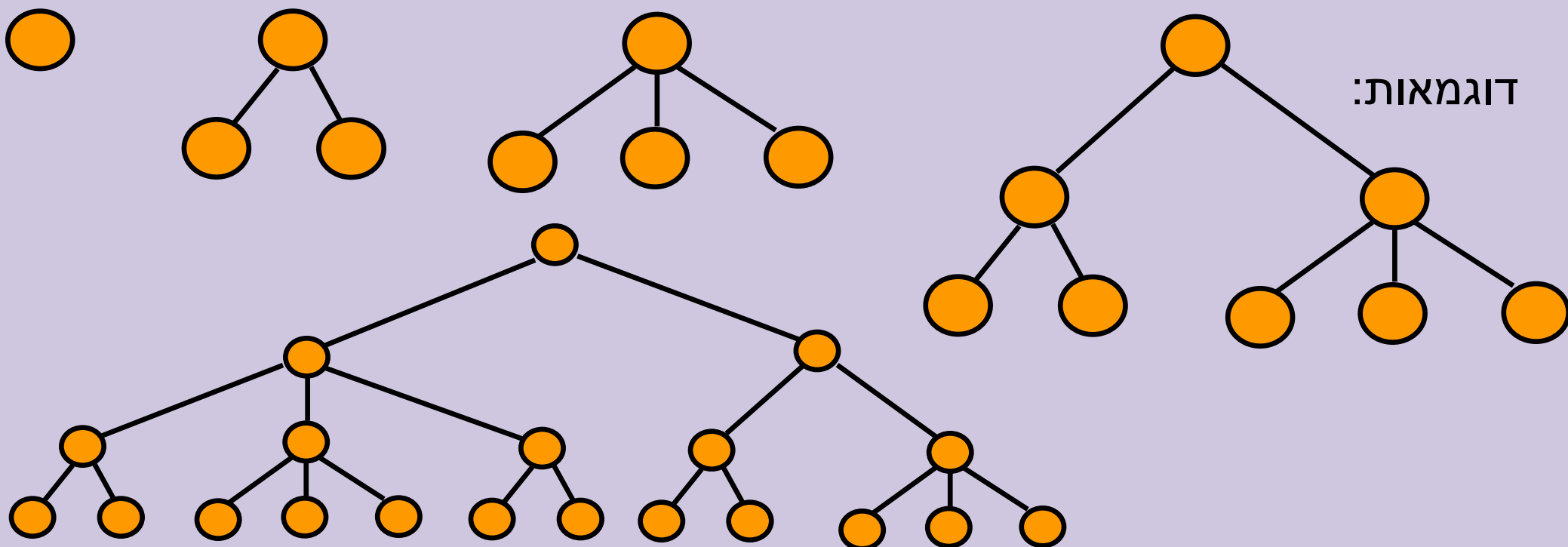
תזכורת: משפחת עצים נקראת מאוזנת אם $h(T) = O(\log n)$.

עצי AVL הם עצים מאוזנים. עצי 2-3 מהווים דוגמא נוספת לעצים מאוזנים.

הגדרה: עץ 2-3 הוא עץ בו

1- כל העלים נמצאים באותה רמה

2- לכל צומת פנימי 2 או 3 בנים.

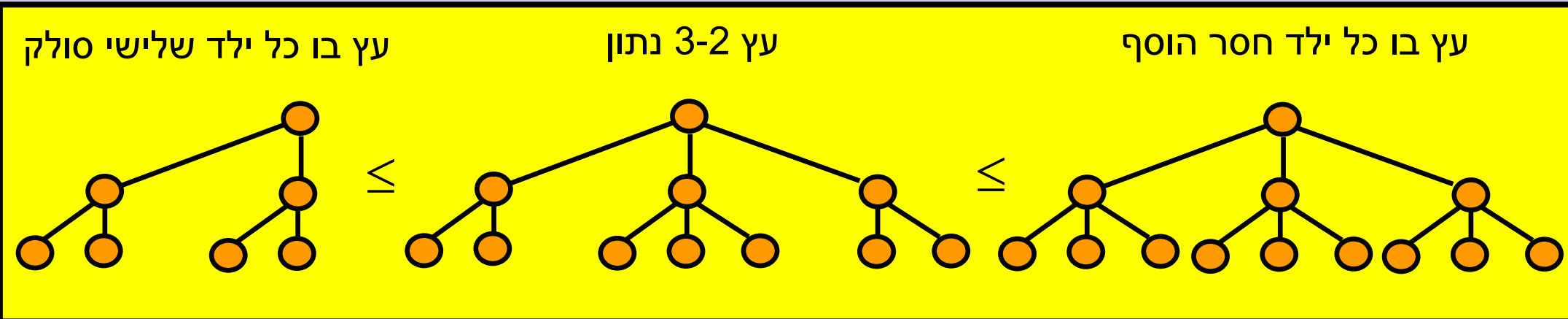


גובה עצי 3-2

מספר העלים L בעץ 3-2 מקיים $2^h \leq n \leq 3^h$ כאשר h הוא גובה העץ.

נימוק: עבור חסם תחתון נבחן את העץ הבינרי השלם הנוצר ע"י סילוק כל ילד שלישי מעץ 3-2.

עבור חסם עליון נבחן את העץ הטרינרי השלם הנוצר ע"י הוספת ילד שלישי לעץ 3-2 בכל מקום בו חסר ילד.



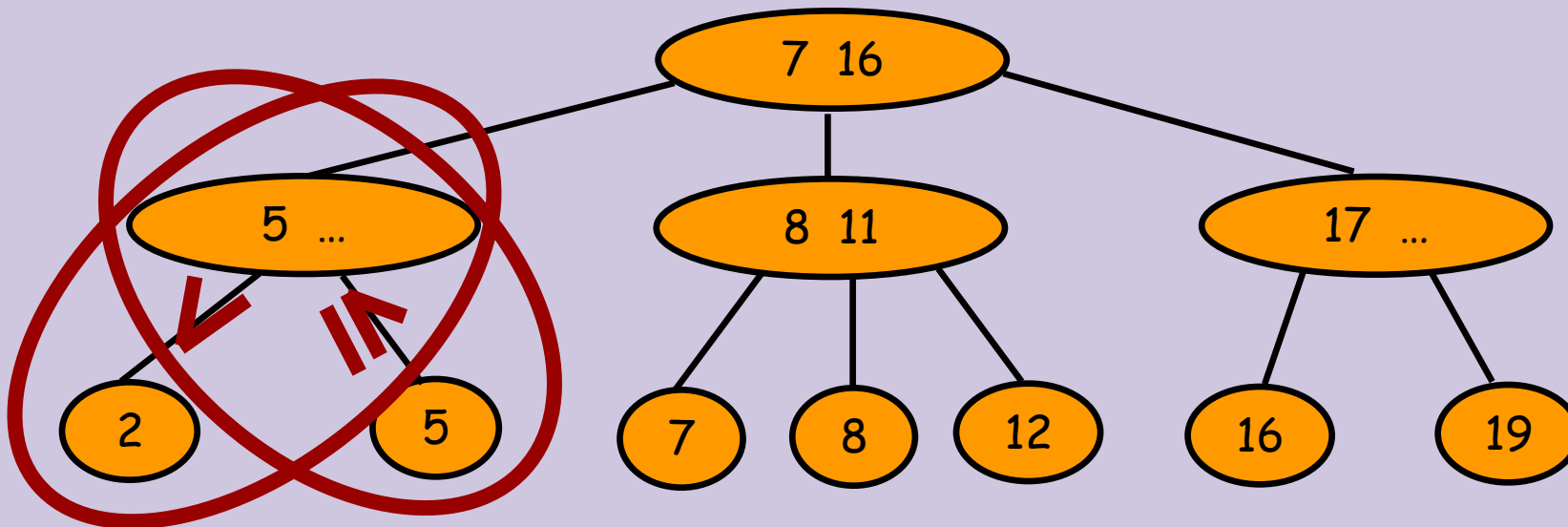
לפיכך הגובה מקיים $\log_3 n \leq h \leq \log_2 n$

כלומר: $h = \Theta(\log n)$

עצי 2-3 כמבני נתונים

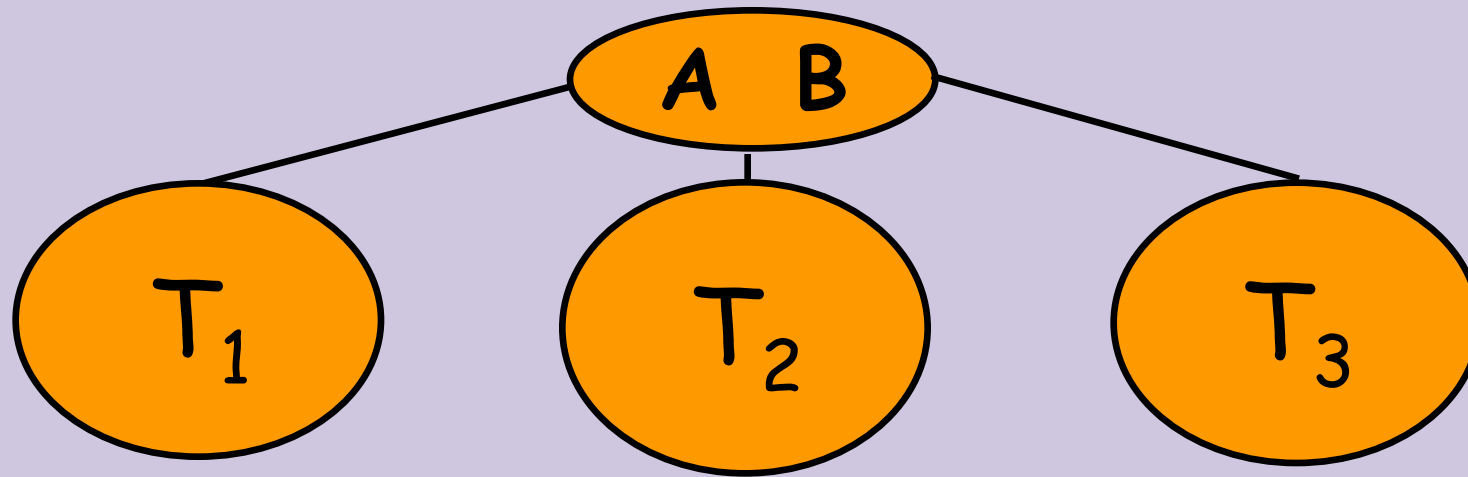
כל עלה מכיל מפתח ורשומה (רק המפתחות נראים בציור).

לכל צמת פנימי $2 \leq d \leq 3$ בנים ויש בו $d-1$ אינדקסים המשמשים לחיפוש הרשומה הנחוצה.

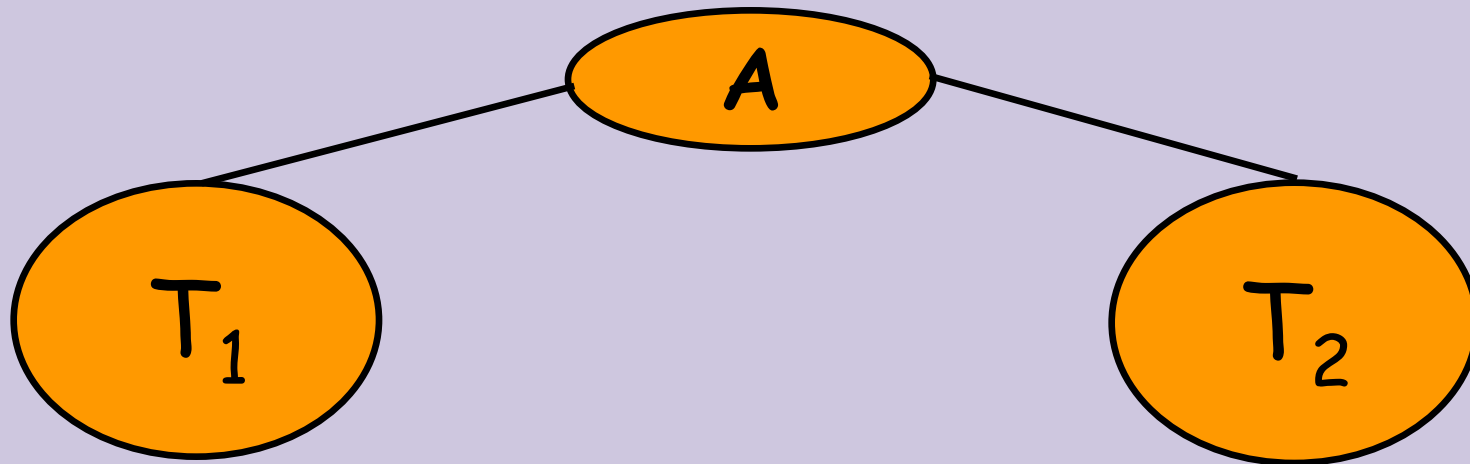


• **בצומת פנימי בעל שני בנים** רשום אינדקס בודד הגדול ממש מהמפתח המקסימלי בתת העץ ששורשו הוא הבן הראשון וכן קטן או שווה למפתח המינימלי בתת העץ ששורשו הוא הבן השני.

• **בצומת פנימי בעל שלושה בנים** רשומים שני אינדקסים. אינדקס ראשון גדול ממש מהמפתחות בעץ הראשון וקטן שווה מהמפתחות בעץ בשני ואינדקס שני גדול ממש מהמפתחות בעץ השני וקטן שווה מהמפתחות בעץ השלישי.



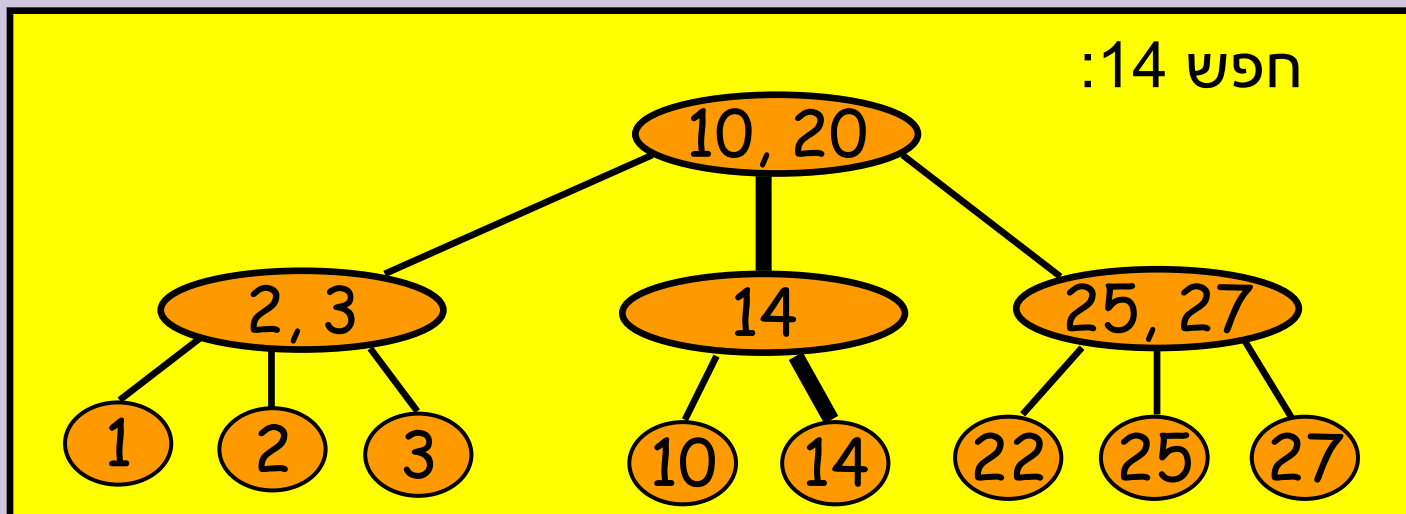
$$T_1 < A \leq T_2 < B \leq T_3$$



$$T_1 < A \leq T_2$$

חיפוש מפתח k בעץ 3-2

- יהי v השורש של העץ
- **אם v עלה**, בדוק אם k נמצא בצומת v .
- יהי k_1 האינדקס המינימלי בצומת v .
- **אם $k < k_1$** :
 - המשך את החיפוש בבן הראשון של v .
 - **אחרת, אם ל- v רק שני בנים או ש- k קטן מהמפתח השני של v** :
 - המשך את החיפוש בבן השני של v .
 - **אחרת, המשך את החיפוש בבן השלישי של v .**



הכנסת מפתח k לעץ 3-2

בהכנסת איבר לעץ 3-2 מבוצעות הפעולות הבאות:

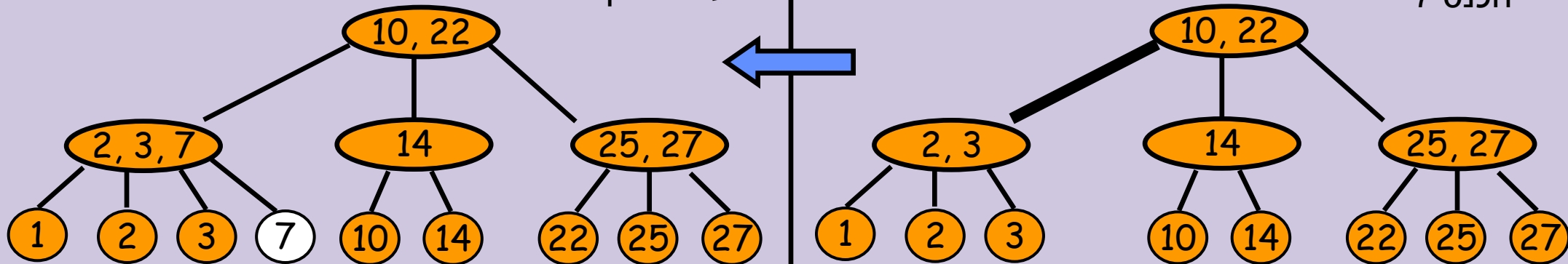
1. חיפוש מקומו של המפתח החדש k .
2. הכנסת עלה למקום החדש וקביעת ערכו k .
3. תיקון העץ כך שלכל צומת יהיו 2 או 3 בנים. התיקון נעשה במסלול החיפוש של k מהעלה שנוסף ועד השורש. בכל רמה נבצע $O(1)$ פעולות.

נמְחִישׁ זאת תחילה ע"י דוגמאות.

דוגמא להכנסה לעץ 3-2

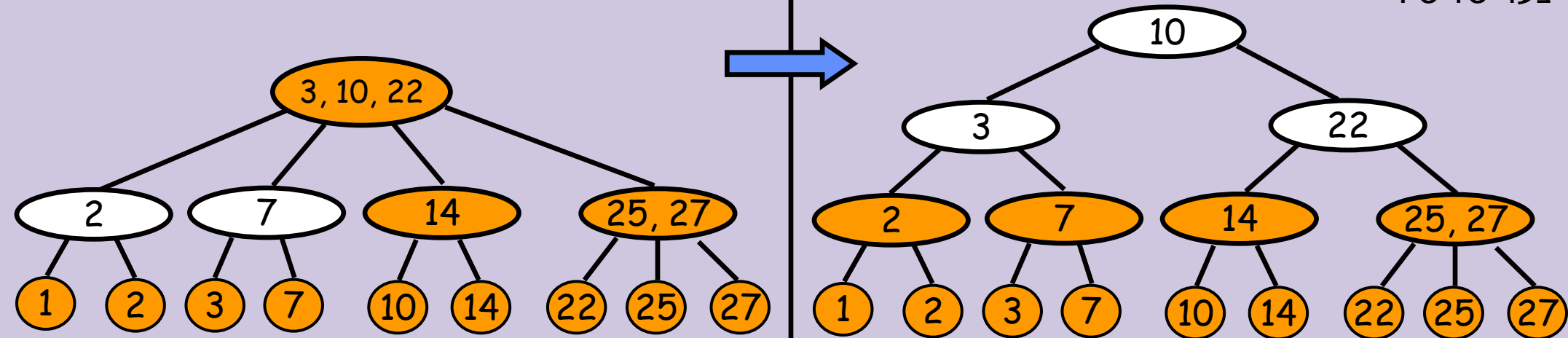
צעד ראשון:

הכנס 7



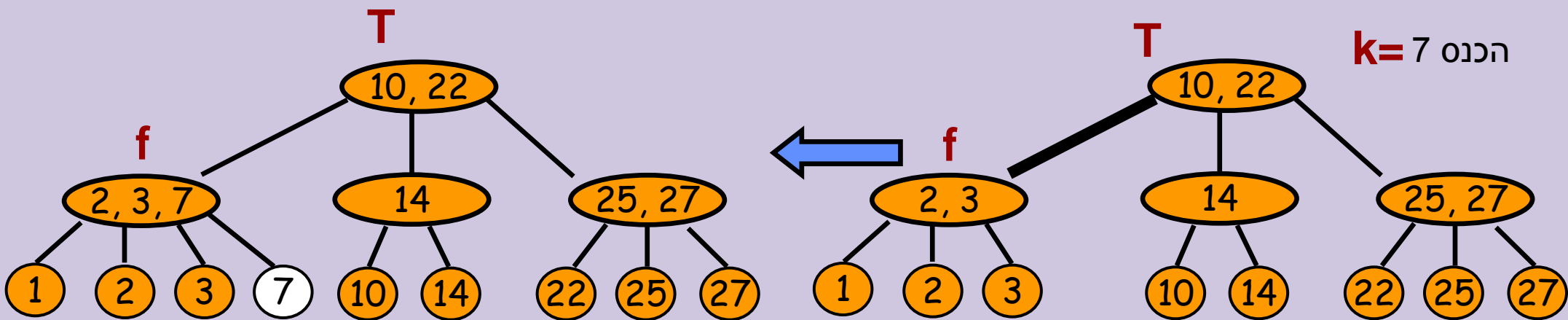
צעד שני:

צעד שלישי:

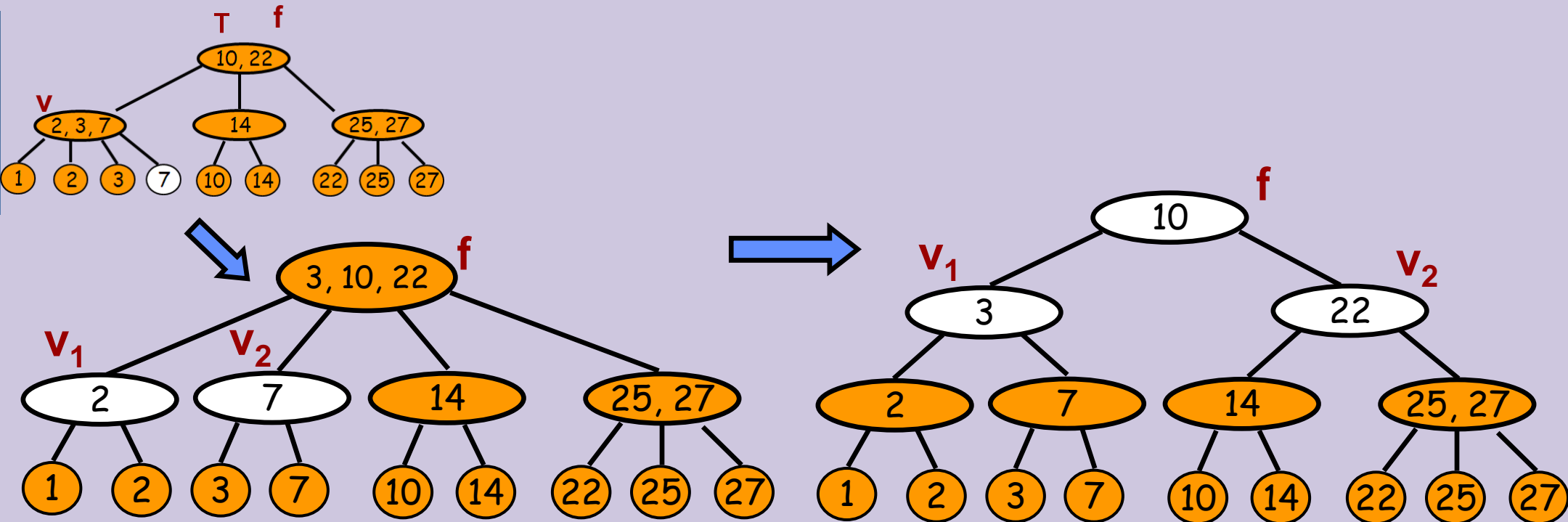


אלגוריתם להכנסת מפתח k (חלק א)

1. חפשו את k בעץ T . אם k נמצא, סיימו.
2. אם k אינו ב- T , יהי f הצומת האחרון, שאינו עלה, במסלול החיפוש.
3. צרו עלה חדש בעל מפתח k , והוסיפו אותו כבן ל- f תוך שמירת הסדר בין הבנים של f . (יתכן וכעת יש ל- f ארבעה בנים).
4. הוסיפו ל- f אינדקס נוסף בהתאם לכללי עץ 2-3. (יתכן וכעת יש שלושה אינדקסים).



אלגוריתם להכנסת מפתח k (חלק ב)



5. $v \leftarrow f$ ו- $f \leftarrow \text{parent}(v)$ - אם ל- v שלושה בנים סיימו

6. אם v הוא שורש צרו צומת f אשר בניו הם הצמתים v_1, v_2 וסיימו.

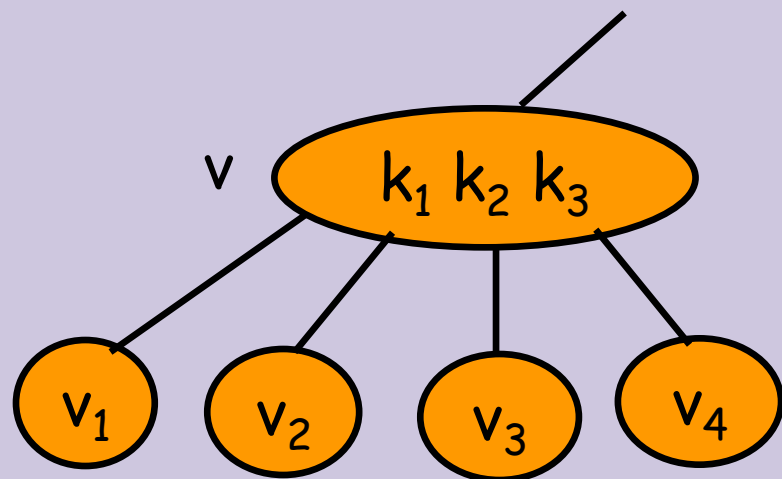
7. אם ל- v ארבעה בנים, פצל את v לשני צמתים v_1, v_2 , וחבר אותם כבנים לאב f תוך שמירה על סדר האינדקסים הנכון.

8. חזרו לצעד 5.

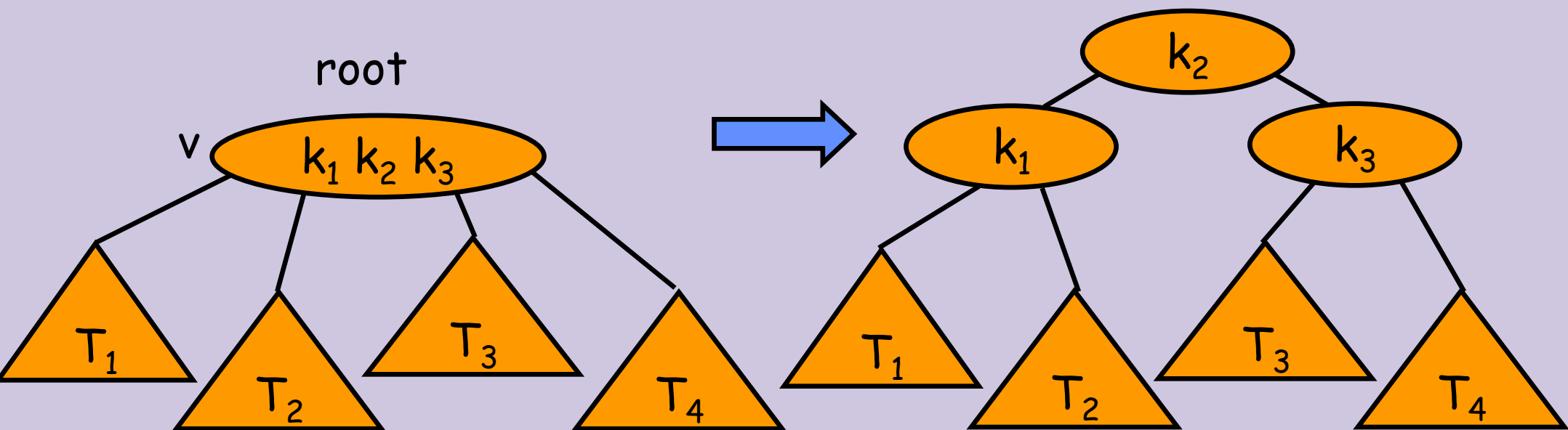
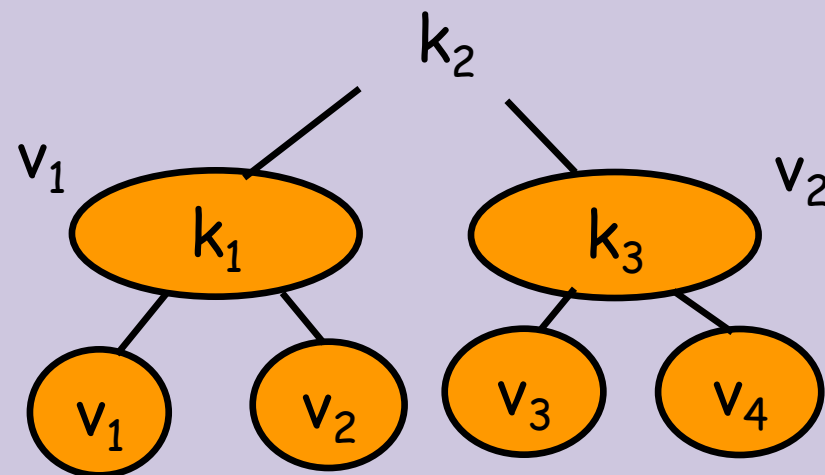
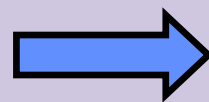
אלגוריתם להכנסת מפתח k (במלואו)

1. חפשו את k בעץ T . אם k נמצא, סיימו.
2. אם k אינו ב- T , יהי f הצומת האחרון, שאינו עלה, במסלול החיפוש.
3. צרו עלה חדש בעל מפתח k , והוסיפו אותו כבן ל- f תוך שמירת הסדר בין הבנים של f . (יתכן וכעת יש ל- f ארבעה בנים).
4. הוסיפו ל- f אינדקס נוסף בהתאם לכללי עץ 2-3. (יתכן וכעת יש שלושה אינדקסים).
5. $v \leftarrow f$ ו- $f \leftarrow \text{parent}(v)$ - אם ל- v שלושה בנים סיימו
6. אם v הוא שורש צרו צומת f אשר בניו הם הצמתים v_1, v_2 וסיימו.
7. אם ל- v ארבעה בנים, פצל את v לשני צמתים v_1, v_2 , וחבר אותם כבנים לאב f תוך שמירה על סדר האינדקסים הנכון.
8. חזרו לצעד 5.

פרוט לפיצול צומת



$$v_1 < k_1 \leq v_2 < k_2 \leq v_3 < k_3 \leq v_4$$



תכונות תהליך ההוספה

- שינויים מתבצעים רק על המסלול מהשורש לעלה שהוסף.
- בזמן פיצול של צומת, הצמתים החדשים הם בעומק שווה לצומת שפוצל.
- בפיצול השורש נוצר צומת חדש שמגדיל ב-1 את העומק של כל הצמתים.

מסקנה 1: לאחר הוספה, כל העלים באותו עומק ולכל צומת פנימי 2-3 בנים.

מסקנה 2: סיבוכיות הכנסת אבר $O(\log n)$.

הוצאת מפתח k מעץ 3-2

בהוצאת איבר לעץ 3-2 מבוצעות הפעולות הבאות:

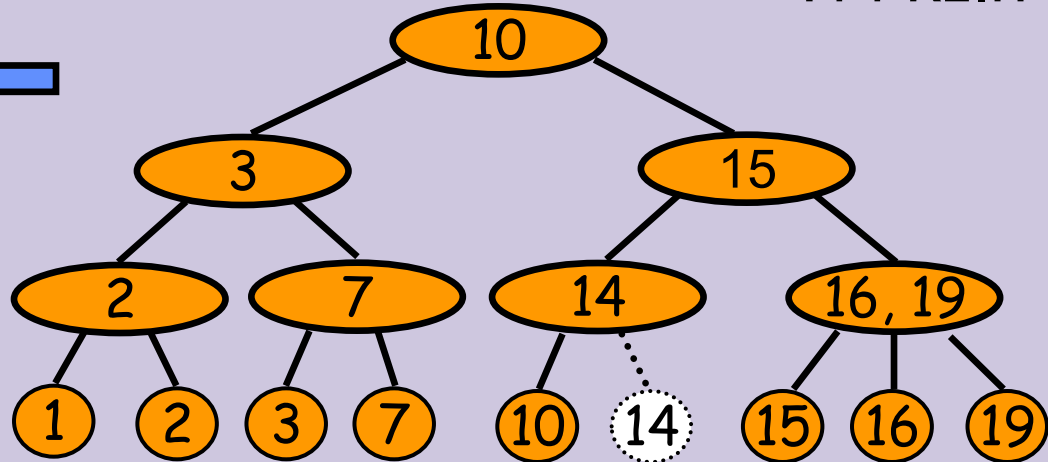
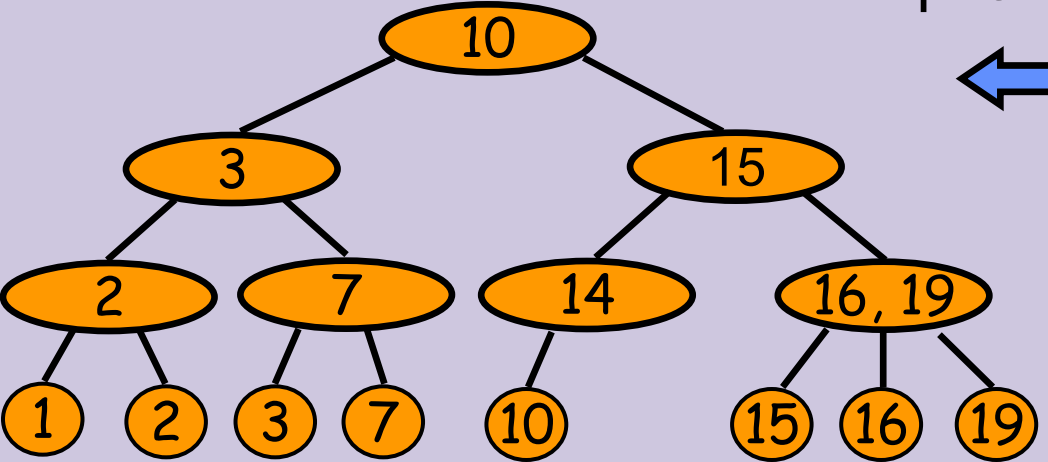
1. חיפוש מקומו של המפתח k .
2. הוצאת העלה שערכו k .
3. תיקון העץ כך שלכל צומת יהיו 2 או 3 בנים. התיקון נעשה במסלול החיפוש של k מהעלה שהוצא ועד השורש. בכל רמה נבצע $O(1)$ פעולות.

נמחיש זאת תחילה ע"י דוגמאות.

דוגמא להוצאה מעץ 3-2 (השאלה בלבד)

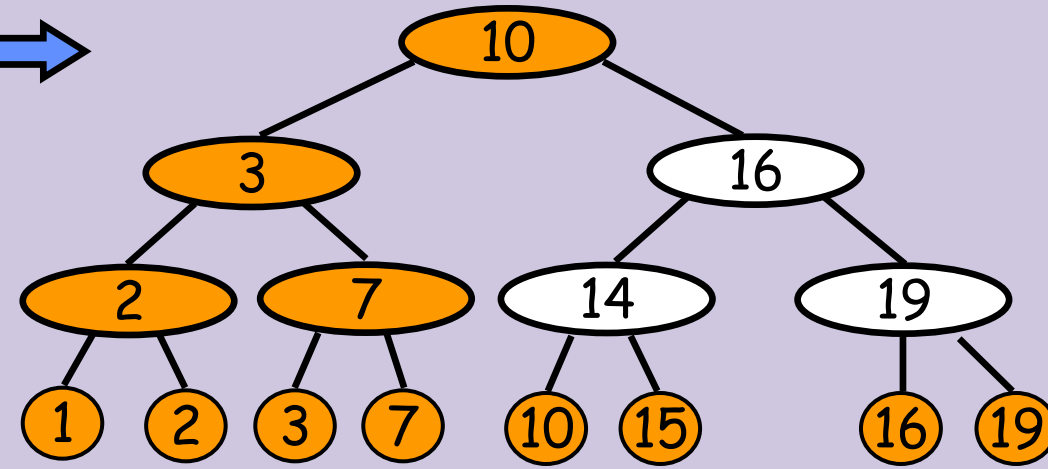
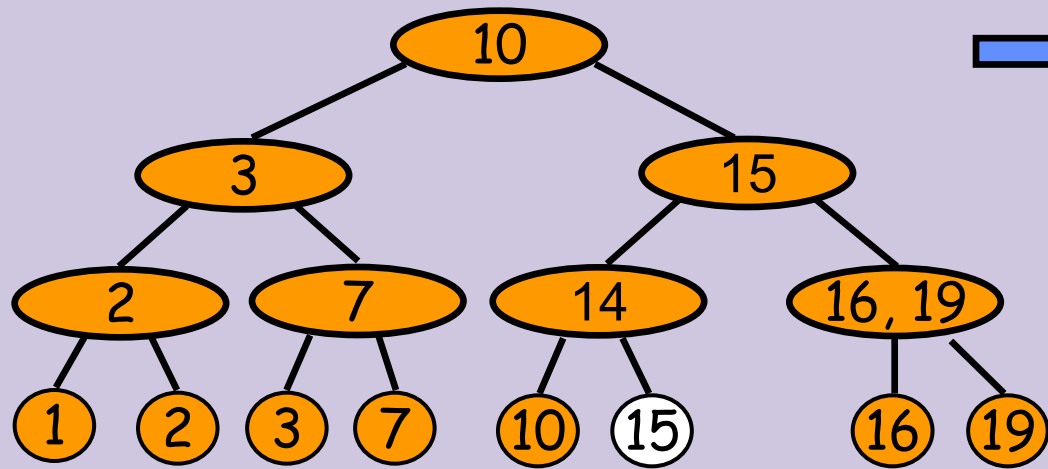
סלק צומת:

הוצא 14:

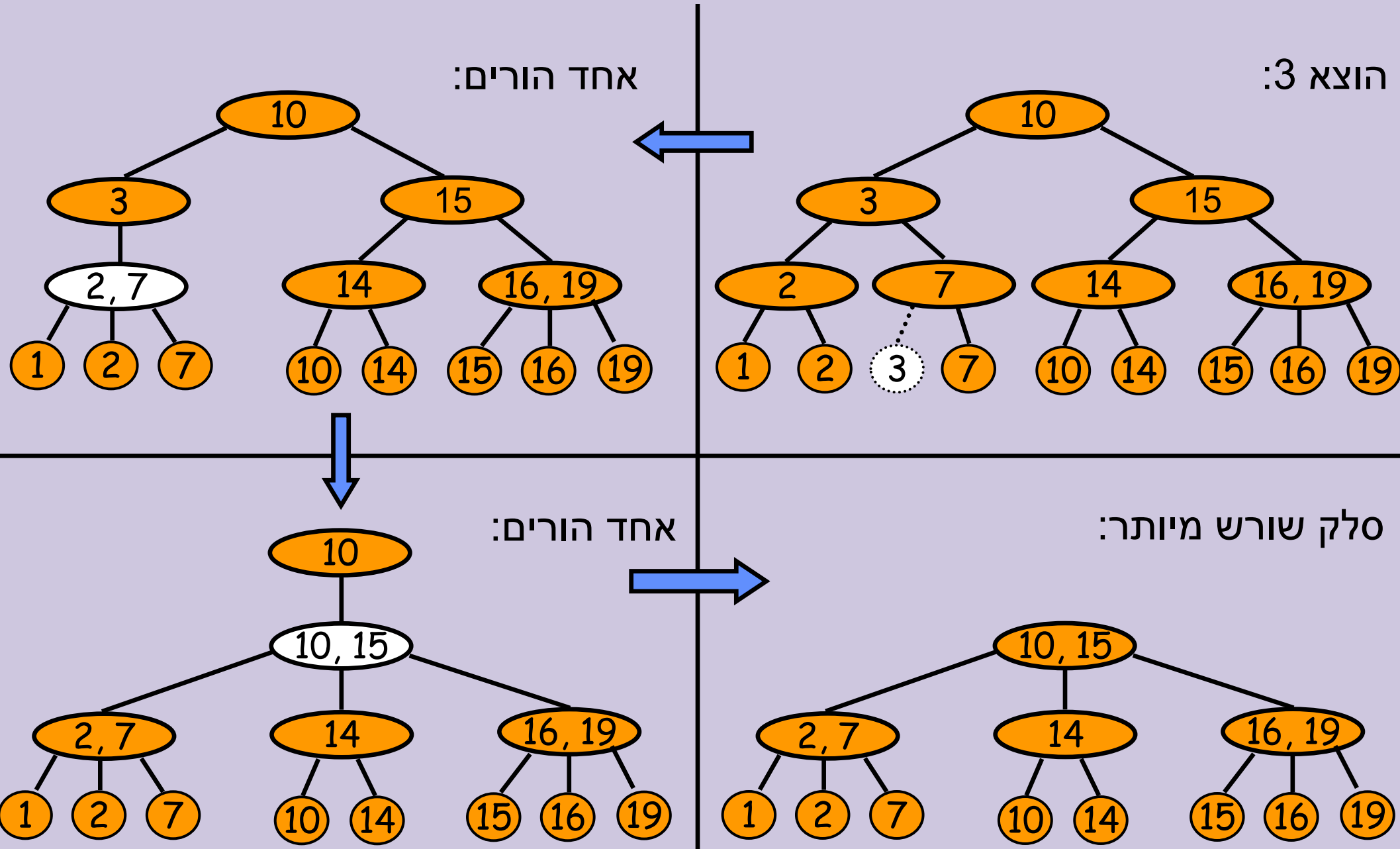


השאל מהאח:

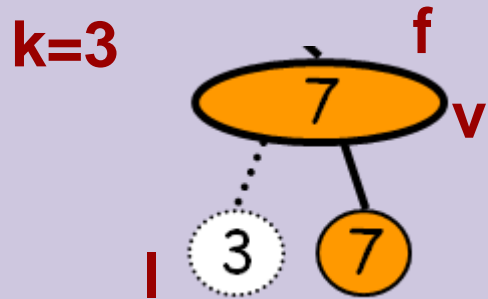
תקן ערכי ההורים:



דוגמא להוצאה מעץ 3-2 (שני איחודים)



אלגוריתם להוצאת מפתח k מעץ 2-3



1. חפש את k בעץ. אם k לא נמצא בעץ, סיים.

2. יהי l העלה שערכו k ויהי f אביו.

3. סלק את l מהעץ.

4. $v \leftarrow f$

5. אם v הוא שורש ולו בן בודד, סלק את v וסיים.

6. אם ל- v נותרו 2 בנים, סיים. (המשך אם ל- v נותר רק בן בודד).

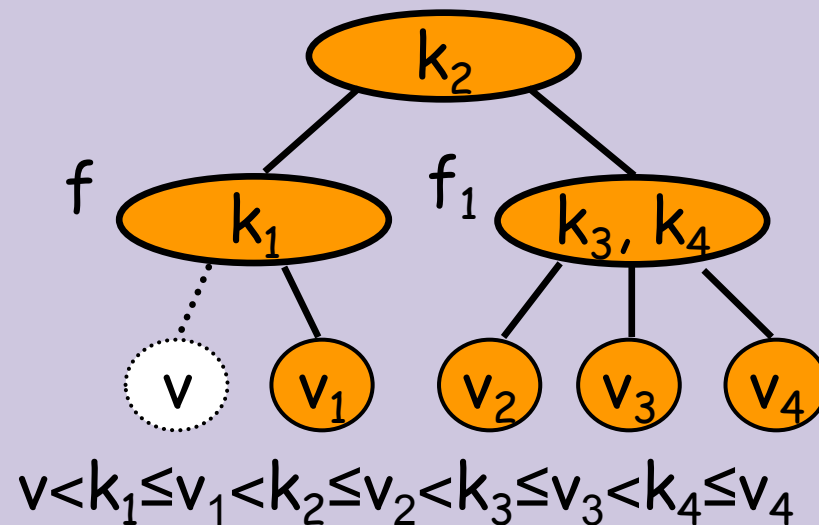
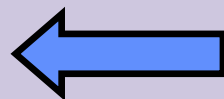
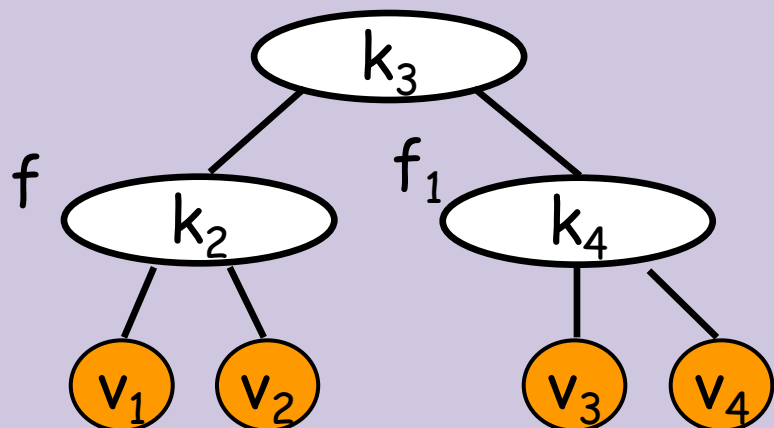
7. מקרה א: אם ל- v יש אח ולו שלושה בנים, שאל בן מהאח וסיים.

מקרה ב: אם לכל אח רק שני בנים, אחד את v עם אח קרוב ל- v ועדכן את המפתחות בצומת שנוצר. יהי f ההורה של v .

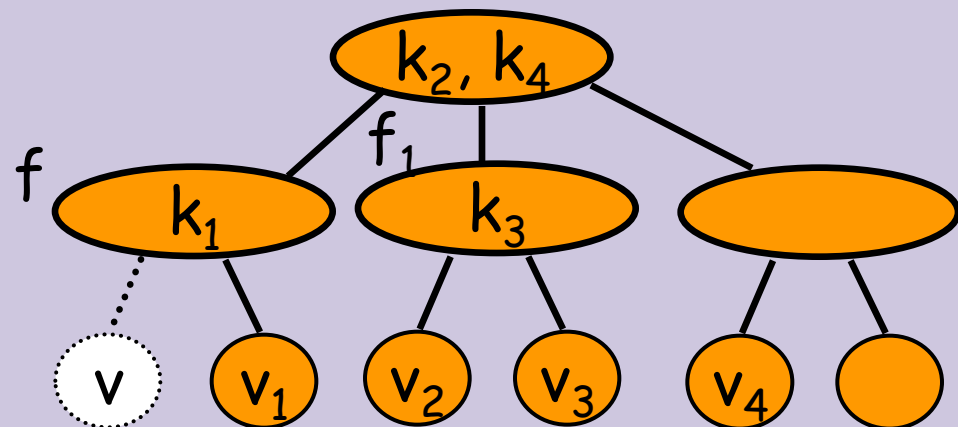
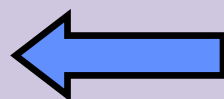
8. חזור לצעד 4.

פרוט פעולת האיחוד בזמן הוצאה

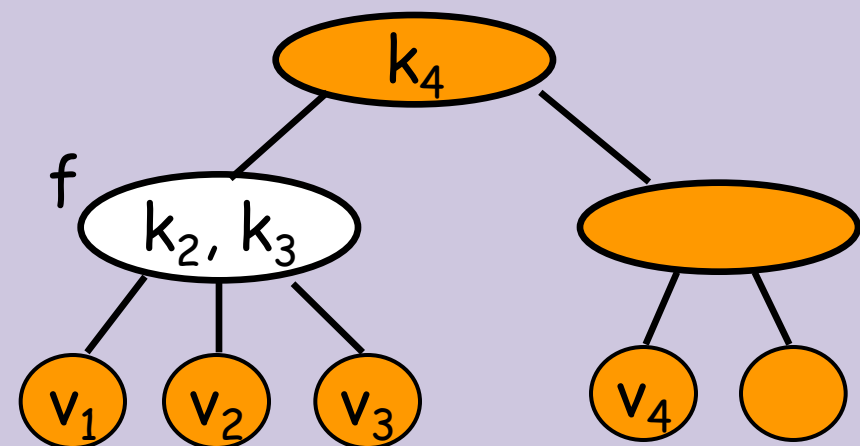
מקרה א:



מקרה ב:



$$v < k_1 \leq v_1 < k_2 \leq v_2 < k_3 \leq v_3 < k_4 \leq v_4$$



ניתוח פעולת ההוצאה

נכונות:

- העומק של שום צומת אינו משתנה בזמן ההוצאה מלבד בסוף התהליך,
- במידה ומתבטל השורש, אז העומק של כל הצמתים קטן באחד. לפיכך כל העלים נותרים בעומק שווה.
- כמו כן לכל צומת פנימי נותרים 2-3 בנים ונשמר יחס הסדר בין האינדקסים כנדרש.

מסקנה 1: לאחר הוצאה, כל העלים באותו עומק ולכל צומת פנימי 2-3 בנים.

מסקנה 2: סיבוכיות הוצאה אבר $O(\log n)$.

סיבוכיות הפעולות בעץ 2-3:

סיבוכיות	function	פעולה
$O(1)$	create(D)	אתחול
$O(\log n)$	find(D,x)	חיפוש
$O(\log n)$	insert(D,x,info)	הוספה
$O(\log n)$	delete(D,x)	הוצאה
$O(\log n) \rightarrow O(1)$	min(D)	מינימום
$O(\log n) \rightarrow O(1)$	next(D,x)	עוקב

עצי B^+ (מדרגה m)

הגדרה: עץ B^+ מדרגה m הוא עץ המקיים את התכונות הבאות:

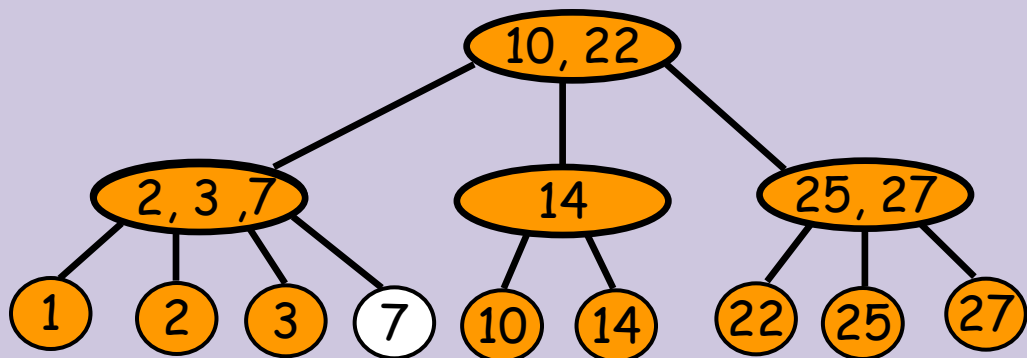
1. כל הערכים נמצאים בעלים, כל העלים באותה רמה.

2. לכל צומת פנימי, פרט אולי לשורש, יש c בנים כאשר $m \geq c \geq \lceil m/2 \rceil$.

לשורש מספר הבנים הוא $m \geq c \geq 2$.

3. לצומת פנימי בעל c בנים יש $c-1$ אינדקסים ממוינים לפי גודלם.

כל המפתחות הנמצאים בתת העץ ה- i קטנים מהאינדקס ה- i וגדולים או שווים לאינדקס ה- $i-1$.
כל המפתחות הנמצאים בתת העץ הימני ביותר גדולים או שווים לאינדקס האחרון.



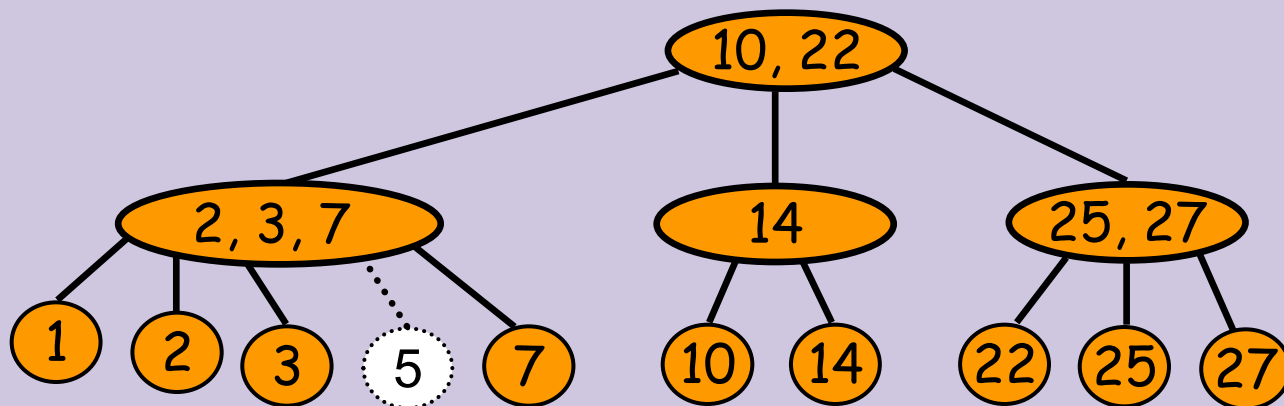
דוגמא: עץ 3-2.

דוגמא: עץ 4-3-2.

חיפוש/הכנסה/הוצאה כמו בעץ 3-2.

כלומר, חיפוש לפי הגדרת עץ B^+ בדרגה m , הכנסה או הוצאה של העלה המבוקש, ותיקון העץ מהעלה ועד השורש במסלול החיפוש כך שמספר הבנים בכל רמה יעמוד בדרישות עץ B^+ בדרגה m .

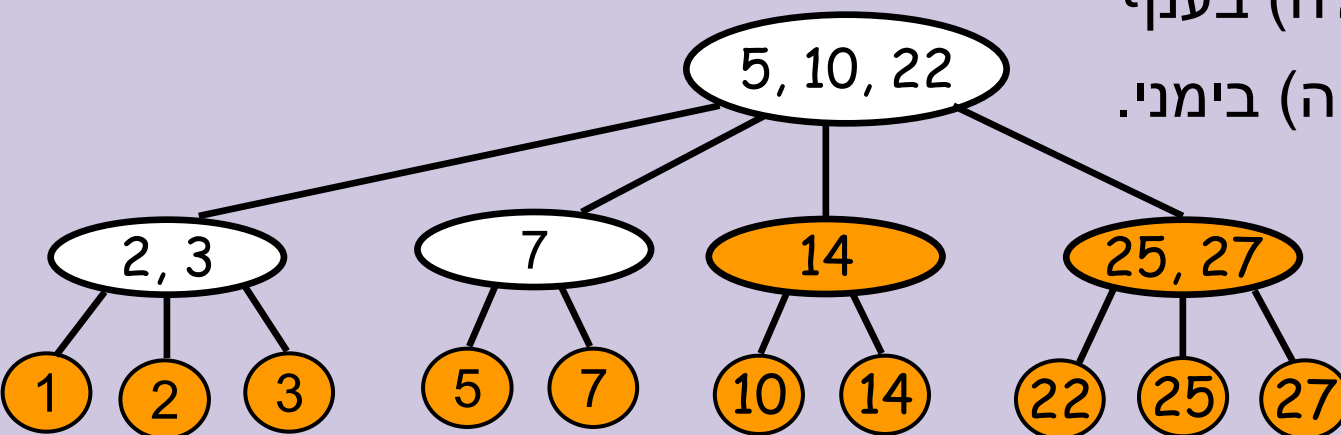
דוגמא להכנסה לעץ B⁺ מדרגה 4



דוגמא: הכנס 5

במקרה זה נדרש פיצול בודד.

מחצית הצמתים (מעוגל כלפי מעלה) בענף השמאלי ומחצית (מעוגל כלפי מטה) בימני.



דוגמאות נוספות בתרגול.

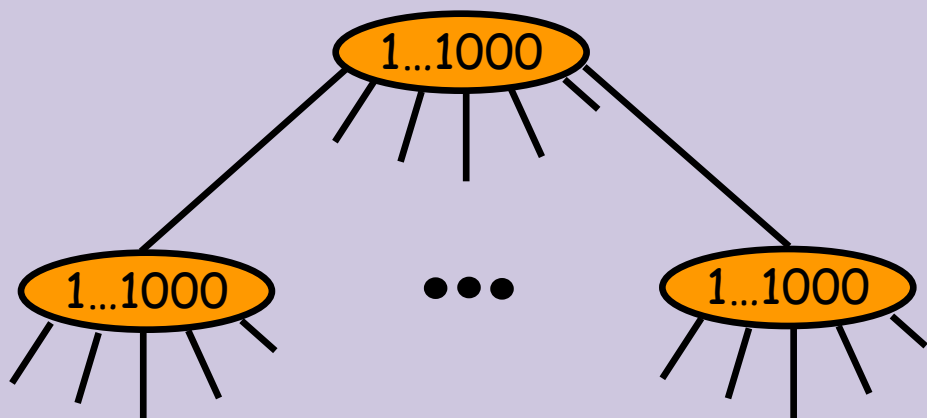
שימוש עיקרי לעץ B⁺

הזיכרון ברוב המחשבים מורכב משני חלקים:

זיכרון ראשי (RAM), מהיר, ללא חלקים נעים, שבב סיליקון בודד.

זיכרון משני (DISK), גדול, איטי יחסית לזיכרון הראשי, בעל חלקים מכניים.

כאשר לא ניתן להכניס את כל מבנה הנתונים לזיכרון הראשי יש לפנות מדי פעם לזיכרון המשני לקבלת אינפורמציה נוספת. זמן הגישה לזיכרון המשני גבוה מאד יחסית לזמן הקריאה של הנתונים והקריאה נעשית בבלוקים.



עצי B⁺ מדרגה גבוהה מספקים פתרון:

בכל צומת פנימי נשמור כמספר הבלוקים של מפתחות שניתן לשמור בזיכרון הראשי.

מספר הגישות לזיכרון המשני תלוי במספר הרמות (וזוהו מספר נמוך) הנקבע ע"י גודל הבלוק.

המחיר יהיה בסקירה ליניארית של המפתחות על מנת לקבוע את המשך החיפוש בעץ (תוך שימוש בזיכרון המהיר).

מידע נוסף בעצי חיפוש

בשימושים רבים של עצי חיפוש כדאי לשמור אינפורמציה נוספת בכל צומת. אינפורמציה זו משמשת להאצת פעולות נוספות הנדרשות מעצי חיפוש. למשל:

הגדרה: האינדקס (rank) של מספר x בקבוצה S הוא מקומו בסדרה ממוינת של איברי S . האינדקס של 5 בקבוצה $\{8,2,7,5\}$ הוא 2.

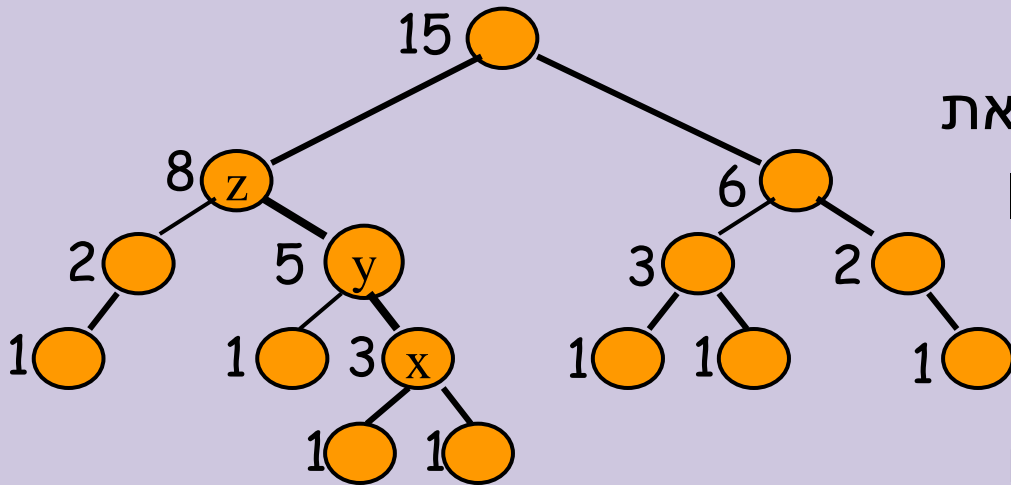
בעיה I: לממש עץ חיפוש התומך, בנוסף לפעולות הכנסה/הוצאה/חיפוש, גם במציאת האינדקס של x בעץ בזמן $O(\text{height})$ כאשר height הוא גובה העץ.

בעיה II: לממש עץ חיפוש התומך במציאת סכום האיברים בעץ הקטנים $m - x$ בזמן $O(\text{height})$.

בבעיות אלה יש לשמור מידע נוסף בעץ החיפוש כדי שנוכל לממש את הפעולות הנוספות ביעילות הנדרשת. אנו ניראה מהו המידע שיש לשמור וכיצד להשתמש בו. אח"כ ניראה כיצד לעדכן את המידע הנוסף כך שהעץ יישאר מאוזן.

עץ דרגות (Rank Tree)

בעיה: לממש עץ חיפוש התומך במציאת האינדקס של איבר x .



פתרון: נשמר בכל צומת v (מלבד המפתח) את מספר הצמתים בתת-העץ ששורשו v . נסמן מספר זה ב- $n(v)$.

נספור את מספר הצמתים על מסלול החיפוש ל- x הקטנים מ- x בתוספת הערכים השמורים בבניהם השמאליים.

$$\text{rank}(x) = 2 + 1 + 1 + |\{z, y\}| = 6$$

$$\text{rank}(x) = \sum_w n(w) + |\{v \in \text{path} \mid v < x\}|$$

(w בן שמאלי של צומת במסלול path)

הגדרה: **עץ דרגות (rank tree)** הוא עץ בו בכל צומת v נשמרים מספר הצמתים בתת-העץ ששורשו v .

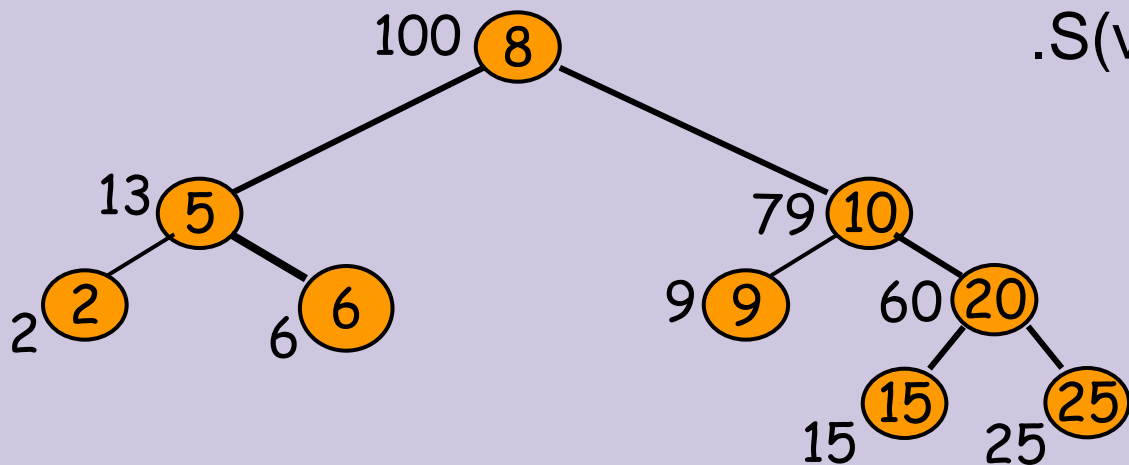
מסקנה: בעץ דרגות מציאת ה- rank דורשת זמן $O(\text{height})$.

עץ דרגות הוא דוגמא למבנה נתונים בו בכל צומת נשמר מידע על תת-העץ שמתחתיו. סוג המידע השמור תלוי בבעיה אותה יש לפתור.

דוגמא למידע נוסף

בעיה: לממש עץ חיפוש התומך במציאת סכום האיברים בעץ הקטנים מ- x בזמן $O(\text{height})$.

פתרון: נשמר בכל צומת v את סכום ערכי הצמתים בתת-העץ ששורשו v . נסמן מספר זה ב- $S(v)$.



נסכם את הערכים על מסלול החיפוש ל- x הקטנים מ- x בתוספת הערכים השמורים בבניהם השמאליים.

דוגמא: סכום הקטן מ-17 מחושב
 $(8+13) + (10+9) + (15+0) = 55$ ע"י

מסקנה: זמן הריצה הוא $O(\text{height})$.

כיצד נבטיח זמן ריצה $O(\log n)$?

עץ דרגות מאוזן (Rank tree)

בעיה: לממש עץ חיפוש מאוזן התומך במציאת האינדקס (rank) של איבר x בזמן $O(\log n)$ (כמה צמתים נמצאים לפני x בעץ).

פתרון: לדוגמא באמצעות עץ AVL. נשמור בכל צומת v (מלבד המפתח) את מספר הצמתים בתת-העץ ששורשו v .

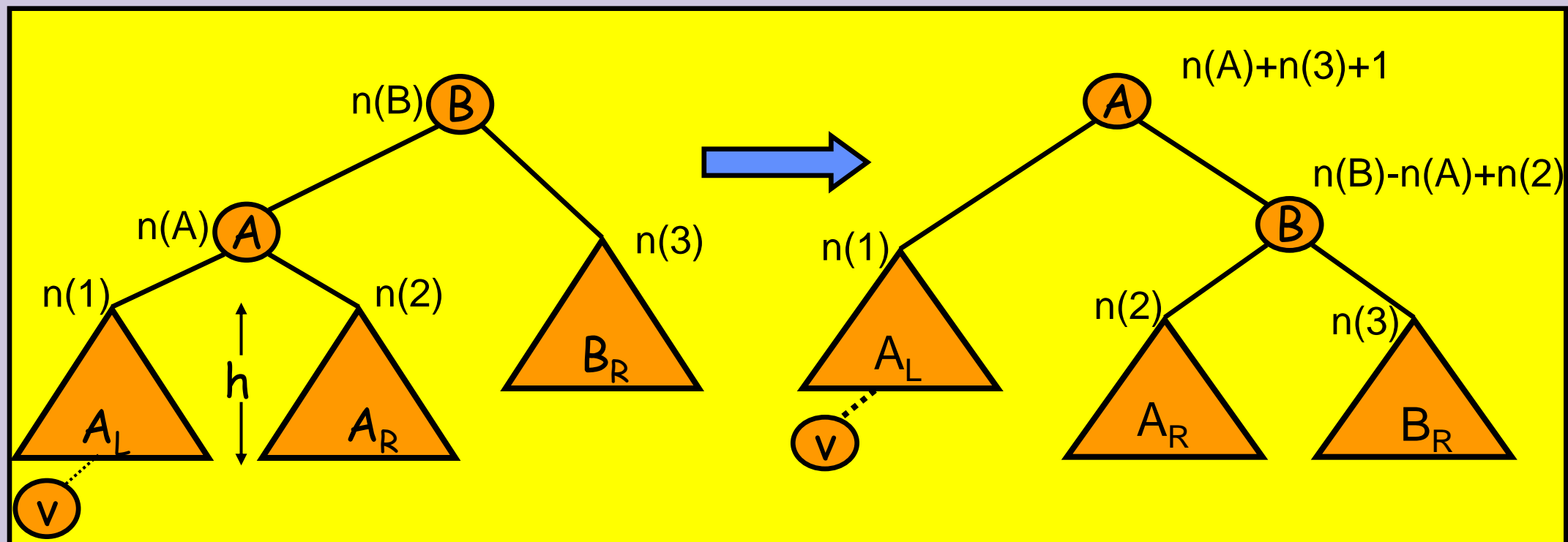
זיכרו שסימנו מספר זה ב- $n(v)$.

נעדכן את $n(v)$ בזמן גלגול.

עדכון השדה הנוסף

כיצד נעדכן את $n(v)$?

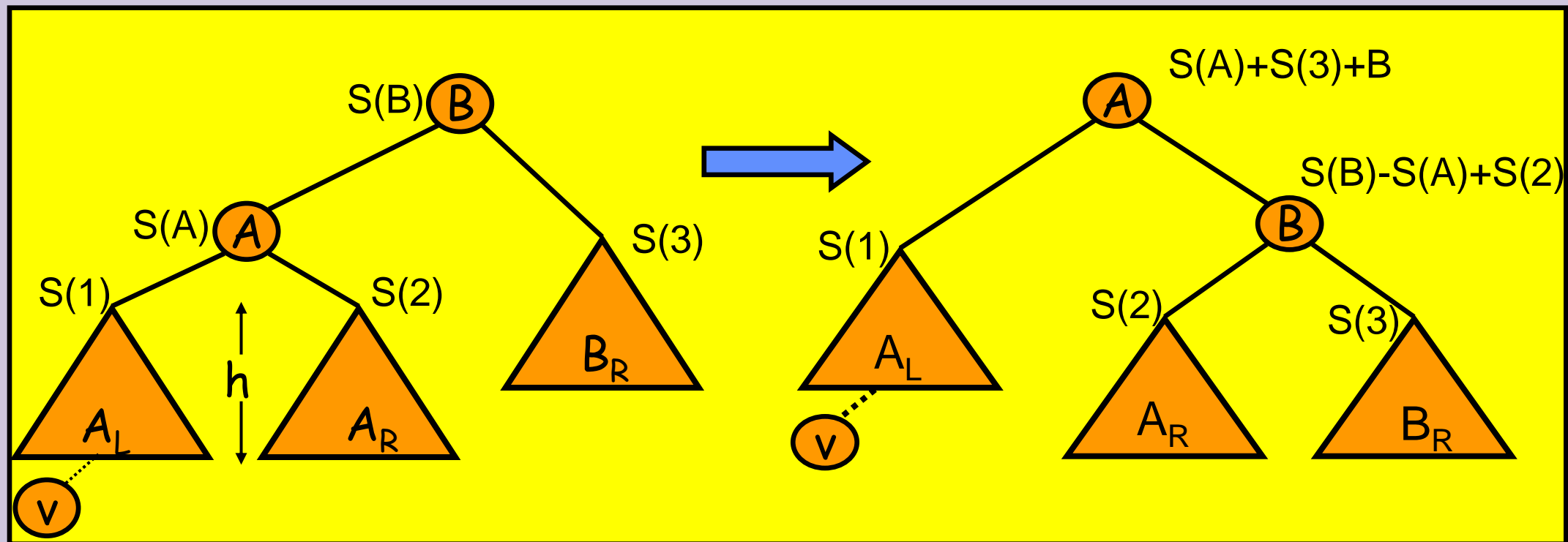
בזמן הכנסה נגדיל את $n(v)$ באחד לאורך המסלול העלה שהוכנס. בזמן גלגול נעדכן את השדה כנדרש. למשל בגלגול LL נעדכן כך:



דוגמא נוספת

כיצד נעדכן את סכום המפתחות בכל תת-עץ של עץ AVL ?

בזמן הכנסה נגדיל את הסכום לאורך המסלול מהשורש לעלה שהוכנס. בזמן גלגול נעדכן את השדה כנדרש. למשל בגלגול LL נעדכן כך:



תרגיל בית: מהו העדכון בגלגולים האחרים ? כיצד מבצעים זאת עם עץ

3-2 ?