מבני נתונים

תרגול 5, סמסטר ב' תשע"ז

עצי AVL

עצי WAVL

עצי AVL / AVL Trees

עץ AVL

- עץ חיפוש בינארי מאוזן
- ההפרש בין הגבהים של שני תתי העצים שלו הוא לכל היותר 1.
- סוגי צמתים חוקיים:
  1,1 או 1,2 או 2,1.
- Rank = height

שאלה 1

- נתון עץ AVL בגובה k.
- הראו כי אורך המסלול הקצר ביותר בין שורש העץ לבין עלה כלשהו בעץ הוא לפחות k/2.

פתרון 1

- נסמן ב a_k את אורך המסלול הקצר ביותר בין צומת v בגובה k לבין עלה צאצא שלו.

- קל לראות כי מתקיימת הנוסחא: $a_k \geq 1 + a_{k-2}$
- כמו כן $a_0 = 0 = 0/2, a_1 = 1 > 1/2$

- באינדוקציה נובע:

$$a_k \geq 1 + a_{k-2} \geq 1 + 1 + a_{k-4} \geq \cdots \geq k/2$$

שאלה 2

- נתון עץ AVL. לכל צומת v, נסמן ב **size(v)** את מספר הצמתים בתת עץ ששורשו v (כולל v).

- יהיו L, R בניו של השורש ונתון שאינם עלים חיצוניים. ונניח כי L צומת בגובה k.

- הראו חסם עליון על היחס בין size(R) ל - size(L).

פתרון 2

- הראינו בהרצאה שמספר הצמתים המינימלי בעץ בגובה k הוא לפחות $\boldsymbol{\phi^k}$ כאשר $\phi \approx 1.618$.
- מספר הצמתים בעץ מלא בגובה k הוא: $1 + 2 + 4 + \cdots + 2^k = \mathbf{2^{k+1}}$
- מניחים ש- L צומת בגובה k <= מתכונת עצי AVL, R הוא צומת בגובה לכל היותר k+1 .
- מכאן ש:

$$\frac{size(R)}{size(L)} \le \frac{2^{k+2}}{\phi^k} = 4\left(\frac{2}{\phi}\right)^k$$

שאלה 3

- נתון עץ AVL ריק. מבצעים עליו n פעולות insert, כאשר בפעולת insert נתון רק מצביע לשורש העץ והאיבר החדש.

(א) מהו זמן הריצה הכולל במקרה הגרוע? (חסם הדוק)

(ב) מהו מספר ה rotations במקרה הגרוע? (חסם עליון)

(ג) מהו מספר ה promotions במקרה הגרוע? (חסם עליון)

פתרון 3(א)

- זמן הריצה של הכנסה בודדת הוא O(logn) ולכן של n הכנסות הוא O(nlogn).

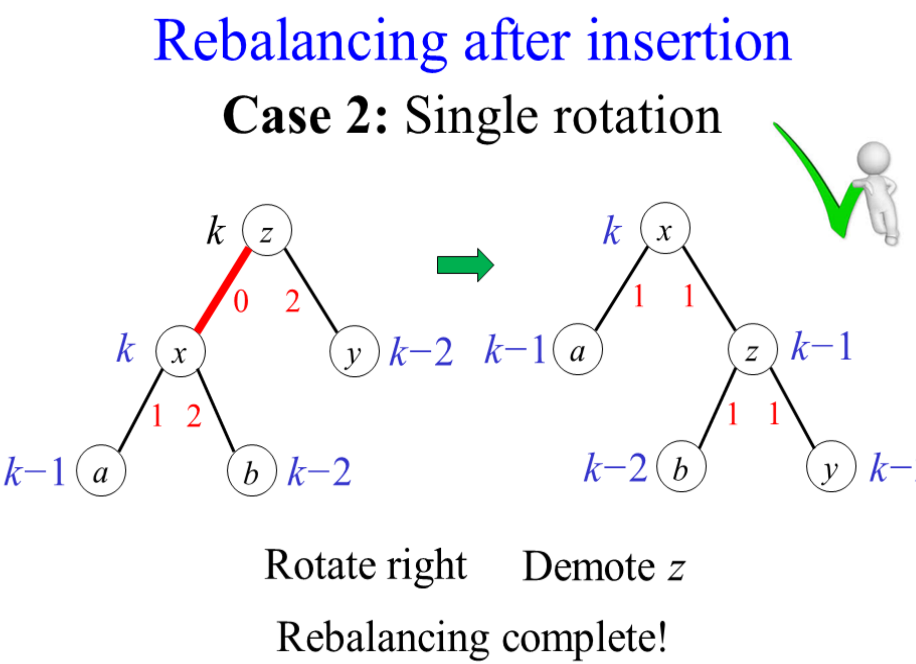
- נראה שחסם זה הדוק:

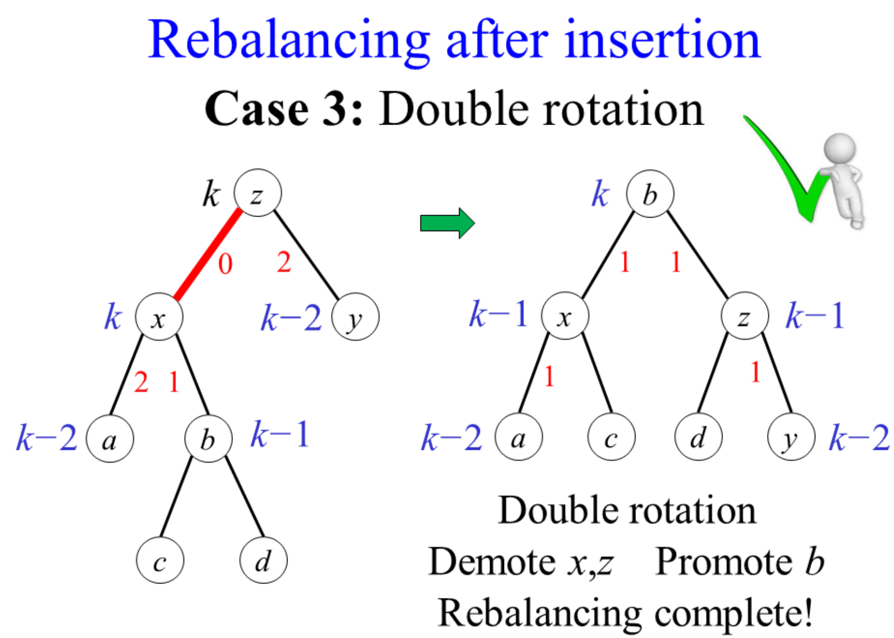
- נסתכל על סדרה שבה כל איבר חדש מוכנס לעלה החיצוני הרחוק ביותר מהשורש.
- מאחר וגובה העץ לוגריתמי במספר הצמתים בעץ, נקבל:

$$\sum_{i=1}^{n} \log i \geq \sum_{i=n/2}^{n} \log i \geq (n/2)\log(n/2) = \Omega(n \log n)$$

פתרון 3(ב)

- בכל פעולת insert מתבצעות לכל היותר שתי קריאות לפונקציה rotate. קיבלנו חסם עליון של O(n).





פתרון 3(ג)

- ראינו (בהרצאה) כי זמן ריצה amortized של n פעולות insert, **כאשר ידוע מקום ההכנסה**, הוא O(1), ומכאן שחסם worst case על n פעולות כאלה הוא O(n).

- מכיוון שהפונקציה promote נקראת רק לאחר שמקום ההכנסה כבר נמצא, חסם זה תקף גם עבור מספר פעולות ה promote שיתבצעו במקרה שלנו.

שאלה 4

- ראינו כי זמן הריצה של פעולת ה insert, כאשר ידוע מקום ההכנסה, הוא O(1) amortized.
- א. הראו כי כאשר מדובר בסדרת פעולות כללית של insert ו delete, גם כאשר ידוע מקום ההכנסה או המחיקה, זמן הריצה amortized יכול להיות $\Omega(\log n)$.
- ב. הראו כי כאשר מדובר בסדרה של n פעולות insert ולאחריהן n פעולות delete (במקומות ידועים), זמן הריצה amortized הוא O(1).

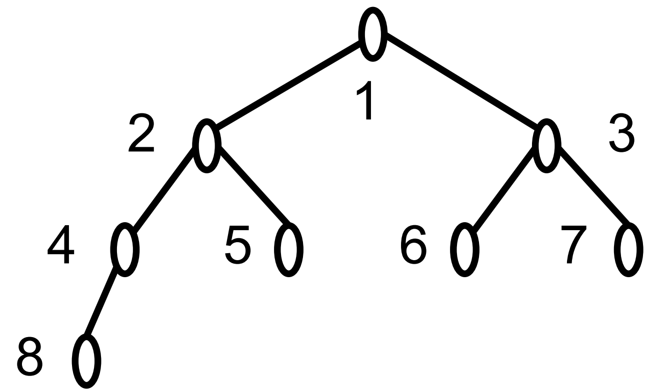
פיתרון 4 – סעיף א

- נראה שקיימת סדרה של n פעולות insert/delete, עם זמן ריצה כולל של $\Theta(n\log n)$.
- זה יראה כי בהכרח זמן ריצה worst case של סדרה כלשהי של n פעולות הוא $\Omega(n\log n)$, וזה מוכיח את הנדרש.

תיאור סדרת הפעולות:

- תחילה נבצע n/2 פעולות הכנסה, כך שיתקבל עץ מלא בגובה log(n/2).
- עובדה חשובה: **לכל** עץ AVL קיימת סדרת הכנסות שיוצרת אותו: יש להכניס את איברי העץ לפי גובהם בעץ שרוצים ליצור, ניתן לראות כי לא יתרחשו גלגולים בדרך – ולכן העץ שיווצר הוא כנדרש.

(המספרים באיור מתארים את סדר ההכנסה של האיברים)



<u>תיאור סדרת הפעולות</u>:

- תחילה נבצע $n/2$ פעולות הכנסה, כך שיתקבל עץ מלא בגובה $\log(n/2)$.

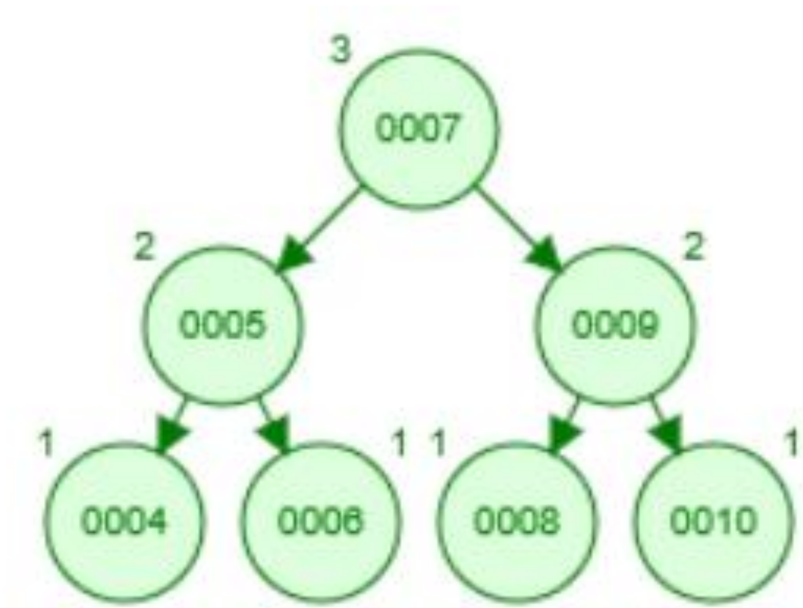
- לאחר מכן, נוסיף ונמחק לסירוגין איבר שערכו קטן יותר מכל האיברים שנמצאים כבר בעץ (סה"כ $n/2$ פעמים).

ניתוח סיבוכיות:

- קל לוודא שכל הכנסה/מחיקה כזאת לוקחת $\Theta(\log n)$ - מתבצע מספר כזה של promote/demote בכל הכנסה/מחיקה.

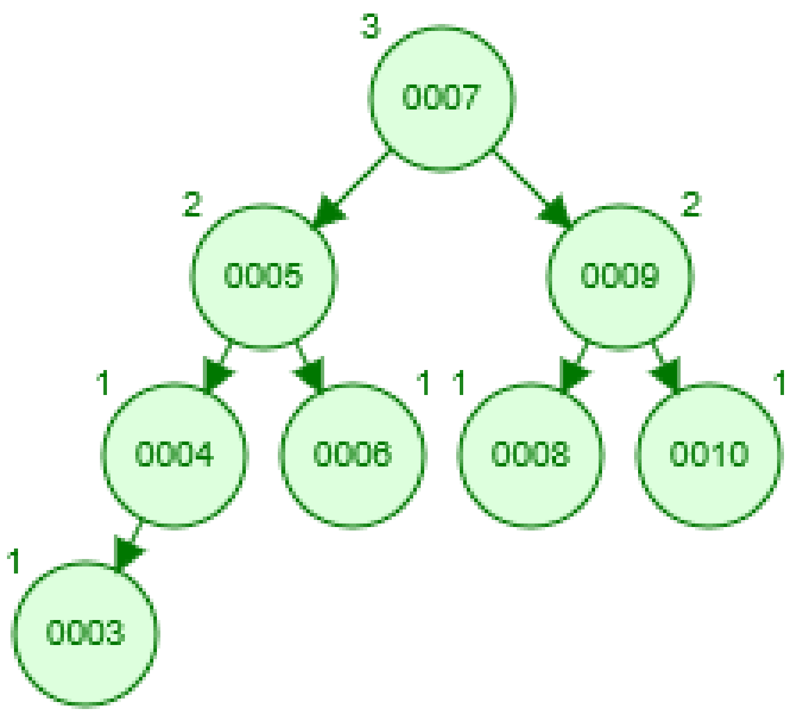
- סה"כ נקבל זמן ריצה .w.c של $\Theta(\log n)$ על כל הסדרה.

Starting with the AVL tree



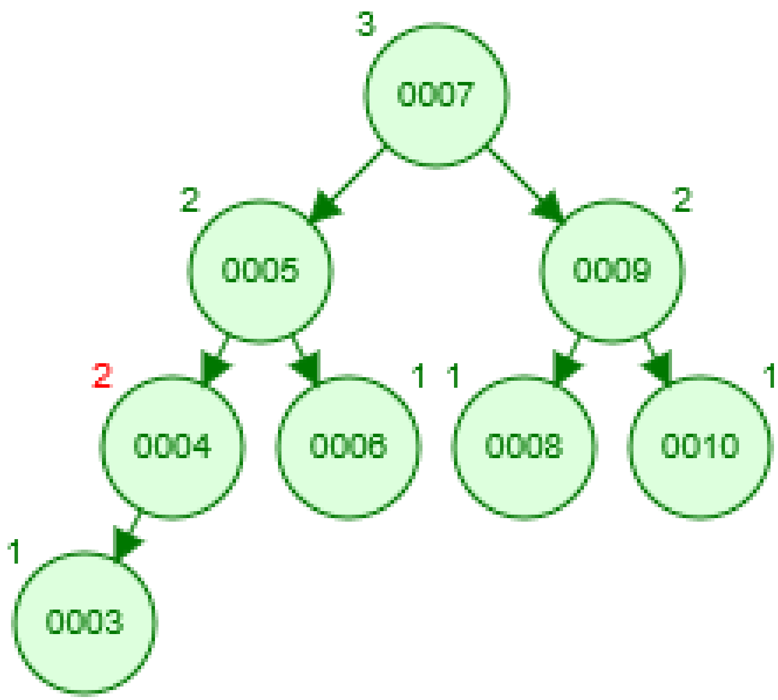
https://www.cs.usfca.edu/~galles/visualization/AVLtree.html

Insert(3)



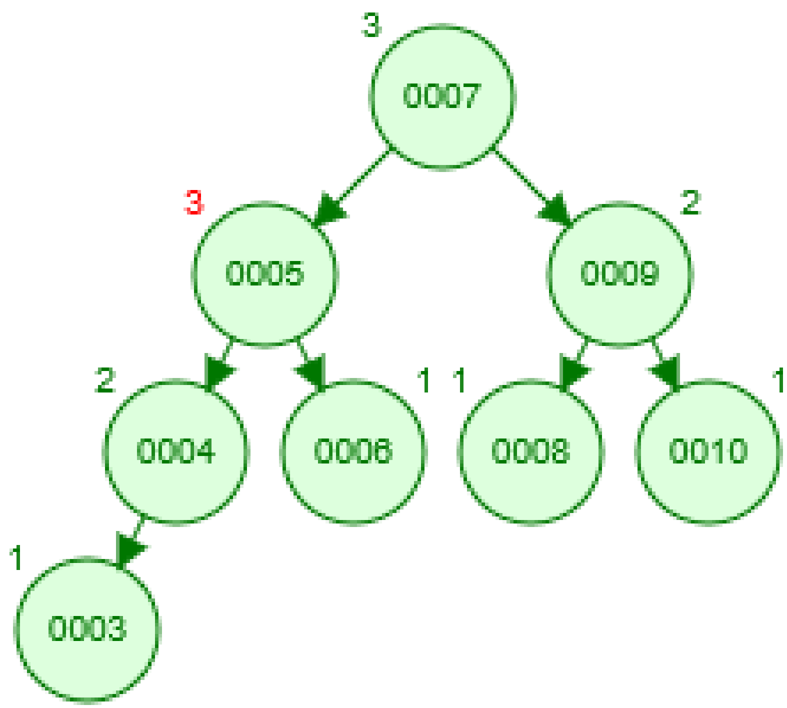
https://www.cs.usfca.edu/~galles/visualization/AVLtree.html

Insert(3)



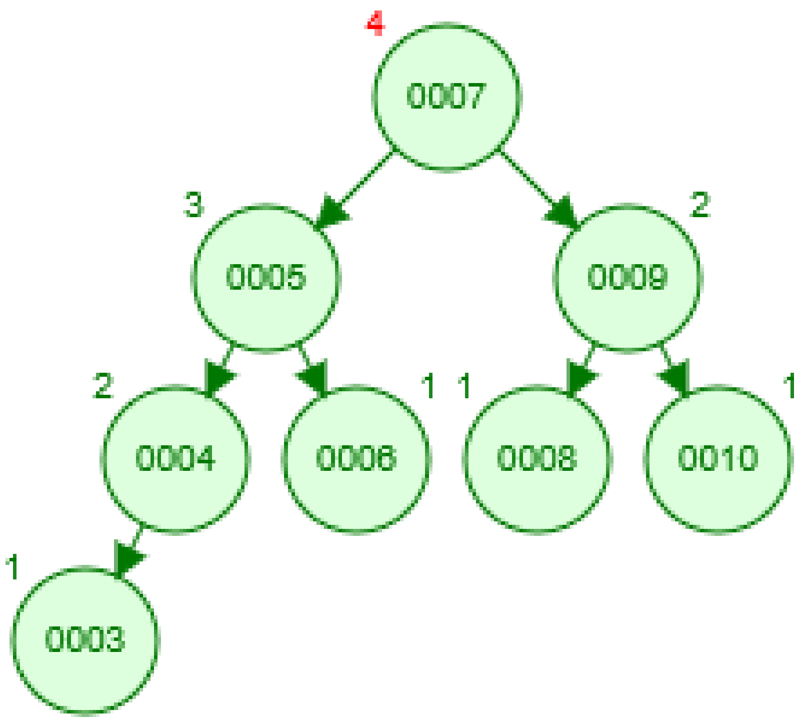
https://www.cs.usfca.edu/~galles/visualization/AVLtree.html

Insert(3)



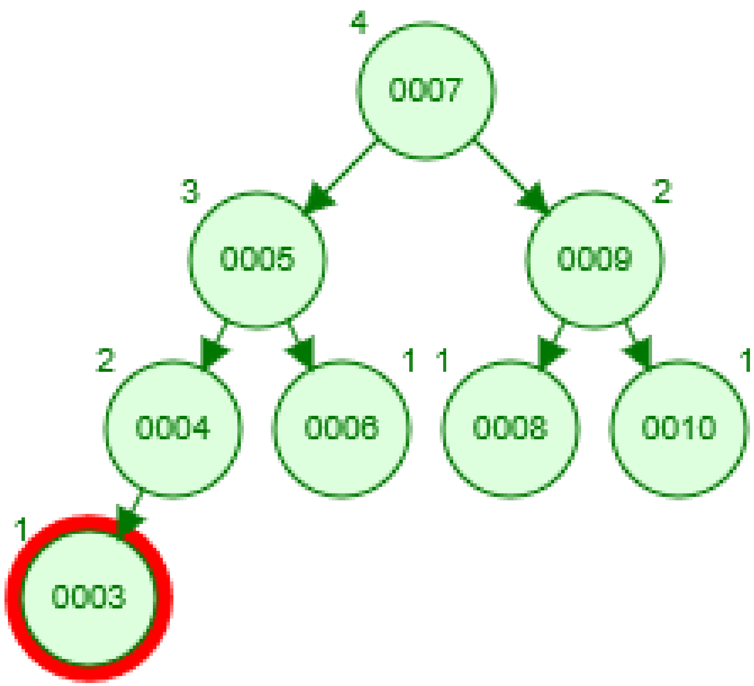
https://www.cs.usfca.edu/~galles/visualization/AVLtree.html

Insert(3)



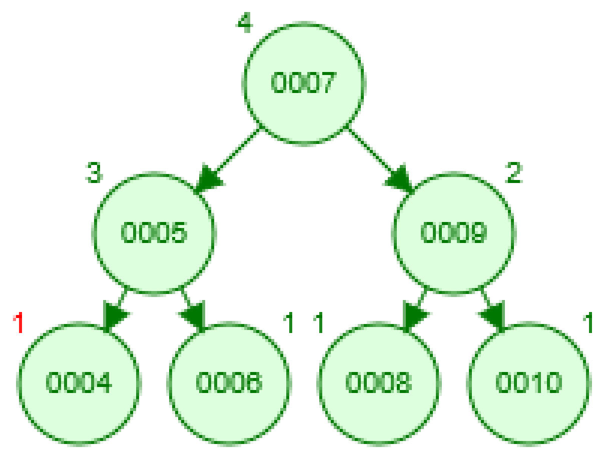
https://www.cs.usfca.edu/~galles/visualization/AVLtree.html

delete(3)



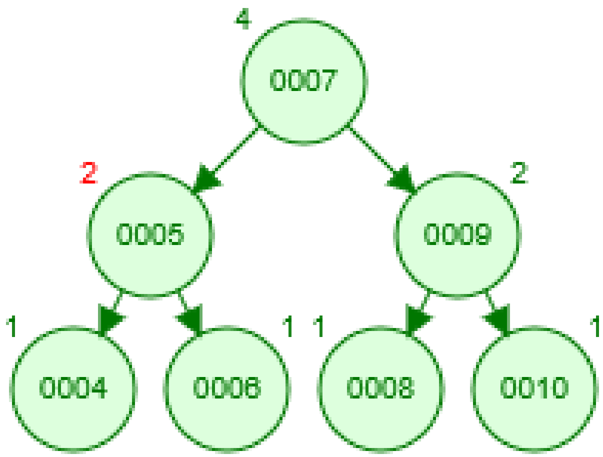
https://www.cs.usfca.edu/~galles/visualization/AVLtree.html

delete(3)



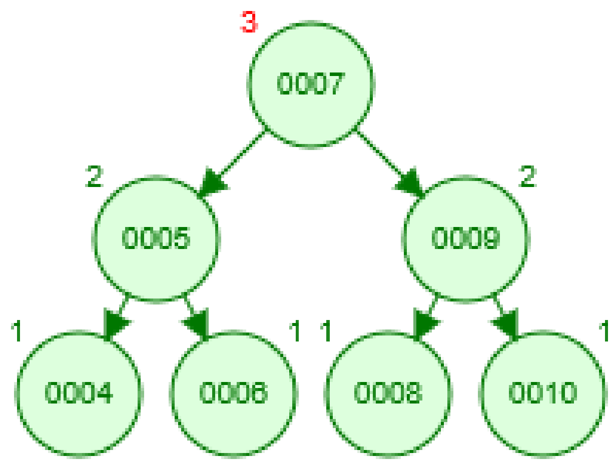
https://www.cs.usfca.edu/~galles/visualization/AVLtree.html

delete(3)



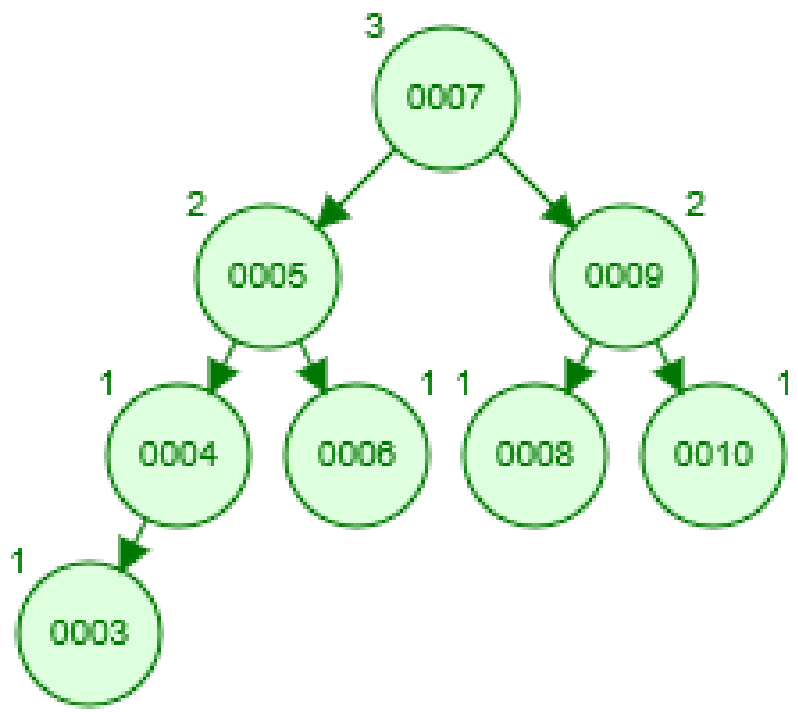
https://www.cs.usfca.edu/~galles/visualization/AVLtree.html

delete(3)



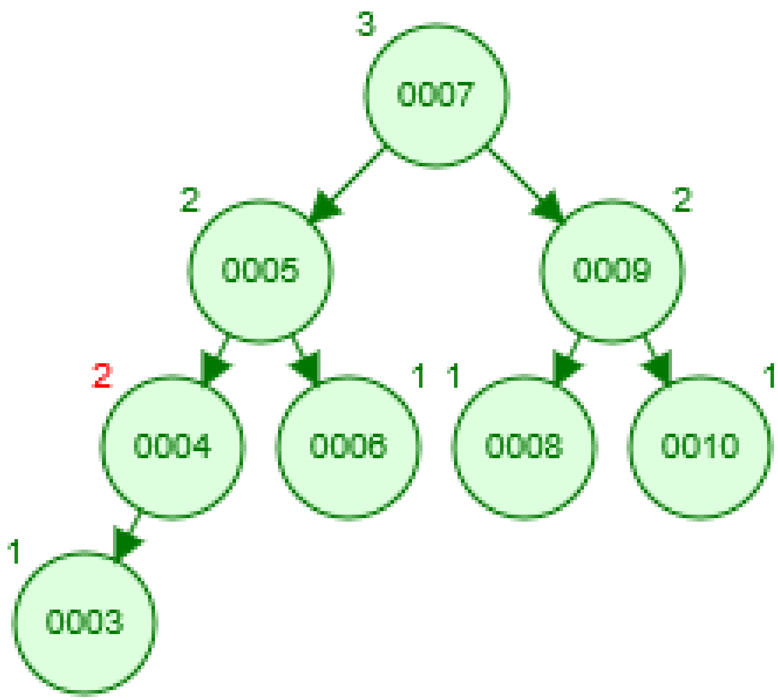
https://www.cs.usfca.edu/~galles/visualization/AVLtree.html

Insert(3)



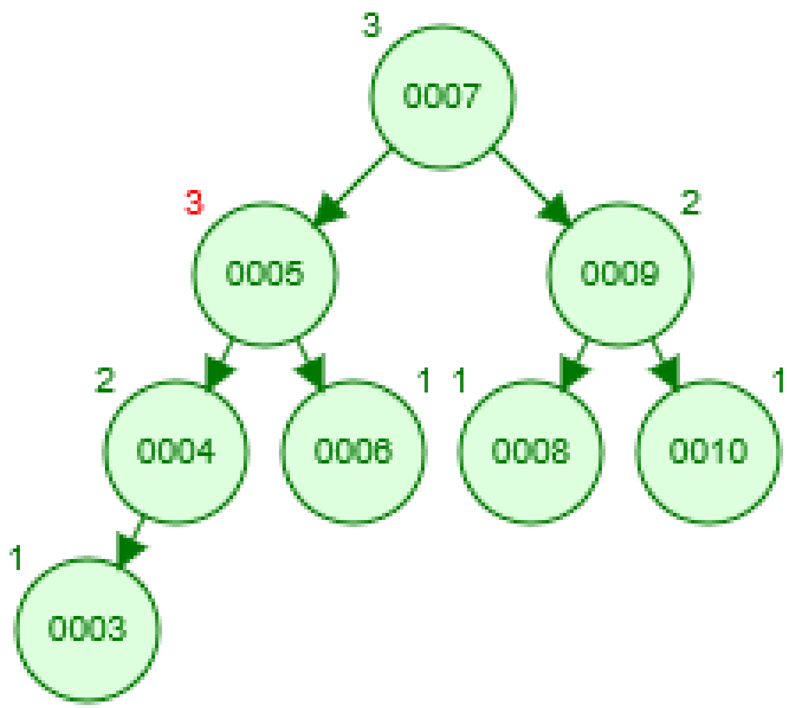
https://www.cs.usfca.edu/~galles/visualization/AVLtree.html

Insert(3)



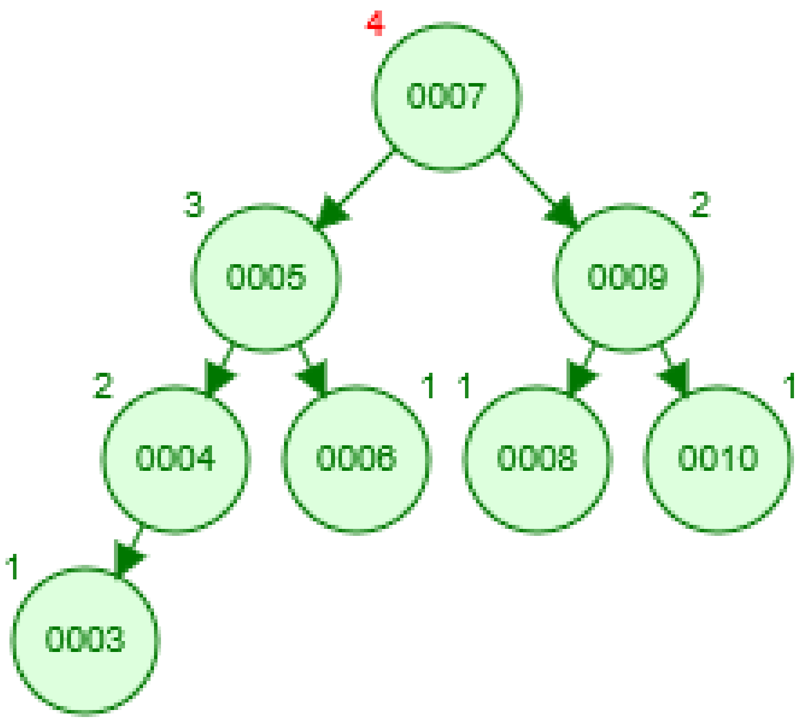
https://www.cs.usfca.edu/~galles/visualization/AVLtree.html

Insert(3)



https://www.cs.usfca.edu/~galles/visualization/AVLtree.html

Insert(3)



https://www.cs.usfca.edu/~galles/visualization/AVLtree.html

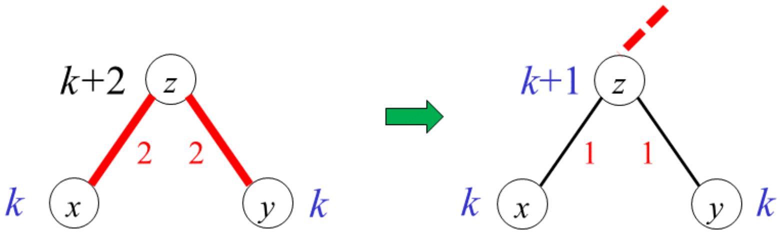
פיתרון 4 – סעיף ב

- ראינו כי סדרה של n הכנסות (במקומות ידועים) תיקח זמן $O(n)$.

- כעת נניח כי יש לנו עץ AVL חוקי בעל n איברים וננסה למצוא פונקציית פוטנציאל מתאימה: צריכים שהפוטנציאל **<u>יקטן</u> בכל פעולת איזון מחדש שהיא לא סופית – וכך הוא ישלם על הפעולה.**
  - זמן הריצה של הפעולות ההתחלתיות והסופיות – כלומר, המחיקה בהתחלה ופעולת האיזון האחרונה - הוא $O(1)$.
  - בפעולות ההתחלתיות והסופיות אנו מרשים שהפוטנציאל יגדל בקבוע.

AVL: Rebalancing after deletion

Case 1: Demote

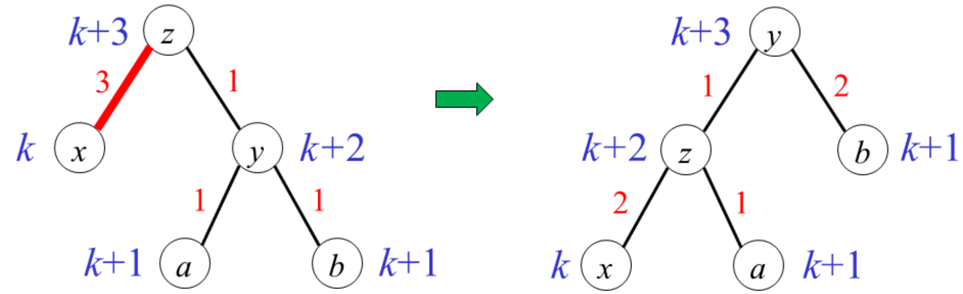


Demote z, i.e., decrease its *rank* by 1

Problem is either fixed or moved up

AVL: Rebalancing after deletion

Case 2: Single Rotation (a)

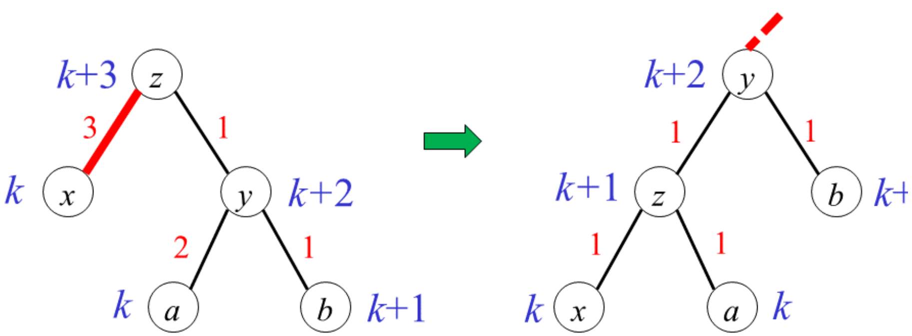


Rotate left

Demote z Promote y

AVL: Rebalancing after deletion

Case 3: Single Rotation (b)



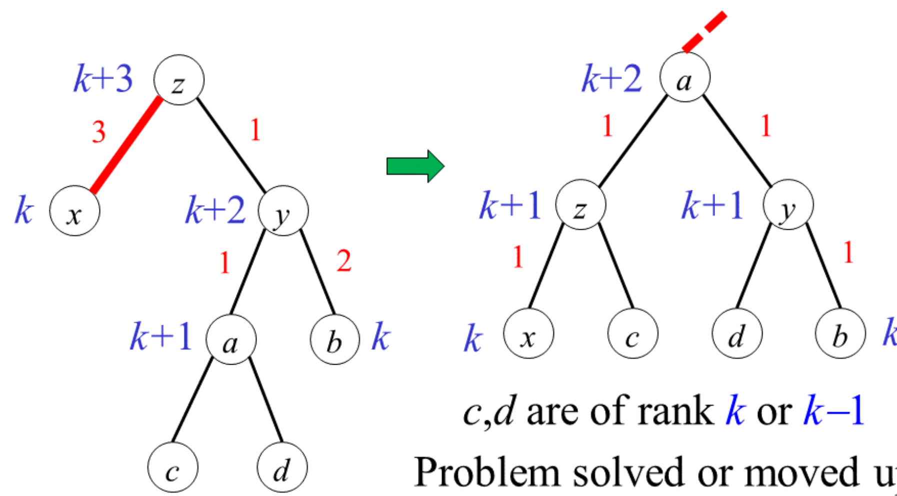
Rotate left

Demote z twice

Problem solved or moved up

AVL: Rebalancing after deletion

Case 4: Double Rotation



c,d are of rank k or $k-1$

Problem solved or moved up

פיתרון 4 – סעיף ב

- המקרים שאינם סופיים בשקף הקודם: 1,3,4.
- נשים לב כי בכל אחד מהמקרים הנ"ל כמות הקדקדים מסוג 2,1 או 1,2 קטנה יותר לאחר האיזון מאשר לפני שקרתה פעולת המחיקה.
  - בכל המקרים, עלינו לתקן קדקדים שהפכו להיות מסוג 1,3 או 2,2 או 1,3 תוך כדי איזון העץ שמתבצע בעקבות פעולת מחיקה.
  - לפני המחיקה, הקדקדים הללו היו מסוג 1,2 או 2,1.
- נגדיר את הפוטנציאל להיות מספר הקדקדים מסוג 2,1 או 1,2.
  - הפוטנציאל יקטן בכל פעם שמבצעים פעולת איזון לא סופית.
  - הפוטנציאל יגדל בקבוע (לכל היותר) בכל פעם שמבצעים פעולה התחלתית או סופית.

פיתרון 4 – סעיף ב

- עדיין לא סיימנו – פונקציית הפוטנציאל שהגדרנו בשקף הקודם לא חוקית!
  - זכרו כי פונקציית פוטנציאל חוקית שווה ל-0 במצב ההתחלתי.
  - כאן במצב ההתחלתי יש לנו n קדקדים בעץ, ולכן הפוטנציאל ההתחלתי לכל היותר n (ולפחות 0).
- אבל אפשר להתמודד עם זה.
  - ניזכר באי השוויון הבא שהוכחנו עבור פונקציית פוטנציאל חוקית.
    בצד שמאל – זמן הריצה ה"פוטנציאלי" ובצד ימין – זמן הריצה האמיתי.

$$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} (c_i + \Phi(D_i) - \Phi(D_{i-1})) = \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0) \geq \sum_{i=1}^{n} c_i$$

  - אצלנו שני השוויונות השמאליים עדיין נכונים, ומתקיים
    $$\phi(D_n) - \phi(D_0) \geq \phi(D_n) - n \geq -n$$
  - לכן זמן הריצה הפוטנציאלי הוא לפחות זמן הריצה האמיתי פחות n.
  - כיוון שזמן הריצה הפוטנציאלי הוא $O(n)$, גם זמן הריצה האמיתי הוא $O(n)$.

WAVL Trees / עצי WAVL

עץ WAVL

- עץ חיפוש בינארי מאוזן
- סוגי צמתים חוקיים:

  1,1 או 1,2 או 2,1 או 2,2
- ה – rank של כל עלה הוא 0
- $\text{height} \le \text{rank} \le 2\text{height}$
- $\text{height} \le \text{rank} \le 2\log n$

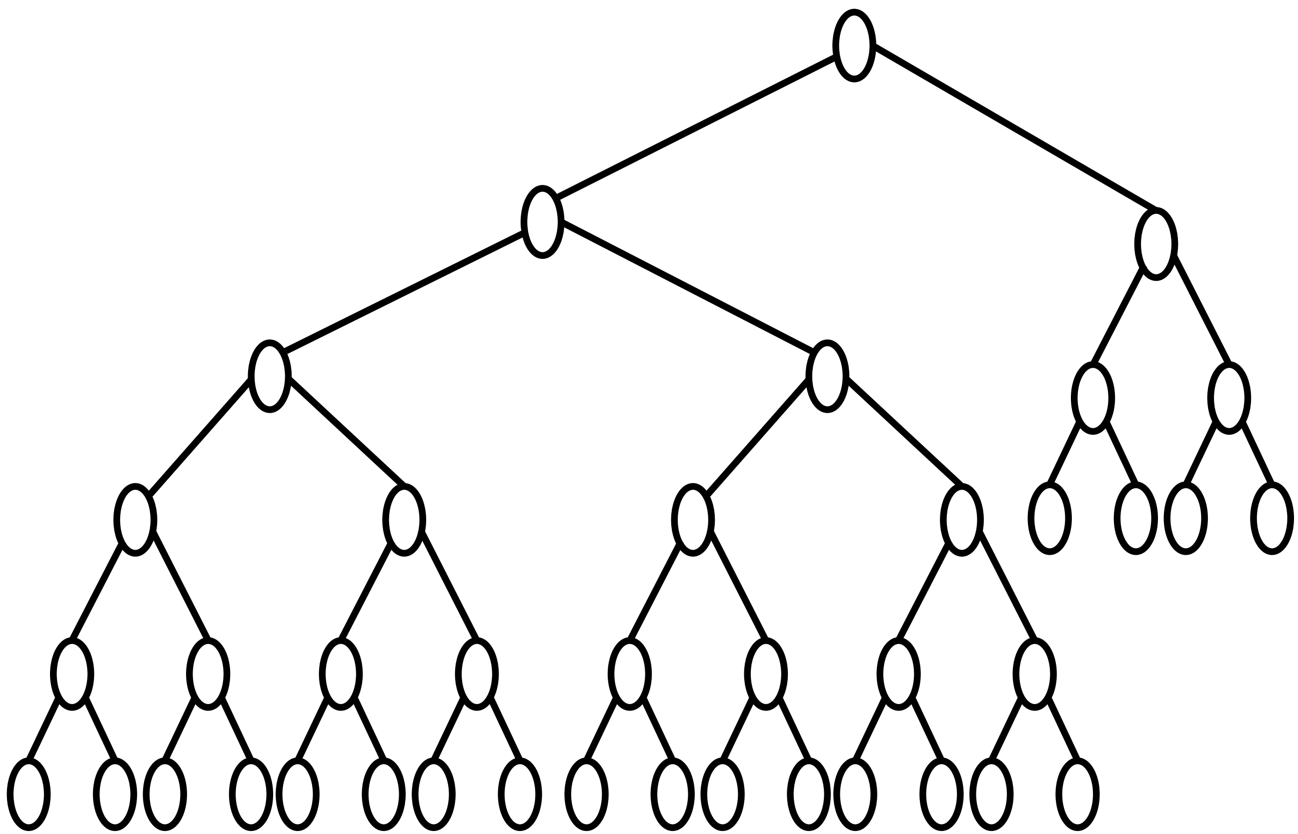
עץ WAVL

- לכל היותר 2 rotation בהכנסה ובהוצאה
- Rebalancing בעלות amortized של O(1)
- Insert פועל באופן זהה ל – AVL (אין צמתי 2,2 חדשים)
- עבור סדרה של הכנסות בלבד אין הבדל בין AVL ו - WAVL

שאלה 5

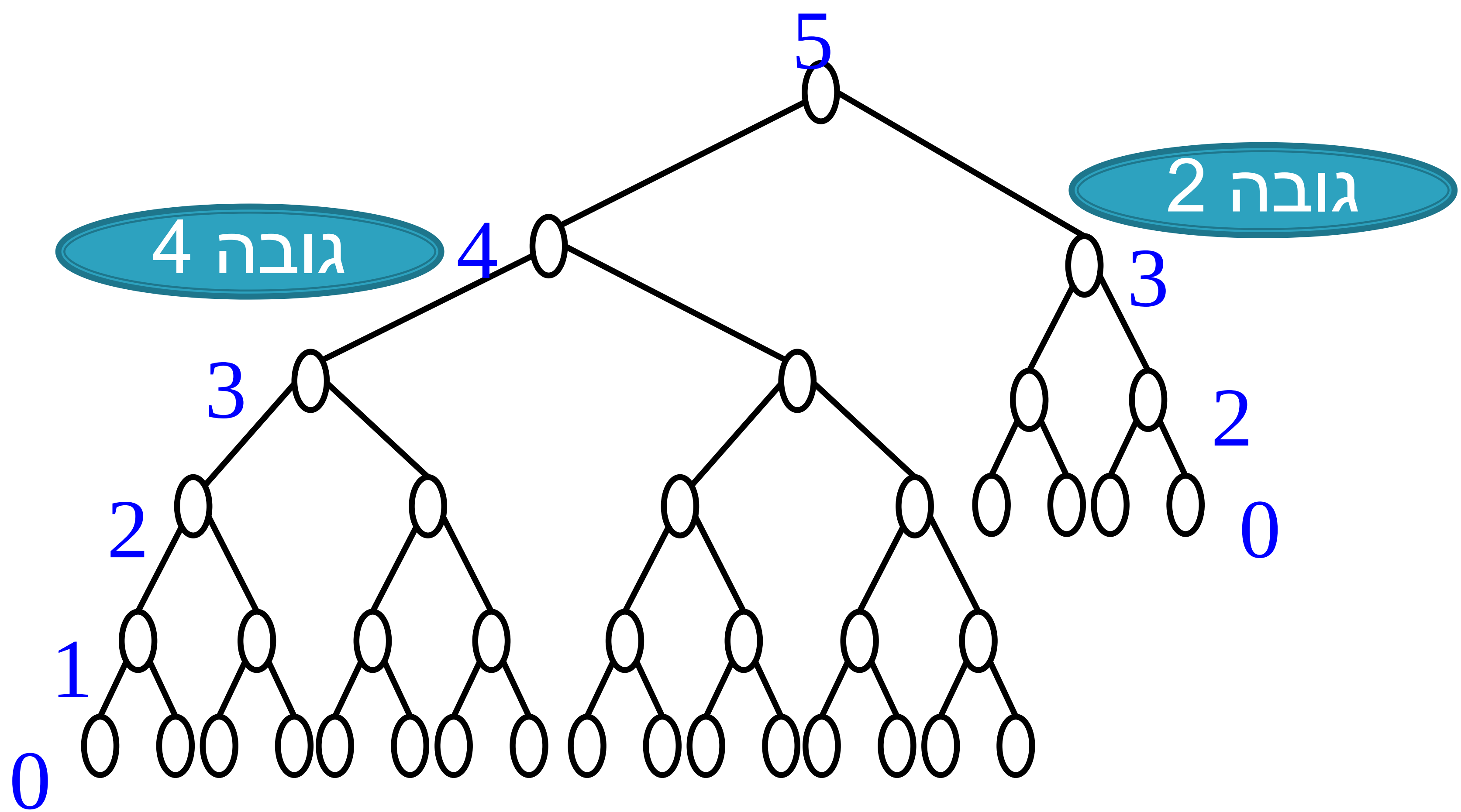
האם קיים עץ WAVL חוקי שאינו עץ AVL חוקי (כאשר מותר לשנות את ערכי ה – rank של הצמתים)?

שאלה 6 - פתרון



שאלה 6 - פתרון

* עץ WAVL חוקי – הנה דרגות מתאימות.
* **<u>אינו</u>** עץ AVL חוקי – הפרש הגבהים של שני תתי העצים של השורש גדול מ- 1.

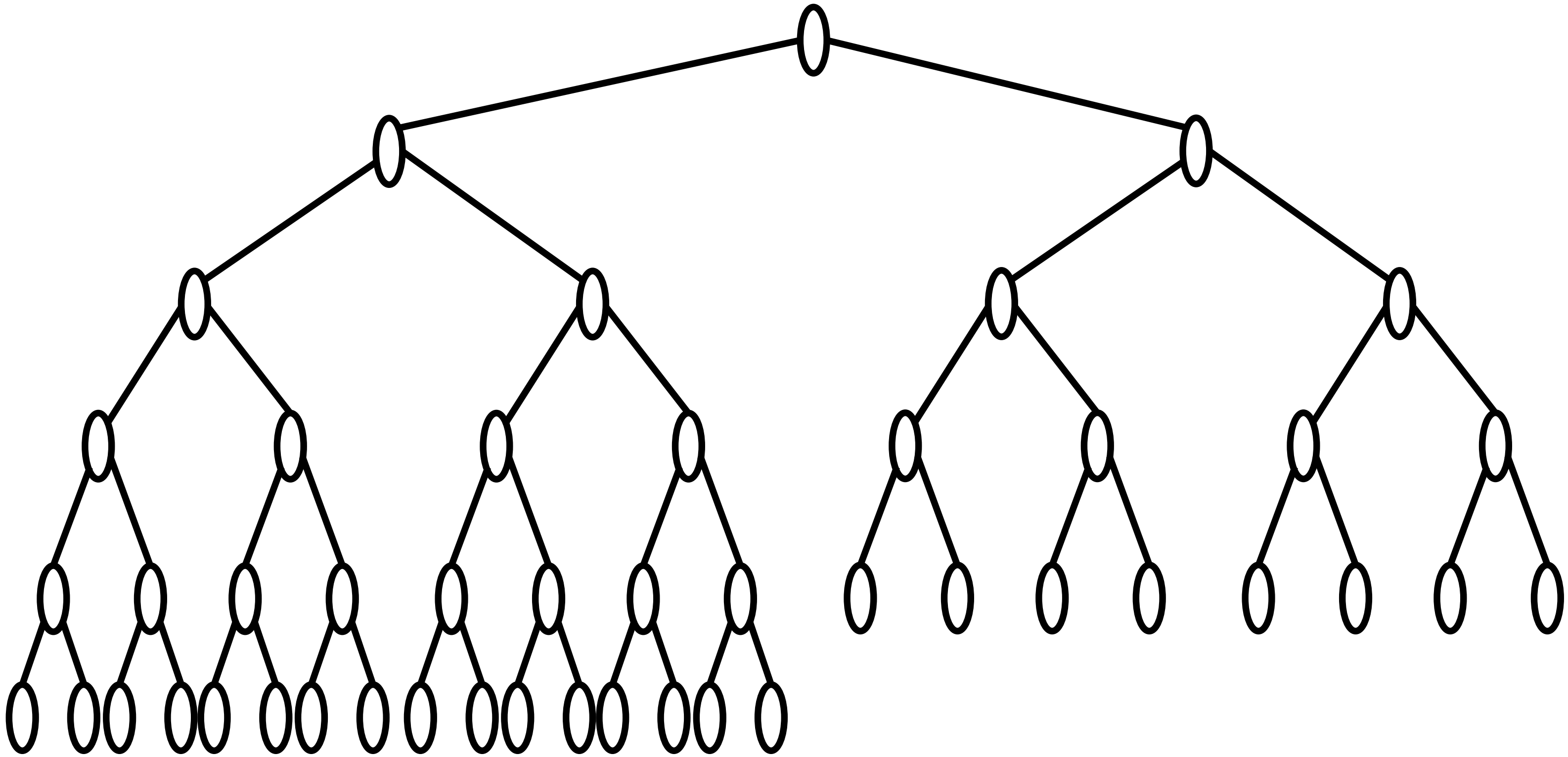


שאלה 6

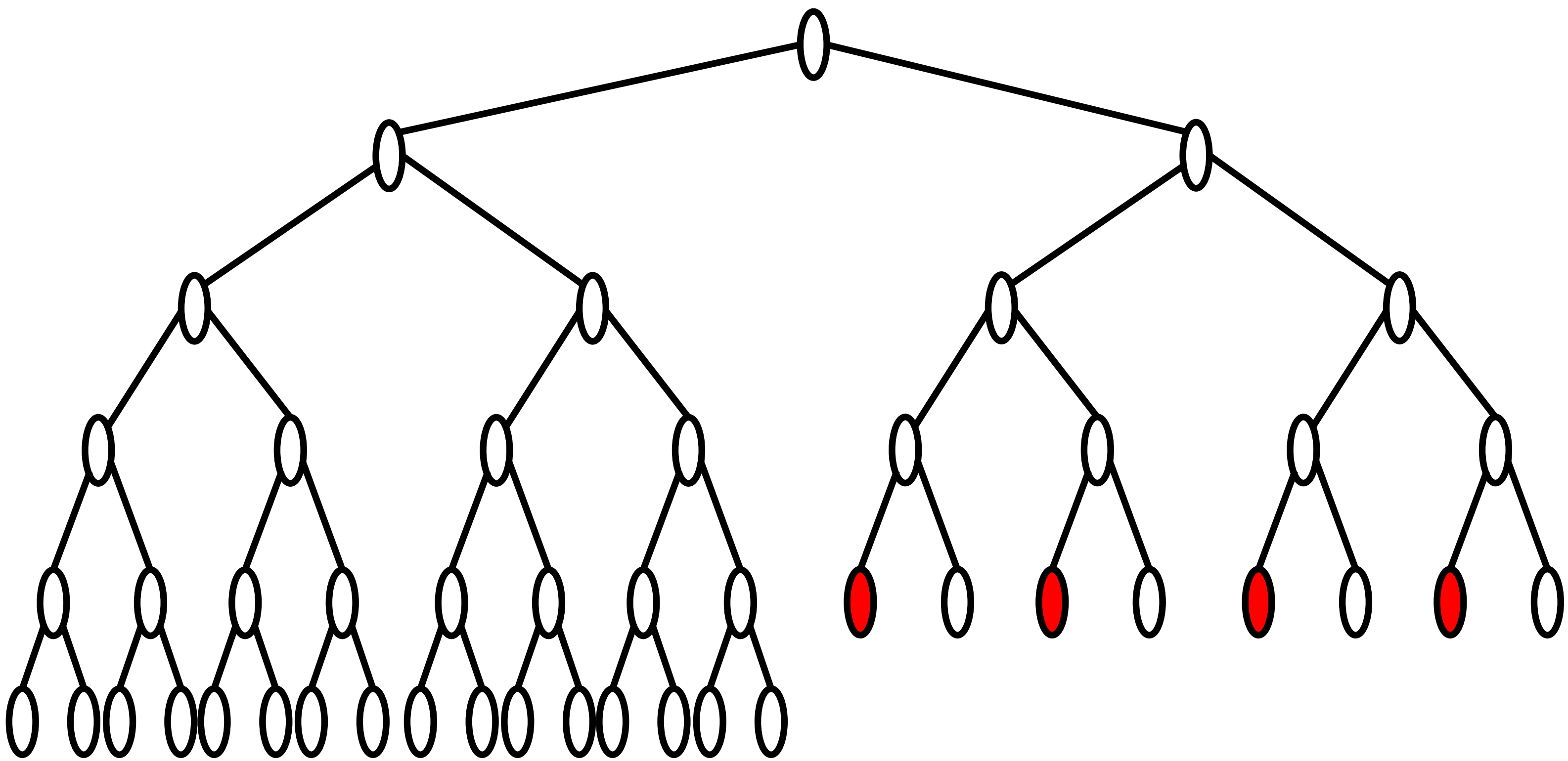
האם ניתן לבנות את העץ מהשאלה הקודמת ע"י סדרה של הכנסות והוצאות בעץ WAVL?

שאלה 6 - פתרון

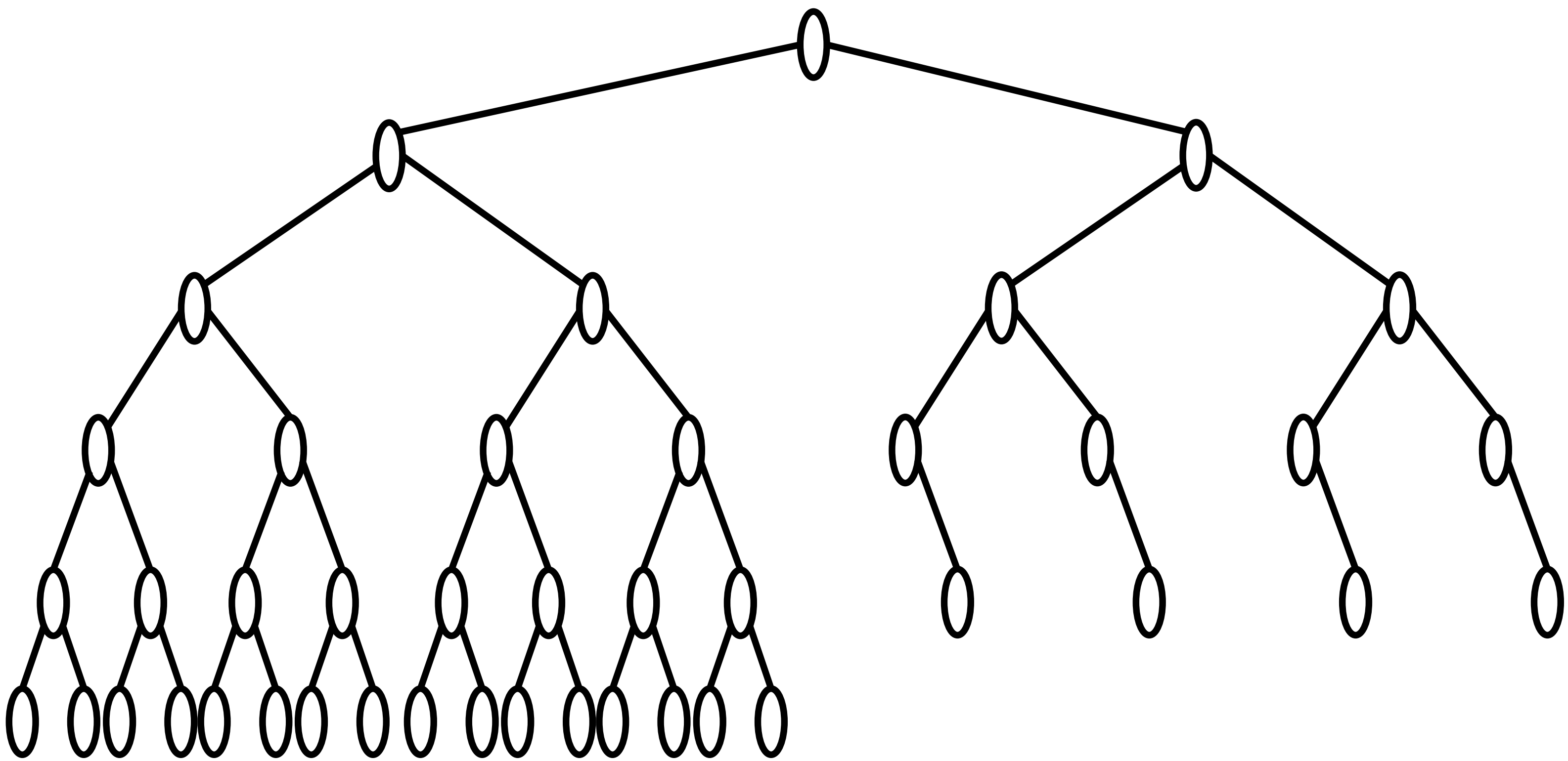
כן. נבנה את העץ AVL הבא באמצעות הכנסות בלבד:



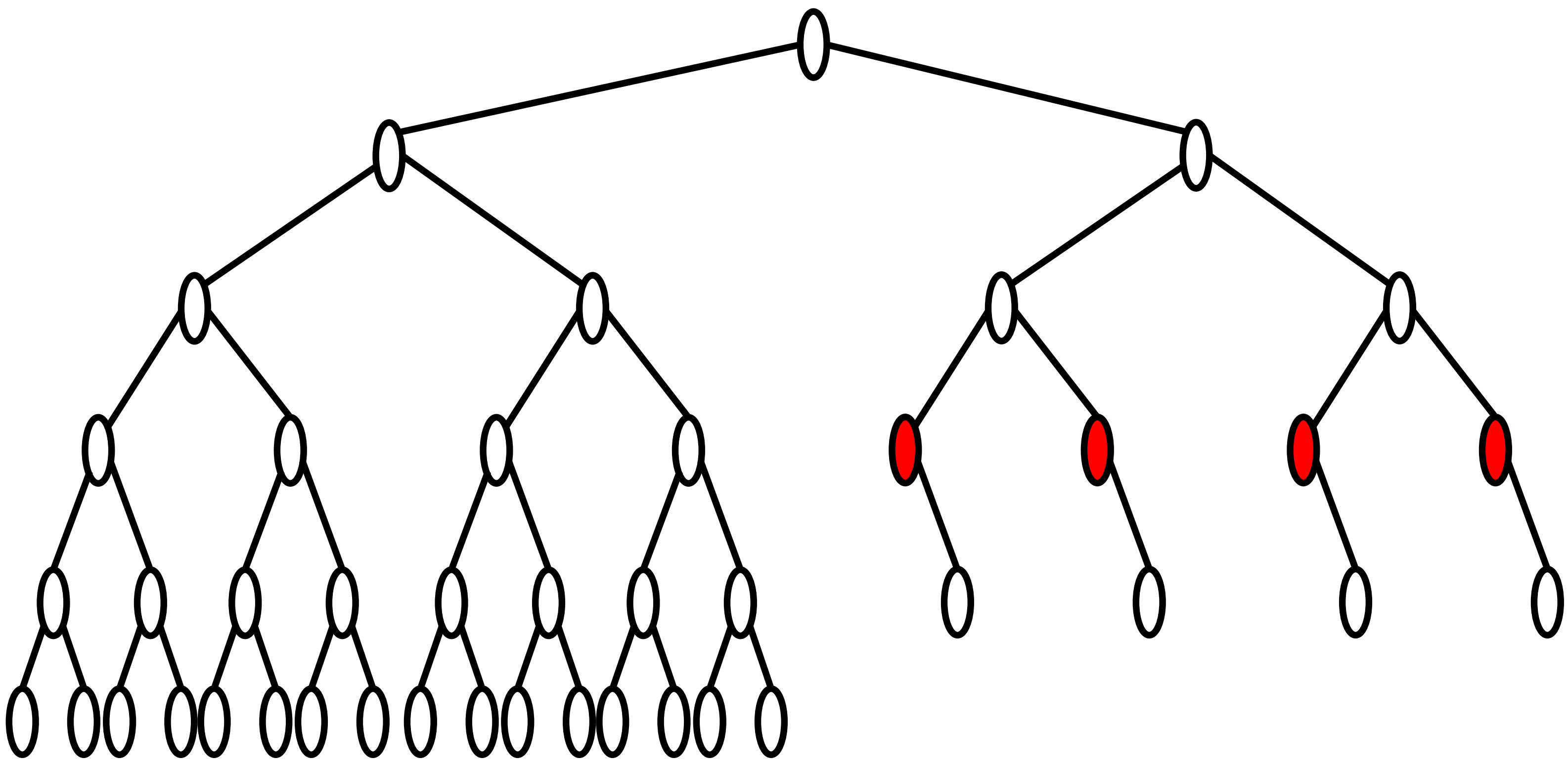
שאלה 6 - פתרון



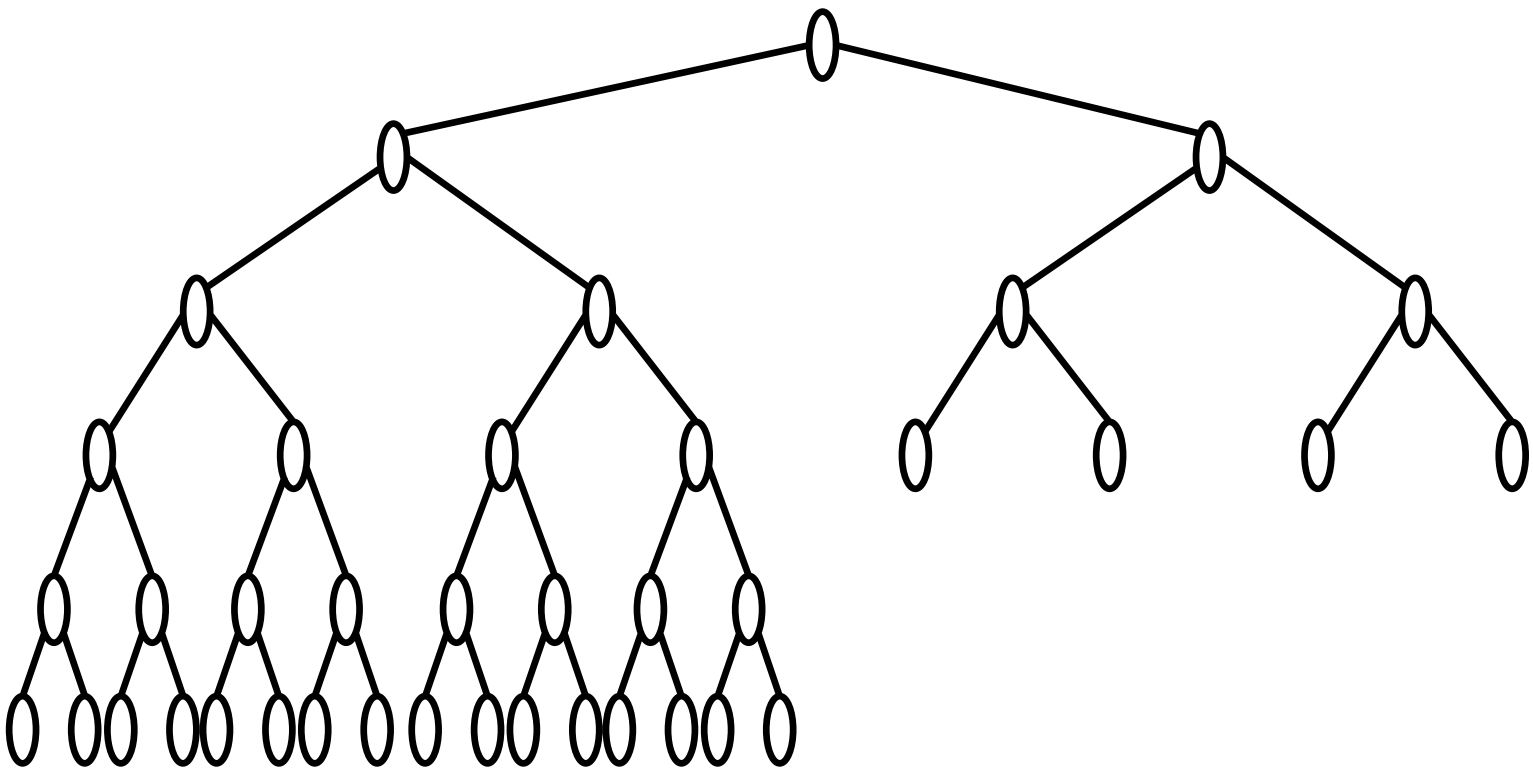
שאלה 6 - פתרון



שאלה 6 - פתרון



שאלה 6 - פתרון



שאלה 7

- תארו אלגוריתם, שבהינתן עץ חיפוש בינארי, קובע האם ניתן להתאים ranks לצמתי העץ כך שיתקבל עץ WAVL

פיתרון 7

<u>תיאור כללי</u>:

- נעבור על העץ מהעלים כלפי מעלה עד השורש.
- בכל צעד נתחזק, בהינתן צומת מסויים v, מהו התווך האפשרי של השמות rank(v), כך שתת העץ ששורשו v מהווה עץ WAVL חוקי (עבור בחירה כלשהי של דרגות לשאר הצמתים בתת העץ).
- אלגוריתם רקורסיבי.

Function Is_WAVL(v) {
If v is an external leaf
return [-1,-1];
If v is an internal leaf
return [0,0];
Else {
$[r_1, R_1] \leftarrow Is\_WAVL(v.left)$;
$[r_2, R_2] \leftarrow Is\_WAVL(v.right)$;
$r \leftarrow \max\{ r_1, r_2 \} + 1$;
$R \leftarrow \min\{ R_1, R_2 \} + 2$;
If $r \leq R$ return $[r, R]$;
Else print('Not a WAVL Tree') and stop.
} }