מבני נתונים ומבוא לאלגוריתמים
מפגש הנחיה מס' 4

מדעי המחשב, קורס מס' 20407

סמסטר 2016ב

מנחה: ג'ון מרברג

האוניברסיטה הפתוחה

מה ראינו עד כה?

- הרבה על אלגוריתמים:
  - תכנון, שיטות
  - ניתוח, הוכחת נכונות
  - השוואת זמני ריצה
  - פתרון נוסחאות נסיגה
  - אלגוריתמים קלאסיים (חיפוש בינרי, מיון מיזוג, ועוד)

מפגש רביעי

- נושאי השיעור
  - פרק 6 בספר – מיון-ערמה
    - הגדרת הערמה
    - מיון-ערמה
    - תור קדימויות
    - תרגילים

מבוסס על מצגת של ברוך חייקין ואיציק בייז

מיון ערימה - הרעיון

- נניח שנוכל לארגן את איברי הקלט בדרך שמאפשרת למצוא את מקום האיבר המקסימלי במהירות.
- כיצד נמיין ?
  - נמצא את המקס' ונחליף אותו עם האיבר שנמצא בסוף המערך.
  - נארגן מחדש את האיברים שנותרו (ללא האחרון)
  - נמצא את המקס' מבין האיברים שנותרו ונחליף אותו עם האיבר שנמצא במקום הלפני אחרון.
  - נמשיך כך עד לסידור כל האיברים במקומם .
- פתרון: ארגון הקלט יהיה בתוך מבנה נתונים שנקרא ערימה

האוניברסיטה הפתוחה

עץ בינארי כמעט שלם - הגדרה

- עץ בינארי כמעט שלם הוא עץ המקיים:
  - לכל צומת לכל היותר שני בנים
  - כל הרמות בעץ מלאות, מלבד אולי הרמה התחתונה (העמוקה ביותר)
  - ברמה התחתונה, העלים נמצאים ברצף משמאל לימין (ללא "חורים") עד לנקודה מסוימת
- תכונות:
  - **עומק** של צומת בעץ מוגדר כמרחק בין השורש לצומת (עומק השורש 0)
  - אוסף כל הצמתים שעומקם k נקראים **רמה** k של העץ.

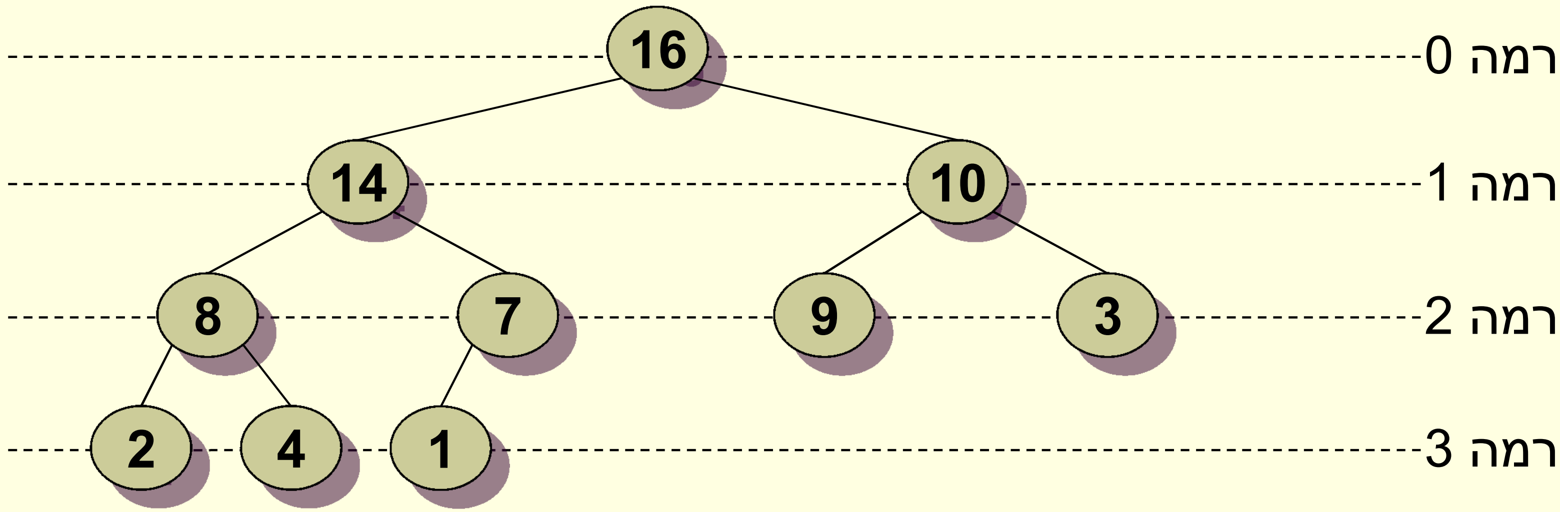
    לתשומת לב: בעץ בינארי כמט מלא, העלים נמצאים ברמה התחתונה, וייתכן וגם ברמה שמעליה
  - **גובה** של צומת בעץ מוגדר כמרחק הארוך ביותר, כלפי מטה, בין הצומת לבין עלה (כל העלים בגובה 0)
  - גובה העץ מוגדר כגובה השורש

ערמה – הגדרה

הגדרת ערמה (heap)

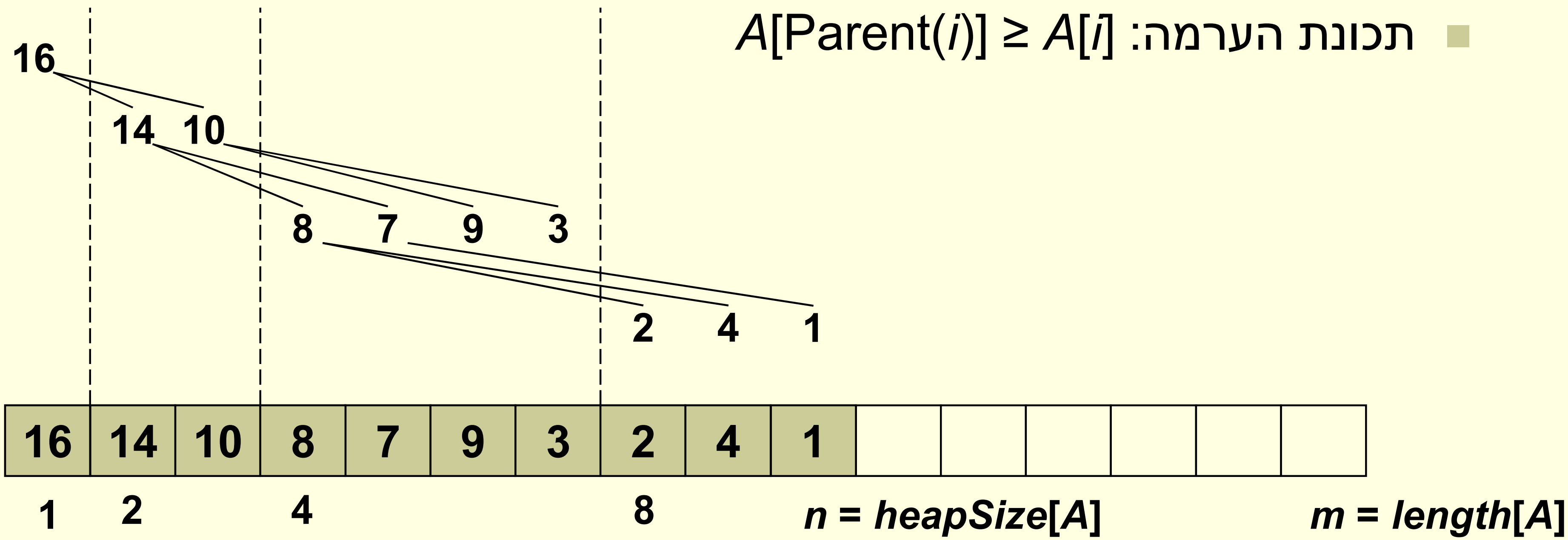
- עץ בינארי כמעט שלם
- מקיים את **תכונת הערמה:**

בכל צומת x פרט לשורש: $key[parent[x]] \geq key[x]$



ייצוג עץ בינארי כמעט שלם ע"י מערך

- **שיכון עץ כמעט שלם בגודל n צמתים במערך $A[1..m]$**
  - $n \le m$ = גודל המערך
  - שורש העץ נמצא ב-$A[1]$
  - הצומת x מיוצג ע"י האיבר ה-i במערך: $key[x] = A[i]$
  - חישוב מיקום צומת האב: $\mathrm{Parent}(i) = \lfloor i/2 \rfloor$
  - חישוב מיקום צומתי הבנים: $\mathrm{Left}(i) = 2i$, $\mathrm{Right}(i) = 2i+1$
  - תכונת הערמה: $A[\mathrm{Parent}(i)] \ge A[i]$



ערמה – תכונות

1. (תרגיל 6.1-1) מהו מספר האיברים בערמה שגובהה h?

$2^h \le n \le 2^{h+1}-1$

$n = \Theta(2^h)$

2. (תרגיל 6.1-2) מהו הגובה של ערמה בת n איברים?

$2^h \le n \le 2^{h+1}-1$

$2^h \le n < 2^{h+1}$

$h = \lg(2^h) \le \lg n < \lg(2^{h+1}) = h+1$

$h = \lfloor \lg n \rfloor$

3. (תרגיל 6.1-3) מה מאפיין את האיבר בשורש הערמה?
   - זהו הערך המקסימאלי בערמה

4. (תרגיל 6.1-7) מהו מספר העלים בערימה בת n איברים?
   - בערמה בת n איברים יש $\lceil n/2 \rceil$ עלים

ערמה – תכונות (המשך)

5. (תרגיל 6.1-4) באיזה מיקום נמצא הערך הקטן ביותר בערמה שכל איבריה שונים זה מזה?
   - הערך נמצא באחד העלים, כלומר בתת-המערך $A[\lfloor n/2 \rfloor + 1..n]$

6. (תרגיל 6.3-3) יהי h גובה של ערמה בת n איברים.
   מהו המספר המקסימאלי n_k של איברים שגובהם k, לכל $0 \le k \le h$?
   - ההוכחה באינדוקציה $n_k \le \left\lceil \frac{n}{2^{k+1}} \right\rceil$

סידור הערמה - Heapify

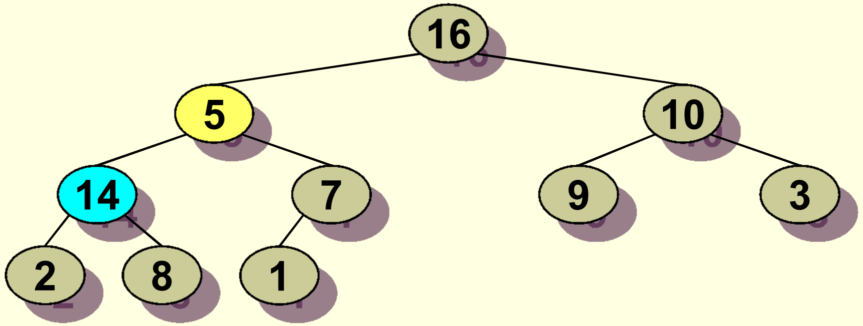
- השגרה Max-Heapify
  - *קלט:* מערך A ואינדקס i, כאשר העצים המושרשים ב-Left(i) וב-Right(i) הם ערמות, אך יתכן ש-$A[i]$ מפר את תכונת הערמה
  - *פלט:* מערך A שבו העץ ששורשו i הוא ערמה חוקית
  - *האלגוריתם:*

Max-Heapify(A, i)

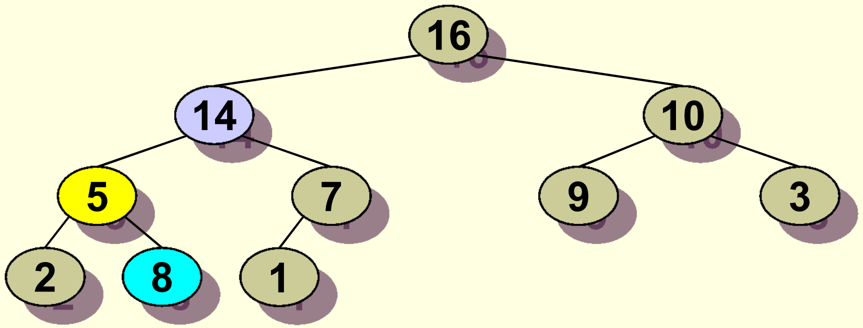
1. $l \leftarrow \text{Left}(i)$, $r \leftarrow \text{Right}(i)$, $n \leftarrow heapSize[A]$
2. $largest \leftarrow i$
3. **if** $l \le n$ **and** $A[l] > A[i]$ **then** $largest \leftarrow l$
4. **if** $r \le n$ **and** $A[r] > A[largest]$ **then** $largest \leftarrow r$
5. **if** $largest \ne i$
6.   **then** exchange $A[i] \leftrightarrow A[largest]$
7.   Max-Heapify(A, $largest$)

האוניברסיטה הפתוחה

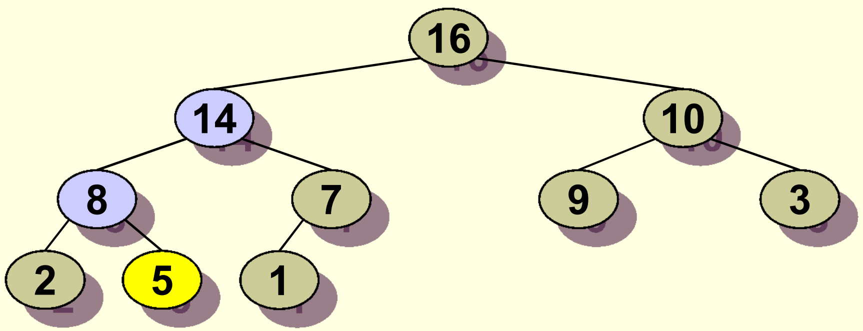
סידור הערמה – דוגמה



קריאה: Max-Heapify(A, 2)
A[2]=5 קטן מהבן השמאלי A[4]=14



קריאה: Max-Heapify(A, 4)
A[4]=5 קטן מהבן הימני A[9]=8



קריאה: Max-Heapify(A, 9)
A[9]=5 הוא עלה, והשגרה עוצרת

סידור הערמה – ניתוח

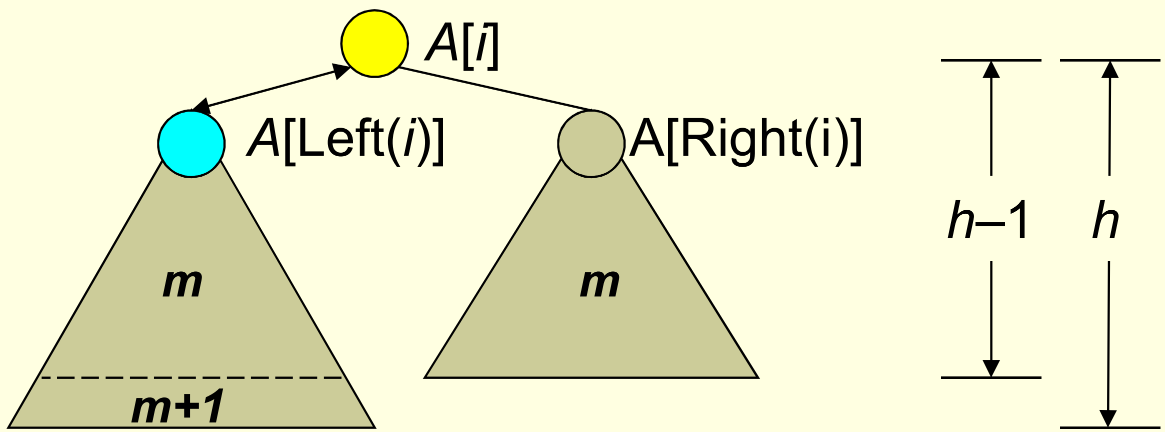
- זמן הריצה של Max-Heapify על עץ בגודל *n* ששורשו *i*
- במקרה הגרוע:
  - בתת-עץ השמאלי יש רמה אחת יותר מאשר בתת-עץ הימני וכולה מלאה

    $n_{left} = n-m-1 = n-(n-2)/3-1 < 2n/3 \quad \leftarrow \quad m=(n-2)/3 \quad \leftarrow \quad n=3m+2$

    מסקנה: בתת העץ השמאלי יש לכל היותר $2n/3$ איברים
  - המפתח של הבן השמאלי הוא הגדול ביותר ← מתבצעת החלפה וקריאה רקורסיבית Max-Heapify(*A*, Left(*i*))
  - נוסחת הנסיגה:

$$T(n) \le T(2n/3) + \Theta(1)$$

$$T(n) = \Theta(\lg n) = \Theta(h)$$



בניית ערמה

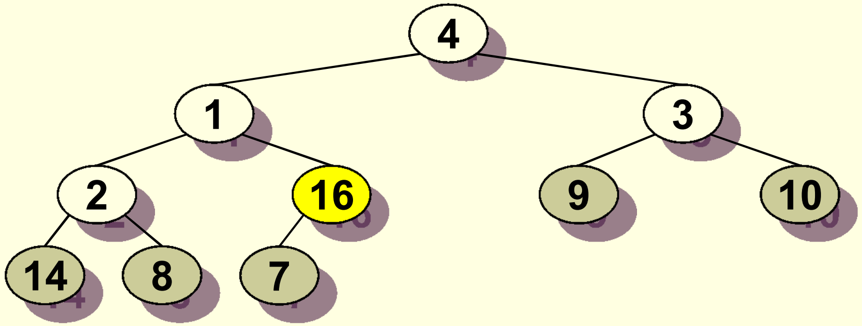
- אלגוריתם נאיבי: מיון של המערך בסדר יורד
- פתרון יעיל יותר: השגרה Build-Max-Heap
  - *קלט:* מערך $A[1..n]$
  - *פלט:* אותם איברים מסודרים כערמה בתוך המערך A
  - *הרעיון:* נהפוך את כל התת-עצים (הלא-טריוויאליים) לערמות באמצעות Max-Heapify, מלמטה למעלה (bottom-up)
  - *האלגוריתם:*

Build-Max-Heap(A)

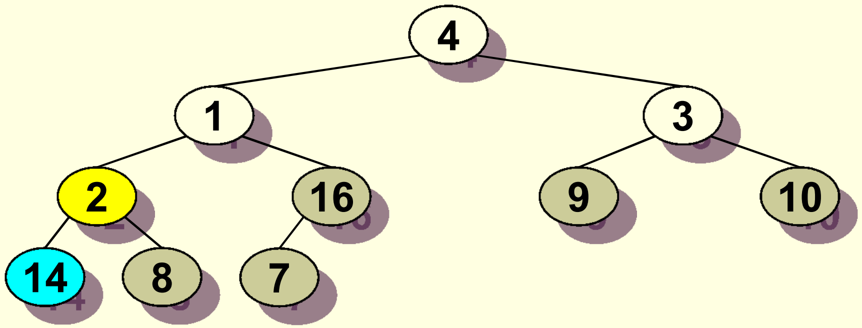
1. $heapSize[A] \leftarrow length[A]$
2. **for** $i \leftarrow \lfloor heapSize[A]/2 \rfloor$ **downto** 1
3. **do** Max-Heapify(A, i)

האוניברסיטה הפתוחה

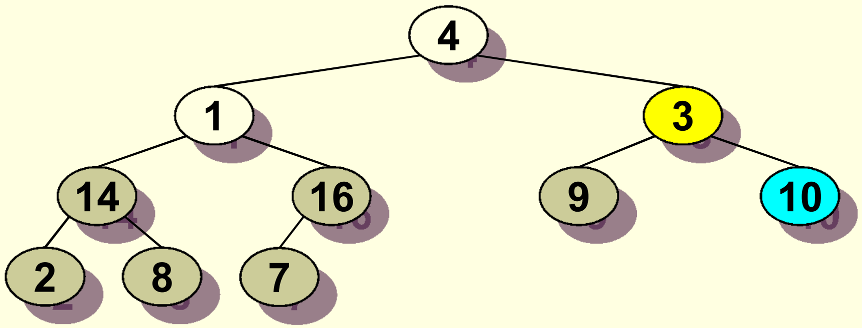
בניית ערמה – דוגמה



4	1	3	2	16	9	10	14	8	7



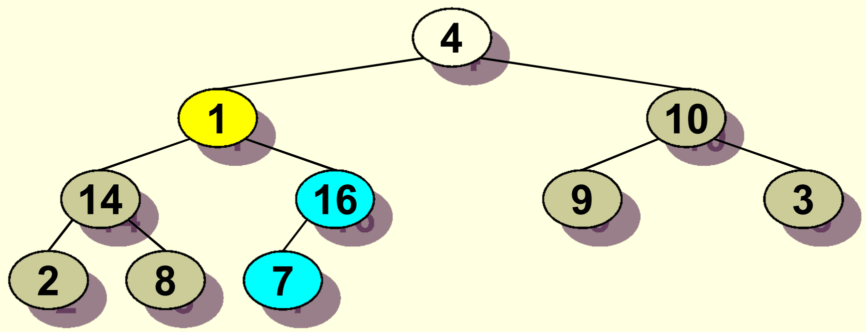
4	1	3	2	16	9	10	14	8	7



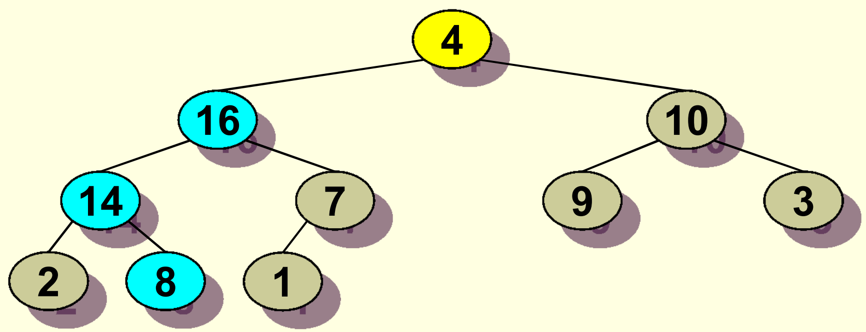
4	1	3	14	16	9	10	2	8	7

האוניברסיטה הפתוחה

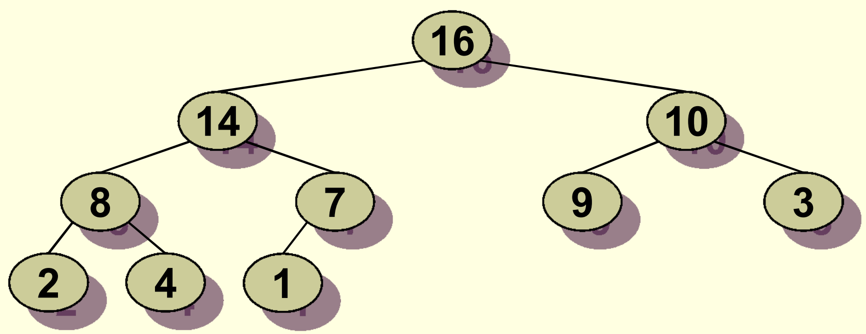
בניית ערמה – דוגמה (המשך)



4	1	10	14	16	9	3	2	8	7



4	16	10	14	7	9	3	2	8	1



16	14	10	8	7	9	3	2	4	1

בניית ערימה – ניתוח

- זמן הריצה של Build-Max-Heap על מערך בגודל n
  - גובה ערמה בת n איברים הוא $h = \lfloor \lg n \rfloor$ (לפי תכונה 2 של ערמה)
  - מספר הצמתים שגובהם k הוא לכל היותר $\lceil n/2^{k+1} \rceil$ (לפי תכונה 6)
    (כאשר גובה הוא המרחק הגדול ביותר מעלה, והעלה הוא בגובה k=0)
  - זמן הריצה של Max-Heapify על צומת בגובה k הוא $O(k)$
  - לכן זמן הריצה הכולל הוא

$$
\begin{aligned}
T(n) &= \Sigma_{k=0}{}^{h}(\lceil n/2^{k+1} \rceil O(k)) \\
&= O(n\Sigma_{k=0}{}^{h}\, k/2^{h}) \\
&= O(n\Sigma_{k=0}{}^{\infty}\, k/2^{k}) \\
&= O(n)
\end{aligned}
$$

  - בחישוב זה נעשה שימוש בשוויון $\Sigma_{k=0}{}^{\infty}\, k/2^{k} = 2$
    (ראה נוסחה א.8 בנספח א' בספר)

בניית ערמה - תרגול

- (תרגיל 6.3-2) מדוע בניית הערמה נעשית מלמטה למעלה?
  - כי השגרה Max-Heapify מניחה ששני תתי-העצים של הבנים של הצומת שהיא מקבלת הם ערמות חוקיות.
- מהו זמן הריצה של Build-Max-Heap על מערך הממוין בסדר הפוך?
  - בכל אחת מ- n/2 האיטרציות, השגרה Max-Heapify תסתיים בקריאה הראשונה, לכן זמן הריצה הוא עדיין $\Theta(n)$, אבל הקבוע החבוי בזמן הריצה הוא קטן יותר.

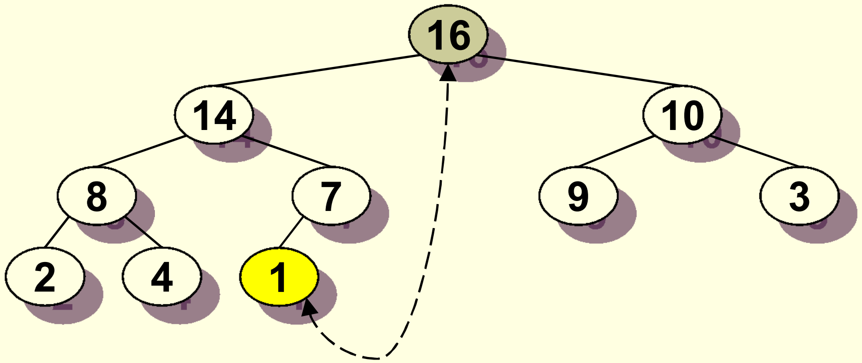
מיון-ערמה

- מיון-ערמה (HeapSort)
  - משתמש במבנה נתונים (ערמה)
    - **זמן ריצה**: רץ במקרה הגרוע בזמן $\Theta(n \lg n)$ (זמן אופטימלי)
    - **זיכרון**: ממיין במקום! (בניגוד למיון-מיזוג)
  - *האלגוריתם:*

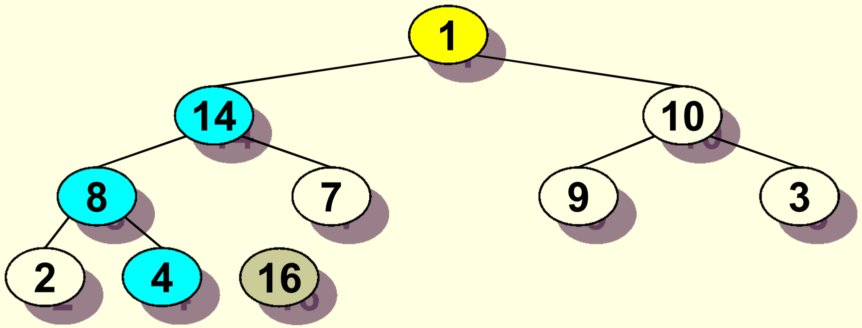
HeapSort(A)
1. Build-Max-Heap(A)
2. **for** $i \leftarrow length[A]$ **downto** 2
3.   **do** exchange $A[1] \leftrightarrow A[i]$
4. $heapSize[A] \leftarrow heapSize[A] - 1$
5.   Max-Heapify(A, 1)

האוניברסיטה הפתוחה

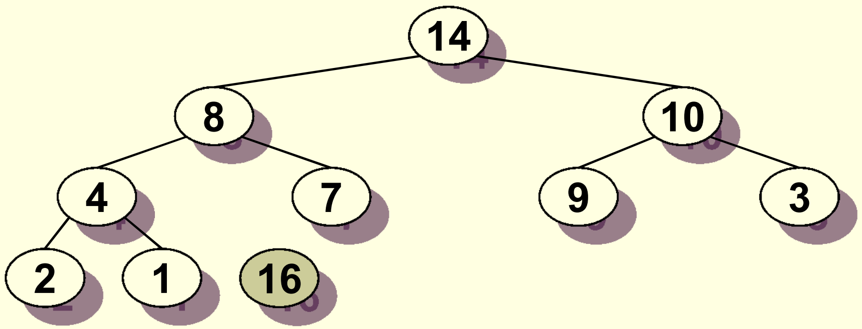
מיון-ערמה – דוגמה



16	14	10	8	7	9	3	2	4	1

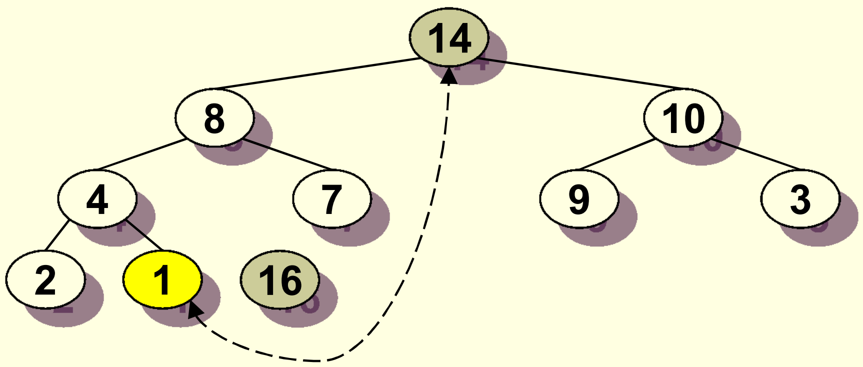


1	14	10	8	7	9	3	2	4		16

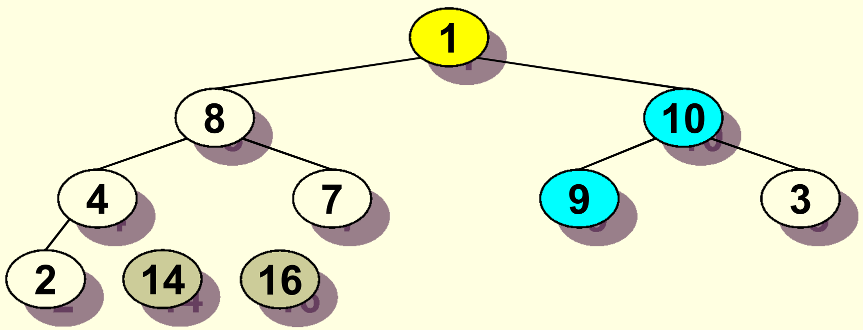


14	8	10	4	7	9	3	2	1		16

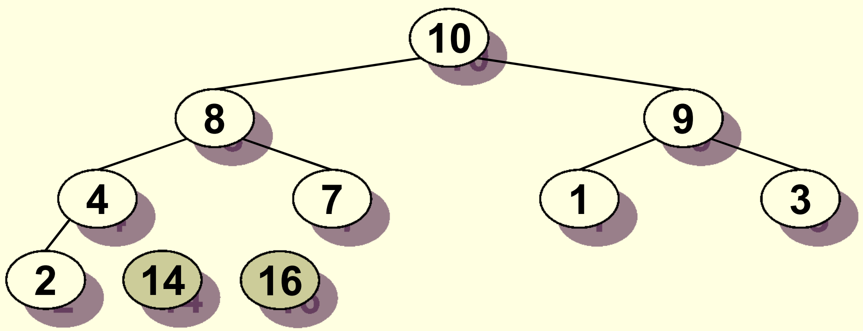
מיון-ערמה – דוגמה (המשך)



14	8	10	4	7	9	3	2	1		16



1	8	10	4	7	9	3	2	14	16



10	8	9	4	7	1	3	2	14	16

מיון-ערמה – ניתוח

- זמן הריצה של HeapSort על מערך בגודל n
  - זמן הריצה של Build-Max-Heap הוא $O(n)$
  - מספר האיטרציות של הלולאה (מספר הקריאות ל- Max-Heapify) הוא $n–1$
  - Max-Heapify רצה בזמן $O(\lg n)$
  - לכן זמן הריצה הכולל הוא

$$\begin{aligned} T(n) &= O(n) + (n-1)O(\lg n) \\ &= O(n \lg n) \end{aligned}$$

  - מכיון ש- $\Omega(n \lg n)$ הוא גם החסם התחתון על זמן הריצה (ראו את תרגיל 6.4-4 בספר), נקבל

$$T(n) = \Theta(n \lg n)$$

תור קדימויות

- שימושים בערמה
  - מיון
  - מימוש תור קדימויות (Priority Queue)
- תור קדימויות – מבנה נתונים לקבוצת איברים עם מפתחות, התומך בפעולות הבאות:
  - הכנסת איבר חדש עם מפתח נתון
  - החזרת האיבר שמפתחו מקסימלי
  - הוצאת האיבר שמפתחו מקסימלי
  - הגדלת המפתח של איבר נתון
- הערה: לפי הגדרה אנו משתמשים בעצם ב"ערמת מקסימום".
  - באופן דומה ניתן להגדיר ערמת מינימום (תכונת הערמה הפוכה) ותור קדימויות מתאים
- דוגמאות לשימוש בתור קדימויות:
  - תזמון משימות (ביצוע עבודות על מעבד, גישה לרשת וכו')
  - סימולציה של אירועים (המפתח: זמן האירוע – בערימת מינימום)

האוניברסיטה הפתוחה

הוצאת מקסימום

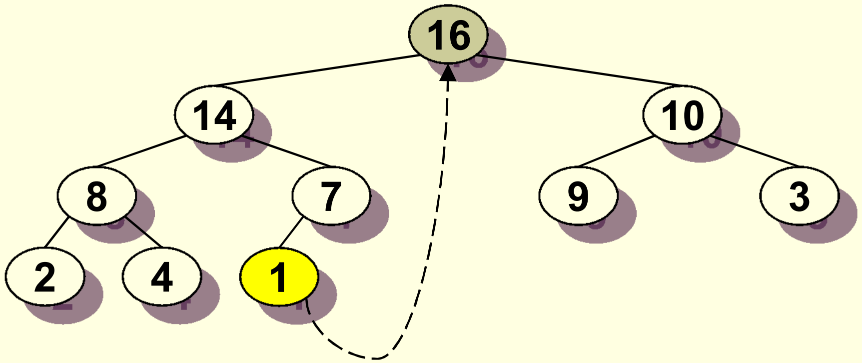
- הרעיון
  - המקסימום הוא $A[1]$; מחזירים את $A[1]$ ואח"כ מסירים את העלה האחרון ומעבירים את המפתח שלו לשורש; הופכים את העץ חזרה לערמה באמצעות קריאה ל- Max-Heapify.
  - *האלגוריתם*:

Heap-Extract-Max(A)

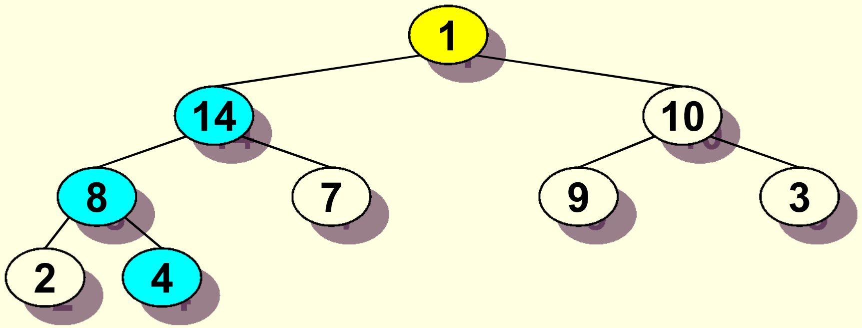
1. $n \leftarrow heapSize[A]$
2. **if** $n = 0$ **then error** "heap underflow"
3. $max \leftarrow A[1]$
4. $A[1] \leftarrow A[n]$
5. $heapSize[A] \leftarrow n-1$
6. Max-Heapify(A, 1)
7. **return** max

- זמן ריצה
  - כמו זמן הריצה של Max-Heapify: $O(\lg n)$

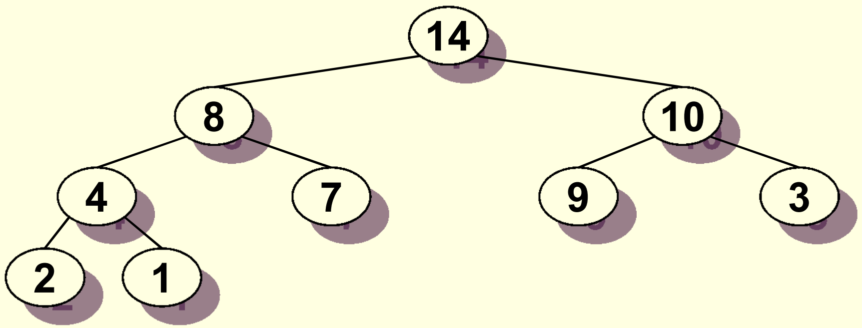
הוצאת מקסימום – דוגמה



16	14	10	8	7	9	3	2	4	1



1	14	10	8	7	9	3	2	4	



14	8	10	4	7	9	3	2	1	

הכנסת איבר

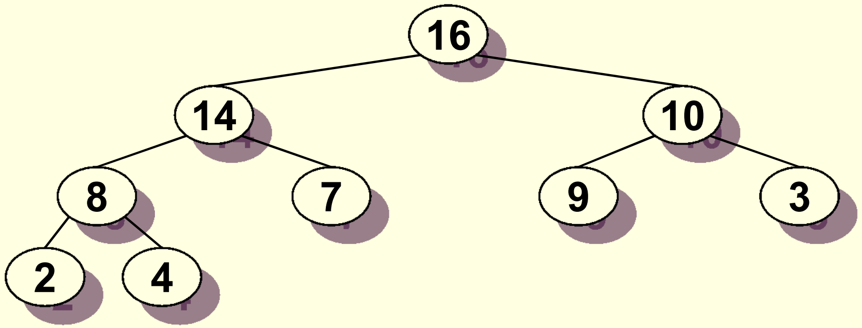
- הרעיון
  - יוצרים צומת חדש כעלה; "דוחפים" את המפתח החדש כלפי מעלה (תוך כדי החלפות) עד שמתקיימת תכונת הערמה.
  - *האלגוריתם:*

Max-Heap-Insert(A, key)

1. $heapSize[A] \leftarrow heapSize[A] + 1$
2. $i \leftarrow heapSize[A]$
3. **while** $i > 1$ **and** $A[\text{Parent}(i)] < key$
4. **do** $A[i] \leftarrow A[\text{Parent}(i)]$
5. $i \leftarrow \text{Parent}(i)$
6. $A[i] \leftarrow key$

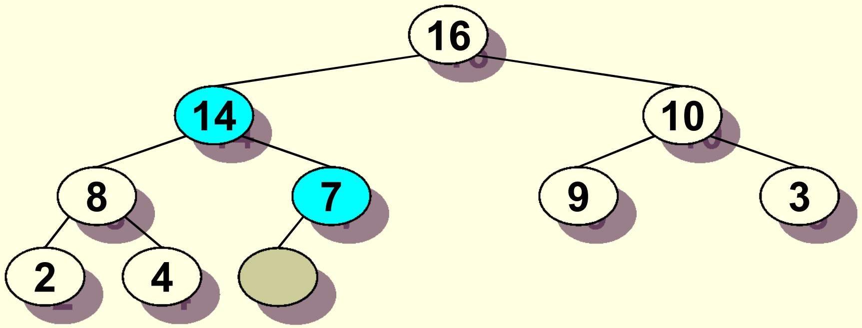
- זמן ריצה
  - במקרה הגרוע המפתח החדש יעלה עד לשורש, לכן זמן הריצה הוא כגובה העץ, $O(\lg n)$

הכנסת איבר – דוגמה

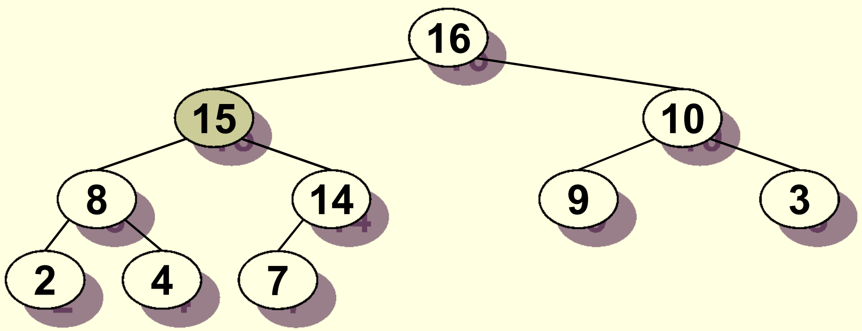


15

16	14	10	8	7	9	3	2	4	



16	14	10	8	7	9	3	2	4	



16	15	10	8	14	9	3	2	4	7

הגדלת ערך איבר

- הרעיון
  - מציבים את הערך החדש באיבר. מכיוון שהערך גדול יותר, יש לבצע תיקון כלפי השורש, בדומה להכנסת איבר
- *האלגוריתם*:

Heap-Increase-Key(A, i, key)

1. **if** $key < A[i]$
2. **then error** "new key is smaller than current key"
3. $A[i] \leftarrow key$
4. **while** $i > 1$ and $A[\text{Parent}(i)] < A[i]$
5. **do** exchange $A[i] \leftrightarrow A[\text{Parent}(i)]$
6. $i \leftarrow \text{Parent}(i)$

הרחבה: מחיקת איבר

- הרעיון
  - מציבים במקום האיבר ה-i את האיבר האחרון (העלה הכי ימני) ומקטינים את גודל הערמה ב-1; אם האיבר *ה-i* החדש גדול מאביו, אז "דוחפים" אותו במעלה הערמה כמו באלגוריתם של הכנסת איבר לערמה; אחרת, קוראים ל-Max-Heapify כדי לטפל במקרה שהאיבר החדש קטן מאחד מבניו (אם האיבר *ה-i* החדש כבר "טיפס" למעלה אז הקריאה ל-Max-Heapify חוזרת מייד).
- *האלגוריתם:*

Heap-Delete(A, i)

$n \leftarrow heapSize[A]$

if $i < 1$ **or** $i > n$ **then error** "index out of bound"

$A[i] \leftarrow A[n]$

$heapSize[A] \leftarrow n - 1$

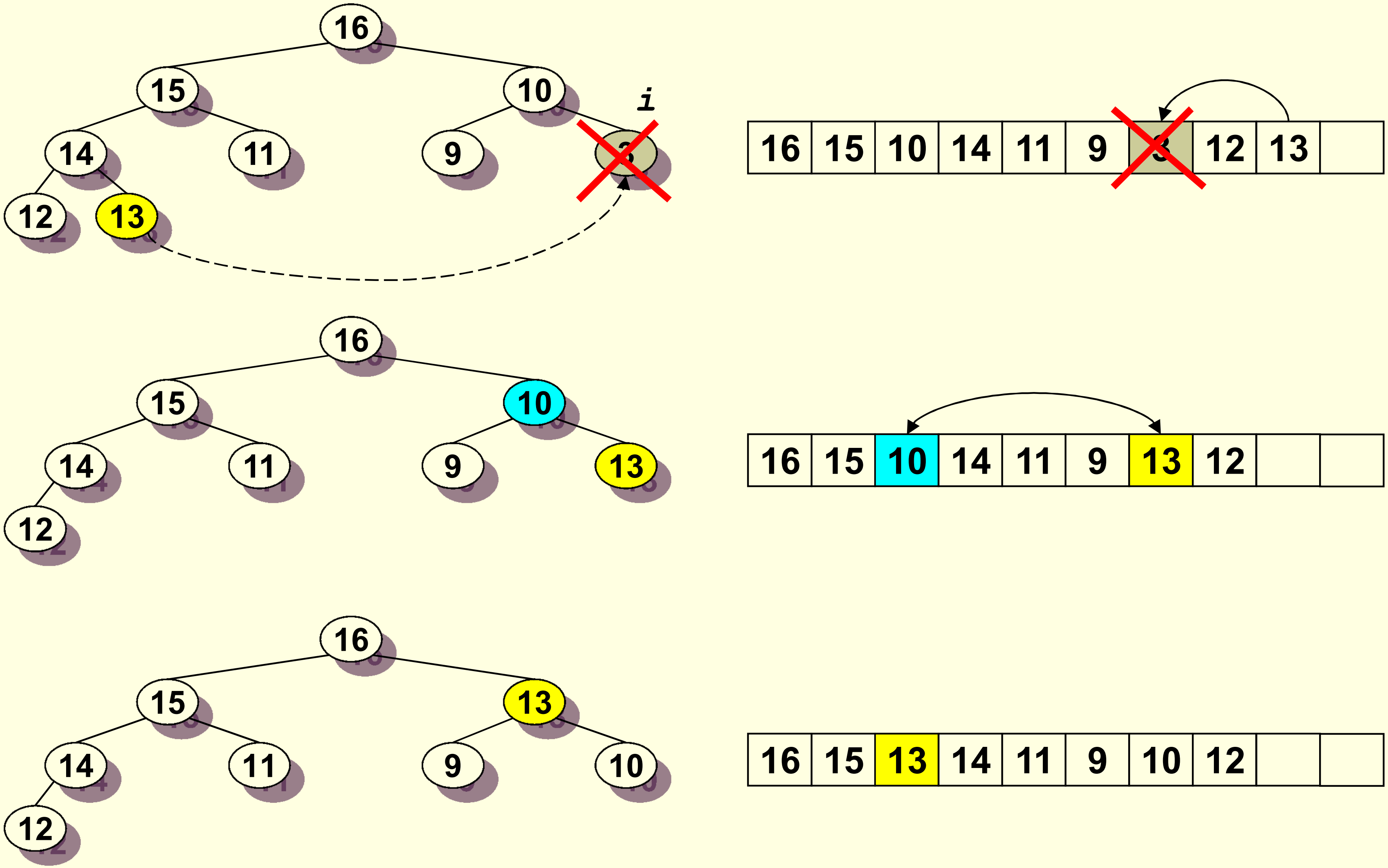
while $i > 1$ and $A[\text{Parent}(i)] < A[i]$

  do exchange $A[i] \leftrightarrow A[\text{Parent}(i)]$

    $i \leftarrow \text{Parent}(i)$

Max-Heapify(A, i)

מחיקת איבר – דוגמה



תרגיל

- באוניברסיטה הפתוחה לומדים n סטודנטים. נתוני הסטודנטים (שם, מס' סטודנט, גיל וכו') נמצאים בקובץ. גודל הזיכרון העומד לרשותנו במחשב האו"פ הוא $n>>m$. מעוניינים למצוא את m הסטודנטים הצעירים ביותר באוניברסיטה הפתוחה. כתבו אלגוריתם יעיל המדפיס את שמותיהם של m הסטודנטים האלה

- נשתמש בערמה בגודל m.

1. נקרא מהקובץ את נתוני m הסטודנטים הראשונים ונבנה מהם ערמת מקסימום, כשהמפתח הוא גיל הסטודנט.
2. נמשיך לעבור בצורה סדרתית על הקובץ, סטודנט אחרי סטודנט; בכל שלב נשווה את גילו של הסטודנט שקראנו מהקובץ עם המפתח של שורש הערמה; אם גילו של הסטודנט צעיר יותר – נציב אותו בשורש הערמה (במקום השורש הקודם), ונתקן את הערמה באמצעות קריאה ל- Max-Heapify.
3. לבסוף, אחרי שעברנו על כל הקובץ, נדפיס את שמותיהם של הסטודנטים שנמצאים בערמה.

- הסבוכיות: שלב 1 ושלב 3: $O(m)$; שלב 2: $(n–m)O(\log m)$
  סה"כ: $O(n\log m)$