# מבני נתונים 08a

תרגול 8

14/2/2008

# המשך ערמות

ליאור שפירא

# ערמות פיבונאצ'י

| Operation | Linked List | Binary Heap | Binomial Heap | Fibonacci Heap † | Relaxed Heap |
|---|---|---|---|---|---|
| make-heap | 1 | 1 | 1 | 1 | 1 |
| is-empty | 1 | 1 | 1 | 1 | 1 |
| insert | 1 | log n | log n | 1 | 1 |
| delete-min | n | log n | log n | log n | log n |
| decrease-key | n | log n | log n | 1 | 1 |
| delete | n | log n | log n | log n | log n |
| union | 1 | n | log n | 1 | 1 |
| find-min | n | 1 | log n | 1 | 1 |

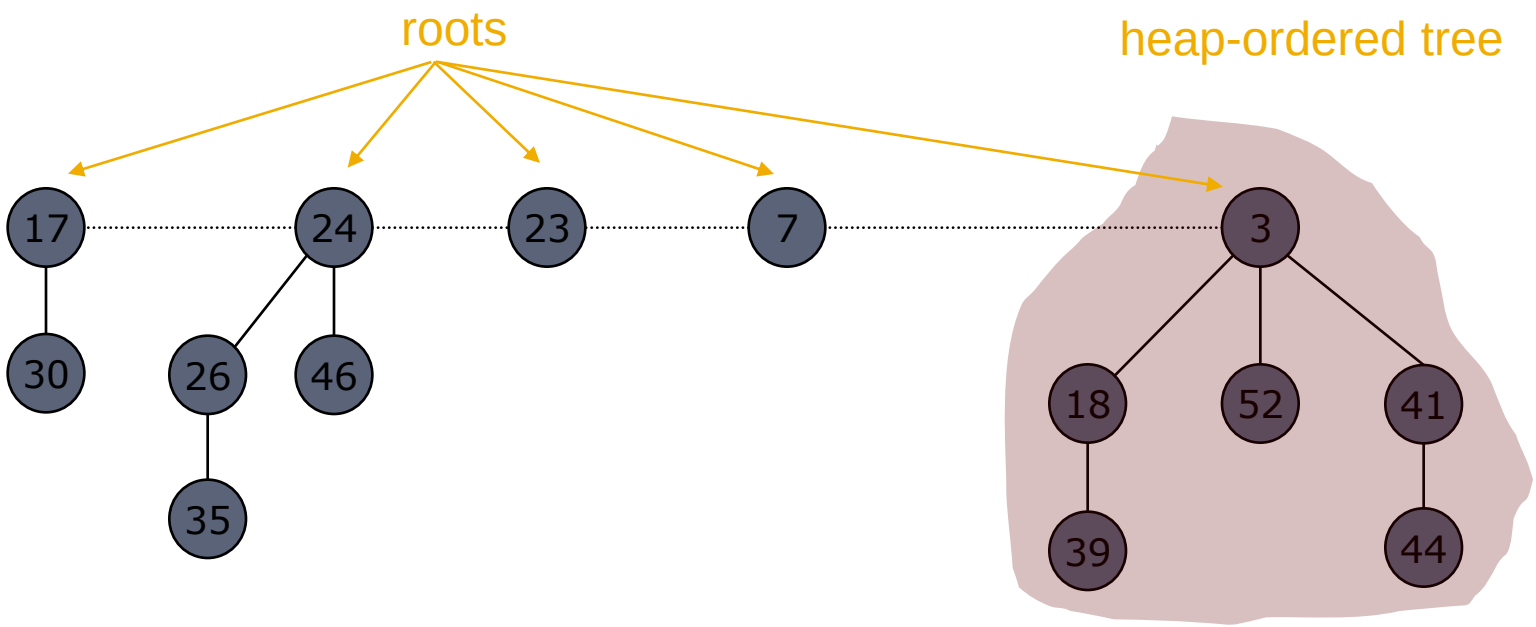
n = number of elements in priority queue

† amortized

# ערימות פיבונאצ'י - מבנה

- Fibonacci heap.
  - Set of heap-ordered trees.
  - Maintain pointer to minimum element.
  - Set of marked nodes.

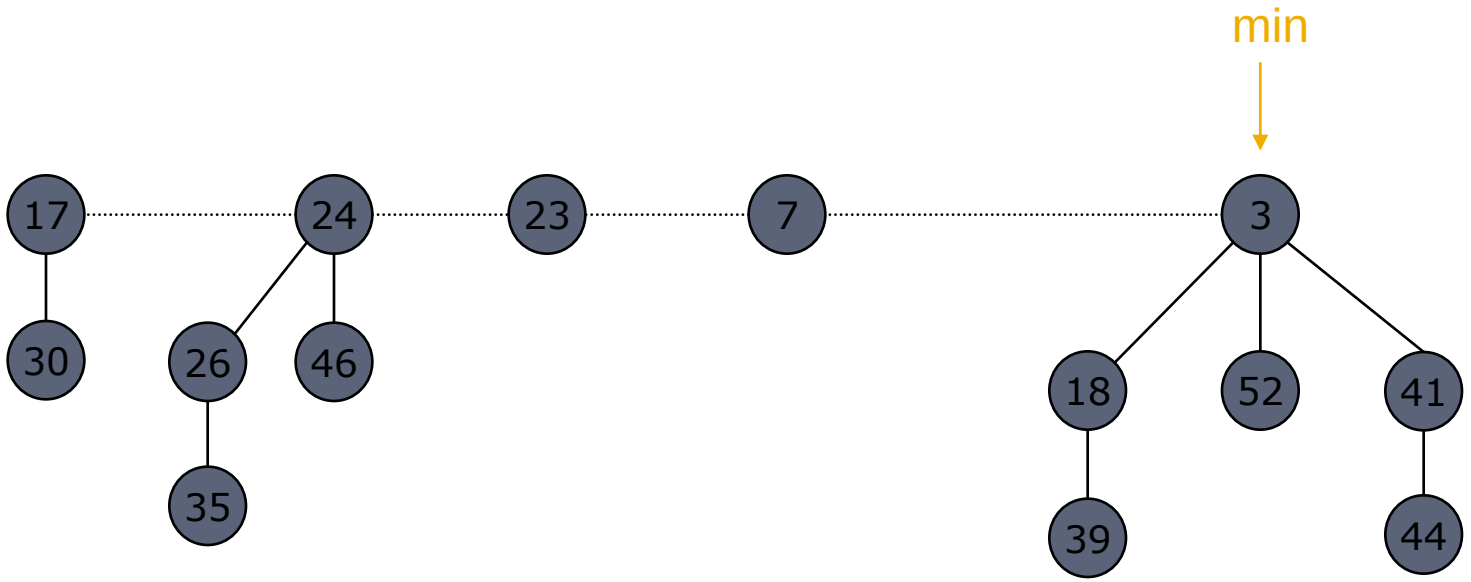
each parent larger than its children



# ערימות פיבונאצ'י - מבנה

- Fibonacci heap.
  - Set of heap-ordered trees.
  - Maintain pointer to minimum element.
  - Set of marked nodes.

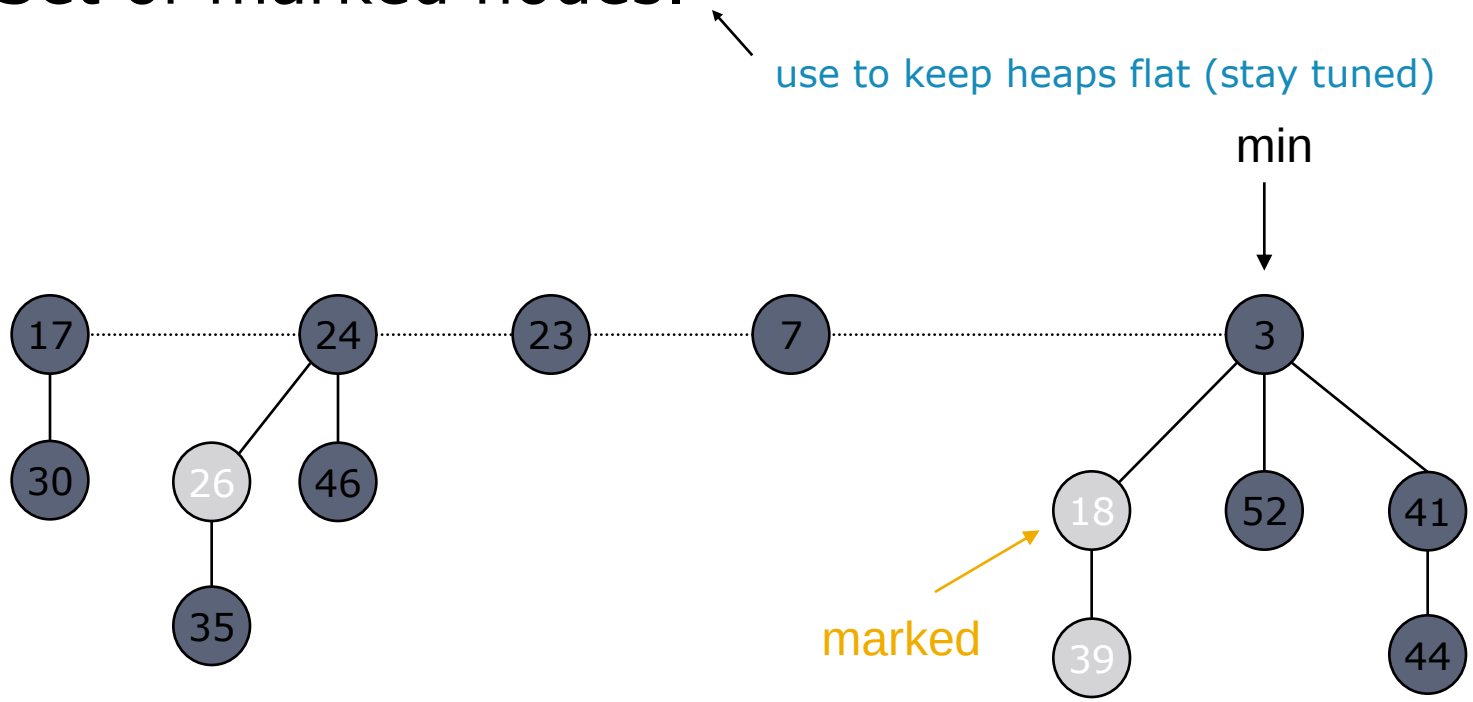
find-min takes O(1) time




Heap H

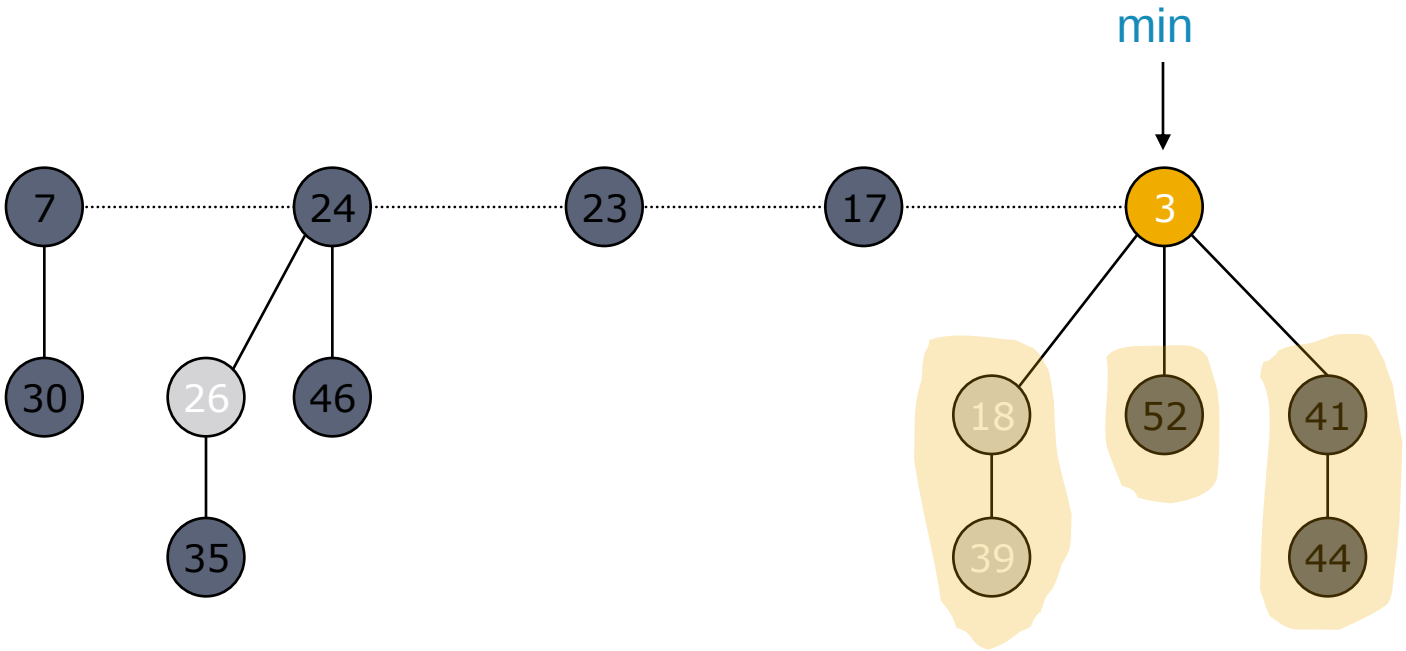
# ערימות פיבונאצ'י - מבנה

- Fibonacci heap.
  - Set of heap-ordered trees.
  - Maintain pointer to minimum element.
  - Set of marked nodes.



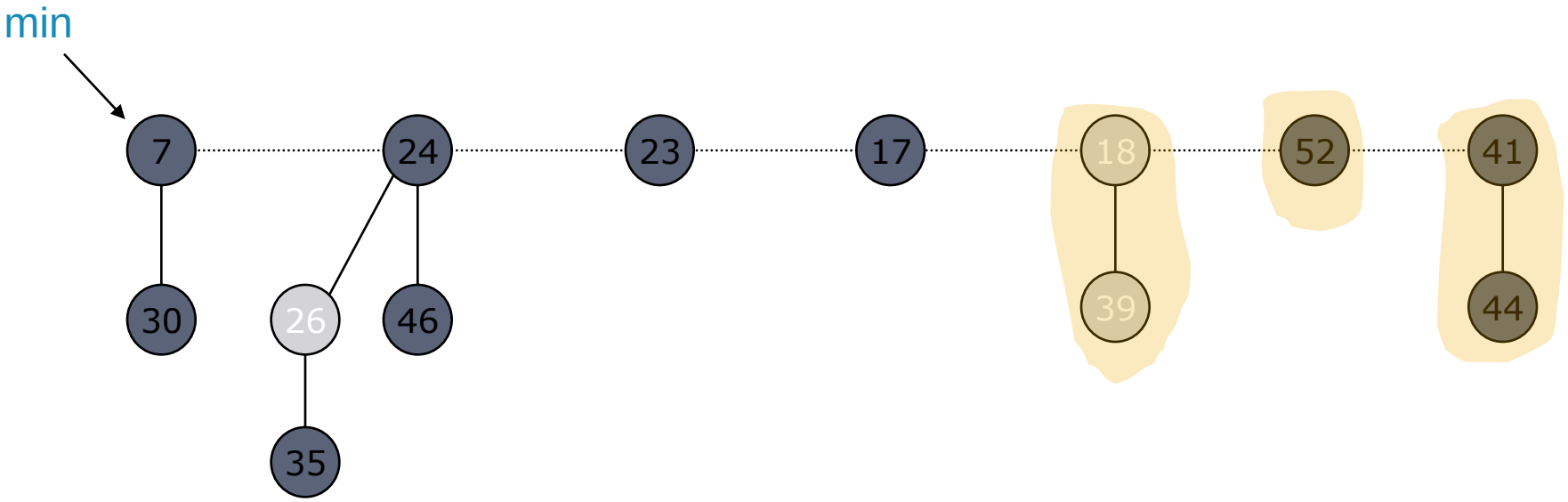
# Cascading cuts & Successive linking

- פעולת delete-min
  - יודעים מי המינימום בערימה
  - נתלה את הבנים שלו כעצים בערימה
  - נבצע successive linking – מכל דרגה עץ יחיד!



# Cascading cuts & Successive linking

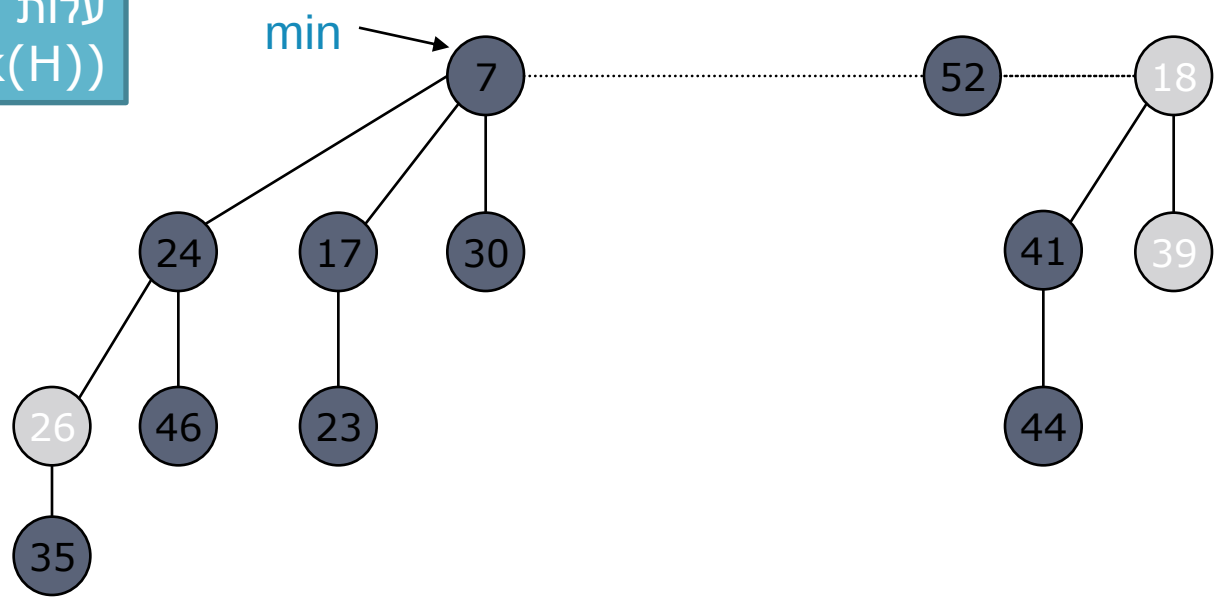
- פעולת delete-min
  - יודעים מי המינימום בערימה
  - נתלה את הבנים שלו כעצים בערימה
  - נבצע successive linking – מכל דרגה עץ יחיד!



# Cascading cuts & Successive linking

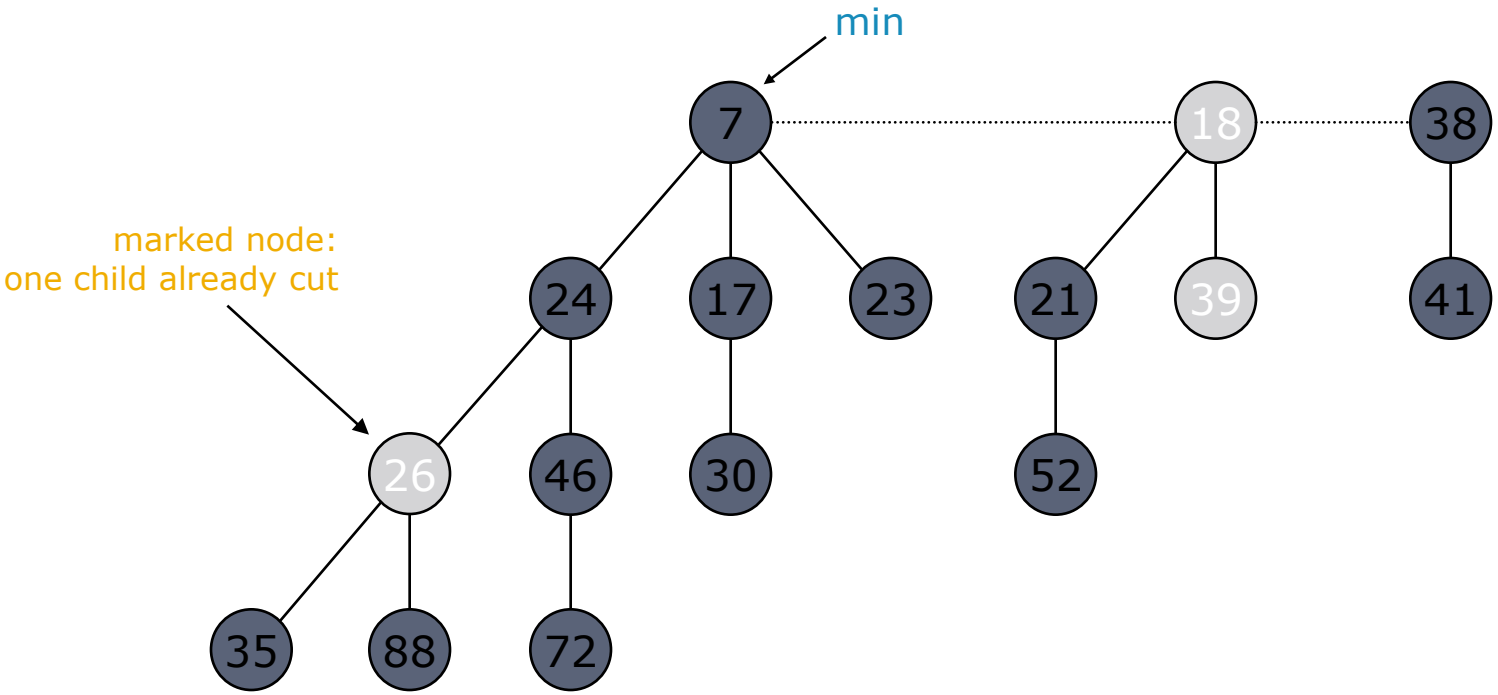
- פעולת delete-min
  - יודעים מי המינימום בערימה
  - נתלה את הבנים שלו כעצים בערימה
  - נבצע successive linking – מכל דרגה עץ יחיד!

עלות הפעולה:
Amortized O(rank(H))



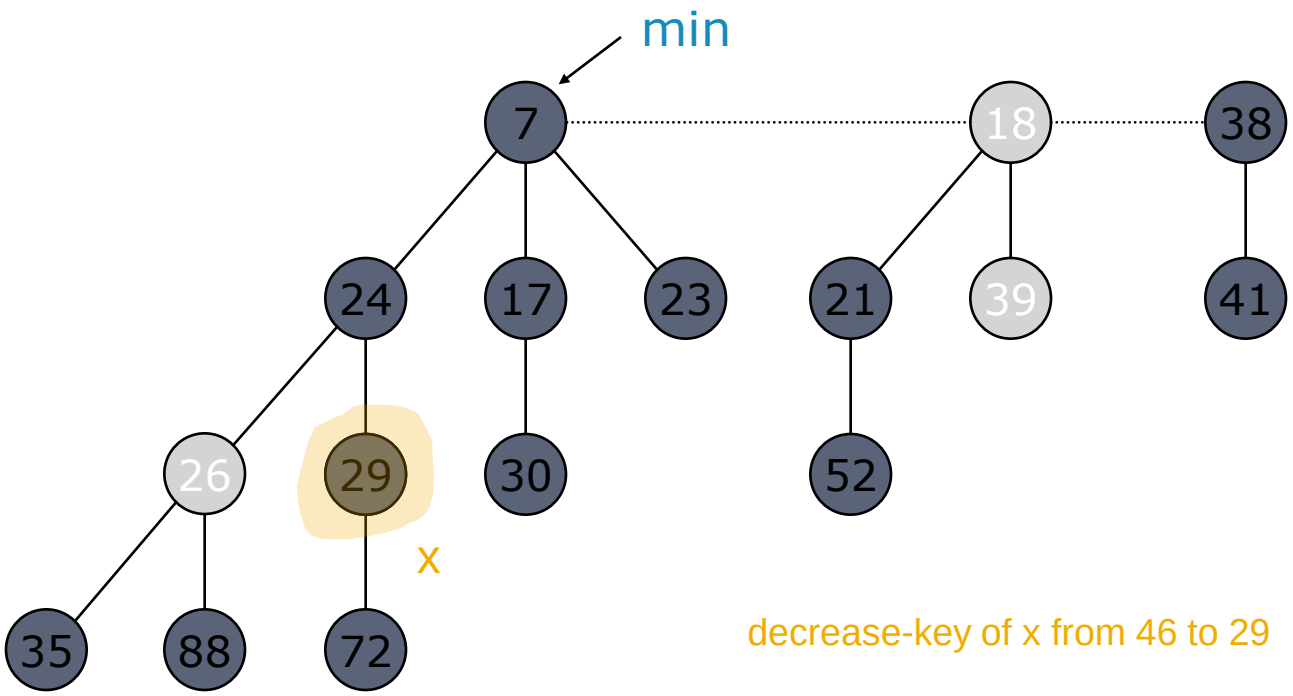
# Cascading cuts & Successive linking

- פעולת decrease-key
  - אינטואיציה:
    - אם חוק הערמה לא מופר, פשוט נשנה את ערך הצומת
    - אחרת נחתוך את תת העץ של הצומת ונתלה כעץ חדש
    - בכדי לשמור על עצים שטוחים יחסית, ברגע שחותכים בן שני לצומת מסוים, גם הוא נתלה כעץ חדש



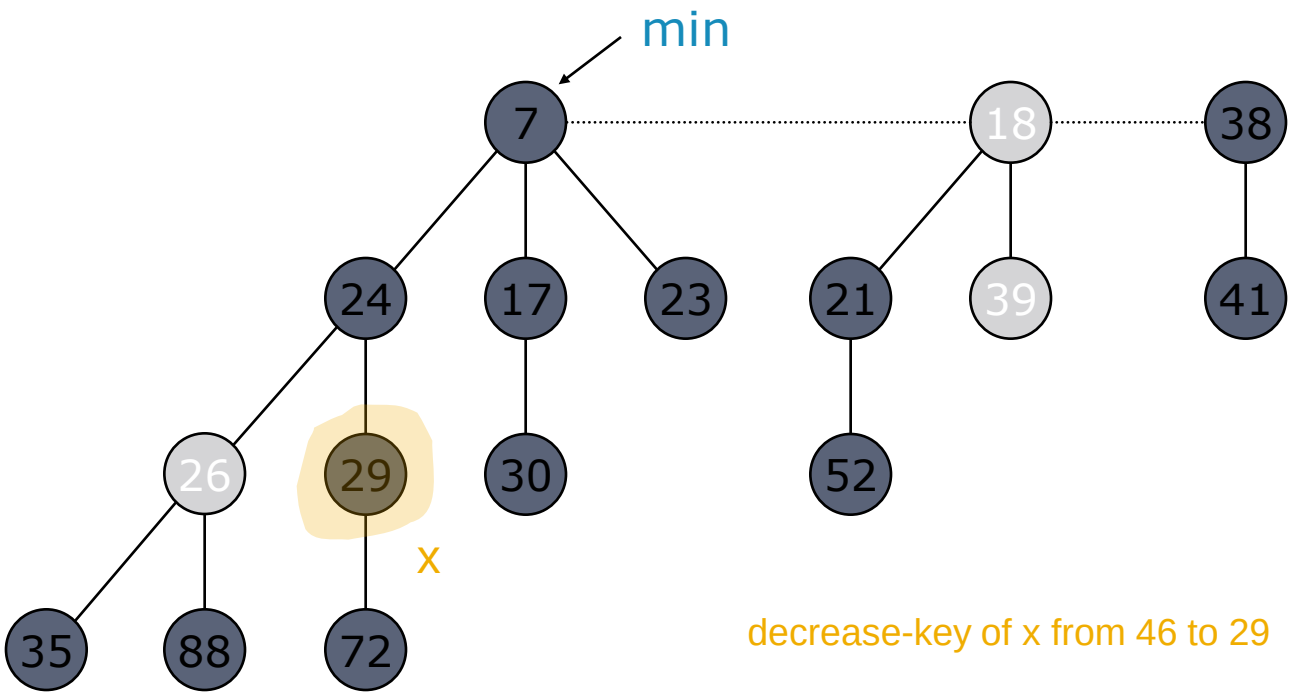
# Fibonacci Heaps: Decrease Key

- Case 1. [heap order not violated]
  - Decrease key of x.
  - Change heap min pointer (if necessary).



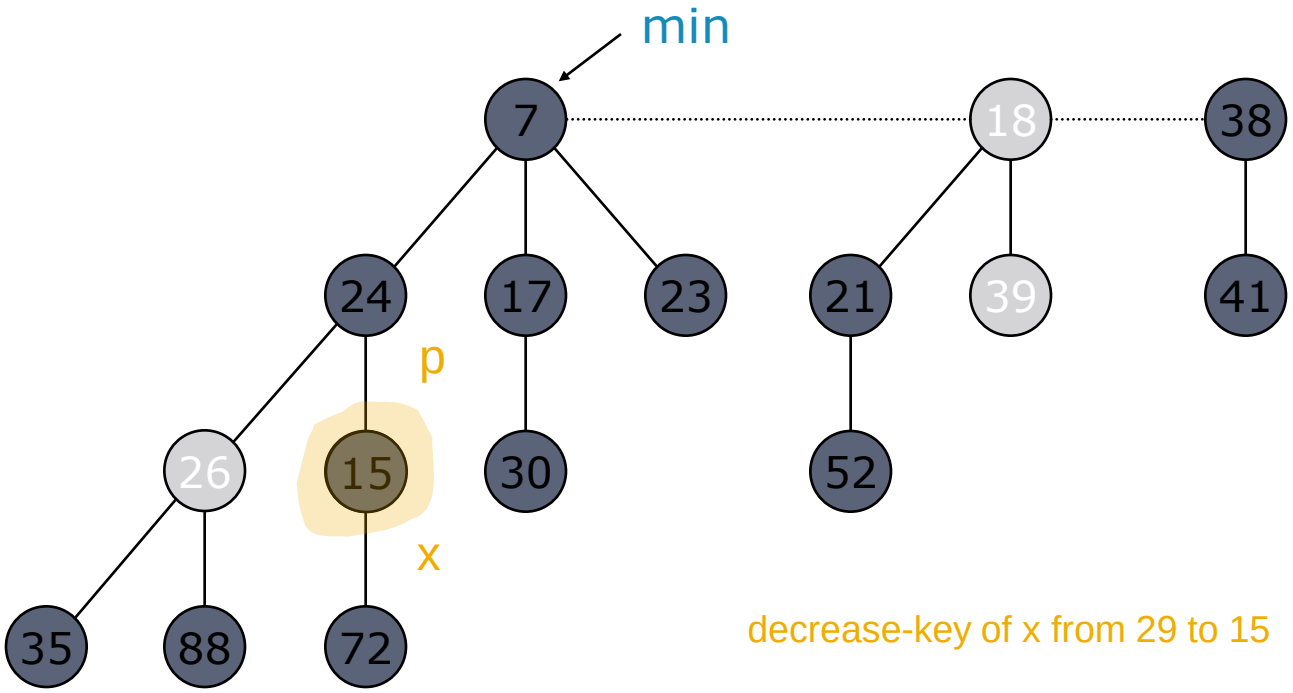
# Fibonacci Heaps: Decrease Key

- Case 1. [heap order not violated]
  - Decrease key of x.
  - Change heap min pointer (if necessary).



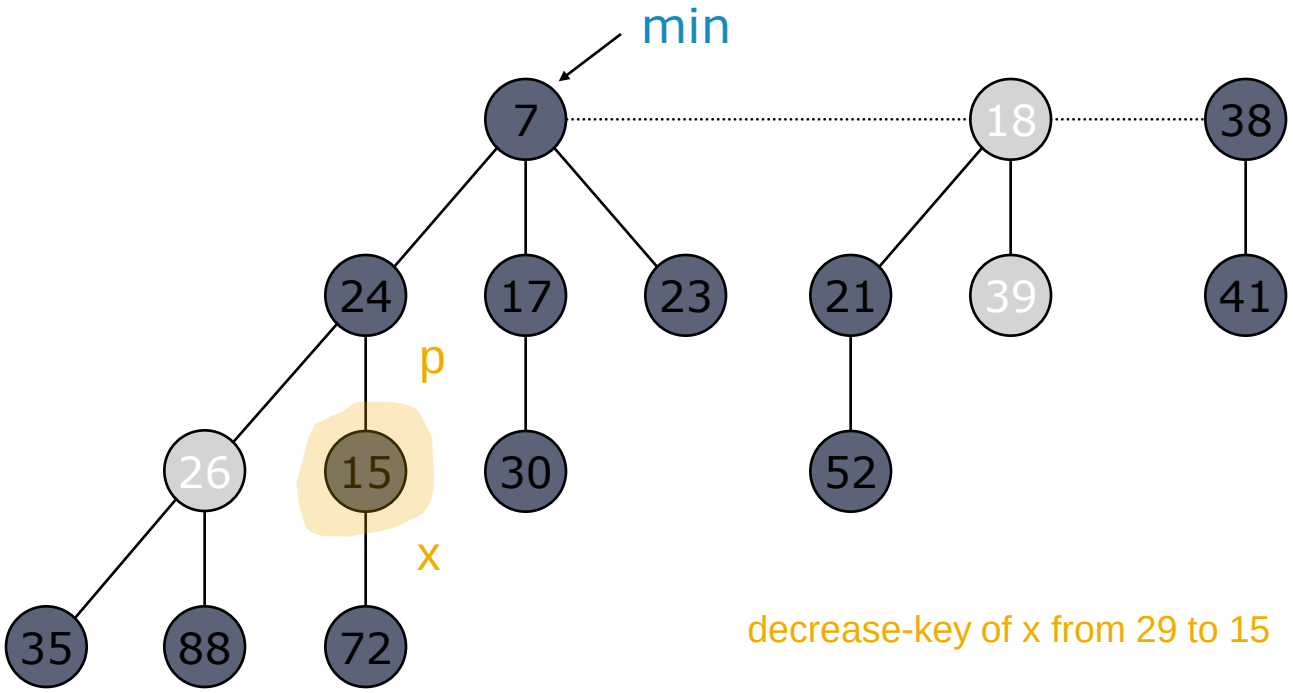
# Fibonacci Heaps: Decrease Key

- Case 2a. [heap order violated]
  - Decrease key of x.
  - Cut tree rooted at x, meld into root list, and unmark.
  - If parent p of x is unmarked (hasn't yet lost a child), mark it;
    Otherwise, cut p, meld into root list, and unmark
    (and do so recursively for all ancestors that lose a second child).



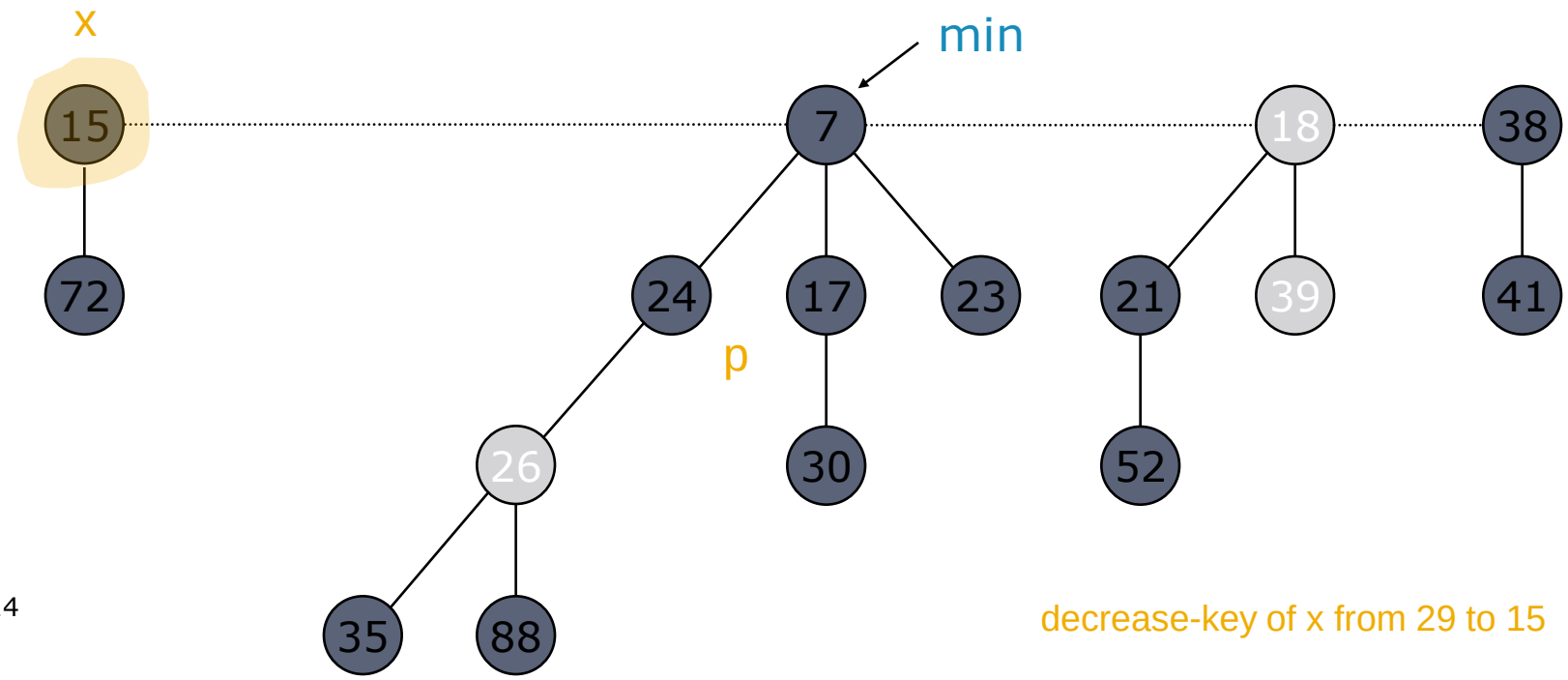
# Fibonacci Heaps: Decrease Key

- Case 2a. [heap order violated]
  - Decrease key of x.
  - Cut tree rooted at x, meld into root list, and unmark.
  - If parent p of x is unmarked (hasn't yet lost a child), mark it;
    Otherwise, cut p, meld into root list, and unmark
    (and do so recursively for all ancestors that lose a second child).



# Fibonacci Heaps: Decrease Key

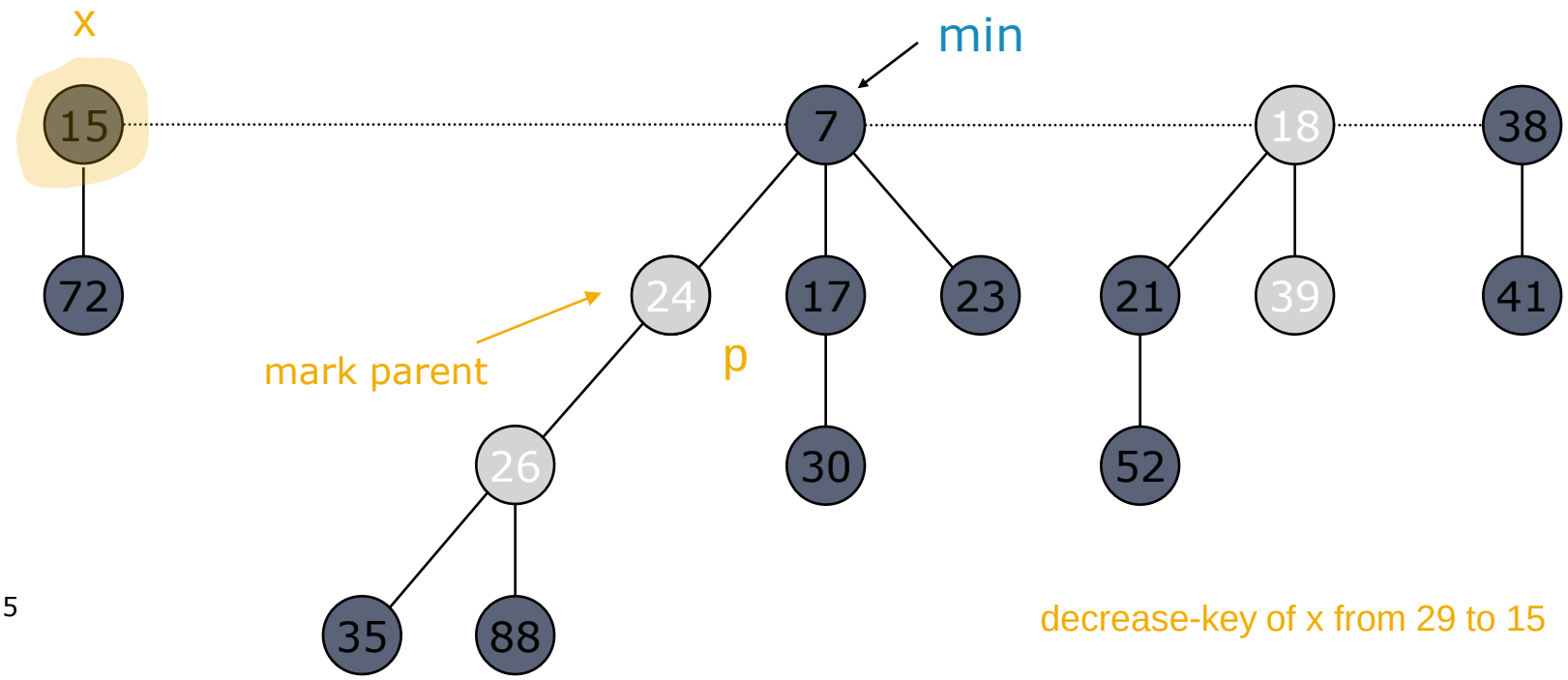
- Case 2a. [heap order violated]
  - Decrease key of x.
  - Cut tree rooted at x, meld into root list, and unmark.
  - If parent p of x is unmarked (hasn't yet lost a child), mark it;
    Otherwise, cut p, meld into root list, and unmark
    (and do so recursively for all ancestors that lose a second child).




decrease-key of x from 29 to 15

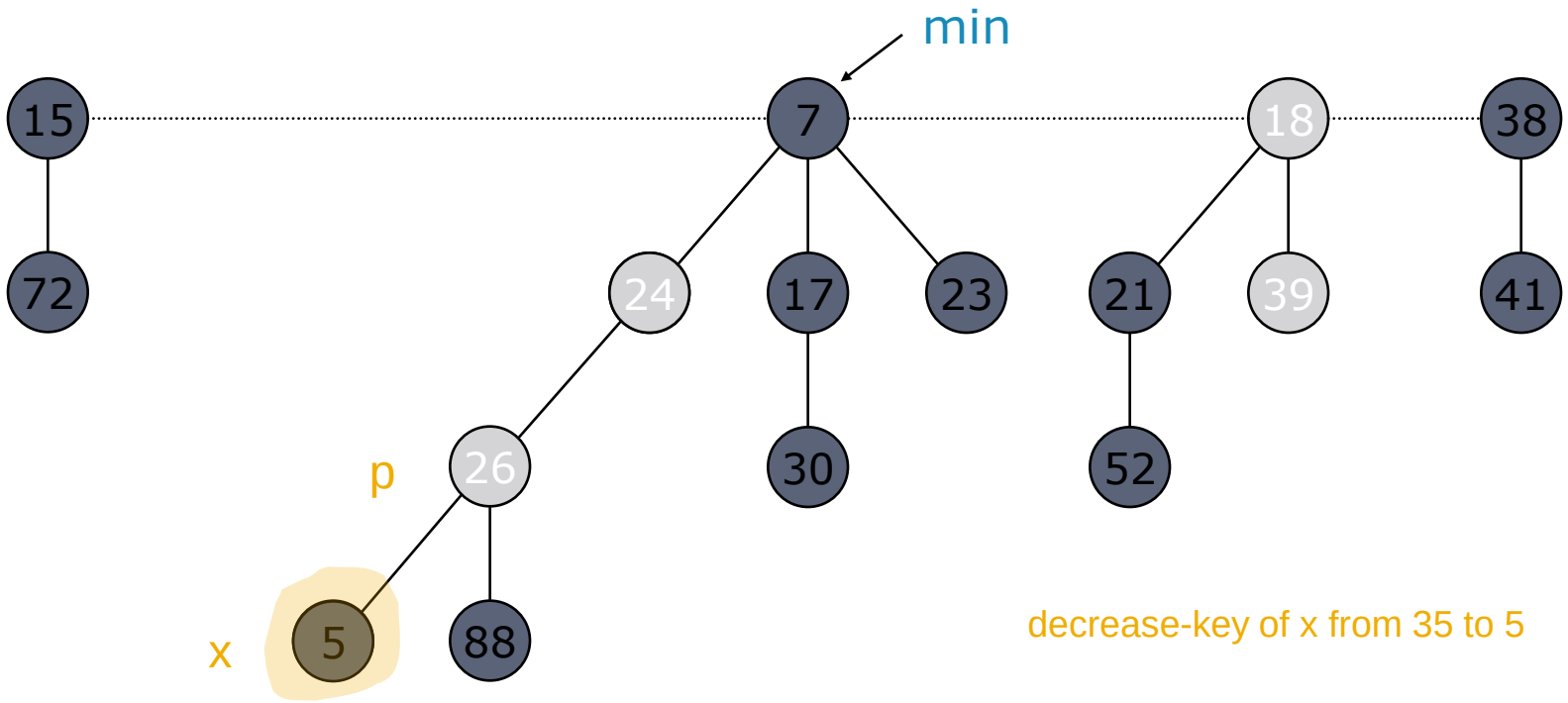
# Fibonacci Heaps: Decrease Key

- Case 2a. [heap order violated]
  - Decrease key of x.
  - Cut tree rooted at x, meld into root list, and unmark.
  - If parent p of x is unmarked (hasn't yet lost a child), mark it;
    Otherwise, cut p, meld into root list, and unmark
    (and do so recursively for all ancestors that lose a second child).



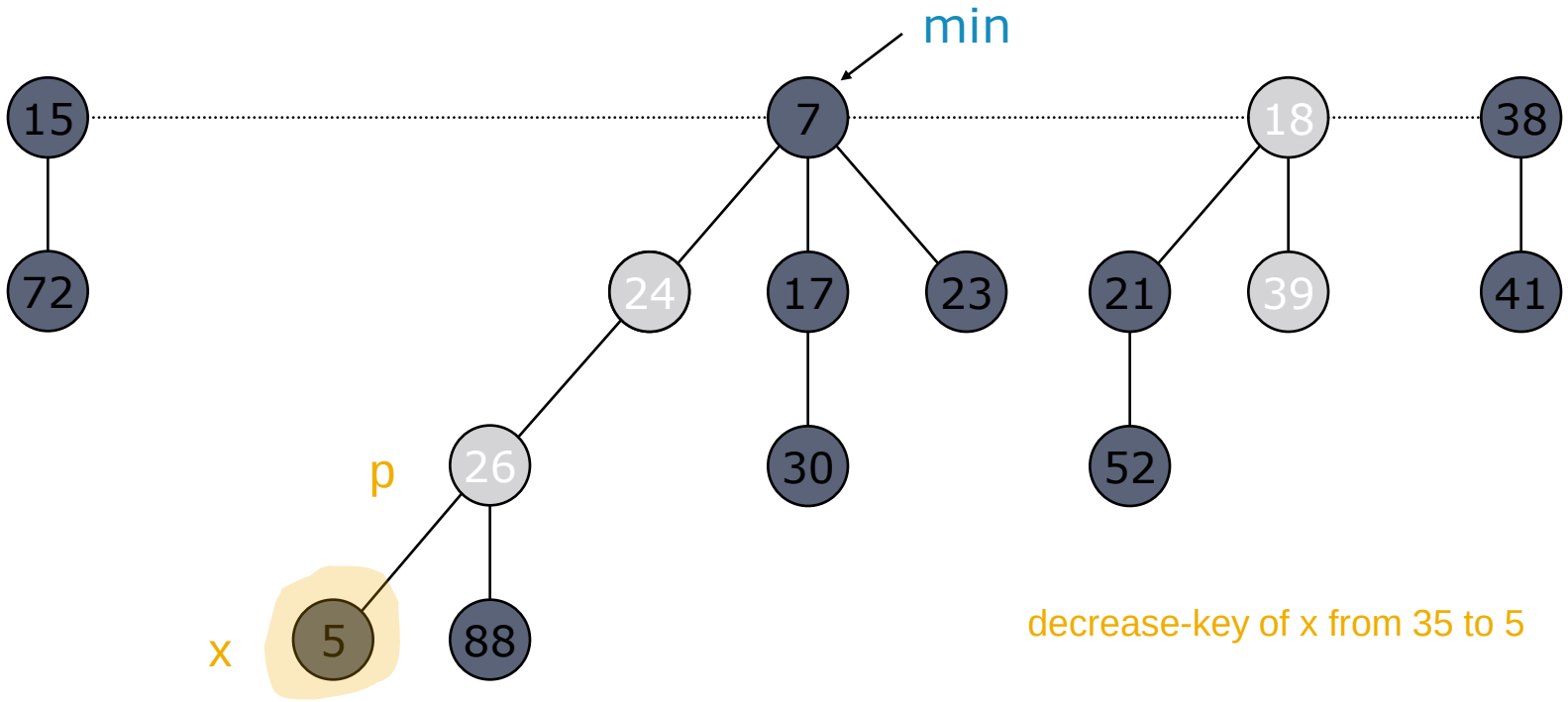
# Fibonacci Heaps: Decrease Key

- Case 2b. [heap order violated]
  - Decrease key of x.
  - Cut tree rooted at x, meld into root list, and unmark.
  - If parent p of x is unmarked (hasn't yet lost a child), mark it;
    Otherwise, cut p, meld into root list, and unmark
    (and do so recursively for all ancestors that lose a second child).



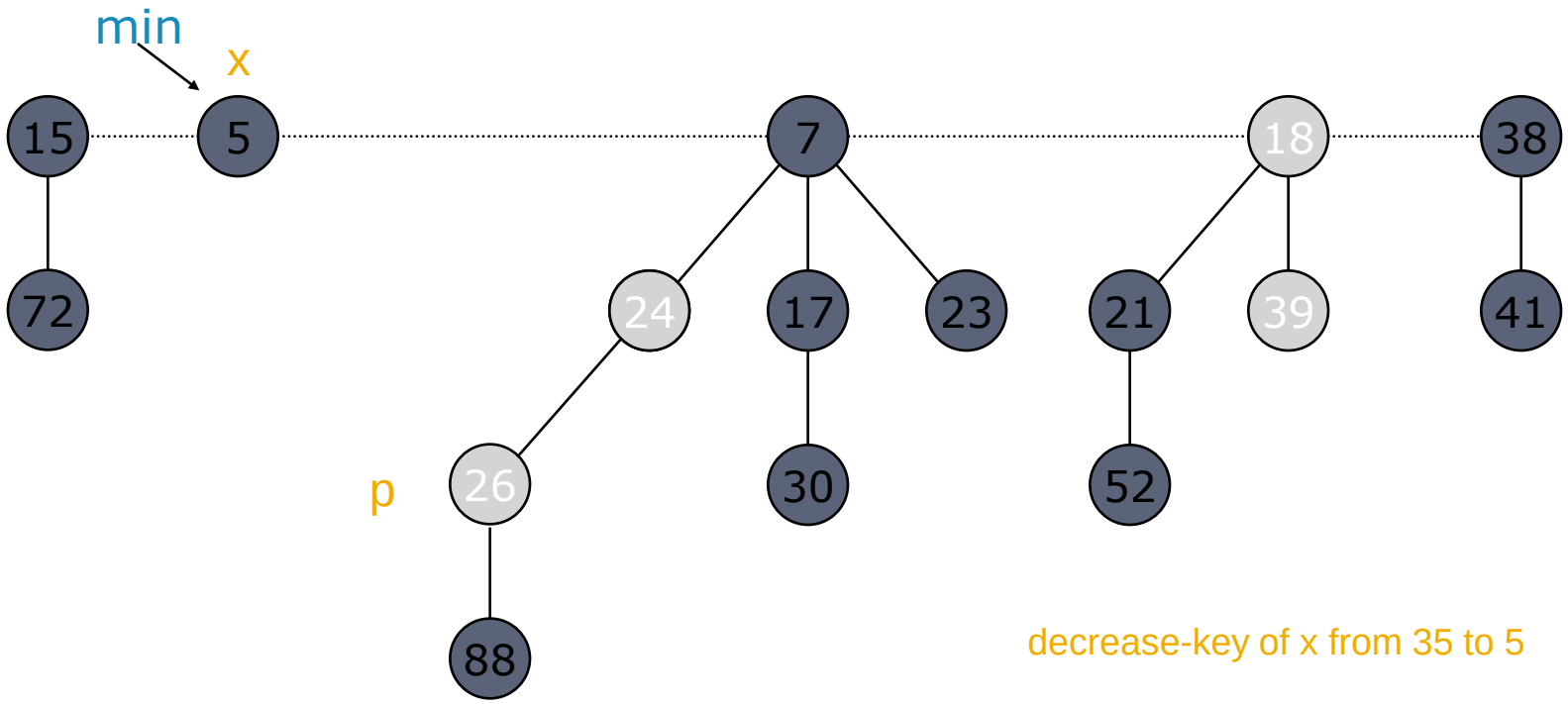
# Fibonacci Heaps: Decrease Key

- Case 2b. [heap order violated]
  - Decrease key of x.
  - Cut tree rooted at x, meld into root list, and unmark.
  - If parent p of x is unmarked (hasn't yet lost a child), mark it;
    Otherwise, cut p, meld into root list, and unmark
    (and do so recursively for all ancestors that lose a second child).



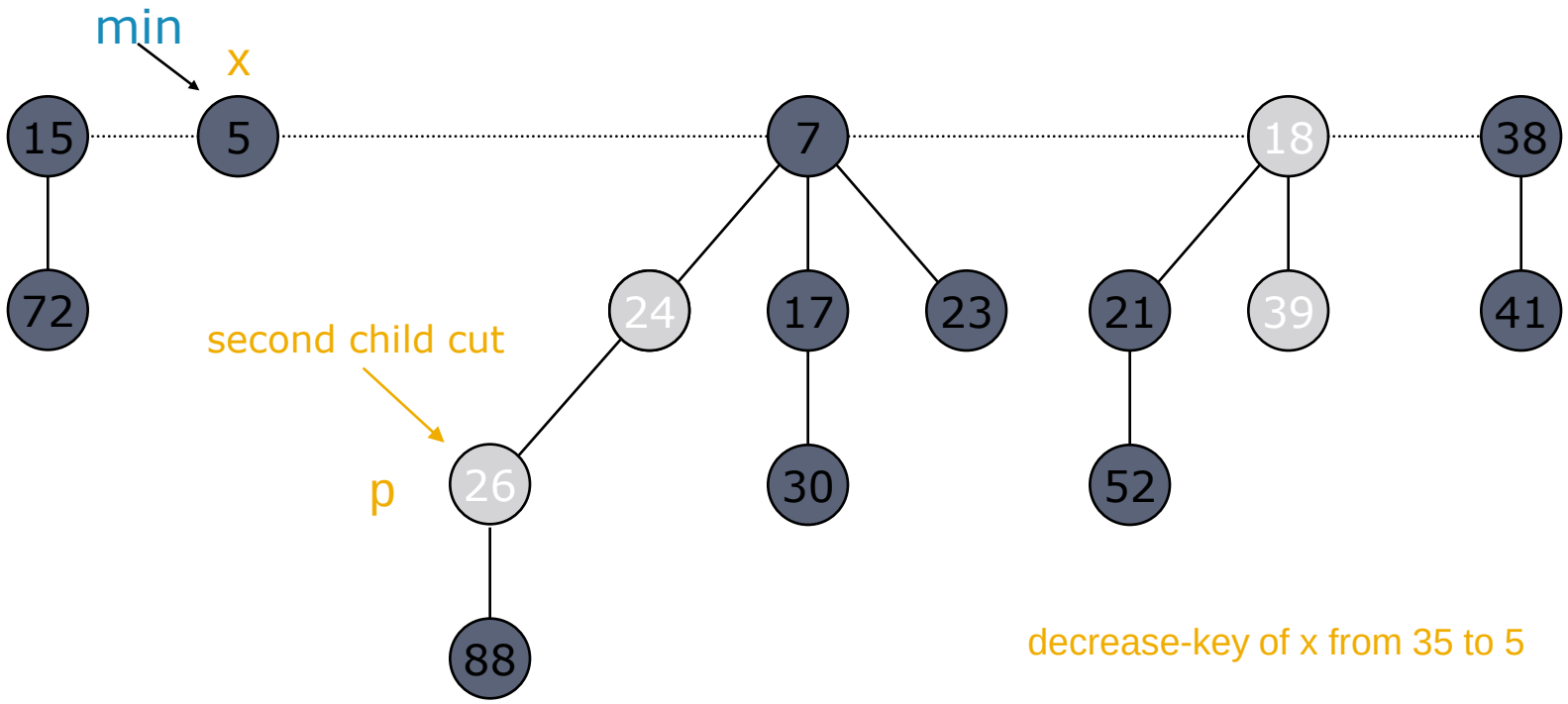
# Fibonacci Heaps: Decrease Key

- Case 2b. [heap order violated]
  - Decrease key of x.
  - Cut tree rooted at x, meld into root list, and unmark.
  - If parent p of x is unmarked (hasn't yet lost a child), mark it;
    Otherwise, cut p, meld into root list, and unmark
    (and do so recursively for all ancestors that lose a second child).



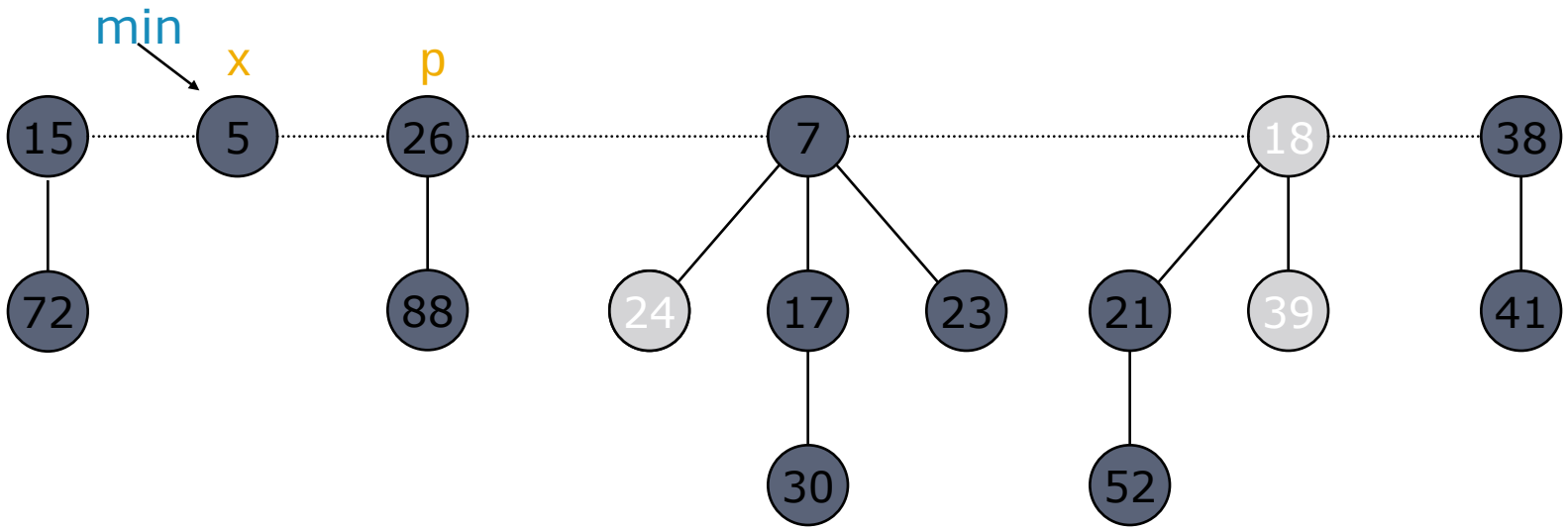
# Fibonacci Heaps: Decrease Key

- Case 2b. [heap order violated]
  - Decrease key of x.
  - Cut tree rooted at x, meld into root list, and unmark.
  - If parent p of x is unmarked (hasn't yet lost a child), mark it;
    Otherwise, cut p, meld into root list, and unmark
    (and do so recursively for all ancestors that lose a second child).



# Fibonacci Heaps: Decrease Key

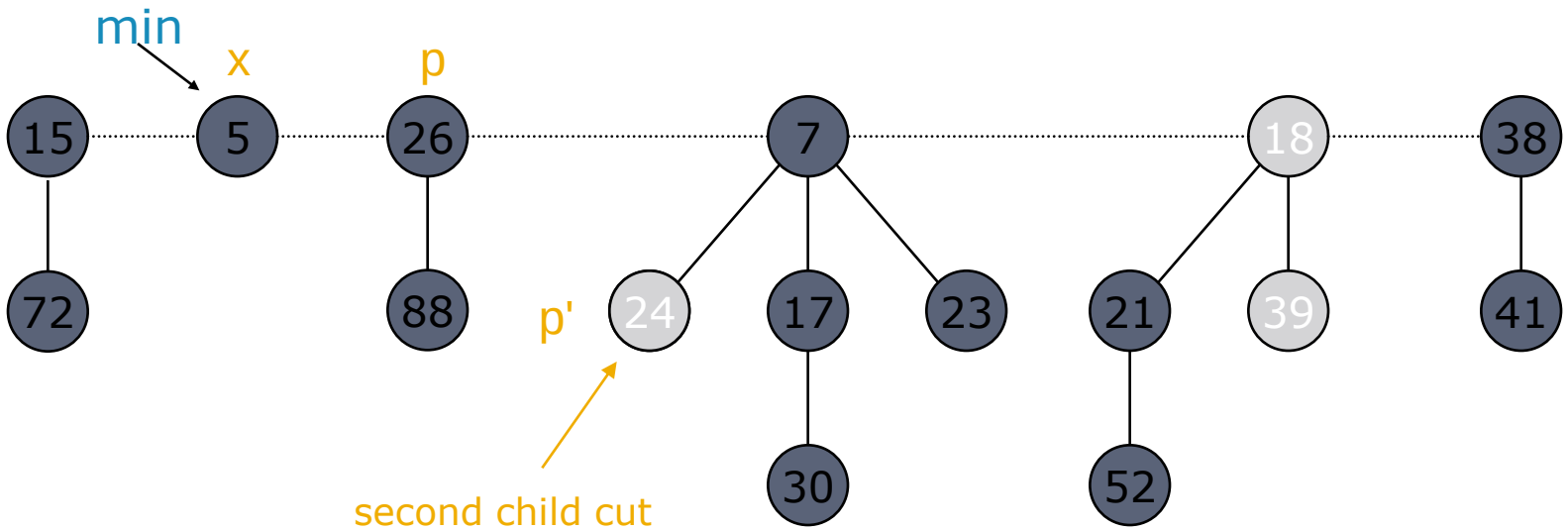
- Case 2b. [heap order violated]
  - Decrease key of x.
  - Cut tree rooted at x, meld into root list, and unmark.
  - If parent p of x is unmarked (hasn't yet lost a child), mark it;
    Otherwise, cut p, meld into root list, and unmark
    (and do so recursively for all ancestors that lose a second child).




decrease-key of x from 35 to 5

# Fibonacci Heaps: Decrease Key

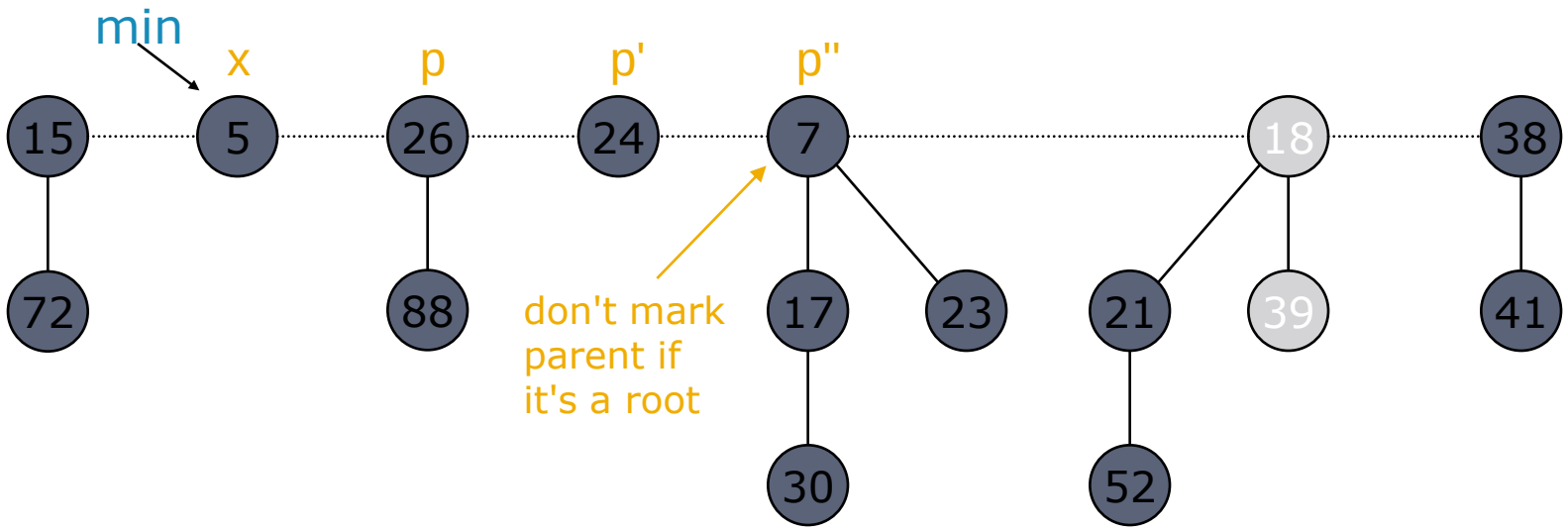
- Case 2b. [heap order violated]
  - Decrease key of x.
  - Cut tree rooted at x, meld into root list, and unmark.
  - If parent p of x is unmarked (hasn't yet lost a child), mark it;
    Otherwise, cut p, meld into root list, and unmark
    (and do so recursively for all ancestors that lose a second child).




decrease-key of x from 35 to 5

# Fibonacci Heaps: Decrease Key

- Case 2b. [heap order violated]
  - Decrease key of x.
  - Cut tree rooted at x, meld into root list, and unmark.
  - If parent p of x is unmarked (hasn't yet lost a child), mark it;
    Otherwise, cut p, meld into root list, and unmark
    (and do so recursively for all ancestors that lose a second child).

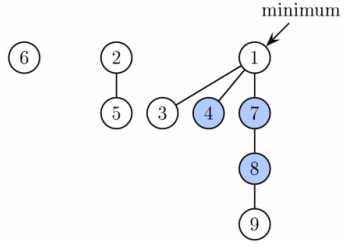



decrease-key of x from 35 to 5

# ערמות פיבונצ'י

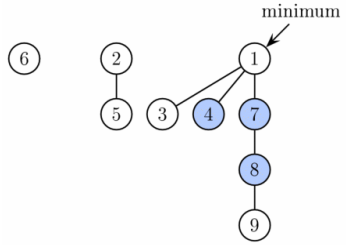
- כמו שראיתם בכיתה:

$$Rank(H) < \log_{\Phi}(n)$$



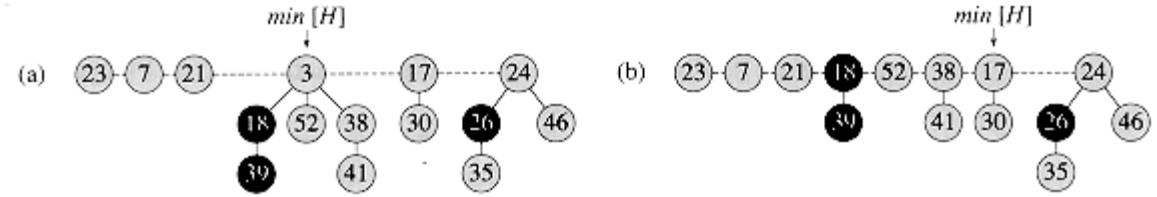

# שאלה משבוע שעבר

- בהינתן מימוש של Fibonacci heaps בו לא מתבצע cascading cuts, הראו שעבור סדרת m פעולות על מקס' n אברים, עלות פעולה ממוצעת גבוהה ככל האפשר
- קצת תזכורת
  - פעולות decrease-key מאד מהירות (זמן קבוע)
  - בעת פעולת extract-min נבצע consolidate
  - דרגת כל צומת הינה D(n) והיא חסומה ע"י O(logn)

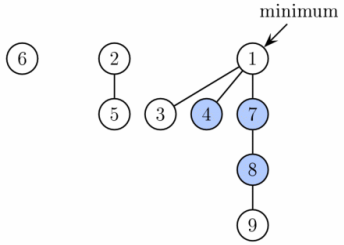



# שאלה משבוע שעבר

- בהינתן שאין cascading cuts, מה משתנה ב"מה שמובטח לנו"? ז.א איפה השתמשנו בידע שלא ניתן להוריד 2 בנים מצומת בלי שהוא ייחתך?
  - בהוכחה שהדרגה המקסימלית O(logn) (חסם על D(n))
- איפה משתמשים בעובדה זו?
  - ב-extract min נעבור על כל השורשים (מקס' D(n))

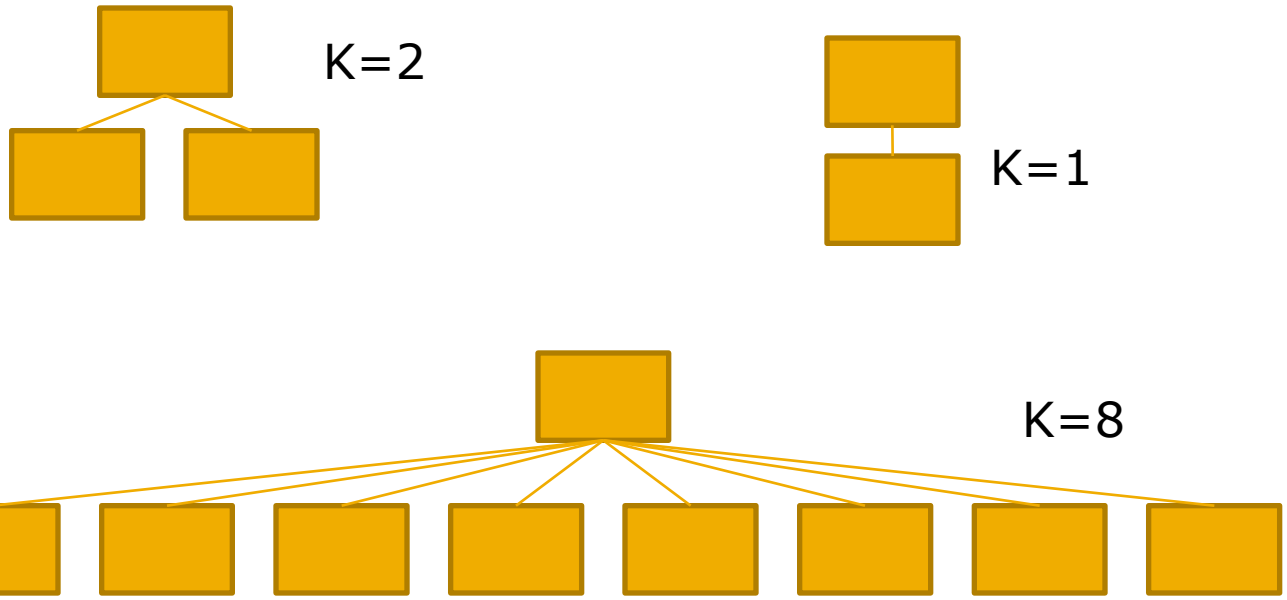


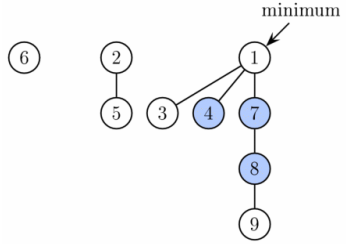

- ז"א אם נגיע למצב שבו יש הרבה מאד עצים בערמה, כל פעולה תהיה יקרה
- מעבר לכך, אם כל עץ מדרגה שונה, פעולת ה-consolidate לא תחבר עצים ביחד




# שאלה משבוע שעבר

- כיצד נגדיר עץ מדרגה k שהוא "ממש גרוע"?
  - נגדיר עץ מיוחד מדרגה k ← k-rank star. זהו שורש עם k בנים, כולם עלים.

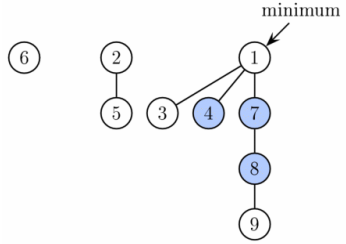





# שאלה משבוע שעבר

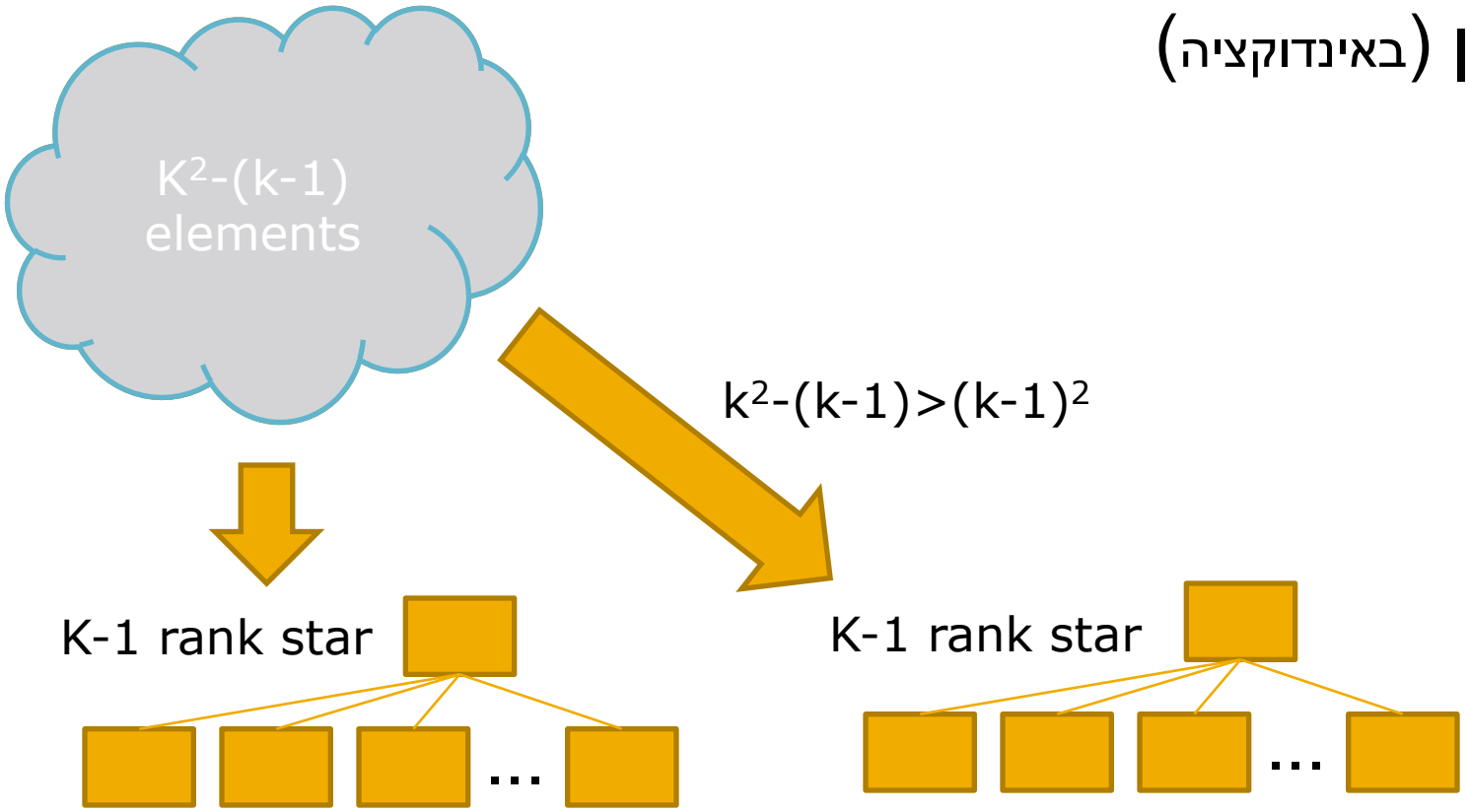
- הפתרון יורכב משני שלבים

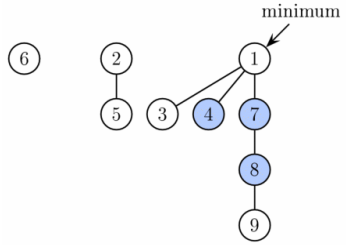
1. בסדרה של f(n) שלבים, בנו ערמת פיבונצ'י כך שיש
   - 1 rank-0 star
   - 1 rank-1 star
   - ...
   - 1 rank-sqrt(n) star
2. חזרו על הפעולות הבאות מספר רב של פעמים (O(m))
   - הכנס ערך X ממש קטן
   - בצע extract-min (מוחק את X)
   - **כל אחת מהפעולות לוקחת $\Omega(\sqrt{n})$ זמן, כך שעבור $m>>n$ פעולה תיקח בממוצע $\sqrt{n}$**




# שאלה משבוע שעבר

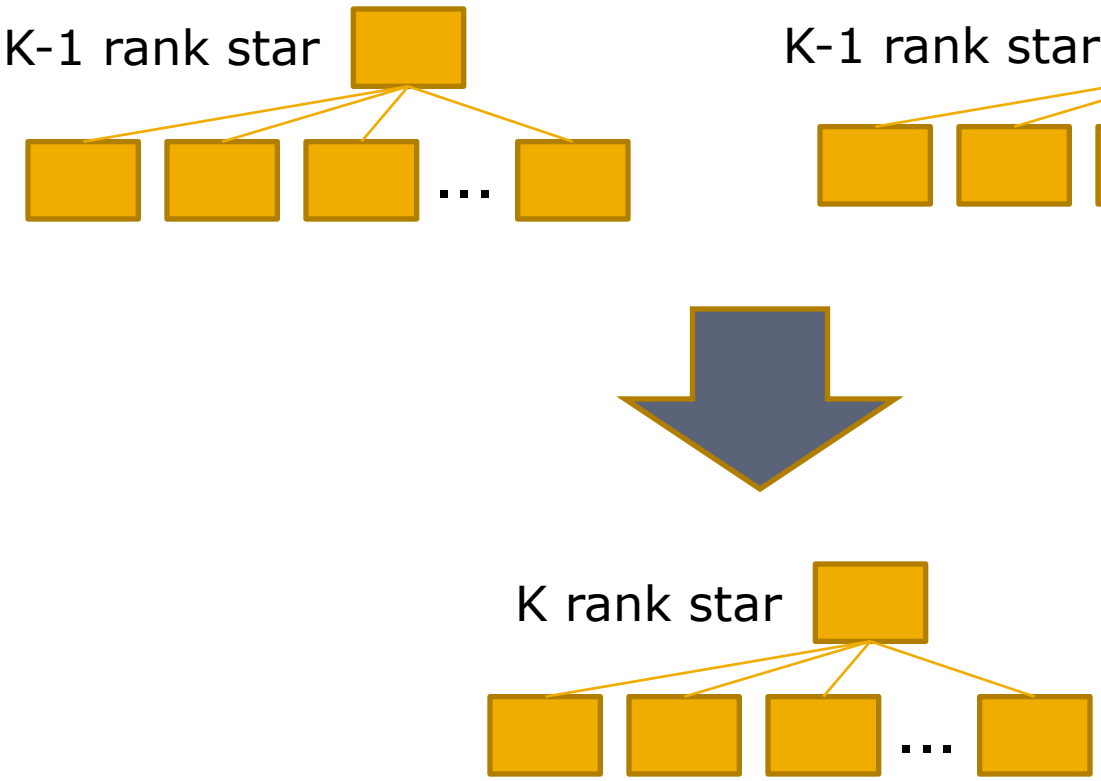
- אך כיצד בונים k-rank star? כיצד נוודא שבשום שלב לא יהיו יותר מ-n אברים בערמה?
- פתרון (באינדוקציה)






# שאלה משבוע שעבר

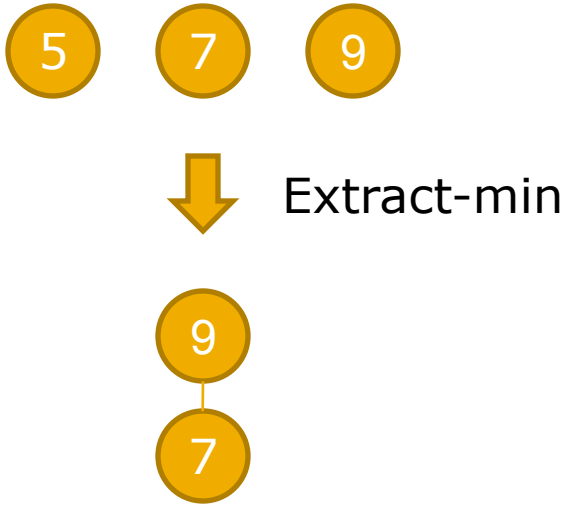
- אך כיצד בונים k-rank star? כיצד נוודא שבשום שלב לא יהיו יותר מ-n אברים בערמה?
- פתרון



# תרגיל 2 – ערמות פיבונאצ'י

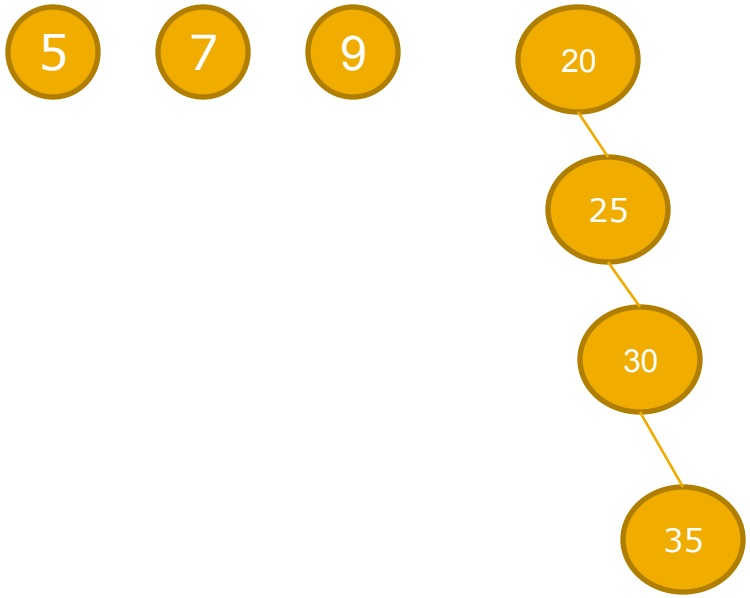
- האם ניתן לבנות ערמת פיבונאצ'י ובה עץ אחד מעומק n?
- פתרון
  - עבור n=1

  - עבור n=2



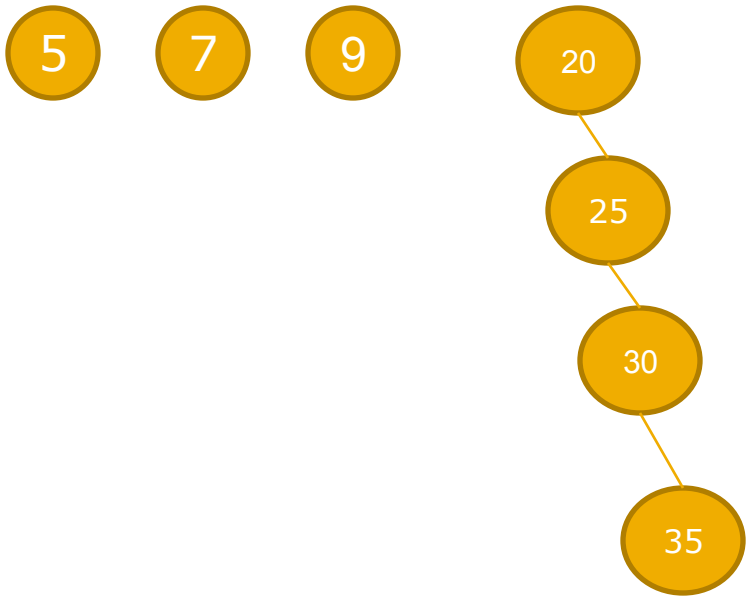
# תרגיל 2 – ערמות פיבונאצ'י

- האם ניתן לבנות ערמת פיבונאצ'י ובה עץ אחד מעומק n?
- פתרון
  - נניח שאנו יודעים לפתור עבור n=k
  - נפתור עבור n=k+1
    - נגדיר x',y',z' כך ש- $key[z'] < key[y'] < key[x'] < \min[H]$
    - נכניס אותם לתוך הערמה



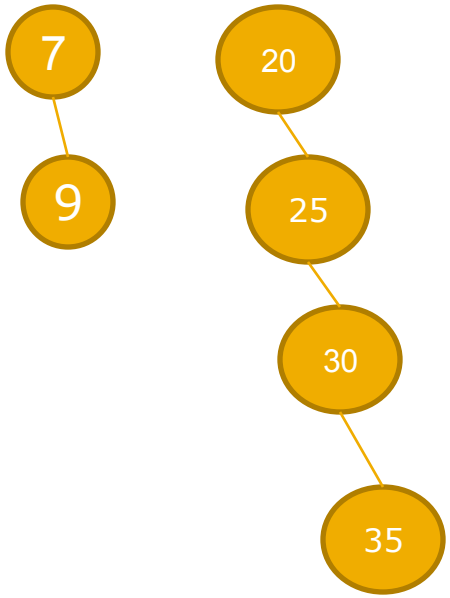
# תרגיל 2 – ערמות פיבונאצ'י

- האם ניתן לבנות ערמת פיבונאצ'י ובה עץ אחד מעומק n?
- פתרון
  - נניח שאנו יודעים לפתור עבור n=k
  - נפתור עבור n=k+1
    - נגדיר x',y',z' כך ש- $key[z'] < key[y'] < key[x'] < \min[H]$
    - נכניס אותם לתוך הערמה
    - נפעיל extract-min



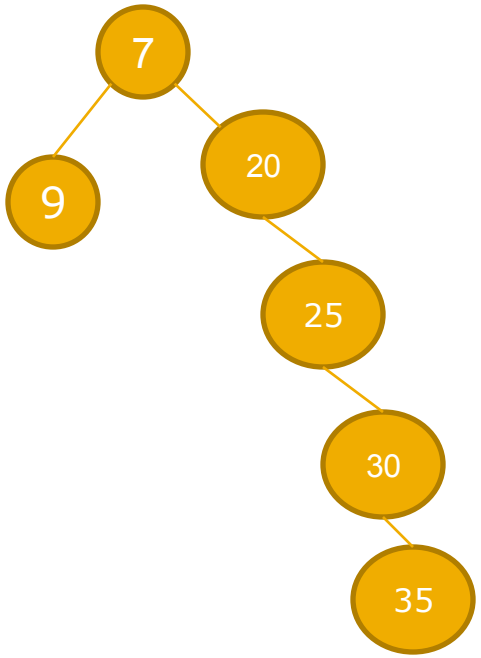
# תרגיל 2 – ערמות פיבונאצ'י

- האם ניתן לבנות ערמת פיבונאצ'י ובה עץ אחד מעומק n?
- פתרון
  - נניח שאנו יודעים לפתור עבור n=k
  - נפתור עבור n=k+1
    - נגדיר x',y',z' כך ש- $key[z'] < key[y'] < key[x'] < \min[H]$
    - נכניס אותם לתוך הערמה
    - נפעיל extract-min



# תרגיל 2 – ערמות פיבונאצ'י

- האם ניתן לבנות ערמת פיבונאצ'י ובה עץ אחד מעומק n?
- פתרון
  - נניח שאנו יודעים לפתור עבור n=k
  - נפתור עבור n=k+1
    - נגדיר x',y',z' כך ש- $key[z'] < key[y'] < key[x'] < \min[H]$
    - נכניס אותם לתוך הערמה
    - נפעיל extract-min
    - נמחק את x' (9)



# תרגיל 3

- בערמות פיבונצ'י, אנחנו מבצעים cascading cuts בצומת v אם הוא איבד צומת בן מאז הפעם האחרונה שהוא נתלה על צומת אחרת. נניח שנבצע CC רק אם v איבד שני בנים מאז, כיצד משתנה הלמה:
  - x צומת בערמת פיב', $y_1,\ldots,y_n$ בנים של x, מסודרים לפי הסדר בו נתלו על x (הכי ישן ועד הכי חדש). אזי $rank(y_i) \geq i-2$ לכל i.

# תרגיל 3

- Answer
  - rank($y_i$)≥i-3
  - Since $y_i$ had the same rank as x when it became a child of x
  - x must have had at least i-1 children at that time, so $y_i$ had at least i-1 rank.
  - It could have lost at most two children since then, therefore rank at least i-3

# הסוף