Skip Lists רשימות דילוגים

Skip lists: A probabilistic Alternative to Balanced Trees, William Pugh, Communications of the ACM, 33(6):668-676, 1990.

המאמר נמצא גם באתר הקורס תחת skip list animation וכן באתר:
www.cs.umd.edu/users/pugh (selected publications)

©Geiger & Itai, 2001

רשימות דילוגים Skip Lists

עצים מאוזנים, שנלמדו בשני השעורים הקודמים, משמשים למימוש פעולות המילון (חיפוש, הכנסה, הוצאה) בזמן O(log n) במקרה הגרוע ביותר.

מימוש הפעולות אינו טריוויאלי, במיוחד כאשר יש צורך לתמוך גם בפעולות הדורשות לשמור בכל צומת אינפורמציה ייחודית (כמו rank) ולעדכן אותה.

נלמד כעת מבנה נתונים בעל מימוש פשוט מאוד, שלא דורש איזונים של מבנה הנתונים לאחר הכנסה/הוצאה. בתמורה : בצוע פעולות המילון יעשה בזמן O(log n) <u>בממוצע</u>.

כמו כן ההסתברות שזמן בצוע פעולה יחרוג בצורה משמעותית (נאמר פי 3) מהזמן הממוצע היא זניחה (פחות מ- 1 ל 100,000,000).

©cs,Technion

מהו "ממוצע" ?

ראינו כבר (בשיעור 3) שעצי חיפוש ללא פעולות איזון יכולים לשמש למימוש פעולות המילון בזמן O(log n) ב"ממוצע".

מהו ממוצע ? בניתוח שעשינו הנחנו שהוכנסו n איברים. לפיכך קיימות !n פרמוטציות להכנסתם. הממוצע של גובה !n העצים שנוצרו הוא O(log n).

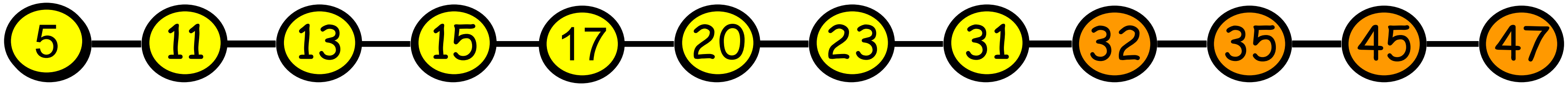
הבעייתיות בניתוח זה נובעת ממספר סיבות. ראשית, בדר"כ ההסתברות להופעת פרמוטציה בקלט אינה אחידה. שנית מבנה הנתונים שנוצר תלוי בסדר הכנסת הנתונים: "יריב" (אדם/תהליך המחבל במערכת adversary) יכול להכניס סדרת נתונים כך שהפעולות יבוצעו בזמן האיטי ביותר.

ברשימות דילוגים בעיות אלה נפתרות. הפעולות וזמן ביצוען תלוי בהגרלות שעושה המחשב ולא בסדר הכנסת הנתונים. הממוצע מחושב לפי ההגרלות ולא לפי הנחות על ההסתברות של הקלטים. אלגוריתם כזה נקרא **אלגוריתם רנדומלי**.

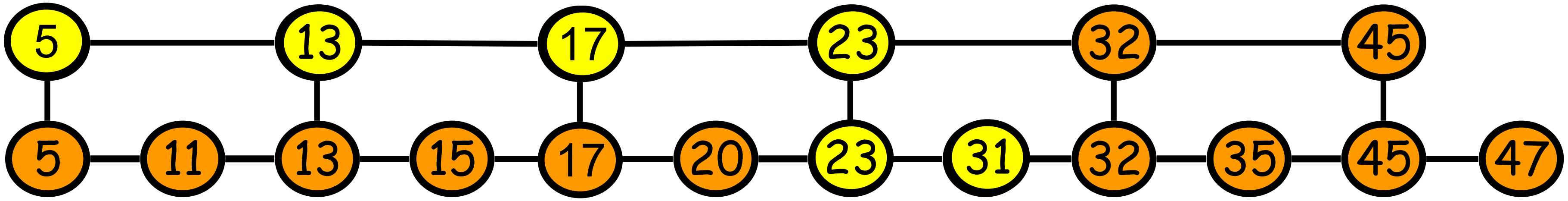
©cs,Technion

הרעיון המרכזי

חיפוש איבר בסוף רשימה הוא יקר.



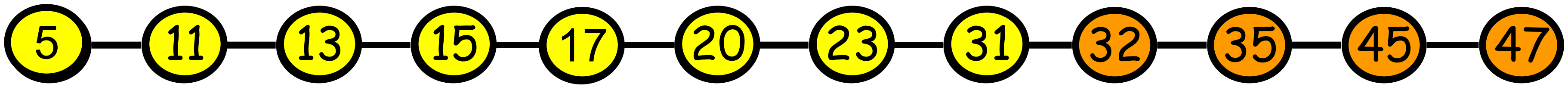
כדי לזרז את החיפוש (פי שניים) נוסיף מדריך- תת רשימה של כל איבר שני:



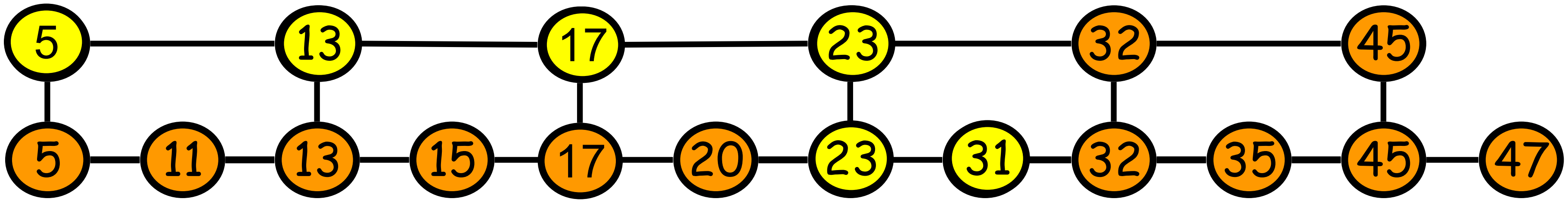
©cs,Technion

הרעיון המרכזי

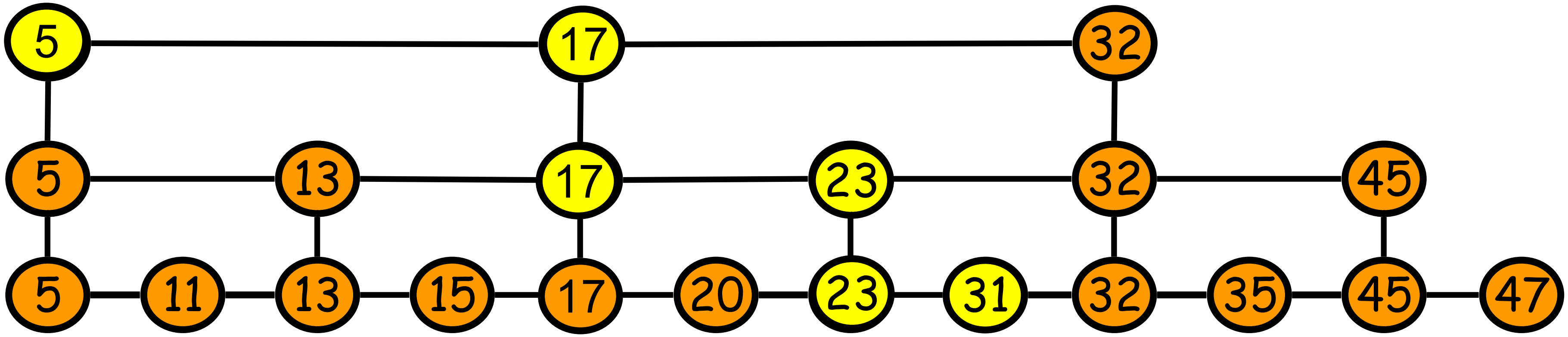
חיפוש איבר בסוף רשימה הוא יקר.



כדי לזרז את החיפוש (פי שניים) נוסיף מדריך- תת רשימה של כל איבר שני:



כדי לזרז את החיפוש במדריך נוסיף רמה נוספת:



וכך הלאה עד לרמה $\lceil \log n \rceil$ ובה איבר בודד.

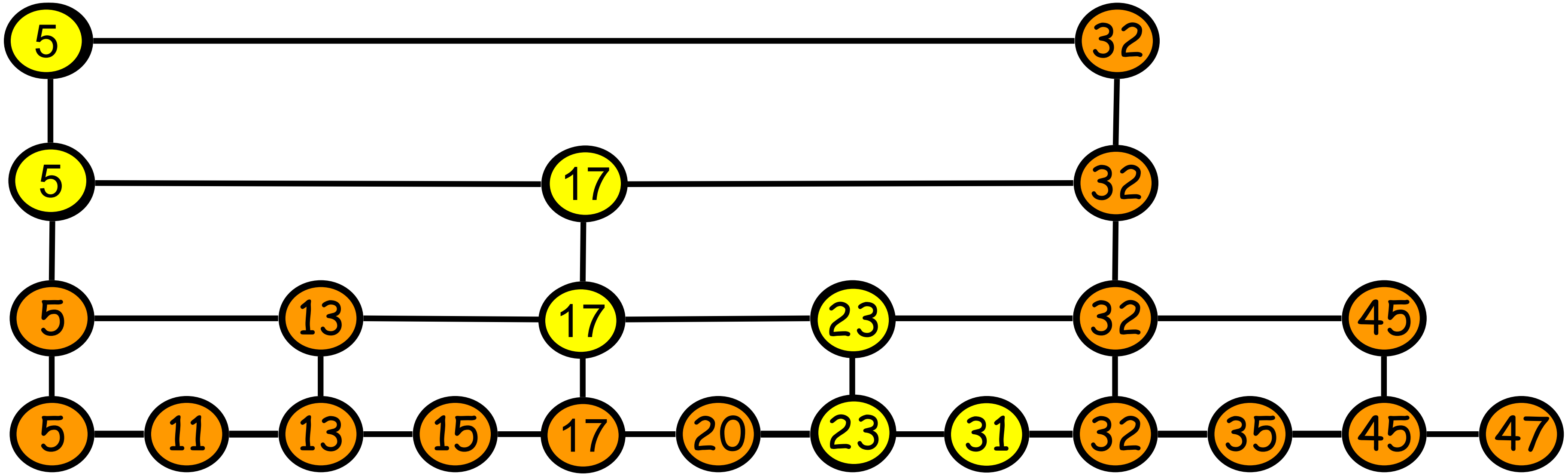
©cs,Technion

אורך מסלול החיפוש

נסתכל על מסלול החיפוש מצומת הסיום ועד צומת ההתחלה.

במעבר מרמה לרמה אנו עוברים על פני 2 מצביעים לכל היותר.

לפיכך אורך המסלול הוא $2log\ n$. מספר הצמתים הכולל אינו עולה על $2n$.

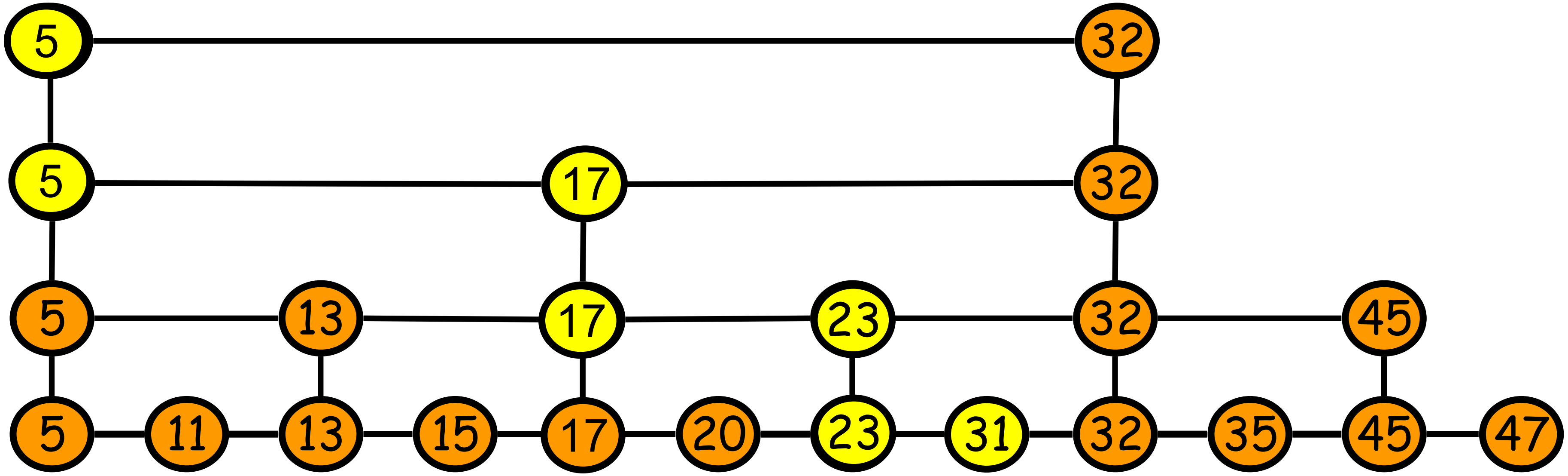


אורך מסלול החיפוש

נסתכל על מסלול החיפוש מצומת הסיום ועד צומת ההתחלה.

במעבר מרמה לרמה אנו עוברים על פני 2 מצביעים לכל היותר.

לפיכך אורך המסלול הוא $2\log n$. מספר הצמתים הכולל אינו עולה על $2n$.



הבעיה שנותר לפתור היא כיצד לשמור על המבנה שיצרנו בעת הכנסה והוצאה מבלי שיהיה צורך לארגן את כל הרמות מחדש.

נשים לב שאלגוריתם החיפוש נכון גם אם מספר הצמתים של רמה i בין כל שני צמתים של רמה i-1 אינו קבוע.

©cs,Technion

הכנסה באמצעות הטלת מטבע

צעדים להכנסת מפתח k.

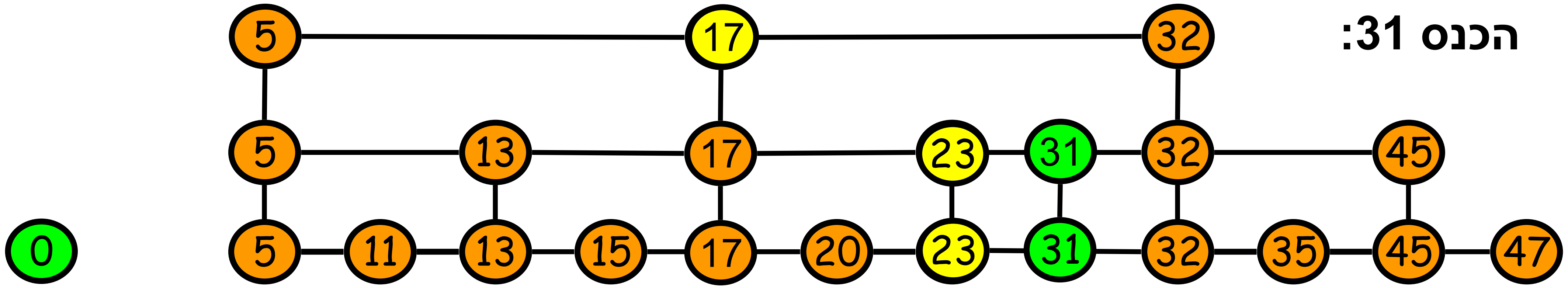
1. חפש את k. אם k נמצא סיים.
2. שמור מצביע לצומת הימני ביותר בכל רמה במסלול החיפוש.
3. הוסף צומת חדש ברמה התחתונה ביותר וקבע את המפתח להיות k.
4. לפי סדר הרמות מלמטה למעלה:

   הגרל מטבע toss( ). אם יוצא 0: הוסף צומת חדש מעל הרמה הנוכחית וקבע את המפתח בו להיות k. אם יוצא 1, עצור.

5. אם ברמה העליונה הוגרל 0, הוסף רמה חדשה.

הגרלה

הכנס 31:



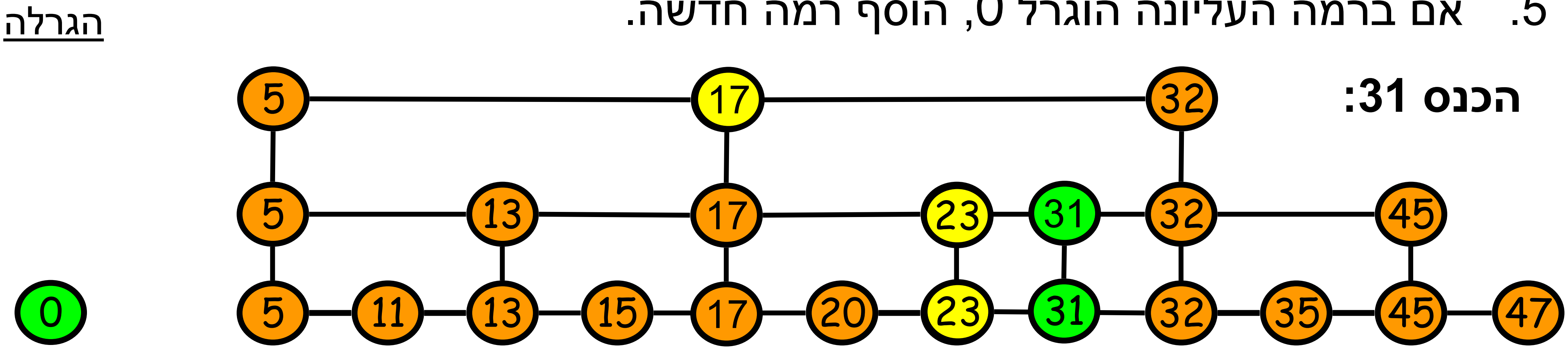
הכנסה באמצעות הטלת מטבע

צעדים להכנסת מפתח k.

1. חפש את k. אם k נמצא סיים.
2. שמור מצביע לצומת הימני ביותר בכל רמה במסלול החיפוש.
3. הוסף צומת חדש ברמה התחתונה ביותר וקבע את המפתח להיות k.
4. לפי סדר הרמות מלמטה למעלה:

   הגרל מטבע toss( ). אם יוצא 0: הוסף צומת חדש מעל הרמה הנוכחית וקבע את המפתח בו להיות k. אם יוצא 1, עצור.

5. אם ברמה העליונה הוגרל 0, הוסף רמה חדשה.



הכנסה באמצעות הטלת מטבע

צעדים להכנסת מפתח k.

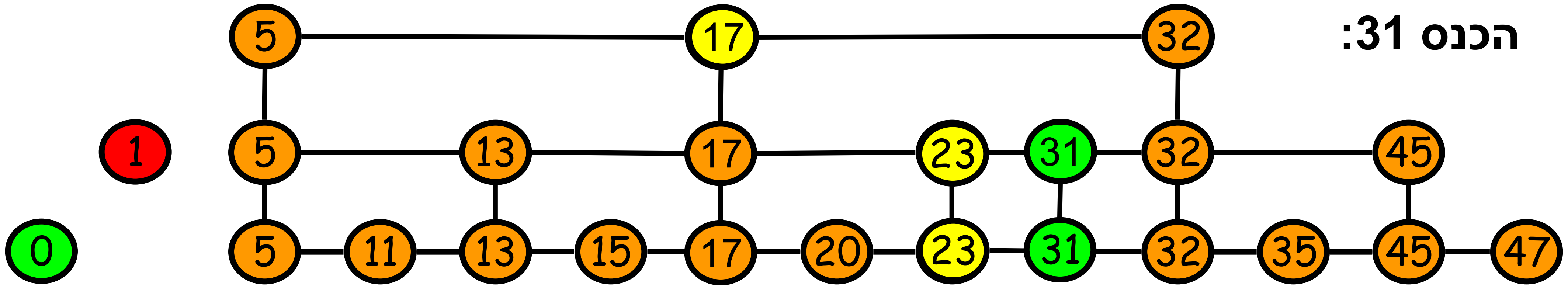
1. חפש את k. אם k נמצא סיים.
2. שמור מצביע לצומת הימני ביותר בכל רמה במסלול החיפוש.
3. הוסף צומת חדש ברמה התחתונה ביותר וקבע את המפתח להיות k.
4. לפי סדר הרמות מלמטה למעלה:

הגרל מטבע toss( ). אם יוצא 0: הוסף צומת חדש מעל הרמה הנוכחית וקבע את המפתח בו להיות k. אם יוצא 1, עצור.

5. אם ברמה העליונה הוגרל 0, הוסף רמה חדשה.

הגרלה

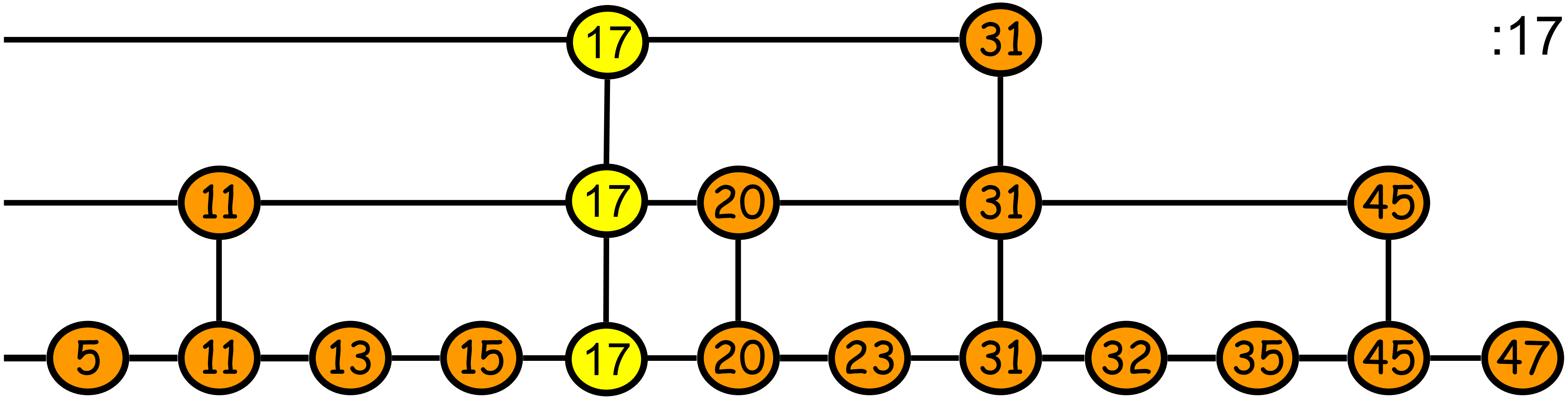
הכנס 31:



הוצאה מרשימת דילוגים

1. מצא את האיבר בעל המפתח k.
2. הוצא איבר זה מכל הרמות בהם הוא מופיע.

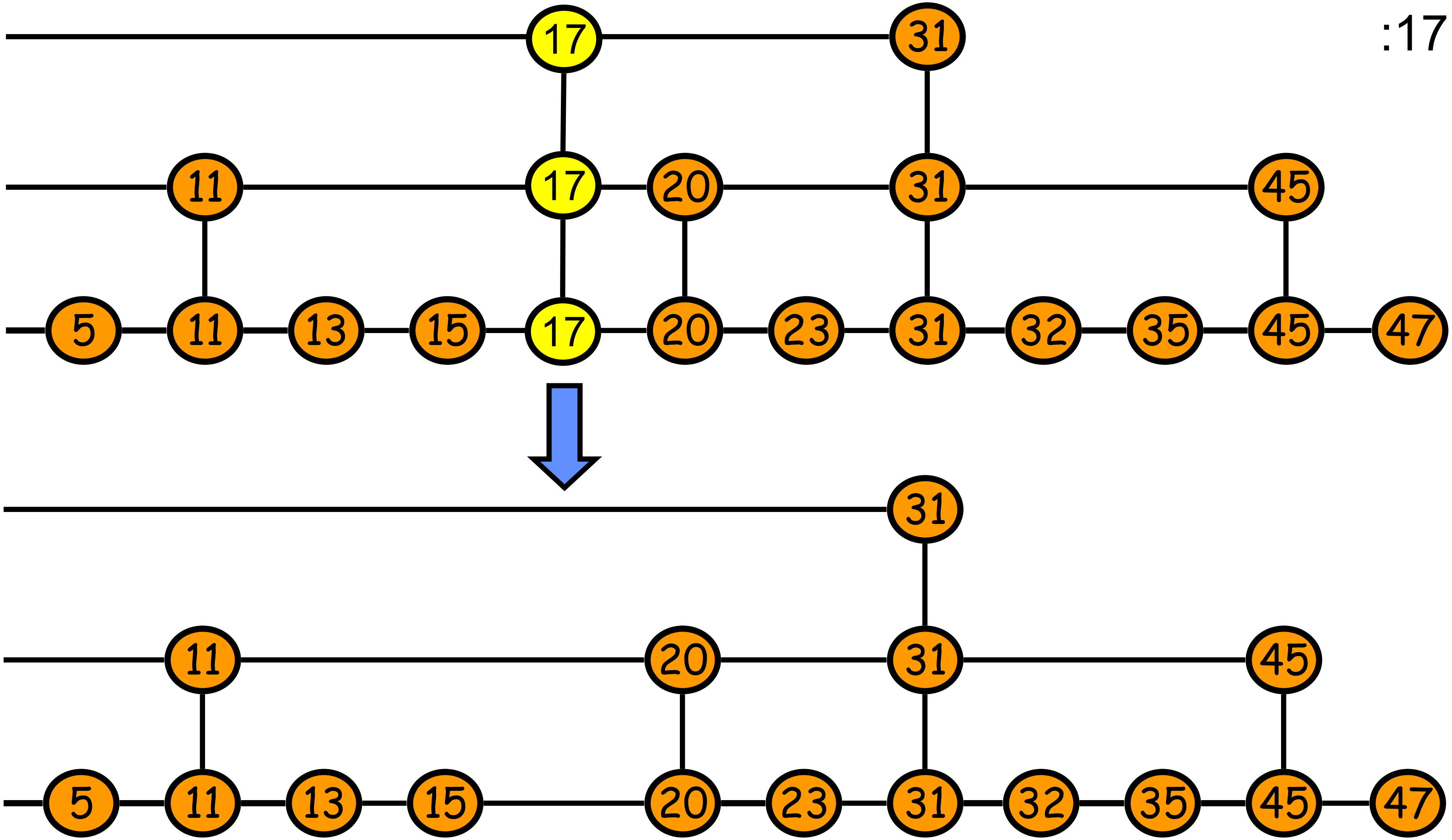
הוצא 17:



הוצאה מרשימת דילוגים

1. מצא את האיבר בעל המפתח k.
2. הוצא איבר זה מכל הרמות בהם הוא מופיע.

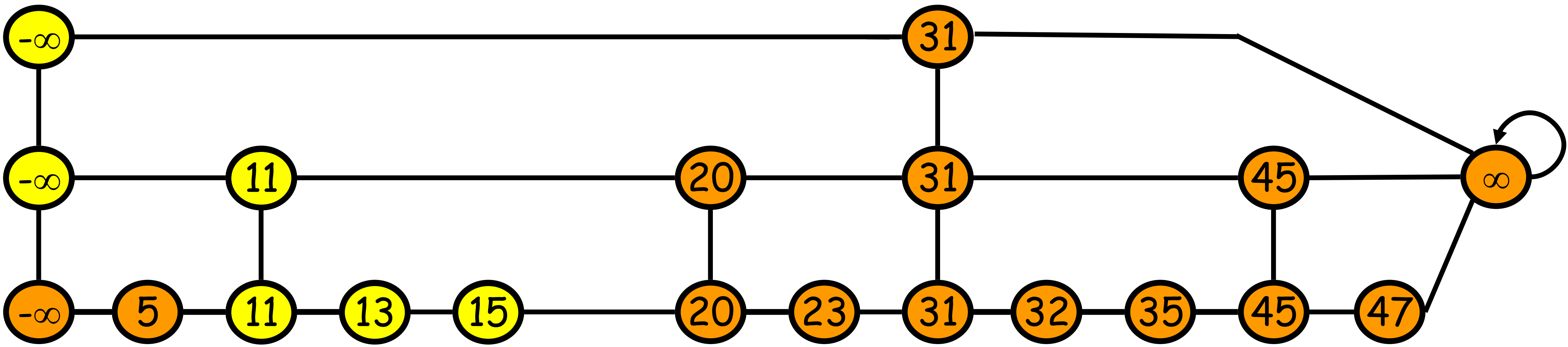
הוצא 17:



הערות לגבי מימוש רשימת דילוגים

נוח להוסיף בכל רמה צומת אחד בהתחלת הרשימה שמפתחו ∞- וצומת אחד בסוף שמפתחו ∞+.

מצא 17: לא נמצא.



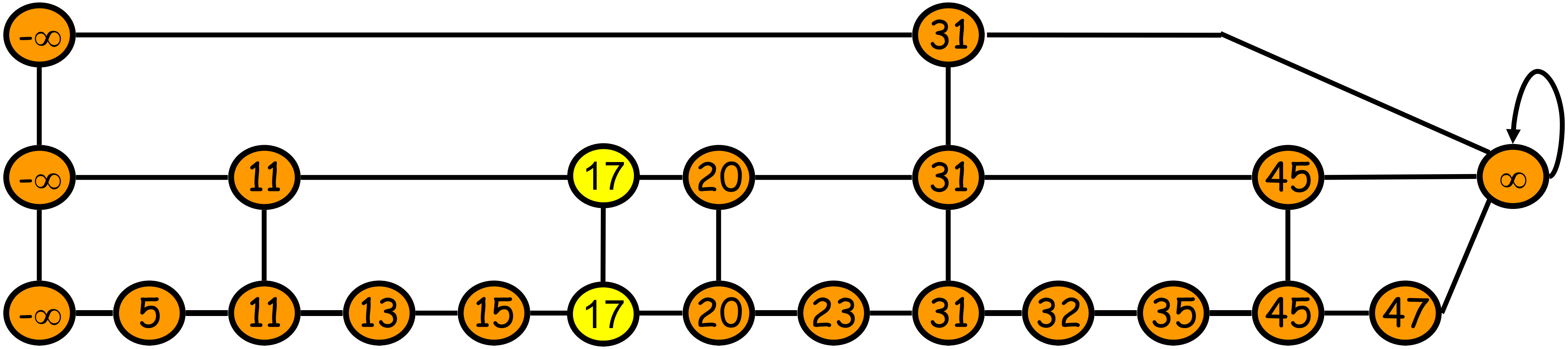
תוכנית החיפוש ההכנסה וההוצאה ניתנות לכתיבה רקורסיבית וגם לא רקורסיבית בקלות רבה. דוגמאות למימוש בחוברת השקפים.

בכמה רמות בממוצע מופיע כל מפתח ?

בכל רמה האלגוריתם מטיל מטבע כדי לקבוע אם יוצר העתק נוסף מעליה.

מספר הרמות הממוצע למפתח שווה למספר הפעמים הממוצע שיש להטיל מטבע הוגן עד שיצא 1.

מספר הפעמים הממוצע שיש להטיל מטבע הוגן עד שיצא 1 הוא 2. (הוכחה בשקף הבא. אינטואיציה:בממוצע מטבע נופל פעם על צד 0 ופעם על צד 1).



מסקנה: מספר הצמתים הממוצע הוא $2n$ כאשר n הוא מספר המפתחות במבנה (כל מפתח מופיע בממוצע פעמיים).

©cs,Technion

הטלת מטבע

נניח שלמטבע הסתברות p לצאת "זנב" (נסמן צד זה באפס) והסתברות 1-p לצאת "ראש" (נסמן צד זה באחד).

כמה פעמים בממוצע L יש להטיל את המטבע עד שנקבל "ראש" ?

נסמן ב- Q_i את ההסתברות שנקבל "ראש" לראשונה בהטלה ה- i (נקרא התפלגות גאומטרית).

הטלת מטבע

נניח שלמטבע הסתברות p לצאת "זנב" (נסמן צד זה באפס) והסתברות 1-p לצאת "ראש" (נסמן צד זה באחד).

כמה פעמים בממוצע L יש להטיל את המטבע עד שנקבל "ראש" ?

נסמן ב- Q_i את ההסתברות שנקבל "ראש" לראשונה בהטלה ה- i (נקרא התפלגות גאומטרית).

מתקיים: $Q_0=0$ $\quad$ $Q_i = p^{i-1}(1-p)$

הטלת מטבע

נניח שלמטבע הסתברות p לצאת "זנב" (נסמן צד זה באפס) והסתברות 1-p לצאת "ראש" (נסמן צד זה באחד).

כמה פעמים בממוצע L יש להטיל את המטבע עד שנקבל "ראש" ?

נסמן ב- Q_i את ההסתברות שנקבל "ראש" לראשונה בהטלה ה- i (נקרא התפלגות גאומטרית).

מתקיים: $Q_0=0$ $\quad Q_i = p^{i-1}(1-p)$

$$L = \sum_{i=1}^{\infty} i \cdot Q_i = \sum_{i=1}^{\infty} i \cdot p^{i-1}(1-p)$$

הטלת מטבע

נניח שלמטבע הסתברות p לצאת "זנב" (נסמן צד זה באפס) והסתברות 1-p לצאת "ראש" (נסמן צד זה באחד).

כמה פעמים בממוצע L יש להטיל את המטבע עד שנקבל "ראש" ?

נסמן ב- Q_i את ההסתברות שנקבל "ראש" לראשונה בהטלה ה- i (נקרא התפלגות גאומטרית).

מתקיים: $Q_0=0$ $Q_i = p^{i-1}(1-p)$

$$L = \sum_{i=1}^{\infty} i \cdot Q_i = \sum_{i=1}^{\infty} i \cdot p^{i-1}(1-p) = (1-p)\sum_{i=1}^{\infty} i \cdot p^{i-1} = \frac{1-p}{(1-p)^2} \qquad (|p|<1)$$

הטלת מטבע

נניח שלמטבע הסתברות p לצאת "זנב" (נסמן צד זה באפס) והסתברות 1-p לצאת "ראש" (נסמן צד זה באחד).

כמה פעמים בממוצע L יש להטיל את המטבע עד שנקבל "ראש" ?

נסמן ב- Q_i את ההסתברות שנקבל "ראש" לראשונה בהטלה ה- i (נקרא התפלגות גאומטרית).

מתקיים: $Q_0=0$ $\quad$ $Q_i = p^{i-1}(1-p)$

$$L=\sum_{i=1}^{\infty} i\cdot Q_i=\sum_{i=1}^{\infty} i\cdot p^{i-1}(1-p) \qquad =(1-p)\sum_{i=1}^{\infty} i\cdot p^{i-1}=\frac{1-p}{(1-p)^2} \qquad (|p|<1)$$

כלומר בממוצע נדרשות $L=\frac{1}{1-p}$ הטלות עד שמקבלים "ראש".

הטלת מטבע

נניח שלמטבע הסתברות p לצאת "זנב" (נסמן צד זה באפס) והסתברות 1-p לצאת "ראש" (נסמן צד זה באחד).

כמה פעמים בממוצע L יש להטיל את המטבע עד שנקבל "ראש" ?

נסמן ב- Q_i את ההסתברות שנקבל "ראש" לראשונה בהטלה ה- i (נקרא התפלגות גאומטרית).

מתקיים: $Q_0=0$ $Q_i = p^{i-1}(1-p)$

$$L=\sum_{i=1}^{\infty} i\cdot Q_i = \sum_{i=1}^{\infty} i\cdot p^{i-1}(1-p) \qquad = (1-p)\sum_{i=1}^{\infty} i\cdot p^{i-1} = \frac{1-p}{(1-p)^2} \qquad (|p|<1)$$

כלומר בממוצע נדרשות $L=\frac{1}{1-p}$ הטלות עד שמקבלים "ראש".

ועבור מטבע הוגן, כאשר $p = \frac{1}{2}$ נדרשות L=2 הטלות בממוצע.

מהו אורך מסלול חיפוש ממוצע ?

משפט: האורך הממוצע L של מסלול חיפוש ברשימת דילוגים עם n מפתחות מקיים $L \leq 2\log_2(n) + 2$.

הוכחה: נסתכל על מסלול חיפוש מנקודת הסיום אחורנית לנקודת ההתחלה. המסלול מורכב מתנועה מעלה (כשאפשר) ותנועה שמאלה (כשחייבים). כאשר מגיעים לקצה השמאלי ביותר, התנועה ממשיכה כלפי מעלה בלבד.

מהו אורך מסלול חיפוש ממוצע ?

משפט: האורך הממוצע L של מסלול חיפוש ברשימת דילוגים עם n מפתחות מקיים $L \le 2\log_2(n) + 2$.

הוכחה: נסתכל על מסלול חיפוש מנקודת הסיום אחורנית לנקודת ההתחלה. המסלול מורכב מתנועה מעלה (כשאפשר) ותנועה שמאלה (כשחייבים). כאשר מגיעים לקצה השמאלי ביותר, התנועה ממשיכה כלפי מעלה בלבד.

נסמן ב- $c(k)$ את אורך המסלול הממוצע המטפס k רמות.

מתקיים: $c(0) = 0$

$$c(k) \le \tfrac{1}{2}\,(1+ c(k)) + \tfrac{1}{2}\,(1 + c(k-1))$$

הסבר: צעד שמאלה בהסתברות $\frac{1}{2}$ (אם אפשר) וצעד מעלה בהסתברות $\frac{1}{2}$.

מהו אורך מסלול חיפוש ממוצע ?

משפט: האורך הממוצע L של מסלול חיפוש ברשימת דילוגים עם n מפתחות מקיים $L \leq 2\log_2(n) + 2$.

הוכחה: נסתכל על מסלול חיפוש מנקודת הסיום אחורנית לנקודת ההתחלה. המסלול מורכב מתנועה מעלה (כשאפשר) ותנועה שמאלה (כשחייבים). כאשר מגיעים לקצה השמאלי ביותר, התנועה ממשיכה כלפי מעלה בלבד.

נסמן ב- $c(k)$ את אורך המסלול הממוצע המטפס k רמות.

מתקיים:

$$c(0) = 0$$

$$c(k) \leq \tfrac{1}{2}(1+ c(k)) + \tfrac{1}{2}(1 + c(k-1))$$

הסבר: צעד שמאלה בהסתברות $\tfrac{1}{2}$ (אם אפשר) וצעד מעלה בהסתברות $\tfrac{1}{2}$.

לפיכך:

$$c(k) \leq 2 + c(k-1)$$

מהו אורך מסלול חיפוש ממוצע ?

משפט: האורך הממוצע L של מסלול חיפוש ברשימת דילוגים עם n מפתחות מקיים $L \le 2\log_2(n) + 2$.

הוכחה: נסתכל על מסלול חיפוש מנקודת הסיום אחורנית לנקודת ההתחלה. המסלול מורכב מתנועה מעלה (כשאפשר) ותנועה שמאלה (כשחייבים). כאשר מגיעים לקצה השמאלי ביותר, התנועה ממשיכה כלפי מעלה בלבד.

נסמן ב- $c(k)$ את אורך המסלול הממוצע המטפס k רמות.

מתקיים:
$$c(0) = 0$$

$$c(k) \le \tfrac{1}{2}(1 + c(k)) + \tfrac{1}{2}(1 + c(k-1))$$

הסבר: צעד שמאלה בהסתברות $\tfrac{1}{2}$ (אם אפשר) וצעד מעלה בהסתברות $\tfrac{1}{2}$.

לפיכך:
$$c(k) \le 2 + c(k-1)$$

ולכן:
$$c(k) \le 4 + c(k-2) \le \ldots \le 2k$$

כלומר אורך מסלול ממוצע מרמה 1 לרמה $\log_2(n)$ מקיים: $L \le 2(\log_2(n)-1)$

אורך מסלול חיפוש (המשך)

המשך ההוכחה: נותר לקחת בחשבון את המקרים שבהם מספר הרמות גדול מ-$log_2(n)$.

מעל הרמה ה- $log_2(n)$ מספר הצעדים הממוצע שמאלה חסום ע"י מספר האיברים הממוצע ברשימת הדילוגים שהוכנסו מעל רמה זו. מספרם הממוצע חסום ע"י 2 שכן בכל רמה מספר האיברים הממוצע קטן פי 2.

מהו מספר הצעדים כלפי מעלה?

אורך מסלול חיפוש (המשך)

המשך ההוכחה: נותר לקחת בחשבון את המקרים שבהם מספר הרמות גדול מ- $log_2(n)$.

מעל הרמה ה- $log_2(n)$ מספר הצעדים הממוצע שמאלה חסום ע"י מספר האיברים הממוצע ברשימת הדילוגים שהוכנסו מעל רמה זו. מספרם הממוצע חסום ע"י 2 שכן בכל רמה מספר האיברים הממוצע קטן פי 2.

מהו מספר הצעדים כלפי מעלה?

ניתן להראות שמספר הרמות הממוצע המקסימלי עבור רשימת דילוגים עם n איברים קטן מ- $log_2(n) + 2$

כלומר מספר הצעדים הממוצע כלפי מעלה מעל הרמה $log_2(n)$ הוא לכל היותר 2.

אורך מסלול חיפוש (המשך)

המשך ההוכחה: נותר לקחת בחשבון את המקרים שבהם מספר הרמות גדול מ- $log_2(n)$.

מעל הרמה ה- $log_2(n)$ מספר הצעדים הממוצע שמאלה חסום ע"י מספר האיברים הממוצע ברשימת הדילוגים שהוכנסו מעל רמה זו. מספרם הממוצע חסום ע"י 2 שכן בכל רמה מספר האיברים הממוצע קטן פי 2.

מהו מספר הצעדים כלפי מעלה?

ניתן להראות שמספר הרמות המקסימלי הממוצע עבור רשימת דילוגים עם n איברים קטן מ- $log_2(n) + 2$

כלומר מספר הצעדים הממוצע כלפי מעלה מעל הרמה $log_2(n)$ הוא לכל היותר 2.

לפיכך סך אורכו הממוצע של מסלול חיפוש מקיים: $L \le 2(log_2(n)-1)+4$

כמצוין במשפט.

©cs,Technion

זמן בצוע הפעולות

<u>מסקנה מהמשפט</u>: חיפוש, הכנסה, והוצאה מרשימת דילוגים נעשים בזמן ממוצע $O(\log n)$ כיון שבכל צעד על מסלול החיפוש מתבצעים $O(1)$ צעדים.

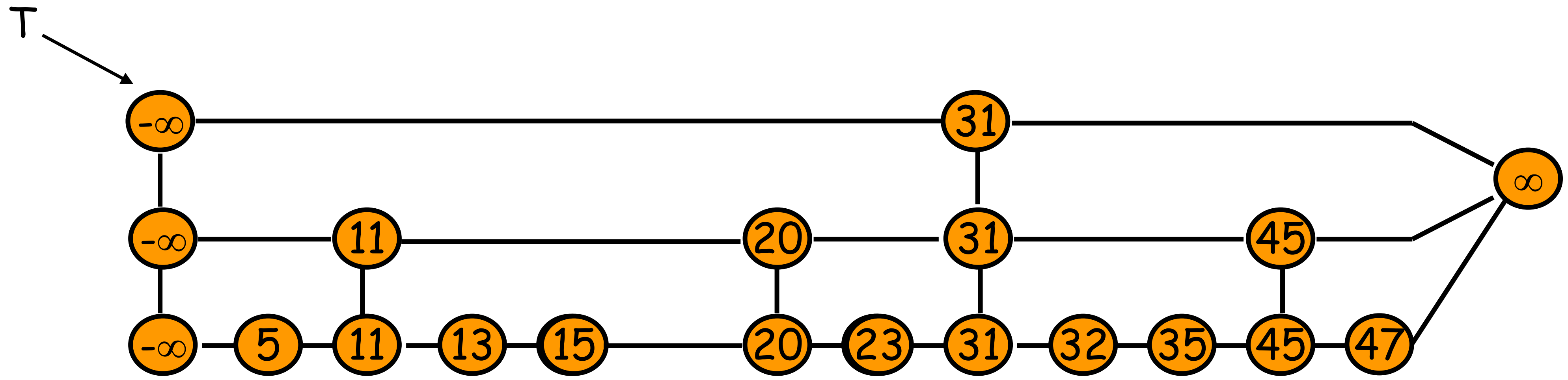
רשימת דילוגים דטרמיניסטית
Deterministic Skip List

נראה כעת כיצד לממש skip list דטרמיניסטי.

מבנה נתונים זה מאפשר פעולות חיפוש/הכנסה/הוצאה כמו בעצים מאוזנים (ובסיבוכיות זמן זהה).

נתאר כעת בפרוטרוט את פעולת ההכנסה.

אח"כ נציג את המבנה בהקשר של עצי 2-3.

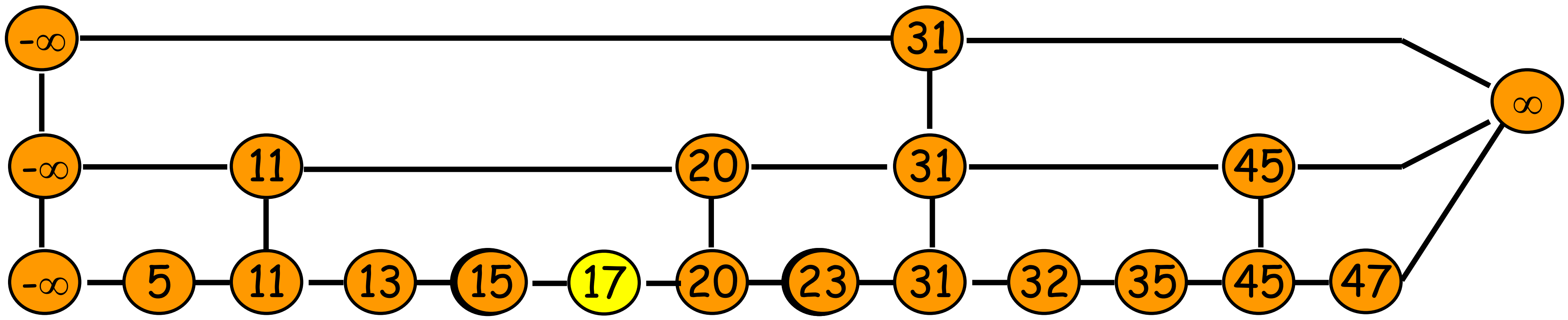


הרעיון העיקרי

יש לשמור שמספר הצמתים של רמה i בין כל שני צמתים של רמה i-1

יהיה 1 או 2. אם מספר הצמתים הוא 3, מוסיפים צומת ברמה ה- (i-1).

דוגמא: הכנס 17



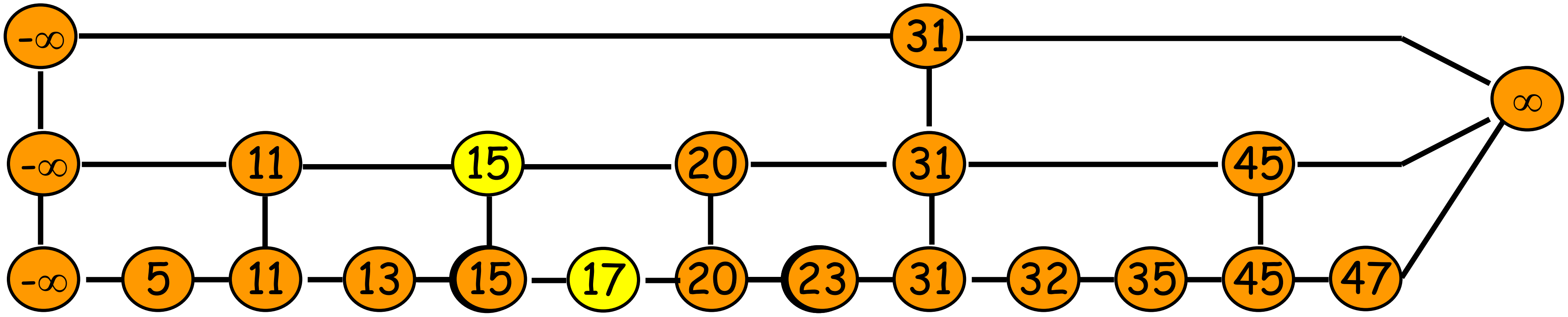
©cs,Technion

הרעיון העיקרי

יש לשמור שמספר הצמתים של רמה i בין כל שני צמתים של רמה i-1

יהיה 1 או 2. אם מספר הצמתים הוא 3, מוסיפים צומת ברמה ה- (i-1).

דוגמא: הכנס 17

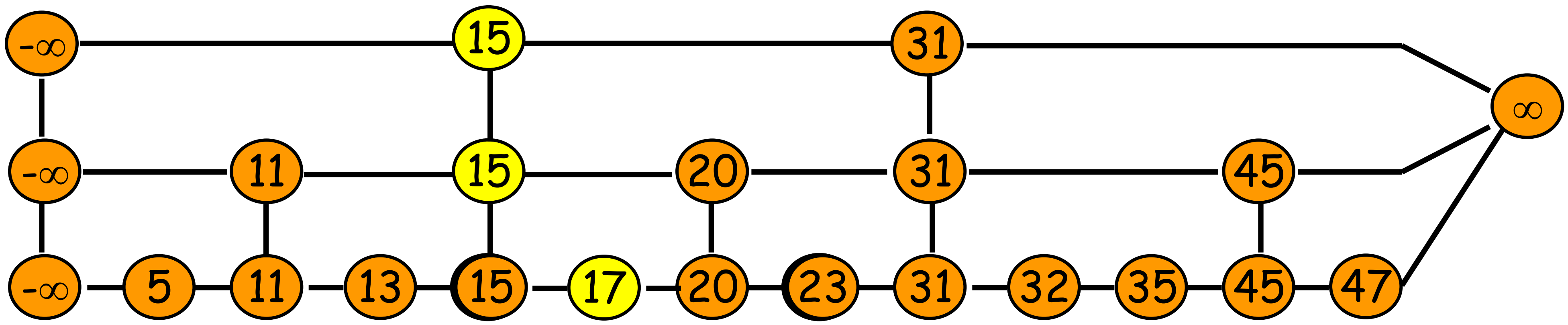


הרעיון העיקרי

יש לשמור שמספר הצמתים של רמה i בין כל שני צמתים של רמה i-1

יהיה 1 או 2. אם מספר הצמתים הוא 3, מוסיפים צומת ברמה ה- (i-1).

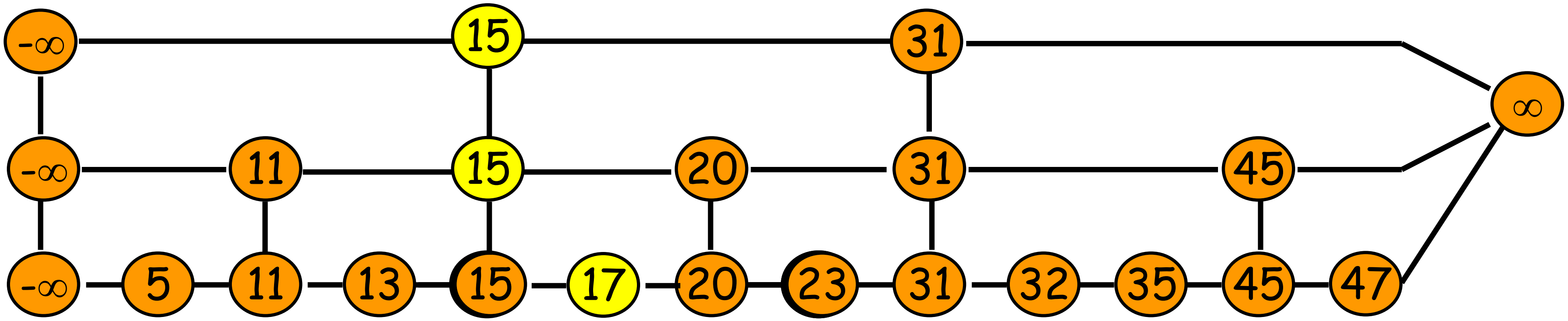
דוגמא: הכנס 17



הרעיון העיקרי

יש לשמור שמספר הצמתים של רמה i בין כל שני צמתים של רמה i-1 יהיה 1 או 2. אם מספר הצמתים הוא 3, מוסיפים צומת ברמה ה- (i-1).

דוגמא: הכנס 17



הערה: בניגוד לרשימת דילוגים (רנדומלית) המפתח שמוכנס אינו בהכרח המפתח שמשוכפל ברמה מעל. בדוגמא הוכנס 17 אך שוכפל המפתח 15.

תוכנית להכנסה ברשימת דילוגים דטרמיניסטית

```
int   insert1(int x, NODE  *T)
{ /* returns 1 iff new node allocated on this level */
NODE * q ;
for ( ; T → next →  value  <= x ; T =  T → next) ;
if (T → down = = NULL){     /* a leaf */
            T → next = get_node(x, T → next, NULL);
            return 1;
}
if (!insert1(x, T → down) )  return 0;
/* check if as result of insertion, there are three nodes
 between T → down and T → next → down  */
q = T → down → next → next ;
if ( q → next → value < T → next → value ) {
      T → next = get_node(q → value, T → next, q);
      return 1 ;
}
return 0 ;
}
```

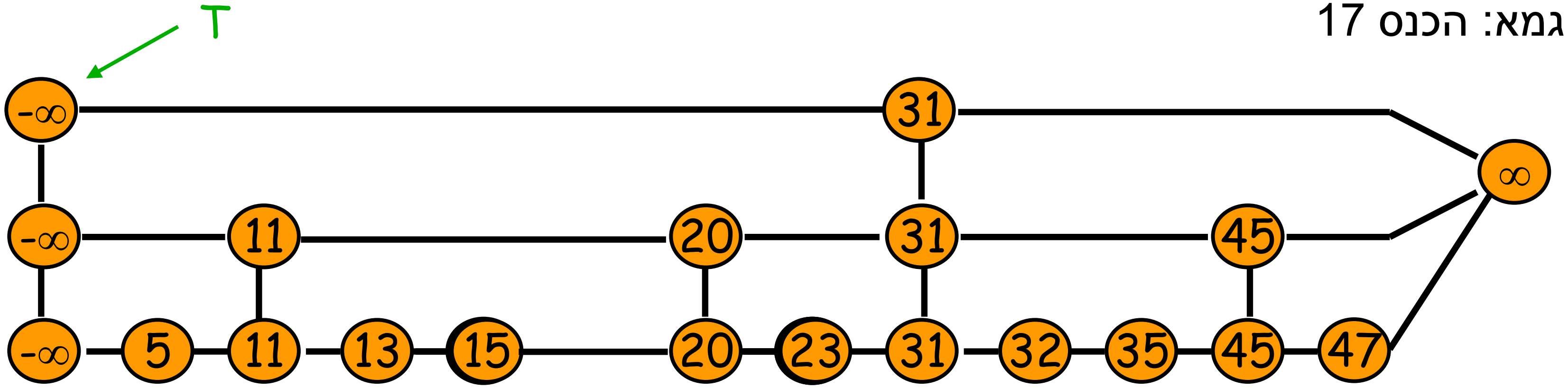
חיפוש ברמה נוכחית.

הוספה ברמה תחתונה.

קריאה רקורסיבית.

חזרה מקריאה רקורסיבית.

דוגמא: הכנס 17



©cs,Technion

תוכנית להכנסה ברשימת דילוגים דטרמיניסטית

```
int   insert1(int x, NODE  *T)
{ /* returns 1 iff new node allocated on this level */
NODE * q ;
for ( ; T → next →  value  <= x ; T =  T → next) ;
if (T → down = = NULL){     /* a leaf */
            T → next = get_node(x, T → next, NULL);
            return 1;
}
if (!insert1(x, T → down) )  return 0;
/* check if as result of insertion, there are three nodes
 between T → down and T → next → down  */
q = T → down → next → next ;
if ( q → next → value < T → next → value ) {
      T → next = get_node(q → value, T → next, q);
      return 1 ;
}
return 0 ;
}
```

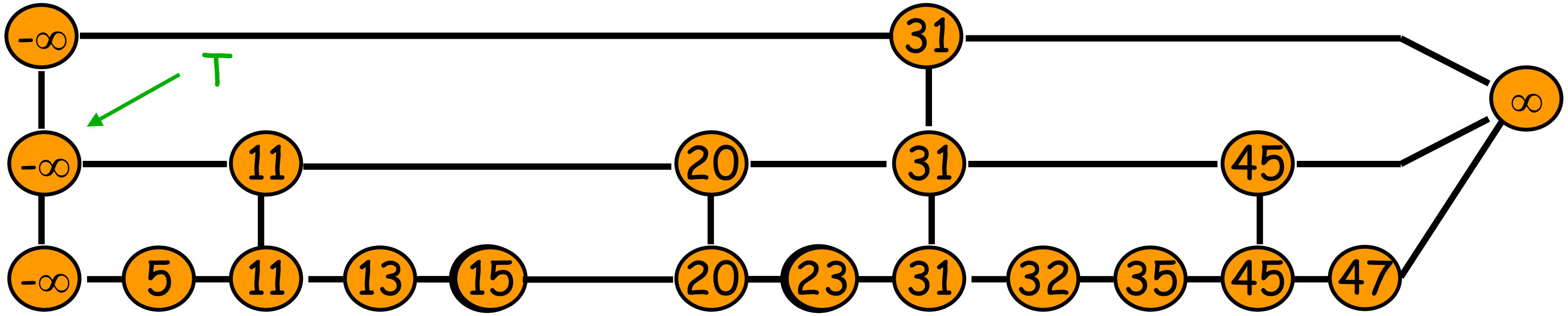
חיפוש ברמה נוכחית.

הוספה ברמה תחתונה.

קריאה רקורסיבית.

חזרה מקריאה רקורסיבית.

דוגמא: הכנס 17



תוכנית להכנסה ברשימת דילוגים דטרמיניסטית

```
int   insert1(int x, NODE  *T)
{ /* returns 1 iff new node allocated on this level */
NODE * q ;
for ( ; T → next →  value  <= x ; T =  T → next) ;
if (T → down = = NULL){     /* a leaf */
            T → next = get_node(x, T → next, NULL);
            return 1;
}
if (!insert1(x, T → down) )  return 0;
/* check if as result of insertion, there are three nodes
 between T → down and T → next → down  */
q = T → down → next → next ;
if ( q → next → value < T → next → value ) {
      T → next = get_node(q → value, T → next, q);
      return 1 ;
}
return 0 ;
}
```

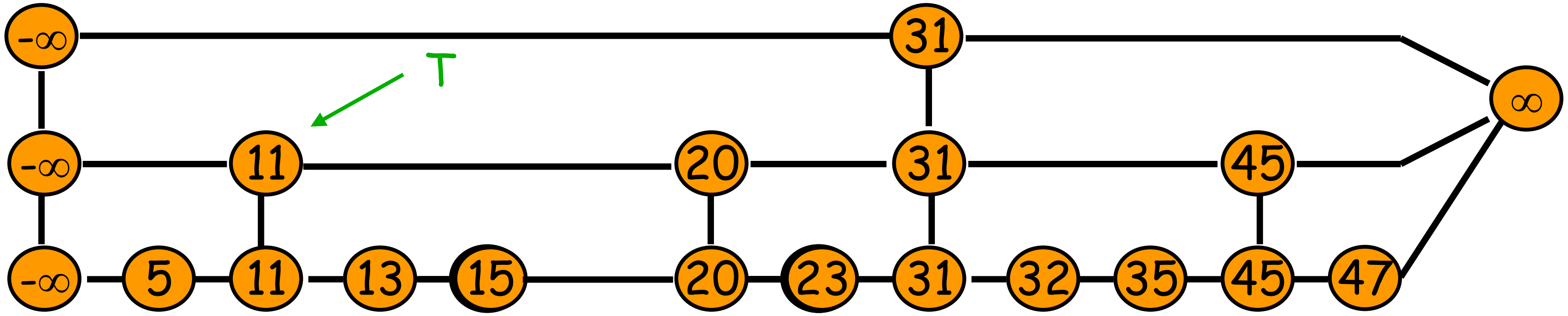
חיפוש ברמה נוכחית.

הוספה ברמה תחתונה.

קריאה רקורסיבית.

חזרה מקריאה רקורסיבית.

דוגמא: הכנס 17



©cs,Technion

תוכנית להכנסה ברשימת דילוגים דטרמיניסטית

```
int   insert1(int x, NODE  *T)
{ /* returns 1 iff new node allocated on this level */
NODE * q ;
for ( ; T → next →  value  <= x ; T =  T → next) ;
if (T → down = = NULL){     /* a leaf */
            T → next = get_node(x, T → next, NULL);
            return 1;
}
if (!insert1(x, T → down) )  return 0;
/* check if as result of insertion, there are three nodes
 between T → down and T → next → down  */
q = T → down → next → next ;
if ( q → next → value < T → next → value ) {
      T → next = get_node(q → value, T → next, q);
      return 1 ;
}
return 0 ;
}
```

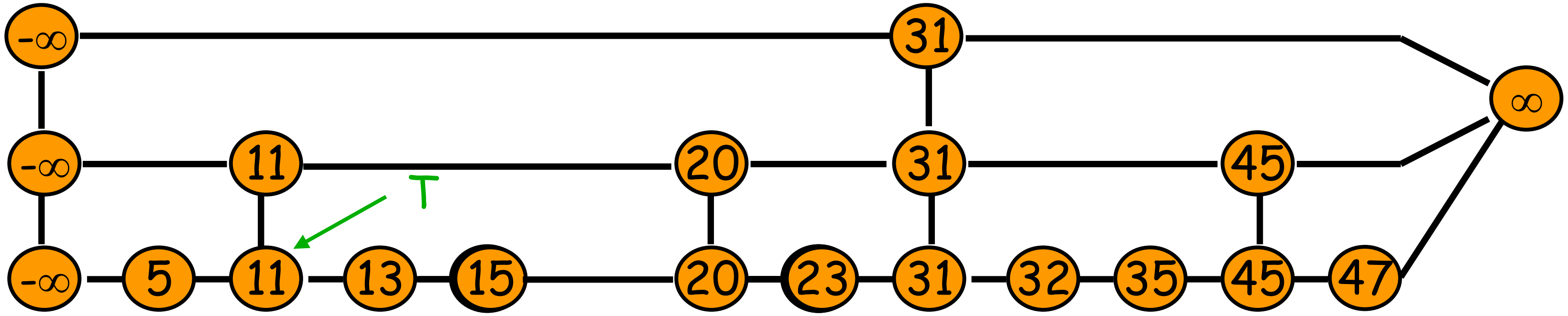
חיפוש ברמה נוכחית.

הוספה ברמה תחתונה.

קריאה רקורסיבית.

חזרה מקריאה רקורסיבית.

דוגמא: הכנס 17



תוכנית להכנסה ברשימת דילוגים דטרמיניסטית

```
int   insert1(int x, NODE  *T)
{ /* returns 1 iff new node allocated on this level */
NODE * q ;
for ( ; T → next →  value  <= x ; T =  T → next) ;
if (T → down = = NULL){     /* a leaf */
            T → next = get_node(x, T → next, NULL);
            return 1;
}
if (!insert1(x, T → down) )  return 0;
/* check if as result of insertion, there are three nodes
 between T → down and T → next → down  */
q = T → down → next → next ;
if ( q → next → value < T → next → value ) {
      T → next = get_node(q → value, T → next, q);
      return 1 ;
}
return 0 ;
}
```

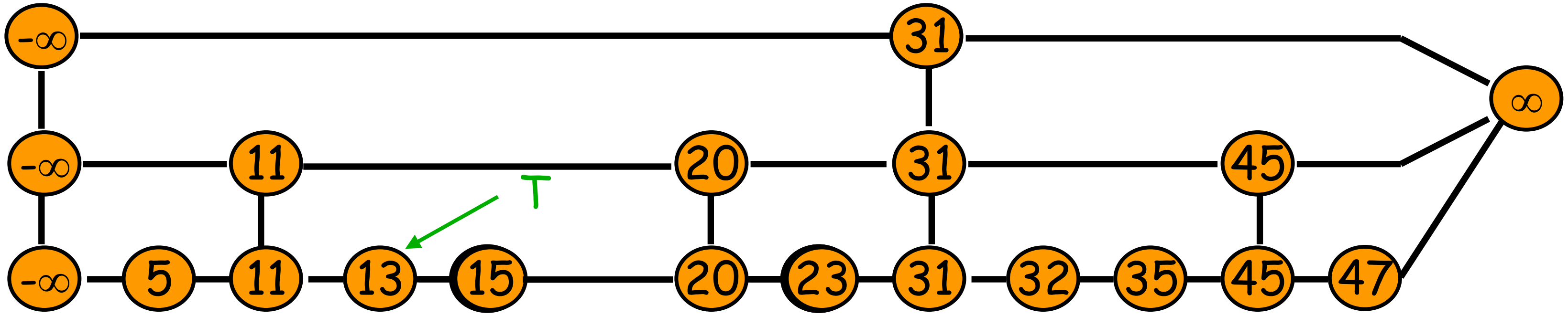
חיפוש ברמה נוכחית.

הוספה ברמה תחתונה.

קריאה רקורסיבית.

חזרה מקריאה רקורסיבית.

דוגמא: הכנס 17



©cs,Technion

תוכנית להכנסה ברשימת דילוגים דטרמיניסטית

```
int   insert1(int x, NODE  *T)
{ /* returns 1 iff new node allocated on this level */
NODE * q ;
for ( ; T → next →  value  <= x ; T =  T → next) ;
if (T → down = = NULL){     /* a leaf */
            T → next = get_node(x, T → next, NULL);
            return 1;
}
if (!insert1(x, T → down) )  return 0;
/* check if as result of insertion, there are three nodes
 between T → down and T → next → down  */
q = T → down → next → next ;
if ( q → next → value < T → next → value ) {
      T → next = get_node(q → value, T → next, q);
      return 1 ;
}
return 0 ;
}
```

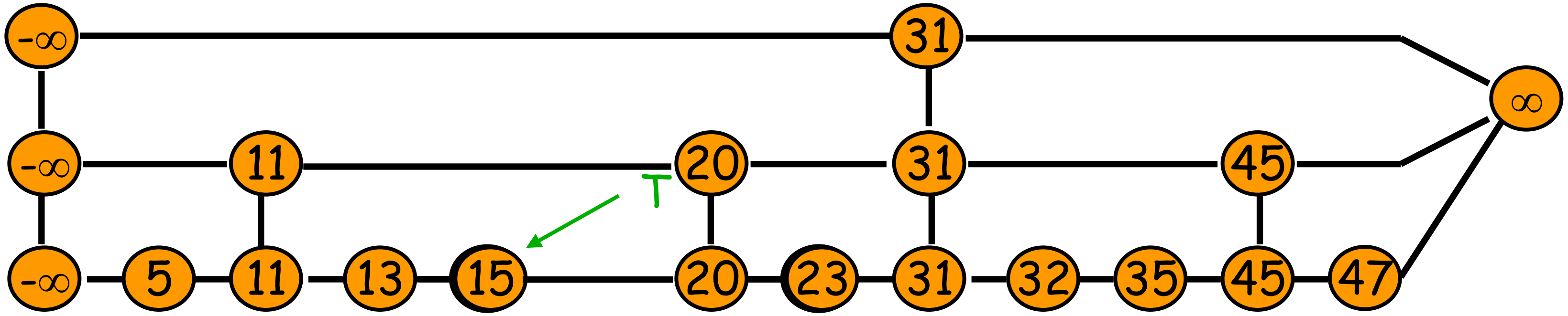
חיפוש ברמה נוכחית.

הוספה ברמה תחתונה.

קריאה רקורסיבית.

חזרה מקריאה רקורסיבית.

דוגמא: הכנס 17



תוכנית להכנסה ברשימת דילוגים דטרמיניסטית

```
int   insert1(int x, NODE  *T)
{ /* returns 1 iff new node allocated on this level */
NODE * q ;
for ( ; T → next →  value  <= x ; T =  T → next) ;
if (T → down = = NULL){     /* a leaf */
            T → next = get_node(x, T → next, NULL);
            return 1;
}
if (!insert1(x, T → down) )  return 0;
/* check if as result of insertion, there are three nodes
 between T → down and T → next → down  */
q = T → down → next → next ;
if ( q → next → value < T → next → value ) {
      T → next = get_node(q → value, T → next, q);
      return 1 ;
}
return 0 ;
}
```

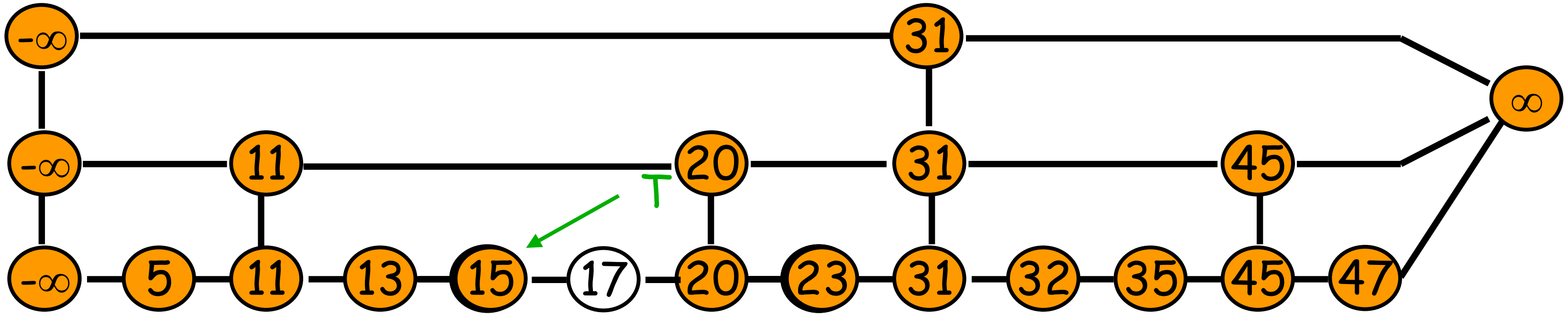
חיפוש ברמה נוכחית.

הוספה ברמה תחתונה.

קריאה רקורסיבית.

חזרה מקריאה רקורסיבית.

דוגמא: הכנס 17



©cs,Technion

תוכנית להכנסה ברשימת דילוגים דטרמיניסטית

```
int   insert1(int x, NODE  *T)
{ /* returns 1 iff new node allocated on this level */
NODE * q ;
for ( ; T → next →  value  <= x ; T =  T → next) ;
if (T → down = = NULL){     /* a leaf */
            T → next = get_node(x, T → next, NULL);
            return 1;
}
if (!insert1(x, T → down) )  return 0;
/* check if as result of insertion, there are three nodes
 between T → down and T → next → down  */
q = T → down → next → next ;
if ( q → next → value < T → next → value ) {
      T → next = get_node(q → value, T → next, q);
      return 1 ;
}
return 0 ;
}
```

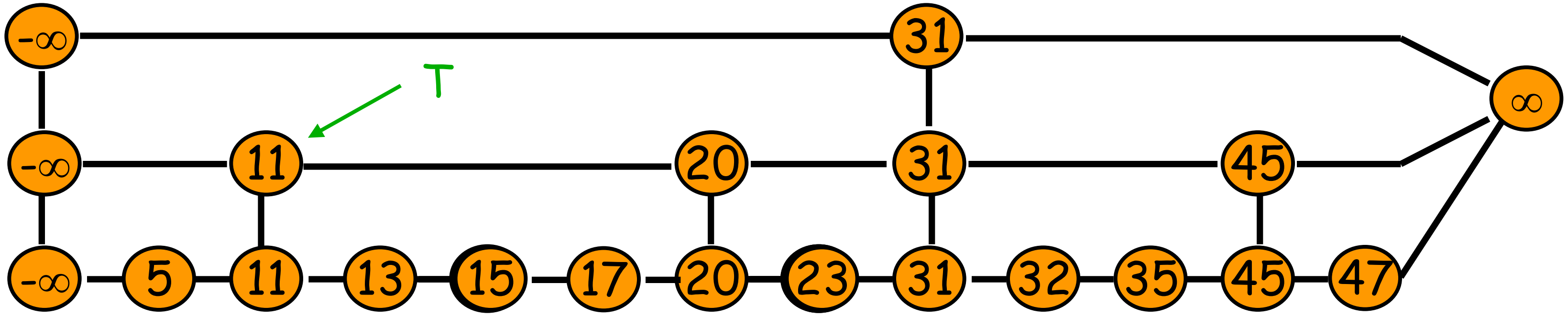
חיפוש ברמה נוכחית.

הוספה ברמה תחתונה.

קריאה רקורסיבית.

חזרה מקריאה רקורסיבית.

דוגמא: הכנס 17



©cs,Technion

תוכנית להכנסה ברשימת דילוגים דטרמיניסטית

```
int   insert1(int x, NODE  *T)
{ /* returns 1 iff new node allocated on this level */
NODE * q ;
for ( ; T → next →  value  <= x ; T =  T → next) ;
if (T → down = = NULL){     /* a leaf */
            T → next = get_node(x, T → next, NULL);
            return 1;
}
if (!insert1(x, T → down) )  return 0;
/* check if as result of insertion, there are three nodes
 between T → down and T → next → down  */
q = T → down → next → next ;
if ( q → next → value < T → next → value ) {
      T → next = get_node(q → value, T → next, q);
      return 1 ;
}
return 0 ;
}
```

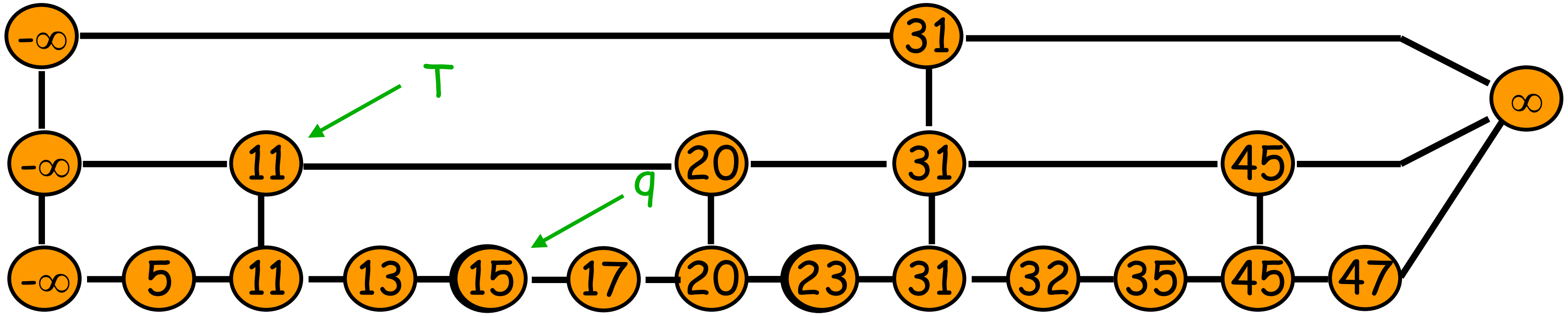
חיפוש ברמה נוכחית.

הוספה ברמה תחתונה.

קריאה רקורסיבית.

חזרה מקריאה רקורסיבית.

דוגמא: הכנס 17



©cs,Technion

תוכנית להכנסה ברשימת דילוגים דטרמיניסטית

```
int   insert1(int x, NODE  *T)
{ /* returns 1 iff new node allocated on this level */
NODE * q ;
for ( ; T → next →  value  <= x ; T =  T → next) ;
if (T → down = = NULL){     /* a leaf */
            T → next = get_node(x, T → next, NULL);
            return 1;
}
if (!insert1(x, T → down) )  return 0;
/* check if as result of insertion, there are three nodes
 between T → down and T → next → down  */
q = T → down → next → next ;
if ( q → next → value < T → next → value ) {
      T → next = get_node(q → value, T → next, q);
      return 1 ;
}
return 0 ;
}
```

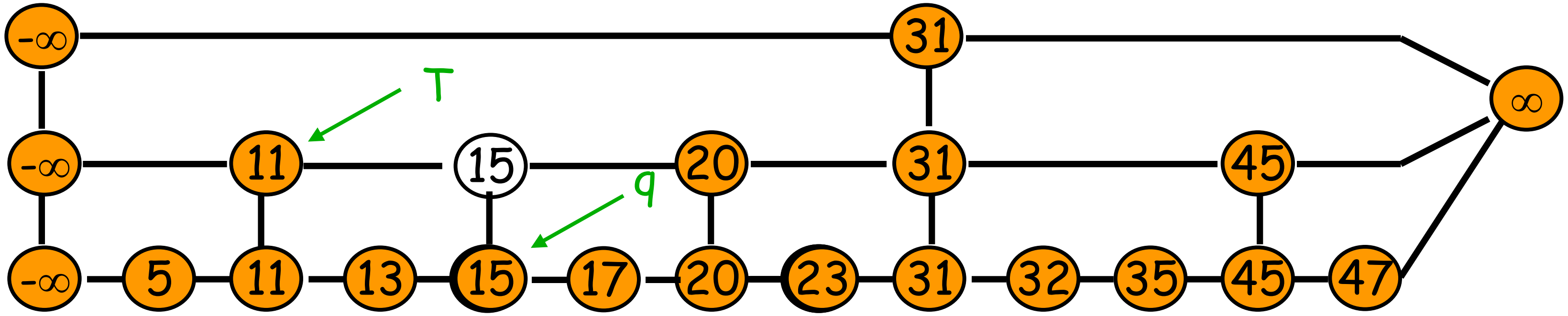
חיפוש ברמה נוכחית.

הוספה ברמה תחתונה.

קריאה רקורסיבית.

חזרה מקריאה רקורסיבית.

דוגמא: הכנס 17



תוכנית להכנסה ברשימת דילוגים דטרמיניסטית

```
int   insert1(int x, NODE  *T)
{ /* returns 1 iff new node allocated on this level */
NODE * q ;
for ( ; T → next →  value  <= x ; T =  T → next) ;
if (T → down = = NULL){     /* a leaf */
            T → next = get_node(x, T → next, NULL);
            return 1;
}
if (!insert1(x, T → down) )  return 0;
/* check if as result of insertion, there are three nodes
 between T → down and T → next → down  */
q = T → down → next → next ;
if ( q → next → value < T → next → value ) {
      T → next = get_node(q → value, T → next, q);
      return 1 ;
}
return 0 ;
}
```

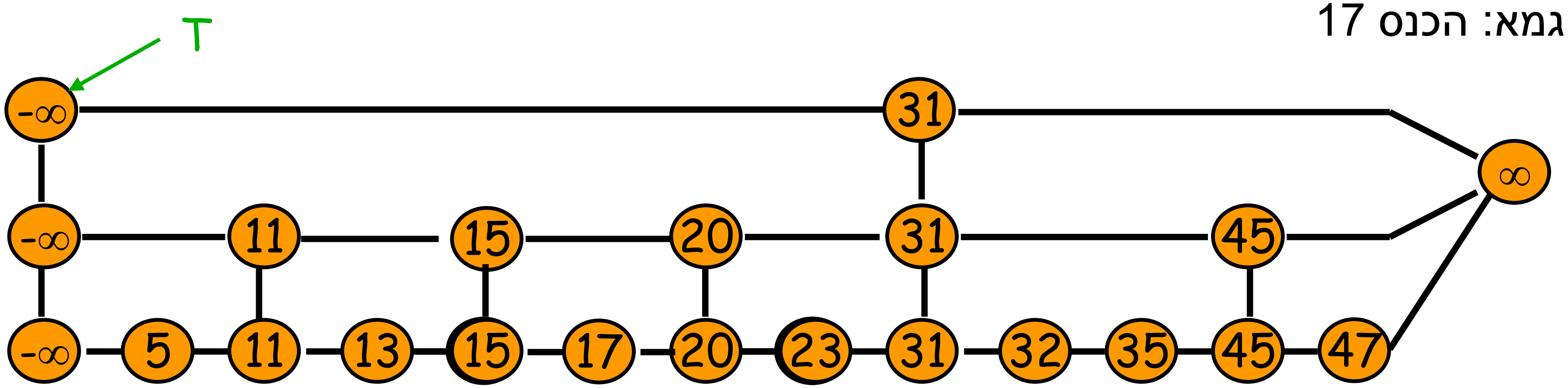
חיפוש ברמה נוכחית.

הוספה ברמה תחתונה.

קריאה רקורסיבית.

חזרה מקריאה רקורסיבית.

דוגמא: הכנס 17



©cs,Technion

תוכנית להכנסה ברשימת דילוגים דטרמיניסטית

```
int   insert1(int x, NODE  *T)
{ /* returns 1 iff new node allocated on this level */
NODE * q ;
for ( ; T → next →  value  <= x ; T =  T → next) ;
if (T → down = = NULL){     /* a leaf */
            T → next = get_node(x, T → next, NULL);
            return 1;
}
if (!insert1(x, T → down) )  return 0;
/* check if as result of insertion, there are three nodes
 between T → down and T → next → down  */
q = T → down → next → next ;
if ( q → next → value < T → next → value ) {
      T → next = get_node(q → value, T → next, q);
      return 1 ;
}
return 0 ;
}
```

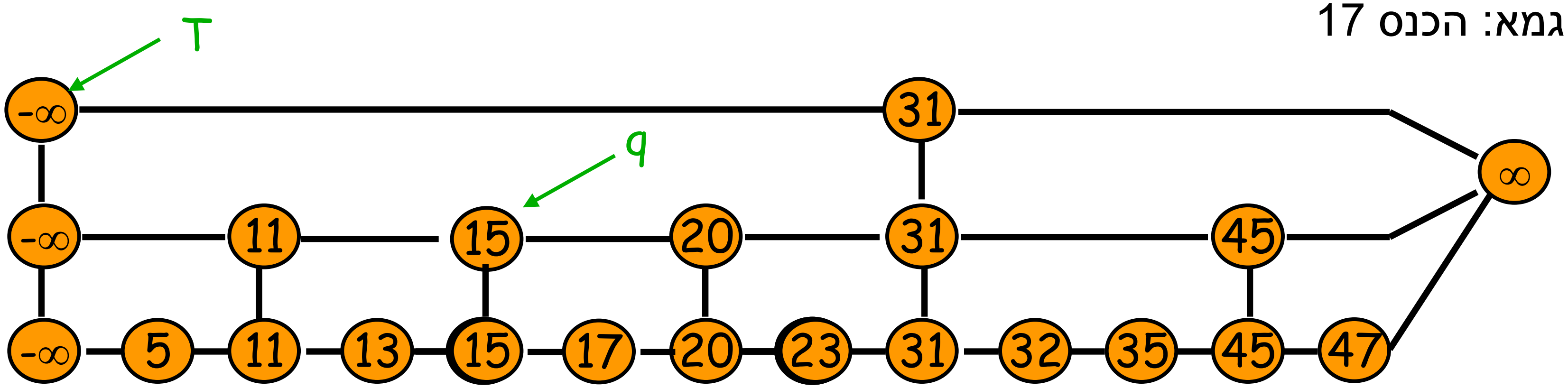
חיפוש ברמה נוכחית.

הוספה ברמה תחתונה.

קריאה רקורסיבית.

חזרה מקריאה רקורסיבית.

דוגמא: הכנס 17



©cs,Technion

תוכנית להכנסה ברשימת דילוגים דטרמיניסטית

```
int   insert1(int x, NODE  *T)
{ /* returns 1 iff new node allocated on this level */
NODE * q ;
for ( ; T → next →  value  <= x ; T =  T → next) ;
if (T → down = = NULL){     /* a leaf */
             T → next = get_node(x, T → next, NULL);
             return 1;
}
if (!insert1(x, T → down) )  return 0;
/* check if as result of insertion, there are three nodes
 between T → down and T → next → down  */
q = T → down → next → next ;
if ( q → next → value < T → next → value ) {
       T → next = get_node(q → value, T → next, q);
       return 1 ;
}
return 0 ;
}
```

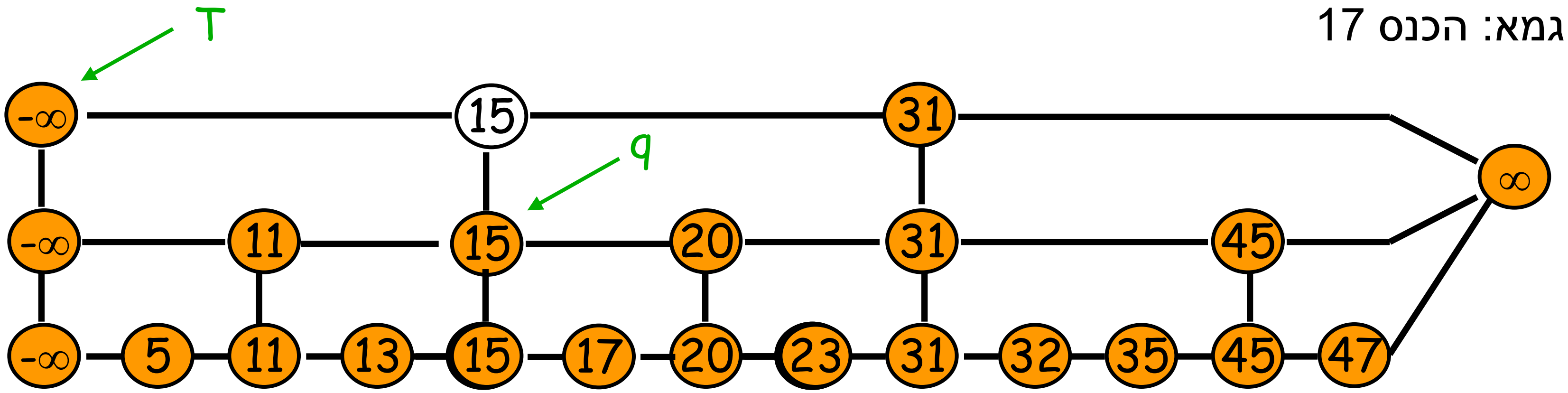
חיפוש ברמה נוכחית.

הוספה ברמה תחתונה.

קריאה רקורסיבית.

חזרה מקריאה רקורסיבית.

דוגמא: הכנס 17



תוכנית להכנסה ברשימת דילוגים דטרמיניסטית

```
int   insert1(int x, NODE  *T)
{ /* returns 1 iff new node allocated on this level */
NODE * q ;
for ( ; T → next →  value  <= x ; T =  T → next) ;
if (T → down = = NULL){     /* a leaf */
            T → next = get_node(x, T → next, NULL);
            return 1;
}
if (!insert1(x, T → down) )  return 0;
/* check if as result of insertion, there are three nodes
 between T → down and T → next → down  */
q = T → down → next → next ;
if ( q → next → value < T → next → value ) {
      T → next = get_node(q → value, T → next, q);
      return 1 ;
}
return 0 ;
}
```

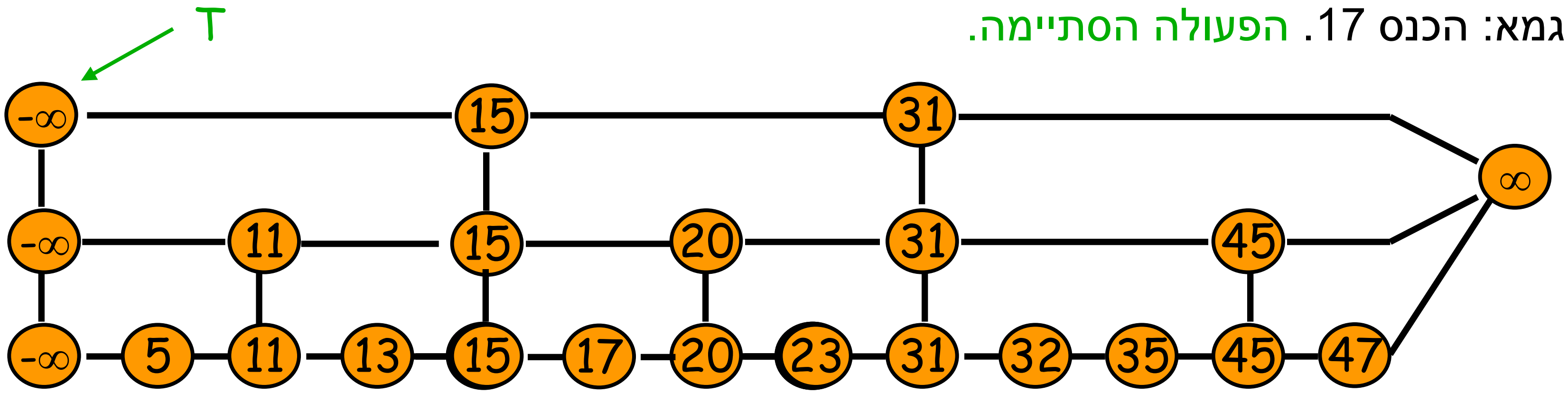
חיפוש ברמה נוכחית.

הוספה ברמה תחתונה.

קריאה רקורסיבית.

חזרה מקריאה רקורסיבית.

דוגמא: הכנס 17. הפעולה הסתיימה.



תוכנית להכנסה ברשימת דילוגים דטרמיניסטית

```
int   insert1(int x, NODE  *T)
{ /* returns 1 iff new node allocated on this level */
NODE * q ;
for ( ; T → next →  value  <= x ; T =  T → next) ;
if (T → down = = NULL){     /* a leaf */
            T → next = get_node(x, T → next, NULL);
            return 1;
}
if (!insert1(x, T → down) )  return 0;
/* check if as result of insertion, there are three nodes
 between T → down and T → next → down  */
q = T → down → next → next ;
if ( q → next → value < T → next → value ) {
      T → next = get_node(q → value, T → next, q);
      return 1 ;
}
return 0 ;
}
```

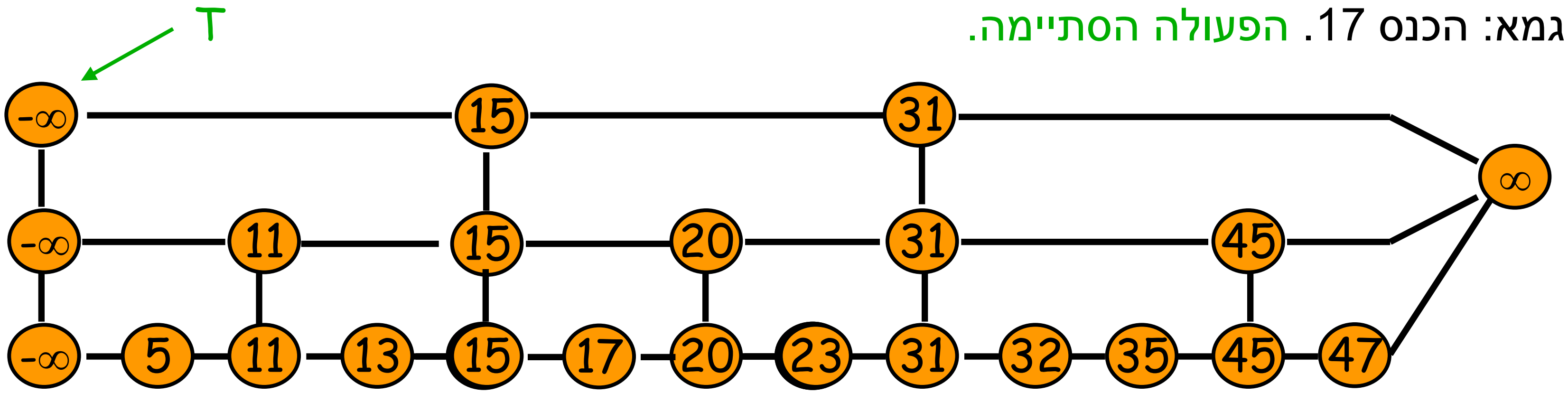
חיפוש ברמה נוכחית.

הוספה ברמה תחתונה.

קריאה רקורסיבית.

חזרה מקריאה רקורסיבית.

דוגמא: הכנס 17. הפעולה הסתיימה.



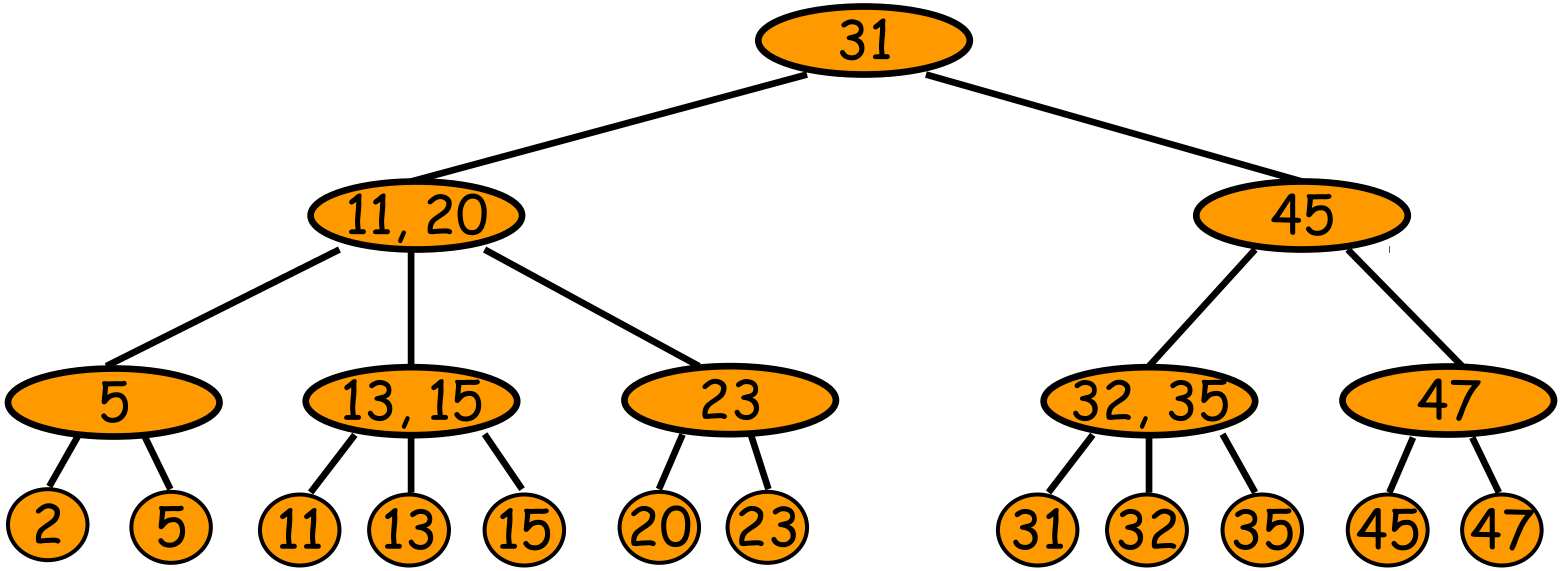
<u>הערה</u>: במידה והרמה העליונה היתה מכילה שני מפתחות אז היה צורך להוסיף רמה חדשה.

הקשר לעצי 3-2

נראה כעת כיצד עץ 3-2 הופך להיות רשימת דילוגים דטרמיניסטית. הטרנספורמציה שנראה נועדה להדגים את הקשר בין שני מבני הנתונים שלמדנו. בשום מצב אין צורך לממש טרנספורמציה זו בתוכנית מחשב.

©cs,Technion

רשימת דילוגים דטרמיניסטית בהקשר לעצי 3-2

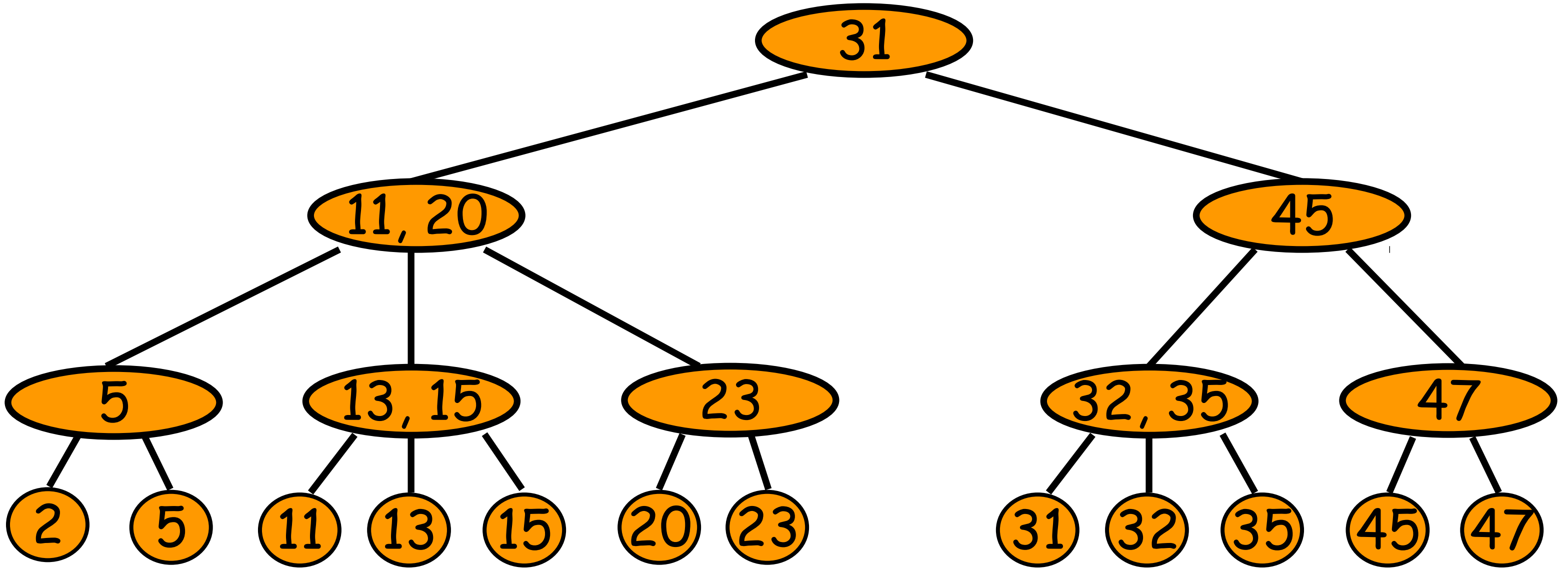


עץ 3-2

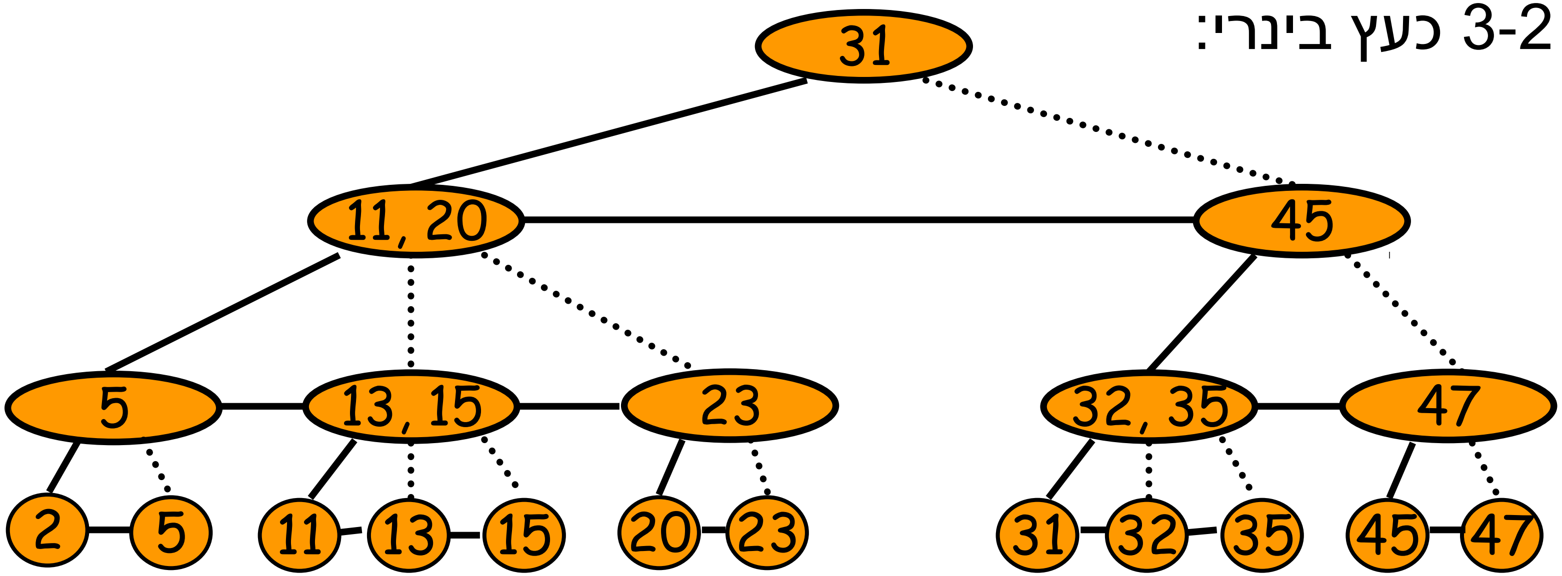
©cs,Technion

רשימת דילוגים דטרמיניסטית בהקשר לעצי 3-2

עץ 3-2

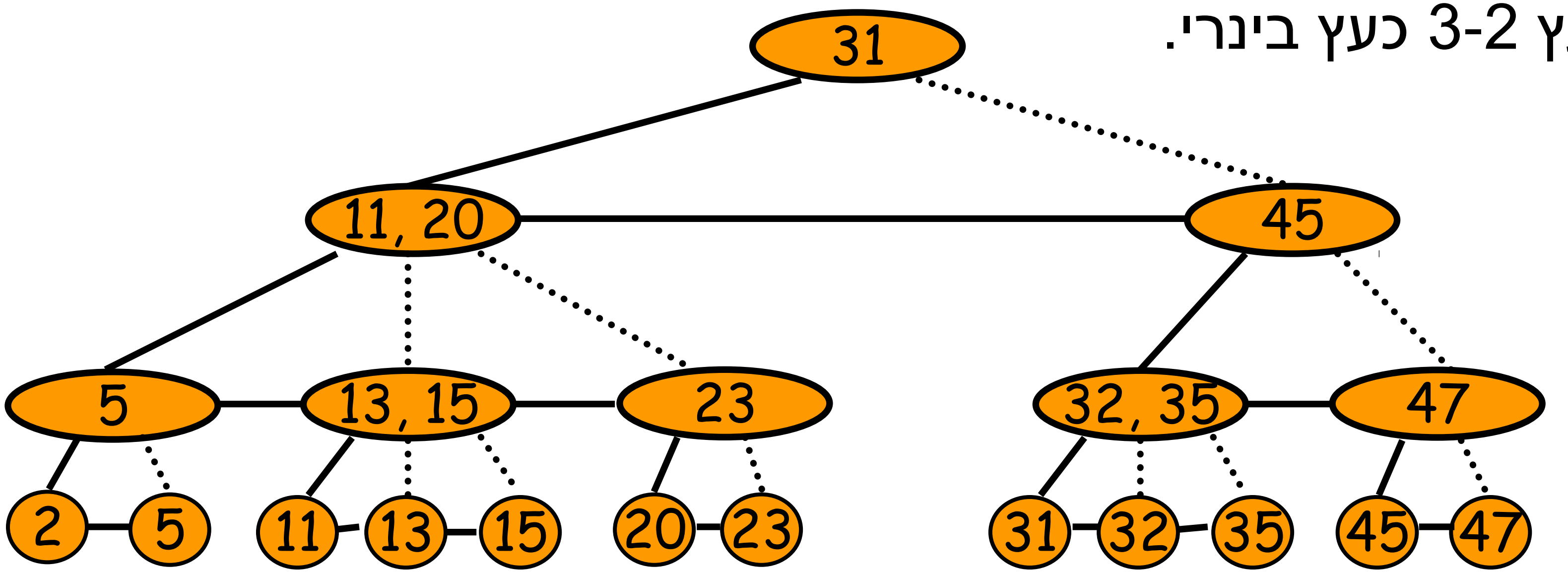


מימוש עץ 3-2 כעץ בינרי:

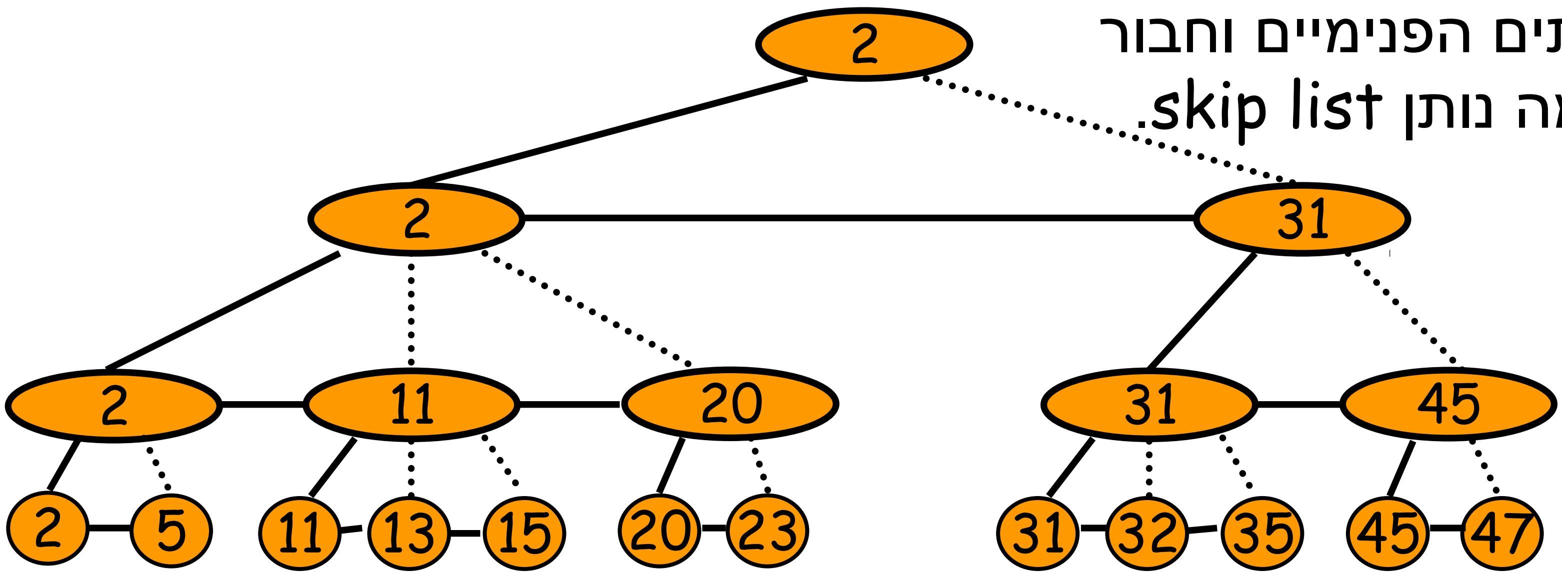


רשימת דילוגים דטרמיניסטית (המשך)

מימשנו עץ 2-3 כעץ בינרי.

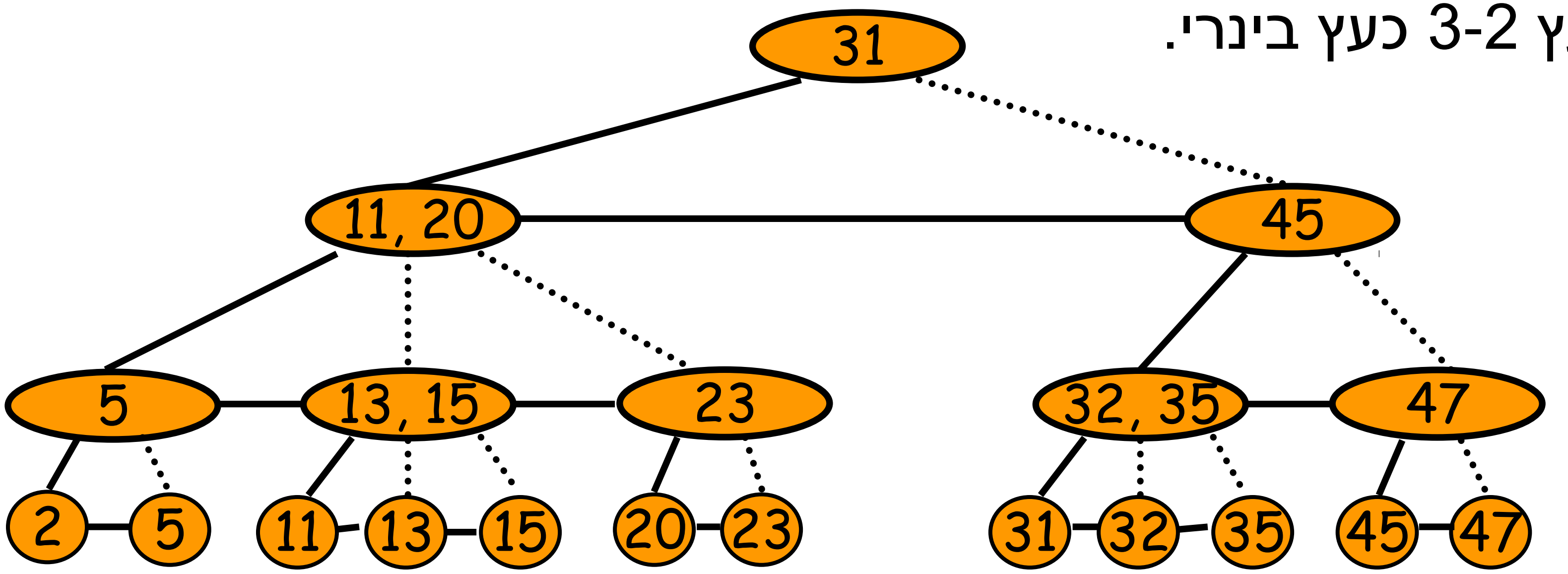


שינוי ערכי הצמתים הפנימיים וחבור הצמתים בכל רמה נותן skip list.

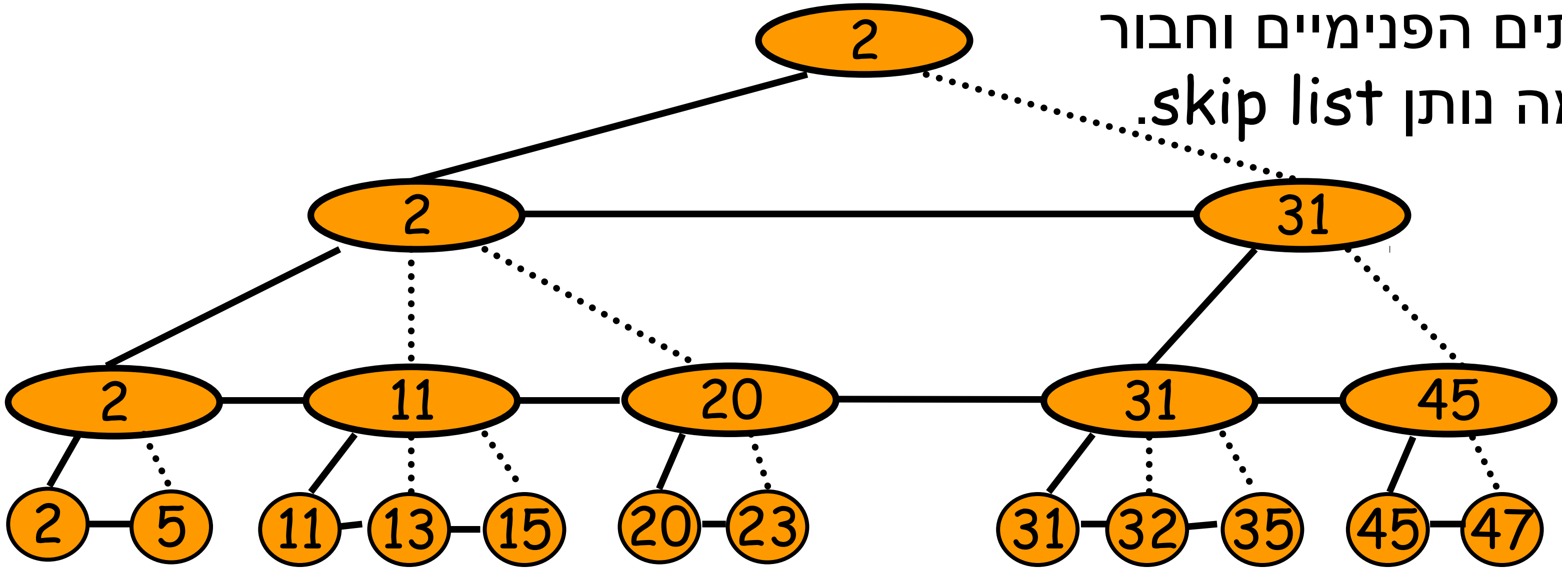


רשימת דילוגים דטרמיניסטית (המשך)

מימשנו עץ 3-2 כעץ בינרי.

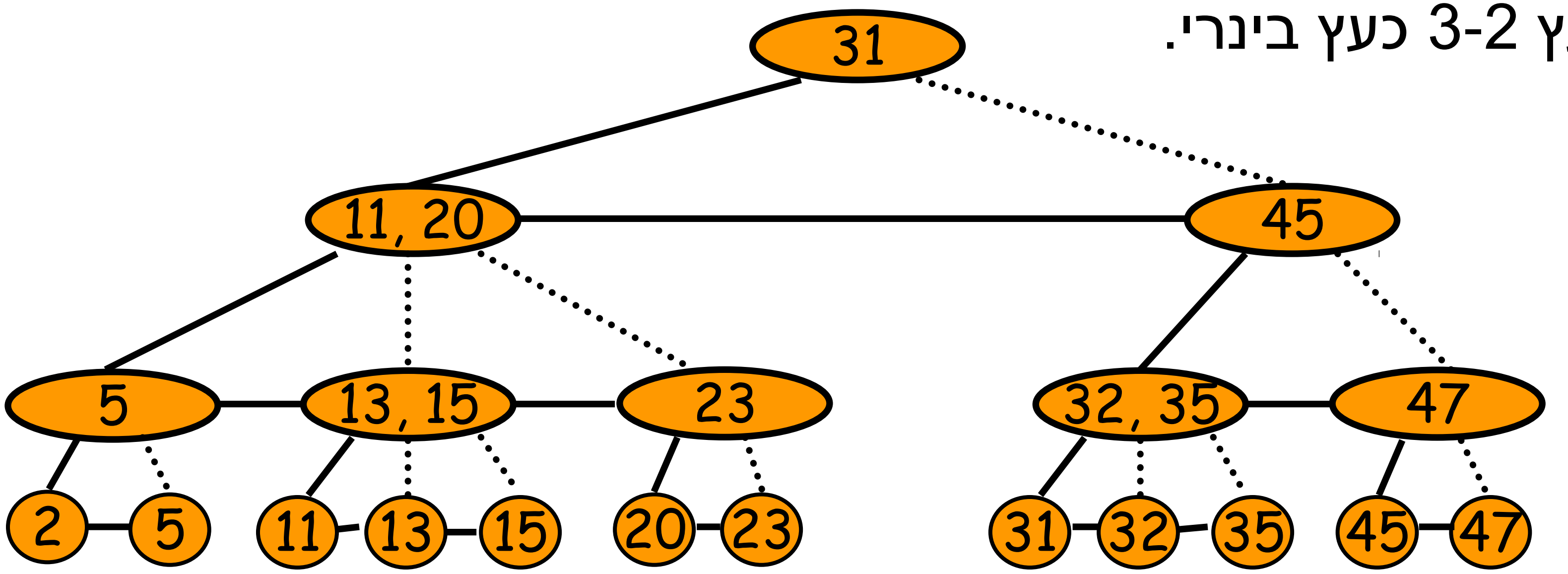


שינוי ערכי הצמתים הפנימיים וחבור הצמתים בכל רמה נותן skip list.

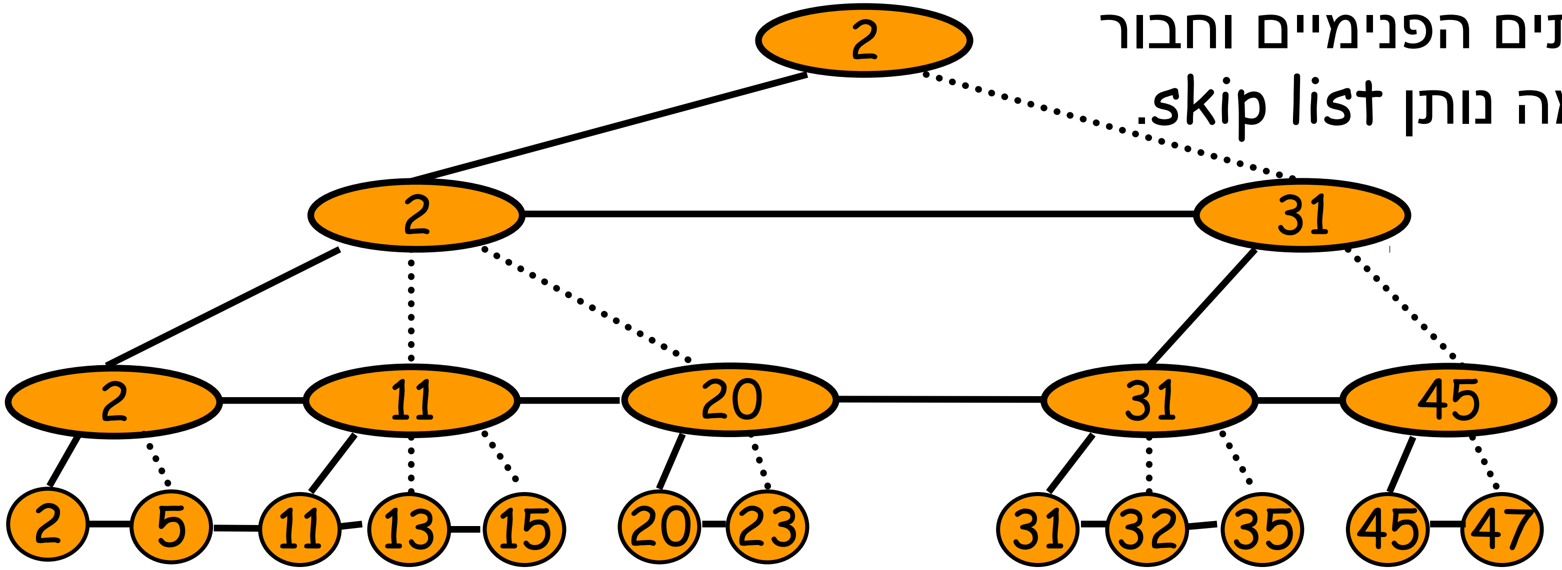


רשימת דילוגים דטרמיניסטית (המשך)

מימשנו עץ 2-3 כעץ בינרי.

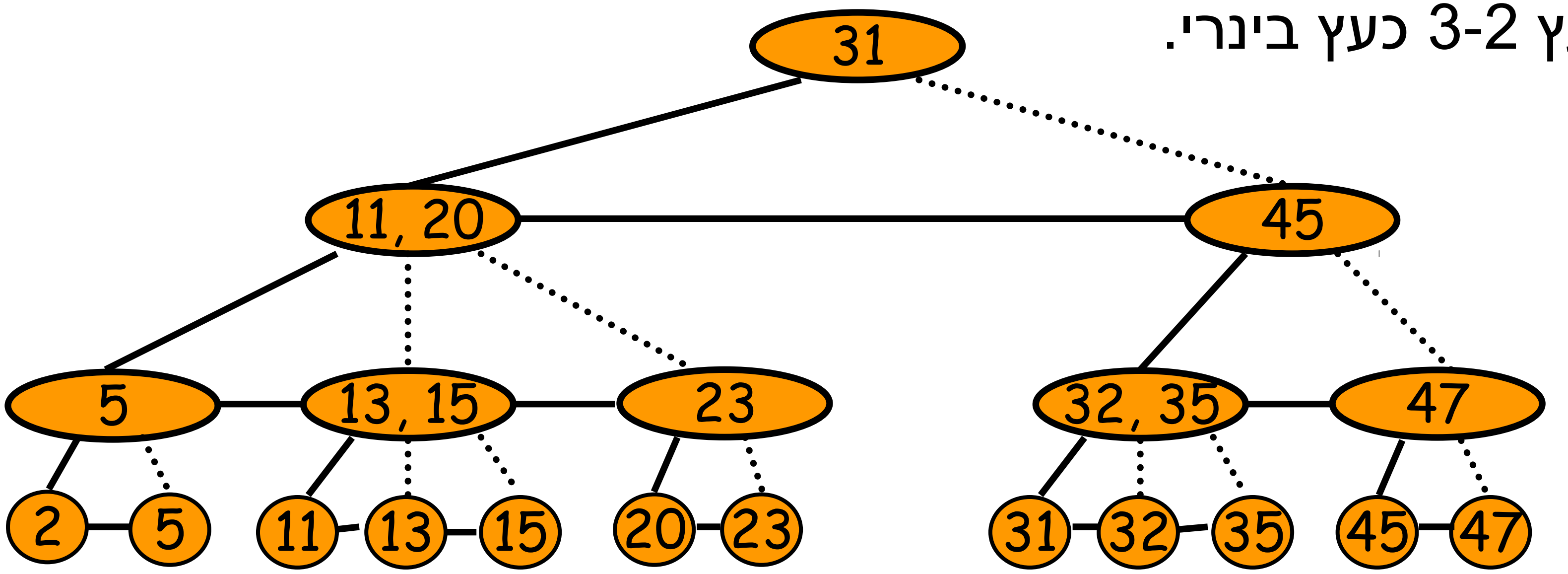


שינוי ערכי הצמתים הפנימיים וחבור
הצמתים בכל רמה נותן skip list.

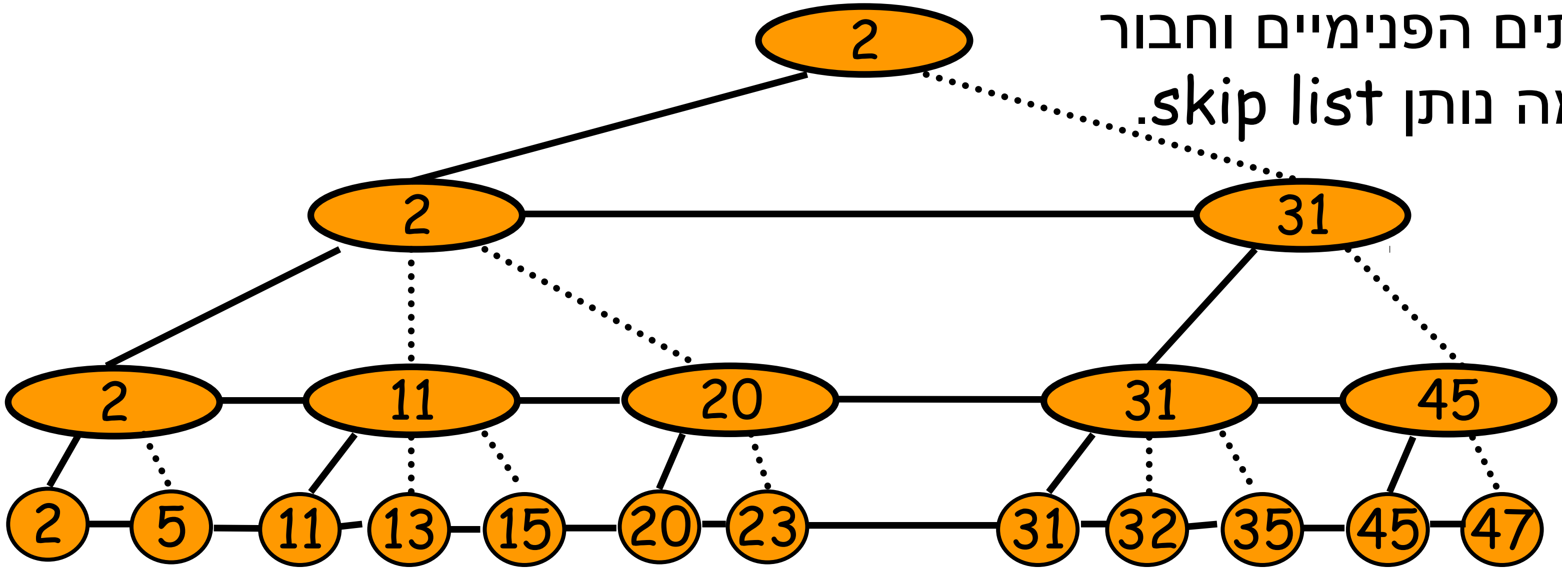


רשימת דילוגים דטרמיניסטית (המשך)

מימשנו עץ 2-3 כעץ בינרי.

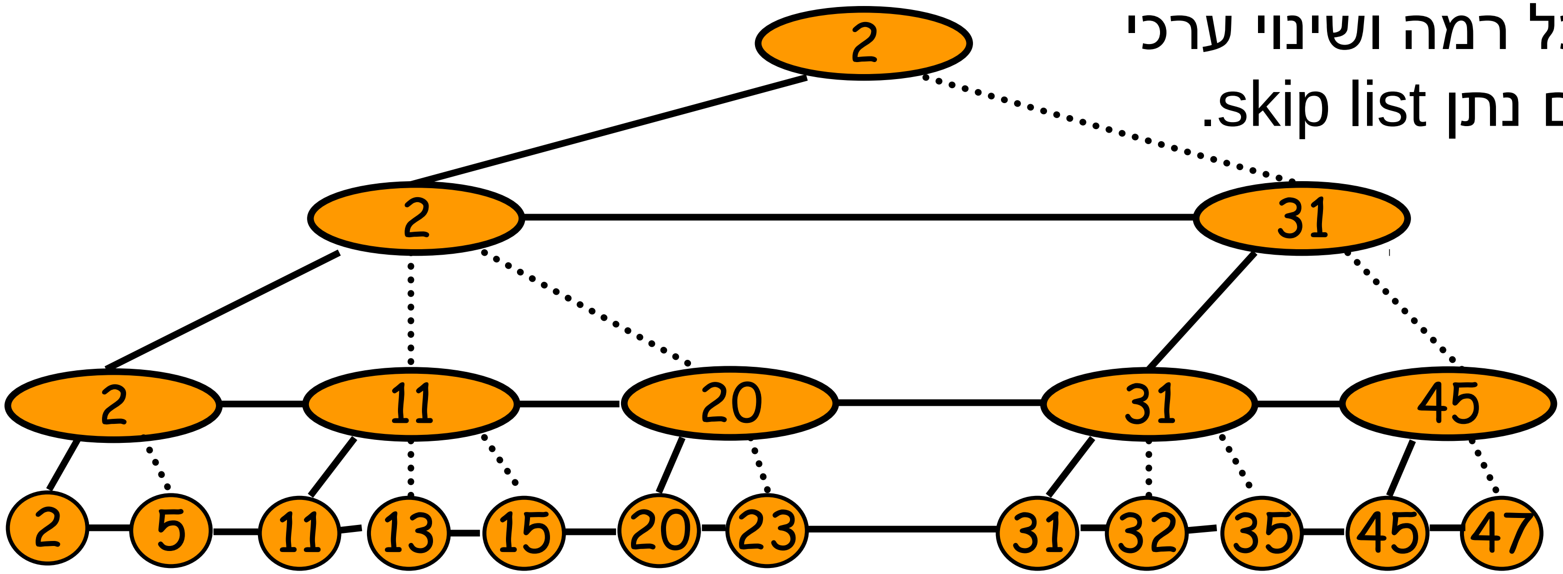


שינוי ערכי הצמתים הפנימיים וחבור
הצמתים בכל רמה נותן skip list.

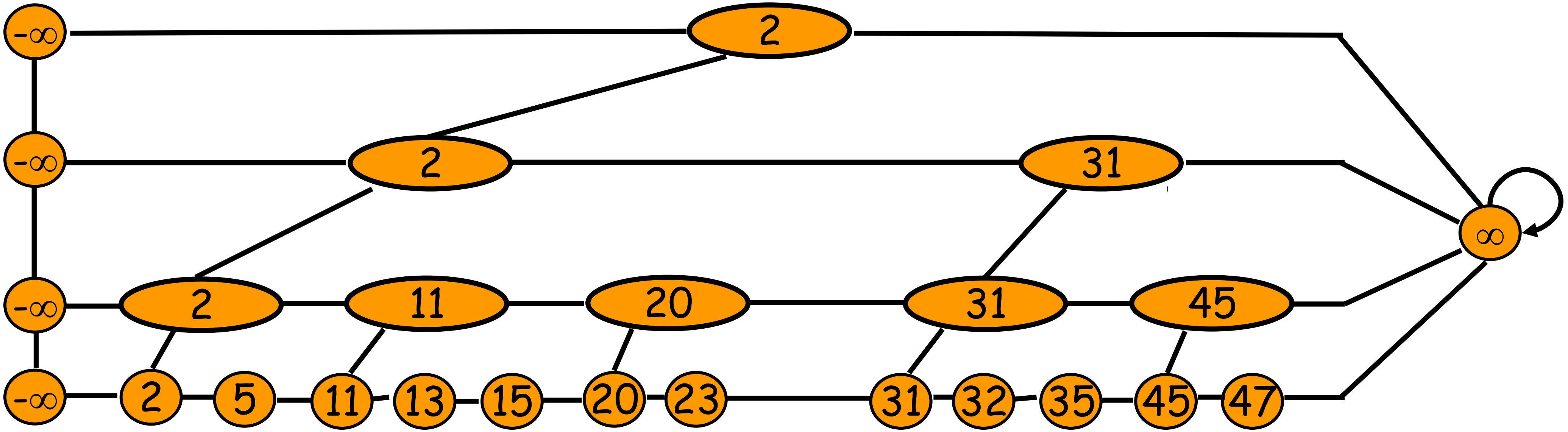


רשימת דילוגים דטרמיניסטית (המשך)

חבור הצמתים בכל רמה ושינוי ערכי
הצמתים הפנימיים נתן skip list.

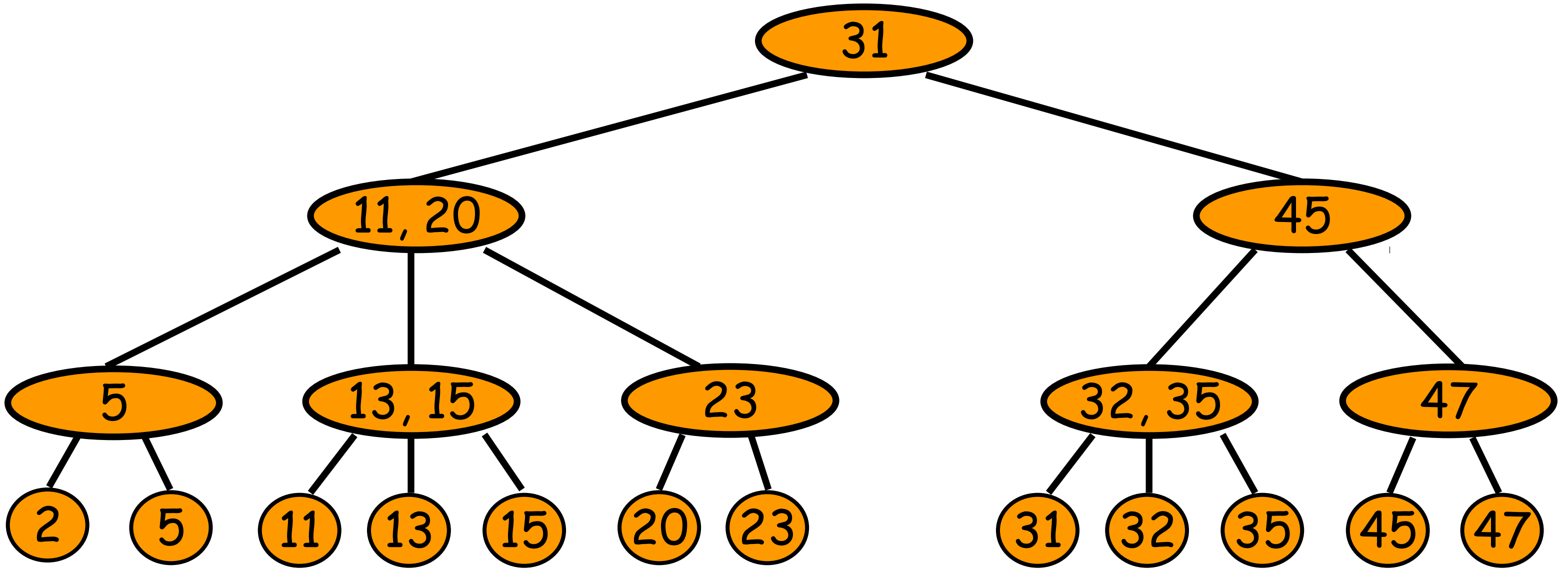


הוספת איבר ראשון ואיבר סופי.

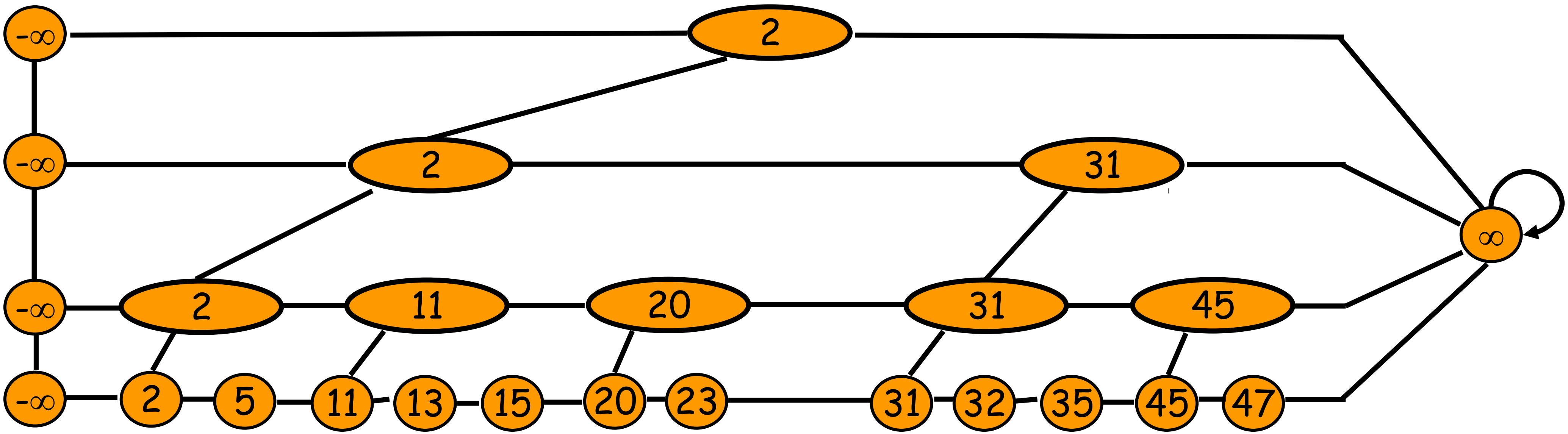


רשימת דילוגים דטרמיניסטית (סיכום)

העץ המקורי.



רשימת דילוגים אקוויוולנטית.



Skip lists