עצי 2-3
ועצי דרגות

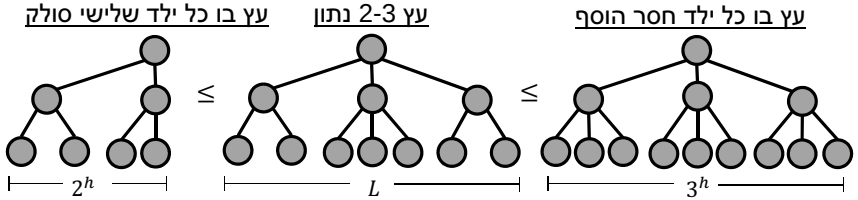
חומר קריאה לשיעור זה

Chapter 19: B trees (381 – 397)
Chapter 15: Augmenting data structures (281 – 290)

גובה עצי 2-3

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

הוכחה: עבור חסם תחתון נבחן את העץ הבינרי השלם הנוצר ע"י סילוק כל ילד שלישי מעץ 2-3.
עבור חסם עליון נבחן את העץ הטרינרי השלם הנוצר ע"י הוספת ילד שלישי לעץ 2-3 בכל מקום בו חסר ילד.

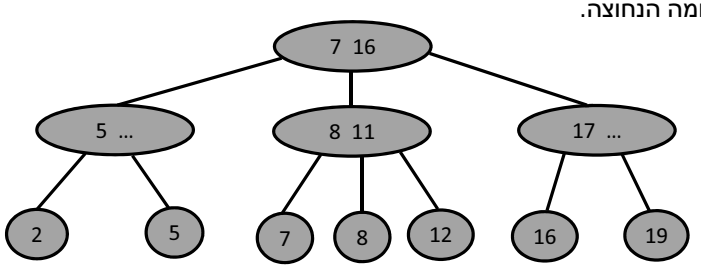


לפיכך הגובה מקיים $\log_3 L \le h \le \log_2 L$.

כלומר: $h = \Theta(\log L)$.

עצי 2-3 כמבני נתונים

כל עלה מכיל מפתח ורשומה (רק המפתחות נראים בציור).
לכל צמת פנימי $2 \le d \le 3$ בנים ויש בו $d-1$ אינדקסים המשמשים לחיפוש הרשומה הנחוצה.



- **בצומת פנימי בעל שני בנים** רשום אינדקס בודד - k_1 - 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 - $k_1 < k_2$. האינדקס הראשון (k_1) גדול ממש מהמפתחות בעץ הראשון וקטן שווה מהמפתחות בעץ בשני והאינדקס השני (k_2) גדול ממש מהמפתחות בעץ השני וקטן שווה מהמפתחות בעץ השלישי.

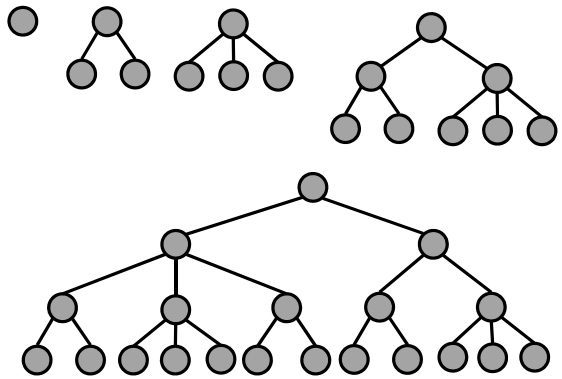
עצים מאוזנים

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 כל העלים נמצאים באותה רמה ולכל צומת פנימי 2 או 3 בנים.

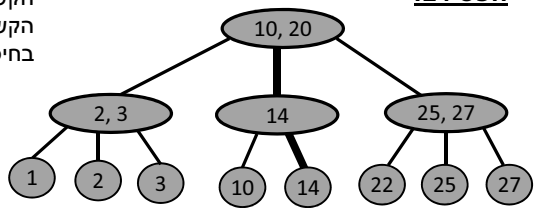
דוגמאות:



חיפוש מפתח k בעץ 2-3

1. יהי v השורש של העץ.
2. אם v עלה, בדוק אם k נמצא בצומת v. החזר תשובה בהתאם.
3. אם $k < k_1$ (k קטן מהמפתח הראשון של v):
   1. המשך את החיפוש בבן הראשון של v.
4. אחרת, אם ל-v רק שני בנים או ש- $k < k_2$ (k קטן מהמפתח השני של v):
   1. המשך את החיפוש בבן השני של v.
5. אחרת, המשך את החיפוש בבן השלישי של v.

חפש 14:

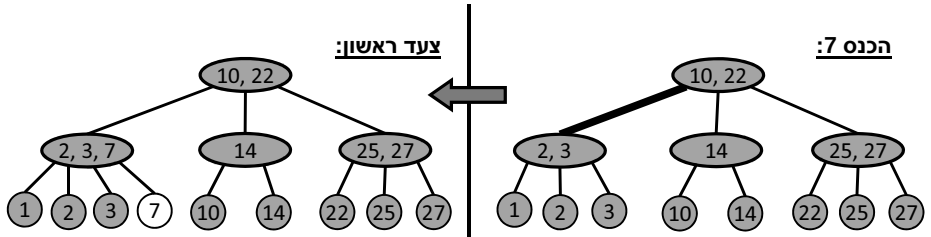


הקשתות המושחרות הן הקשתות בהן יורדים בעת בחיפוש.

דוגמא להכנסה לעץ 2-3

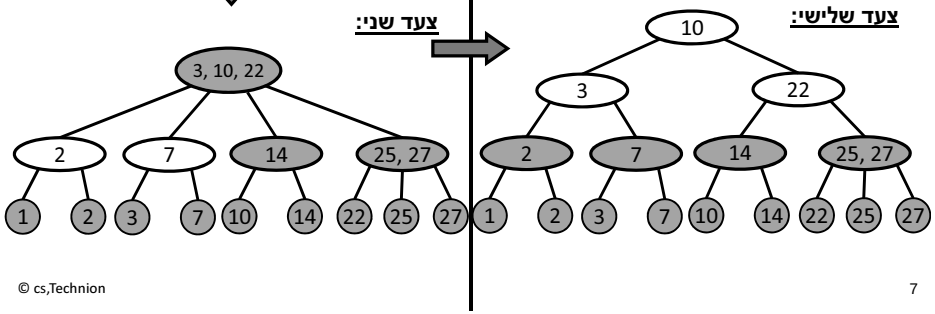
הכנס 7:

צעד ראשון:



צעד שני:

צעד שלישי:



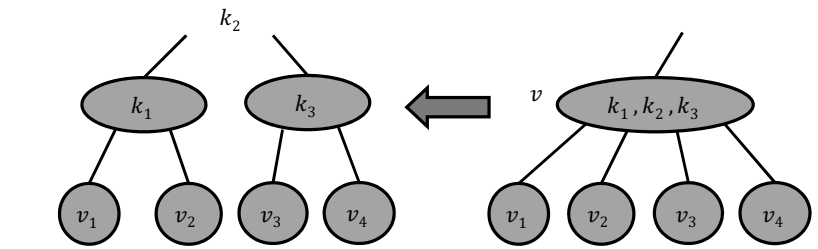
הכנסת מפתח k לעץ 2-3

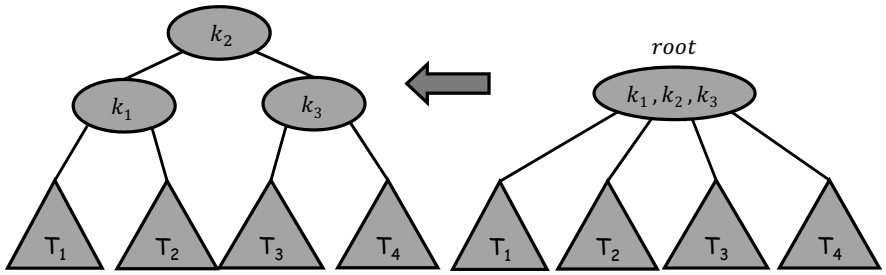
<u>בהכנסת איבר לעץ 2-3 מבוצעות הפעולות הבאות:</u>

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.

נמחיש זאת תחילה ע"י מספר דוגמאות.

פרוט פעולת פיצול צומת

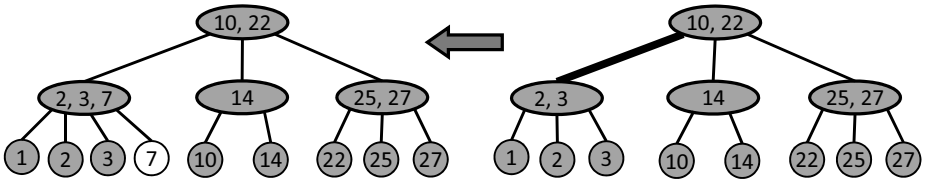




אלגוריתם להכנסת מפתח k (חלק א)

1. חפש את k בעץ T. אם k נמצא, סיים.
2. אם k אינו ב-T, יהי f הצומת האחרון, שאינו עלה, במסלול החיפוש. (לו k היה ב-T אזי f היה אבי העלה שבו k נמצא).
3. צור עלה חדש בעל מפתח k, והוסף אותו כבן ל-f תוך שמירת הסדר בין הבנים של f. (יתכן וכעת יש ל-f ארבעה בנים).
4. הוסף ל-f אינדקס נוסף בהתאם לכללי עץ 2-3. (יתכן וכעת יש ל-f שלושה אינדקסים).

הכנס 7:



אלגוריתם להכנסת מפתח k (במלואו)

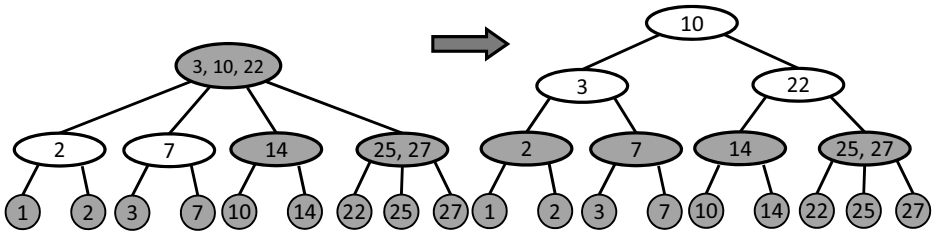
1. חפש את k בעץ T. אם k נמצא, סיים.
2. אם k אינו ב-T, יהי f הצומת האחרון, שאינו עלה, במסלול החיפוש. (לו k היה ב-T אזי f היה אבי העלה שבו k נמצא).
3. צור עלה חדש בעל מפתח k, והוסף אותו כבן ל-f תוך שמירת הסדר בין הבנים של f. (יתכן וכעת יש ל-f ארבעה בנים).
4. הוסף ל-f אינדקס נוסף בהתאם לכללי עץ 2-3. (יתכן וכעת יש שלושה אינדקסים).
5. $v \leftarrow f$.
6. אם ל-v ארבעה בנים, פצל את v לשני צמתים v_1, v_2, וחבר אותם כבנים לאב f של צומת v תוך שמירה על סדר האינדקסים הנכון. חזור לצעד 5. אם v הוא שורש, צור צומת f אשר בניו הם הצמתים v_1 ו-v_2 וסיים. אם ל-v שלושה בנים, סיים.

תכונות תהליך ההוספה

1. שינויים עולים $O(1)$, ומתבצעים רק על המסלול מהשורש לעלה שהוסף.
2. בזמן פיצול של צומת, הצמתים החדשים הם בעומק שווה לצומת שפוצל.
3. בפיצול השורש נוצר צומת חדש שמגדיל ב-1 את העומק של כל הצמתים.

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

אלגוריתם להכנסת מפתח k (חלק ב)



5. $v \leftarrow f$.
6. אם ל-v ארבעה בנים, פצל את v לשני צמתים v_1, v_2, וחבר אותם כבנים לאב f של צומת v תוך שמירה על סדר האינדקסים הנכון. חזור לצעד 5. אם v הוא שורש, צור צומת f אשר בניו הם הצמתים v_1 ו-v_2 וסיים. אם ל-v שלושה בנים, סיים.

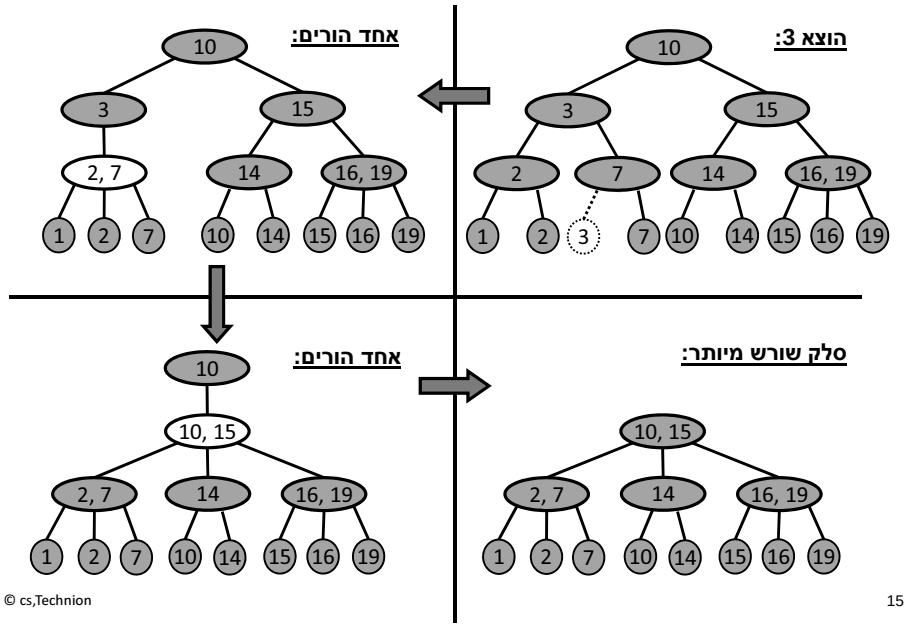
הוצאת מפתח k מעץ 2-3

בהוצאת איבר מעץ 2-3 מבוצעות הפעולות הבאות:

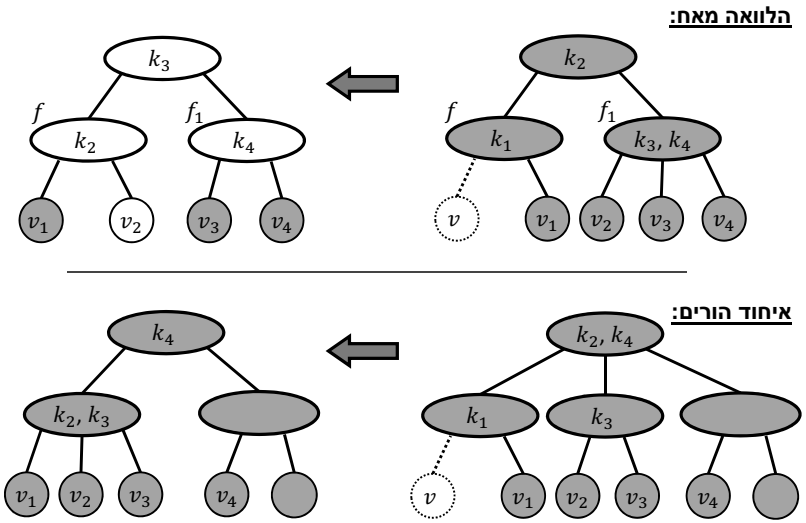
1. חיפוש מקומו של המפתח k.
2. הוצאת העלה שערכו k.
3. תיקון העץ כך שלכל צומת יהיו 2 או 3 בנים. התיקון נעשה במסלול החיפוש של k מהעלה שהוצא ועד השורש. בכל רמה נבצע $O(1)$ פעולות.

נמחיש זאת תחילה ע"י מספר דוגמאות.

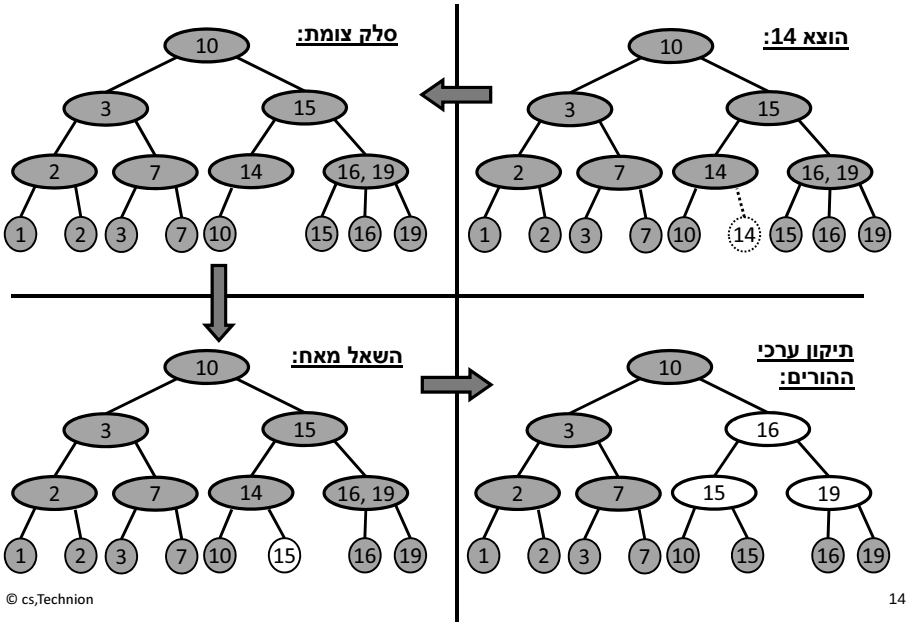
דוגמא להוצאה מעץ 2-3 (שני איחודים)



פרוט פעולות תיקון בזמן הוצאה



דוגמא להוצאה מעץ 2-3 (השאלה בלבד)



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. <u>מקרה א (השאלה מאח):</u> אם ל-v יש אח ולו שלושה בנים, שאל בן מהאח וסיים.

   <u>מקרה ב (איחוד הורים):</u> 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.

ניתוח פעולת ההוצאה

<u>נכונות</u>:

העומק של שום צומת אינו משתנה בזמן ההוצאה מלבד בסוף התהליך, במידה ומתבטל השורש, ואז 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.

<u>ניתוח זמן</u>:

1. כל פעולת חיפוש, הכנסה, והוצאה מעץ 2-3 דורשת $O(h)$ צעדי חישוב כאשר h הוא גובה העץ, וזאת כי בכל רמה מבוצעות $O(1)$ פעולות.
2. נזכר שעקב תכונות עץ 2-3 מתקיים כי $h = O(\log L)$, כאשר $L = n$ הוא מספר המפתחות.

סה"כ נקבל כי כל פעולה לוקחת $O(\log L) = O(\log n)$ צעדי חישוב.

השוואה בין עצי 2-3 ועצי AVL

שני סוגי העצים תומכים בפעולות המילון בזמן $O(\log n)$.
בעץ 2-3 רשומות נשמרות <u>רק בעלים</u>.
בעץ AVL רשומות נשמרות <u>גם בצמתים פנימיים וגם בעלים</u>.

לכל שימוש בעץ 2-3 ישנה כנראה אנלוגיה גם בעץ AVL ולהפך. לרוב השימושים הבחירה בין השניים אינה עקרונית.

מושג השמורה - Invariant

מושג השמורה משמש במדעי המחשב בשביל להוכיח נכונות של קוד ושאר תכונות מועילות שלו. מעשית, זו תכונה של הנתונים שנשמרת בין פעולה לפעולה, ומניבה תועלת כלשהי.

<u>דוגמאות אותן ראינו:</u>

- <u>שמורת עץ חיפוש בינארי:</u> כל צומת גדול מהצמתים בתת העץ השמאלי שלו וקטן מכל הצמתים בתת העץ הימני שלו.
  <u>תועלת:</u> חיפוש, הכנסה והוצאה בזמן $O(h)$, כש- h גובה העץ.

- <u>שמורת עץ AVL:</u> לכל צומת גורם האיזון בתחום $[-1,0,1]$.
  <u>תועלת:</u> $h = O(\log n)$. יחד עם שמורת עץ החיפוש הבינארי ותיקונים בזמן $O(h)$ אחרי הכנסה\הוצאה, מאפשר חיפוש, הכנסה והוצאה זמן $O(\log n)$.

- <u>שמורות עץ 2-3:</u> כל עלה באותה רמה, לכל צומת פנימי בין 2 ל-3 בנים.
  <u>תועלת:</u> חיפוש, הכנסה והוצאה ניתנים לביצוע בזמן $O(\log n)$ בעץ 2-3 (אם מעדכנים באמצעות איחוד, פיצול והלוואה מאח אחרי עדכונים).

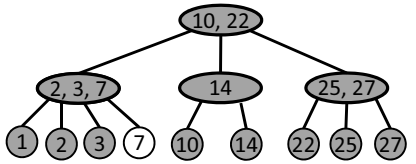
הכללה - עצי B^+ (מדרגה m)

<u>הגדרה</u>: 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 $\lceil m/2 \rceil \leq c \leq m$.

   לשורש מספר הבנים הוא $2 \leq c \leq m$.
3. לצומת פנימי בעל c בנים יש $c-1$ אינדקסים ממוינים לפי גודלם.
4. 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.

<u>דוגמא: עץ 2-3.</u>
<u>דוגמא: עץ 2-3-4.</u>

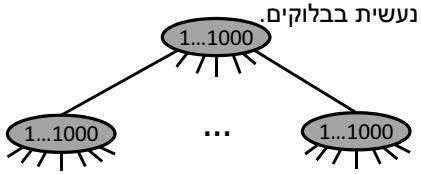


<u>חיפוש/הכנסה/הוצאה:</u> כמו בעץ 2-3.
כלומר, חיפוש לפי הגדרת עץ B^+ מדרגה m, הכנסה או הוצאה של העלה המבוקש, ותיקון העץ מהעלה ועד השורש במסלול החיפוש כך שמספר הבנים בכל רמה יעמוד בדרישות עץ B^+ בדרגה m.

שימוש עיקרי לעץ B^+ (מסדי נתונים)

הזיכרון ברוב המחשבים מורכב משני חלקים:
זיכרון ראשי (RAM), קטן אך מהיר.
זיכרון משני (DISK), גדול, אך איטי יחסית לזיכרון הראשי.

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.



<u>עצי B^+ מדרגה גבוהה מספקים פתרון:</u>
בכל צומת פנימי נשמור כמספר הבלוקים של מפתחות שעדיין מאפשרים חיפוש מהיר בזיכרון הראשי.
מספר הגישות לזיכרון המשני תלוי במספר הרמות (וזהו מספר נמוך) הנקבע ע"י גודל הבלוק.
המחיר יהיה בסקירת המפתחות על מנת לקבוע את המשך החיפוש בעץ (תוך שימוש בזיכרון המהיר).

עצי דרגות - 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:

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

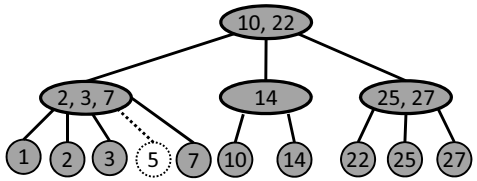
<u>בעיה I</u>: לממש עץ חיפוש התומך, בנוסף לפעולות הכנסה\הוצאה\חיפוש, גם במציאת האינדקס של x בעץ בזמן $O(h)$ כאשר h הוא גובה העץ.

<u>בעיה II</u>: לממש עץ חיפוש התומך בחישוב סכום האיברים בעץ הקטנים מ– x בזמן $O(h)$.

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.

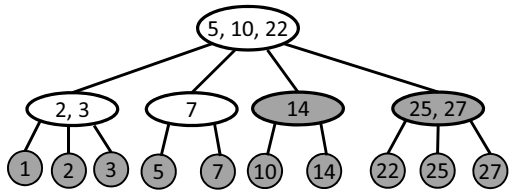
דוגמא להכנסה לעץ B^+ מדרגה 4 (עץ 2-3-4)

<u>דוגמא: הכנס 5:</u>



במקרה זה נדרש פיצול בודד.

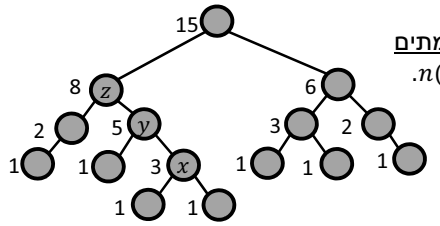
מחצית הצמתים (מעוגל כלפי מעלה) בענף השמאלי ומחצית (מעוגל כלפי מטה) בימני.



דוגמאות נוספות בתרגול.

עץ דרגות (Rank Tree)

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



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

- ערך זה מופיע משמאל לצמתים באיור

יהא $path$ מסלול החיפוש של x. לכל צומת v שקטן או שווה מ- x ב- $path$ נספור את v וכל הצמתים בתת העץ השמאלי של v. בעזרת השדה $n(v)$ ניתן לחשב זאת כך:

$rank(x) = |\{z, y, x\}| + 2 + 1 + 1 = 7.$

$rank(x) = \sum_{v \in path,\ v \le x} (1 + n(v.left)).$

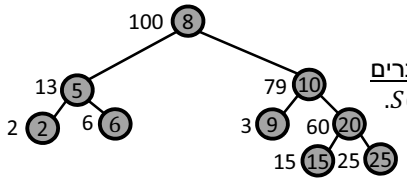
הגדרה: **עץ דרגות** (**rank tree**) הוא עץ בו בכל צומת v נשמרים מספר הצמתים בתת-העץ ששורשו v.

מסקנה: בעץ דרגות מציאת ה- rank דורשת זמן $O(h)$.

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

עץ דרגות (Rank Tree)

בעיה II: לממש עץ חיפוש התומך במציאת סכום האיברים הקטנים מאיבר x בזמן $O(h)$.



פתרון: נשמור בכל צומת v גם את סכום האיברים בתת-העץ ששורשו v. נסמן מספר זה ב- $S(v)$.

יהא $path$ מסלול החיפוש של x. לכל צומת v שקטן או שווה מ- x ב- $path$ נוסיף את ערכו של v ושל כל הצמתים בתת העץ השמאלי של v. בעזרת השדה $S(v)$ ניתן לחשב זאת כך:

דוגמא: סכום האיברים הקטנים מ- 17 בעץ הנ"ל הוא:
$(8 + 13) + (10 + 9) + (15 + 0) = 55$

$SumTo(x) = \sum_{v \in path,\ v \le x} (v + S(v.left)).$

מסקנה: זמן הריצה הוא $O(h)$.
כיצד נבטיח זמן ריצה $O(\log n)$?

עץ דרגות מאוזן (Balanced Rank tree)

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

פתרון: לדוגמא, באמצעות עץ AVL. נשמור בכל צומת v את מספר הצמתים בתת-העץ ששורשו v. זיכרו שסימנו מספר זה ב- $n(v)$.

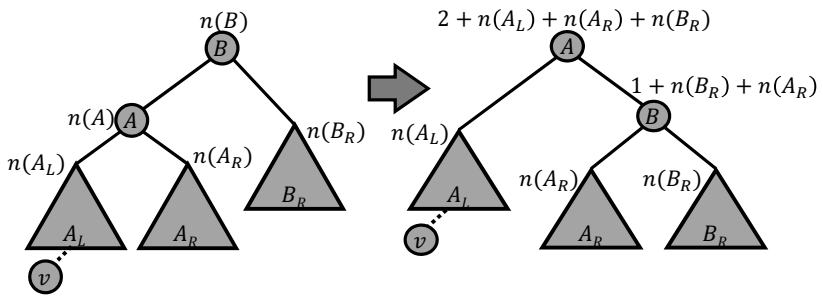
נדאג לעדכן את $n(v)$ בזמן הכנסות והסרות ובזמן גלגולים.

עדכון השדה הנוסף

כיצד נעדכן את $n(v)$ – מספר הצמתים בתת העץ – בכל תת-עץ של עץ AVL ?

1. בזמן הכנסה נגדיל את $n(v)$ ב-1 לאורך המסלול מהשורש לעלה שהוכנס.
   בזמן הסרה נקטין את $n(v)$ ב-1 לאורך המסלול מהשורש לצומת שהוסר.
2. בזמן גלגול נעדכן את השדה כנדרש, תוך שימוש בעובדה ש-$n(v)$ תלוי רק בערכי $n(v)$ של תת העץ, שכבר תוקנו.

למשל בגלגול LL נעדכן כך:

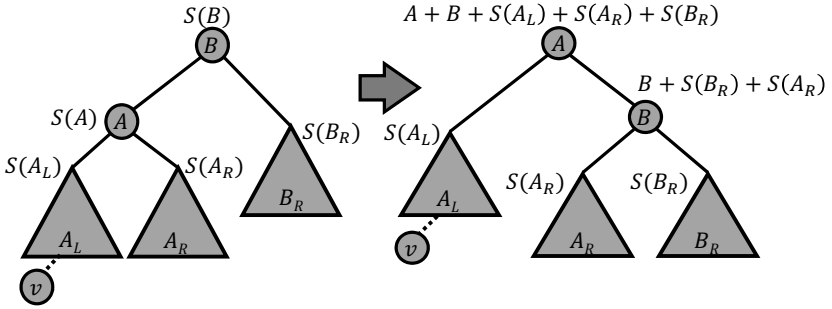


דוגמא נוספת

כיצד נעדכן את $S(v)$ – <u>סכום</u> המפתחות בתת העץ – בכל תת-עץ של עץ AVL ?

באופן דומה לאופן עדכון $n(v)$ בשקף הקודם - בזמן הכנסה\הסרה נעדכן את הסכום לאורך המסלול מהשורש לעלה שהוכנס\צומת שהוסר. בזמן גלגול נעדכן את השדה כנדרש.

<u>למשל בגלגול LL נעדכן כך:</u>



<u>תרגיל בית</u>: מהו העדכון בגלגולים האחרים? כיצד ניתן לתחזק שדות $n(v)$ ו-$S(v)$ בעץ 3-2?

© cs,Technion

© cs,Technion