

ערימה

Heap

מה נלמד?

- **מבנה נתונים ערים**
 - **הגדלה**
 - **פועלות**
 - **שימושים**





נתונה קבוצה דינמית של איברים S , כאשר לכל איבר מצורף מפתח key

key *Satellite data*

מינימלי

מקסימום

ערימת מקסימום היא מבנה נתונים מבוסס עץ שמאפשר גישה מהירה לערך מקסימאלי ב- S

או מינימום או מקסימום, **לא** בו זמן ניתן



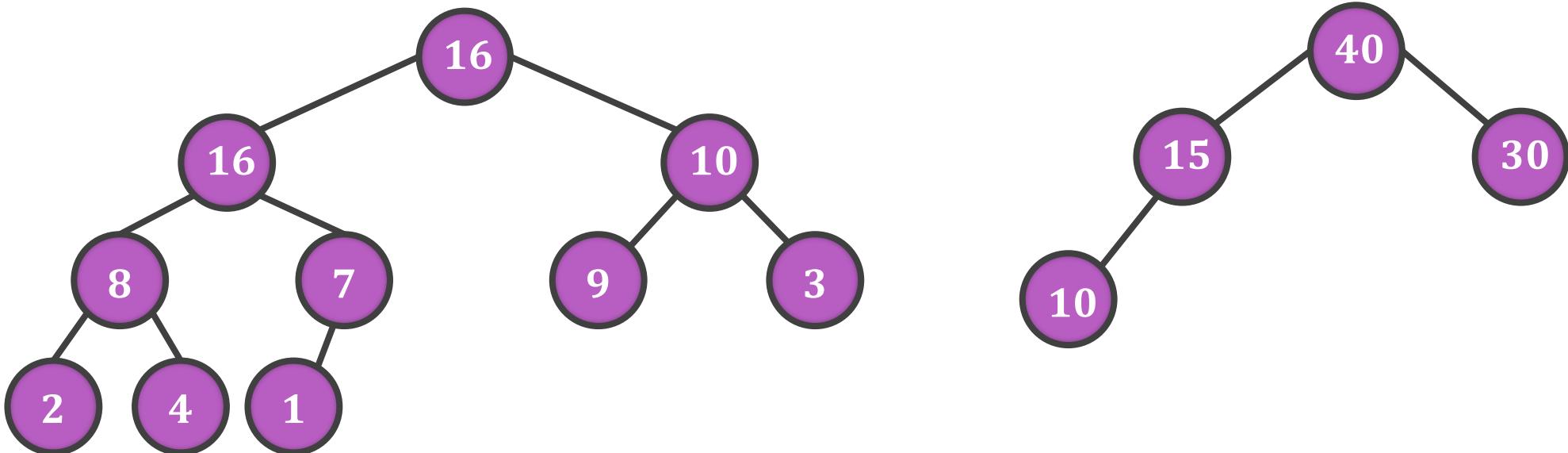
הגדירה - המשך

עリימה מקיימת את תכונת הערימה:

עリימת מקסימום:

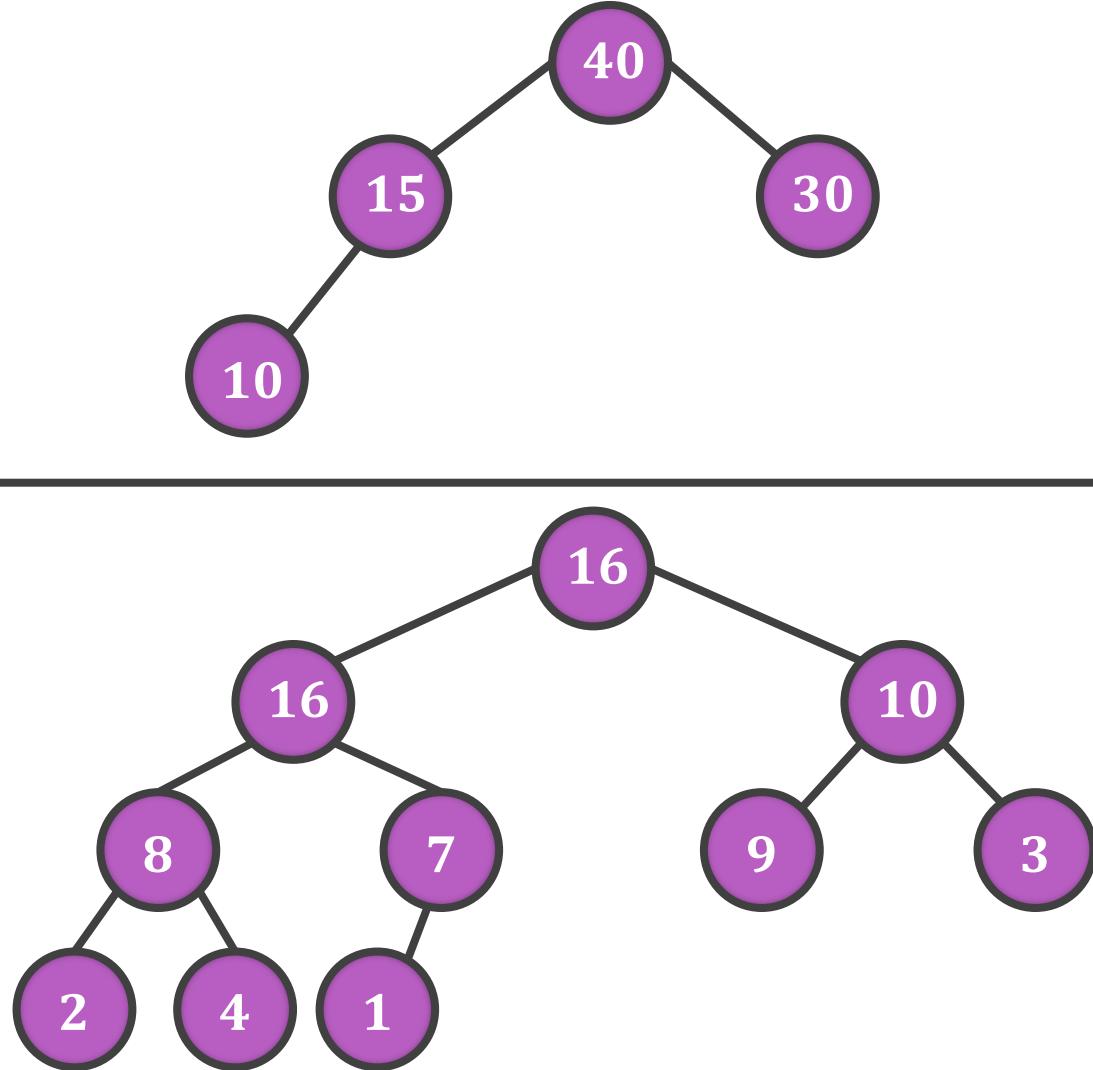
לכל צומת x :

$x.key \geq keys\ of\ its\ children$



?

איפה בערימת מקסימום נמצא איבר בעל מפתח מקסימלי?



באחד העלים

.1

בשורש

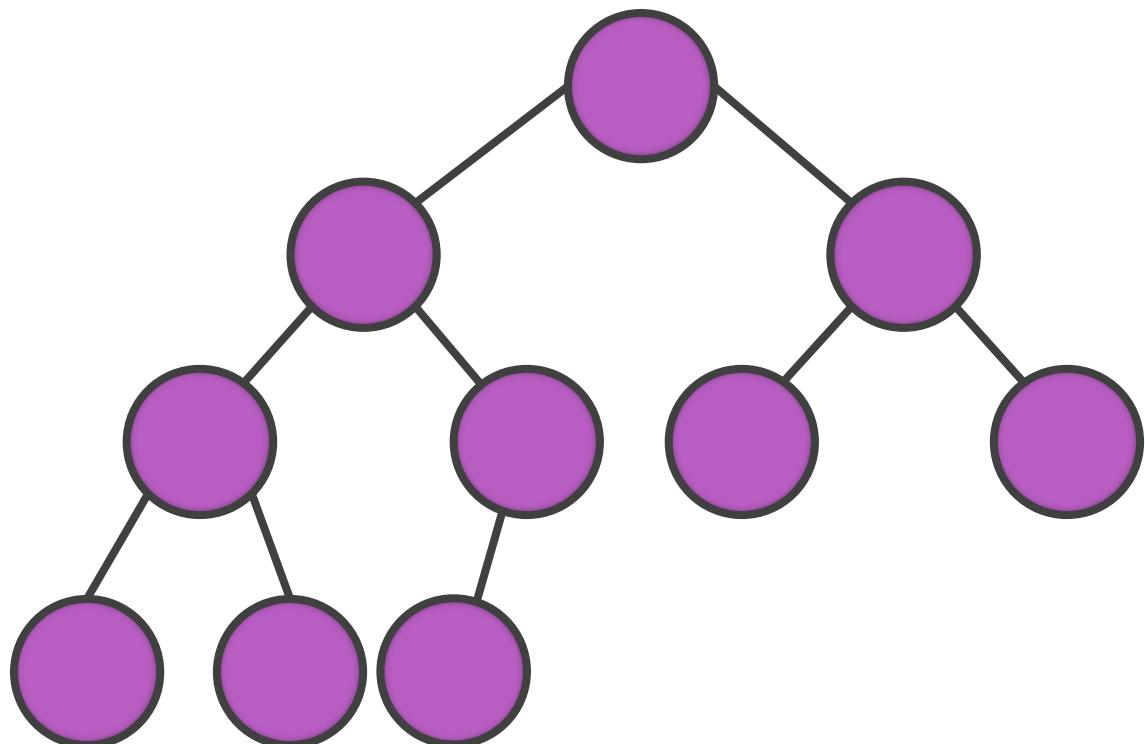
.2

לא ניתן לדעת בזדאות

.3



הגדירה - המשך



עדרינה מקיימת את תוכנת העדרינה:

עדרינה מקסימום:
לכל צומת x :

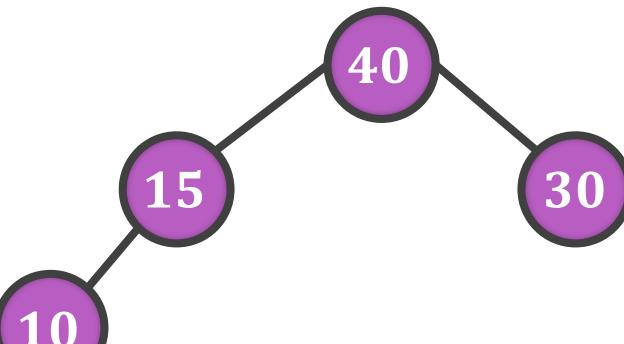
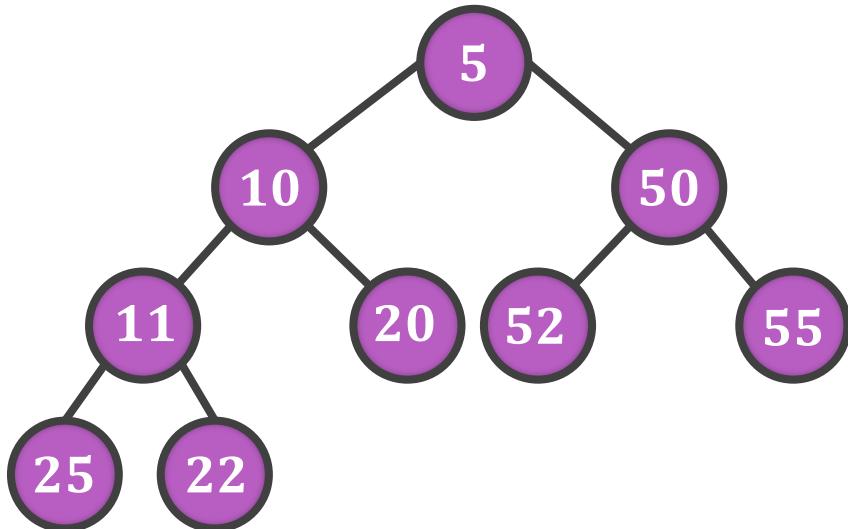
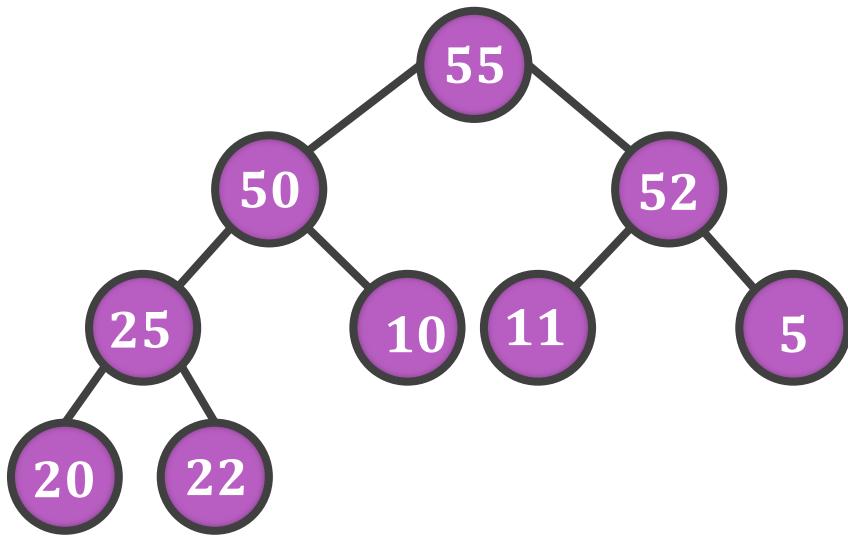
$$x.key \geq keys\ of\ its\ children$$

עדרינה מינימום
לכל צומת x :

$$x.key \leq keys\ of\ its\ children$$

עדרינה תיוצג על ידי **עץ שלם**

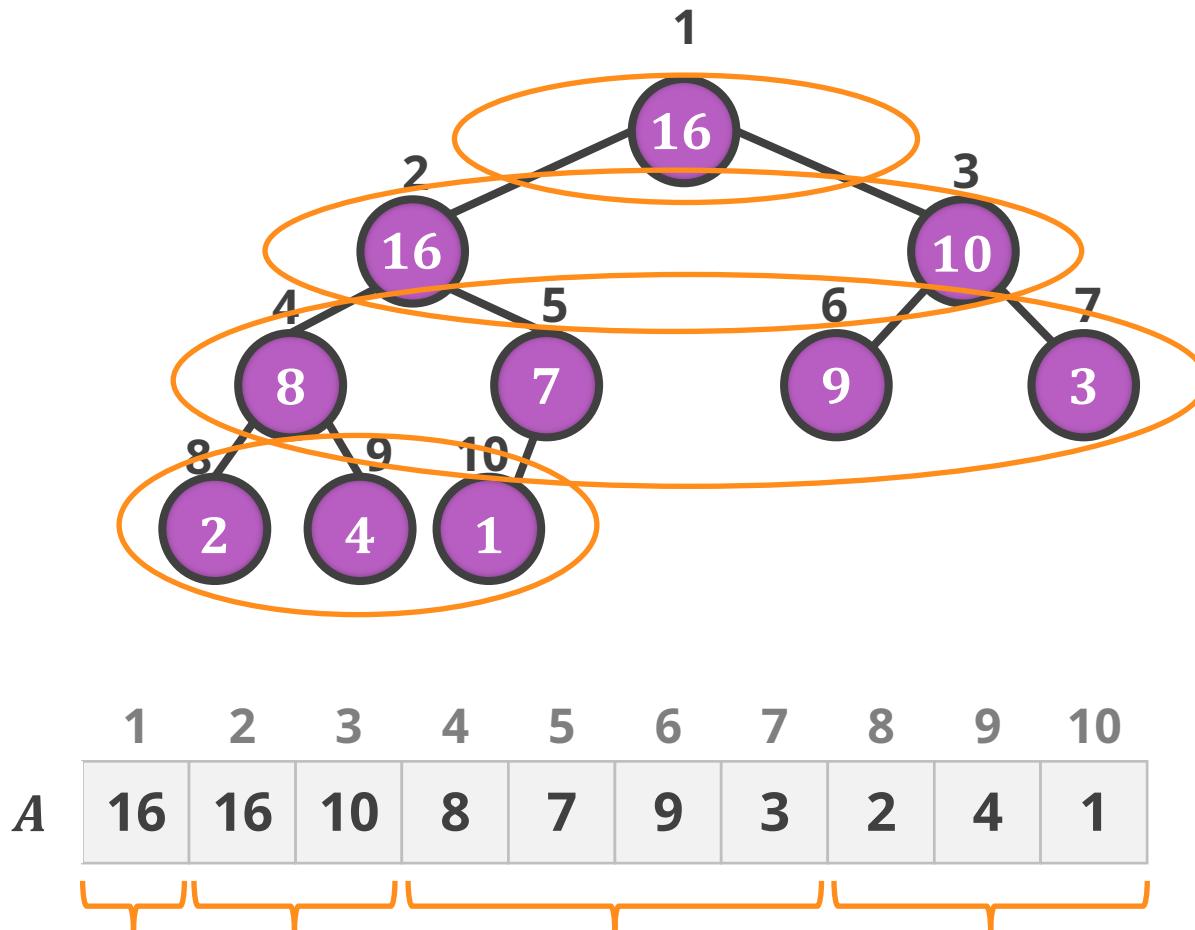
דוגמאות



ערימת **מקסימום**:

ערימת **מינימום**:

הגדירה - המשן



נמחש עירימה באמצעות מערך
שורש העץ - $[1] A$
בהתנאי אינדקס i של ה策ומת

left(i)
return $2i$

right(i)
return $2i + 1$

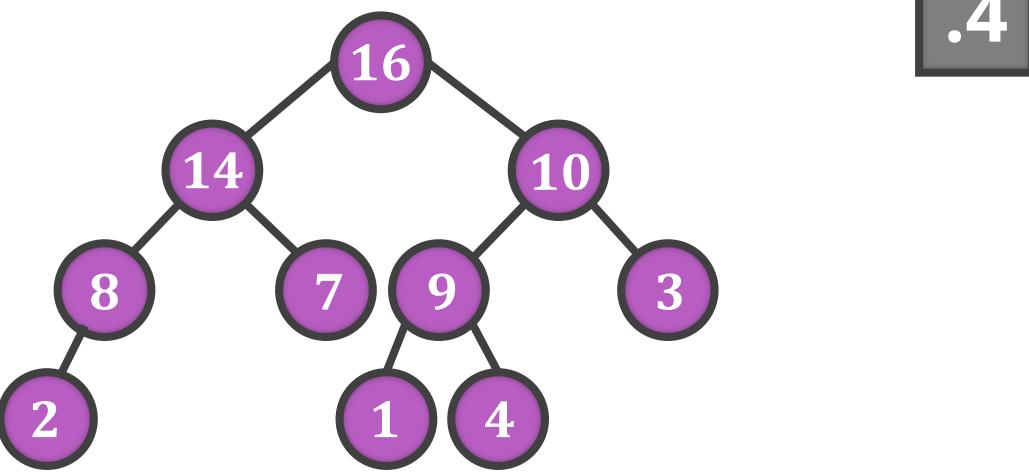
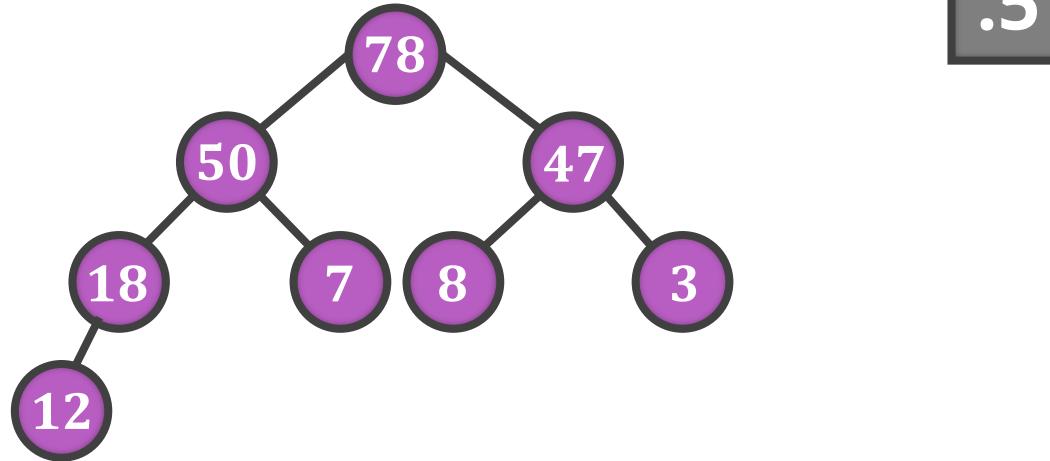
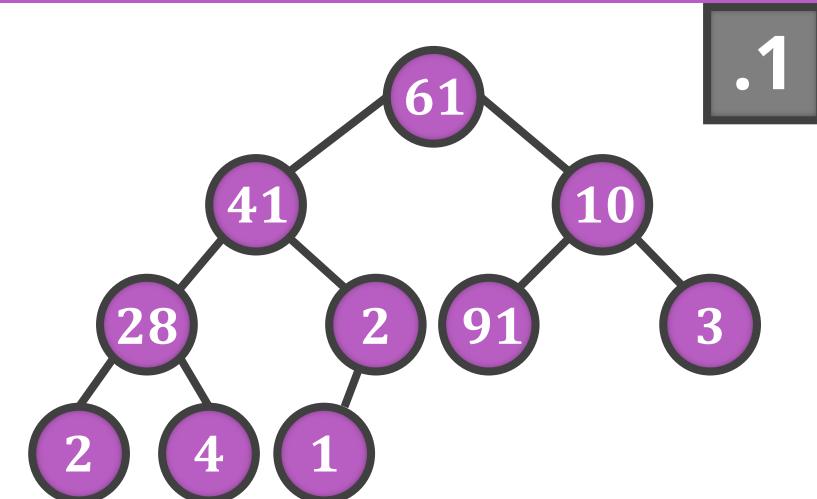
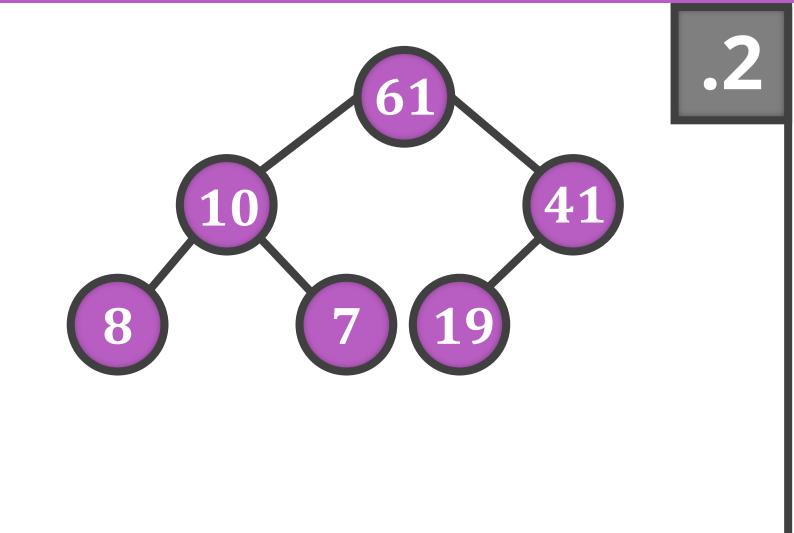
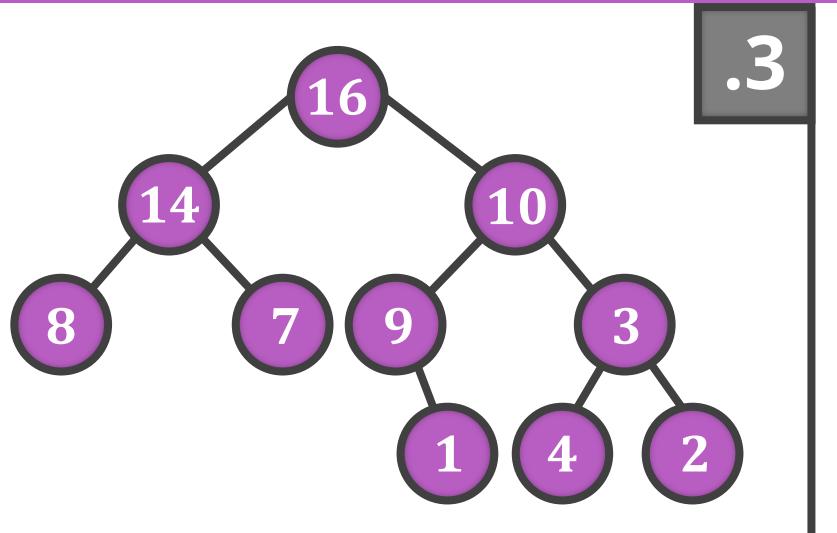
parent(i)
return $\lfloor i/2 \rfloor$

למערך A המיצג עירימה שני מאפיינים:

- גודל המערך - $A.length$
- גודל העירימה - $A.heap_size$

גובה של עירימה בעלת n איברים הוא $[n \log n]$

אילו מה העצים הבאים הם ערימת מקסימום?

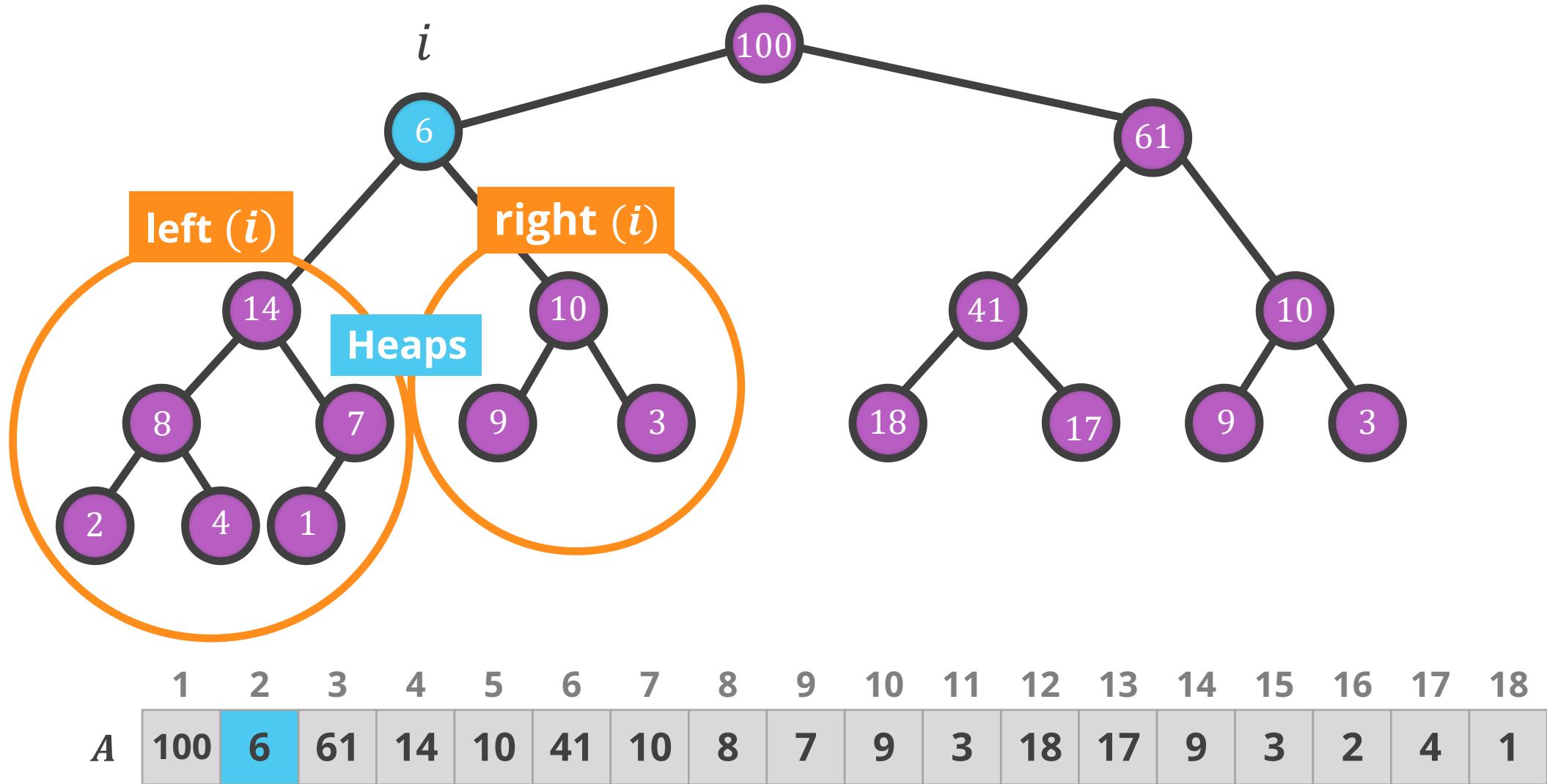




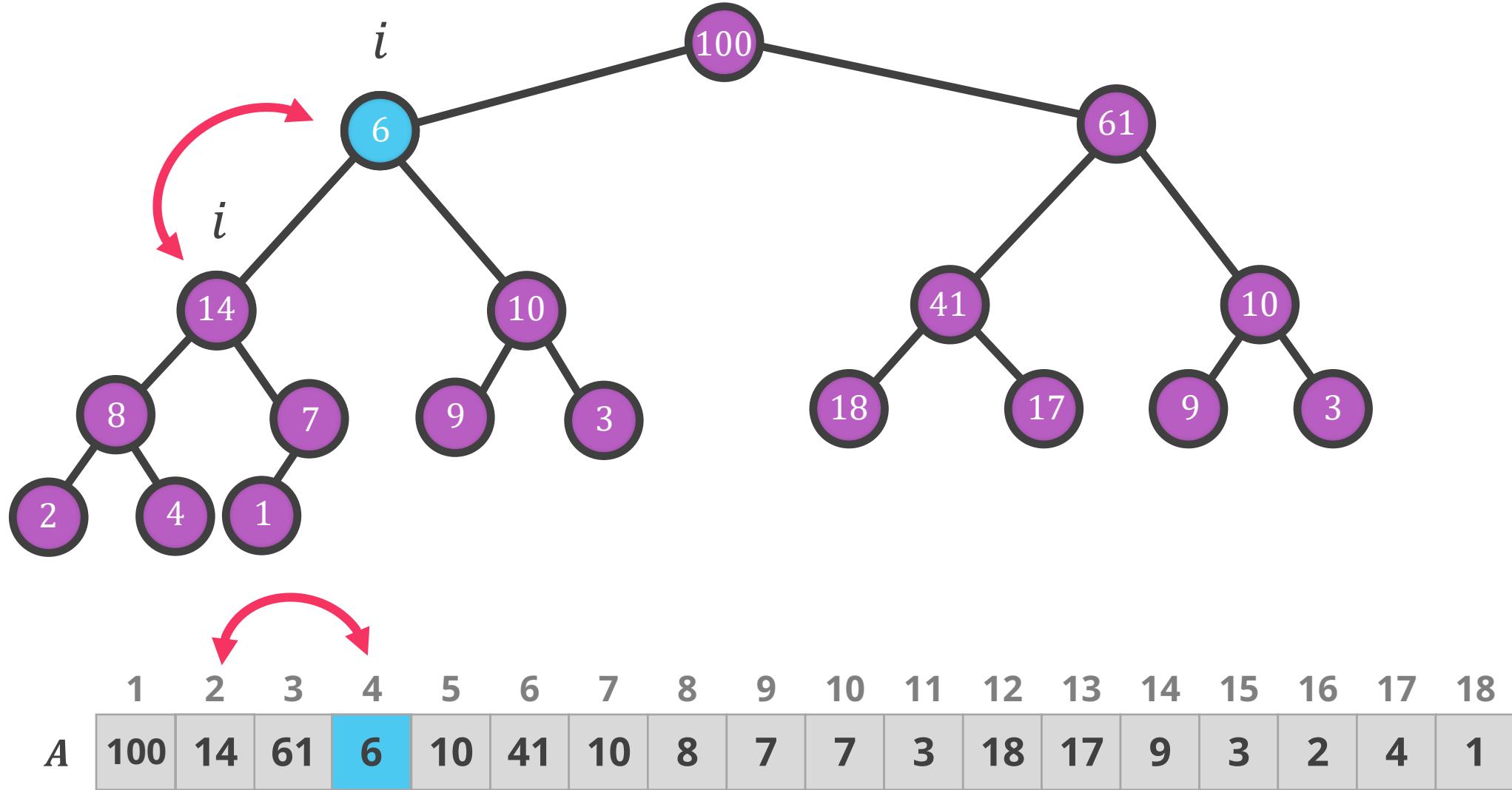
פעולות המוגדרות על ערימה

- **יצרת ערימה ממארך קלט בלתי ממויין** - *Build-Max-Heap*
- **החזרת איבר בעל מפתח מקסימלי בערימת מקסימים** - *Heap-Max*
- **מחיקת איבר בעל מפתח מקסימאלי בערימת מקסימים** - *Heap-Extract-Max*
- **הגדלת מפתח** - *Heap-Increase-Key*
- **הכנסת איבר חדש** - *Max-Heap-Insert*
- **שמירה על תכונת הערימה** - *Max-Heapify*

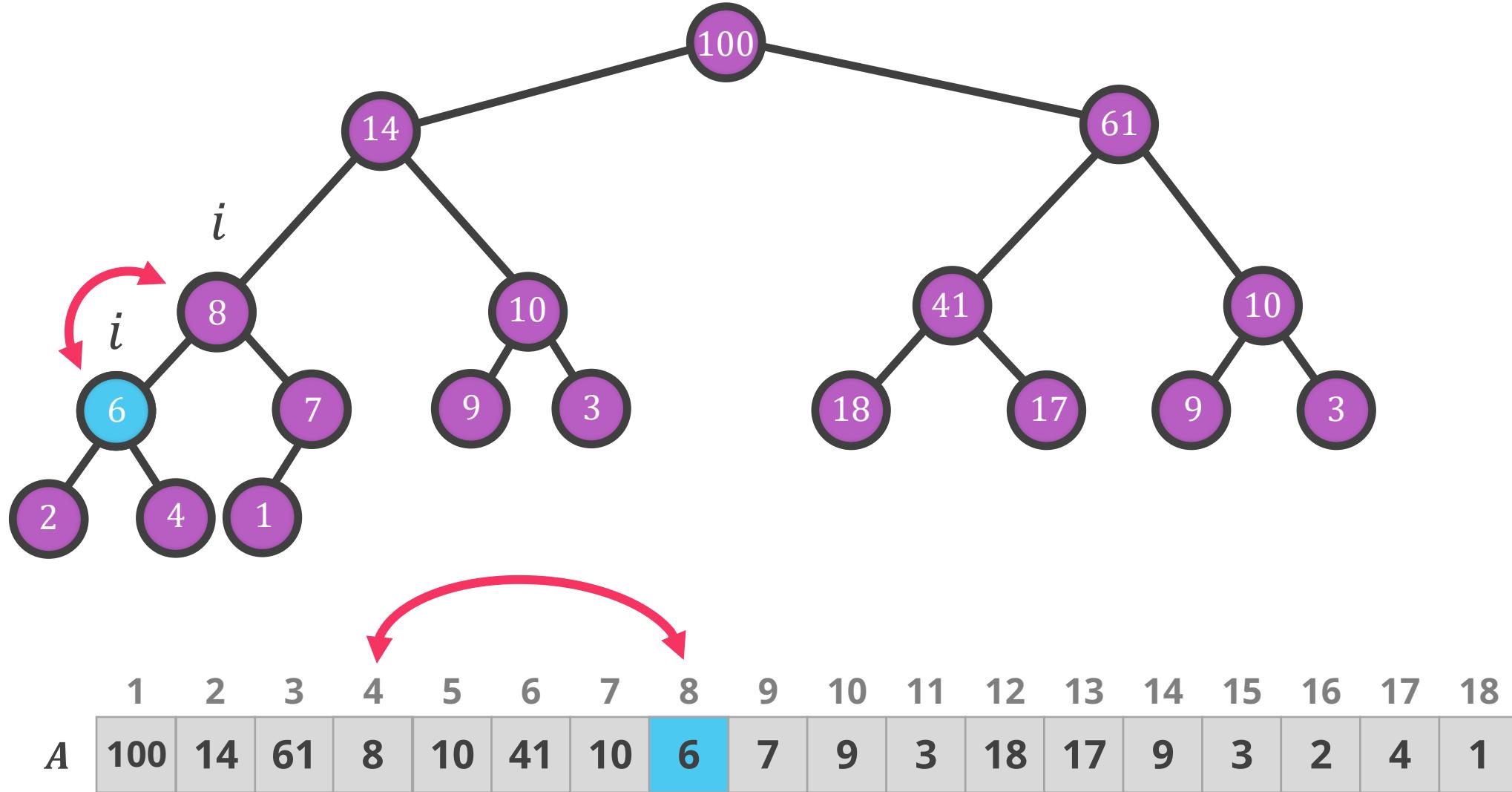
MAX-HEAPIFY



MAX-HEAPIFY



MAX-HEAPIFY



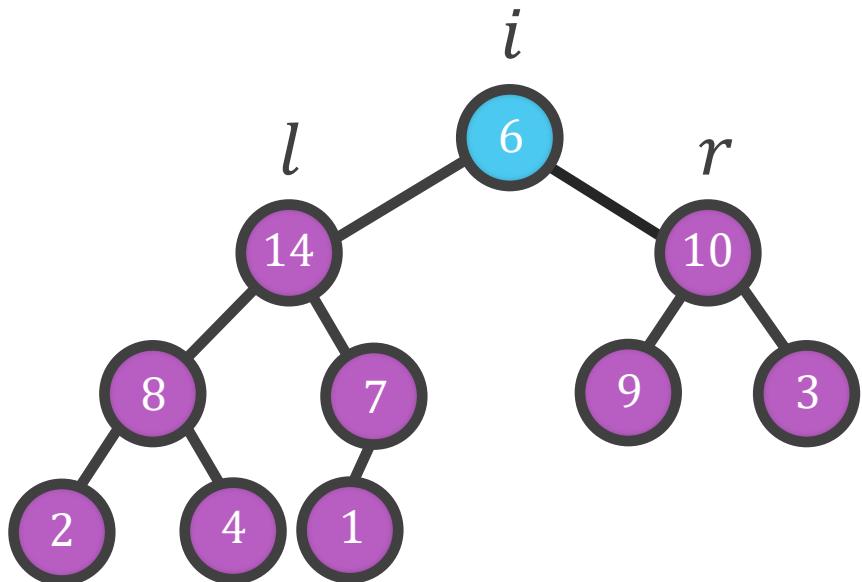
MAX-HEAPIFY

- **קלט:** מערך A ואינדקס i
- **הנחה:** העצים הבינאריים המושרשים ב- (i) -*right* (i)-*left* (i) הם עריםות תקיןות, אבל אולי $[i]A$ קטן מבניו, ובכך מפר את תכונות העריםה
- **פלט:** העץ שורשו $[i]A$ הופך לעירמת מקסימום

Max – Heapify (A, i)

```
1   l  $\leftarrow$  left(i )
2   r  $\leftarrow$  right(i )
3   if l  $\leq$  A.heap_size and A[l] > A[i ]
4       largest  $\leftarrow$  l
5   else largest  $\leftarrow$  i
6   if r  $\leq$  A.heap_size and A[r] > A[largest]
7       largest  $\leftarrow$  r
8   if largest  $\neq$  i
9       exchange A[i ]  $\leftrightarrow$  A[largest]
10  Max-Heapify(A, largest)
```

MAX-HEAPIFY



?

מה זמן הריצה של Max-Heapify

$O(\log n)$

.1

$O(n^2)$

.2

$O(n)$

.3

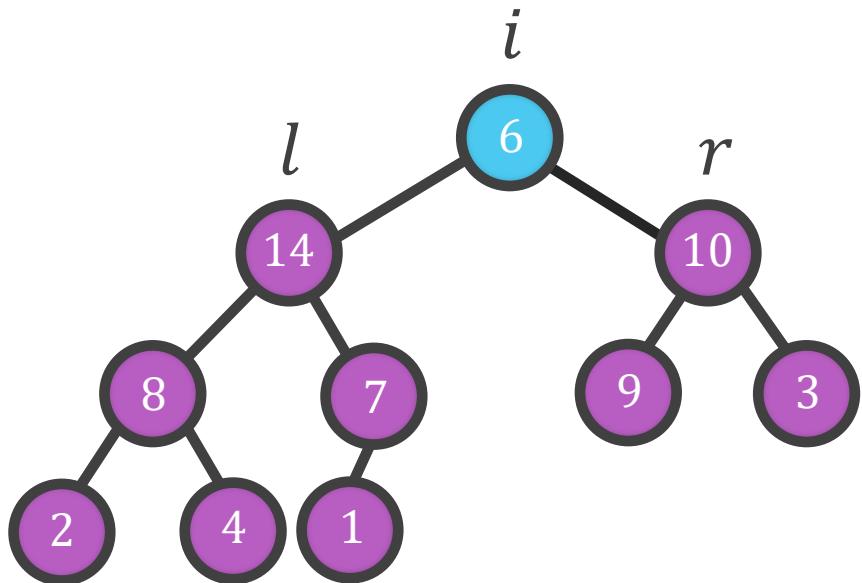
$O(n \log n)$

.4

Max – Heapify (A, i)

```
1    $l \leftarrow left(i)$ 
2    $r \leftarrow right(i)$ 
3   if  $l \leq A.heap\_size$  and  $A[l] > A[i]$ 
4        $largest \leftarrow l$ 
5   else  $largest \leftarrow i$ 
6   if  $r \leq A.heap\_size$  and  $A[r] > A[largest]$ 
7        $largest \leftarrow r$ 
8   if  $largest \neq i$ 
9       exchange  $A[i] \leftrightarrow A[largest]$ 
10  Max-Heapify( $A, largest$ )
```

MAX-HEAPIFY



זיהוי ריצחה:

$$T(n) = O(h) = O(\log n)$$

פעולות המוגדרות על עירימה

- יוצרת עירימה ממערך קלט בלתי ממוקם *Build-Max-Heap*

- החזרת איבר בעל מפתח מקסימלי בערימת מקסימום *Heap-Max* •
- מחיקת איבר בעל מפתח מקסימאלי בערימת מקסימום *Heap-Extract-Max* •
- הגדלת מפתח *Heap-Increase-Key* •
- הכנסת איבר חדש *Max-Heap-Insert* •
- שמירה על תכונת העירימה *Max-Heapify* •

Build-Max-Heap

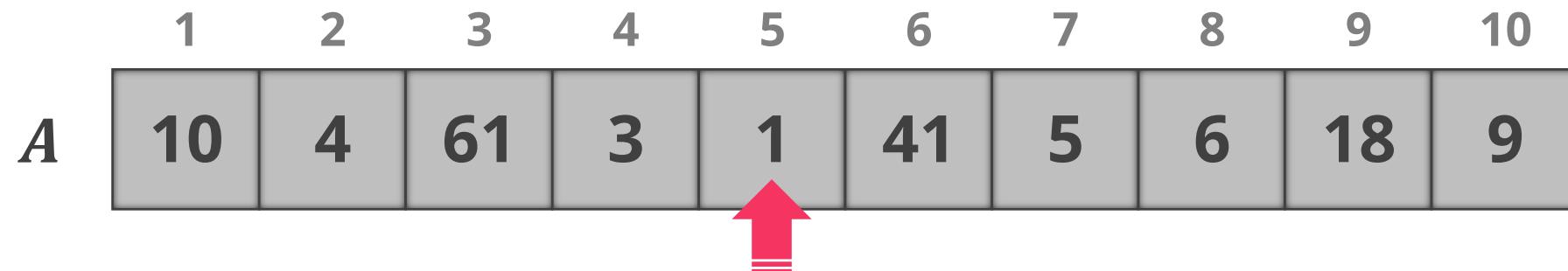
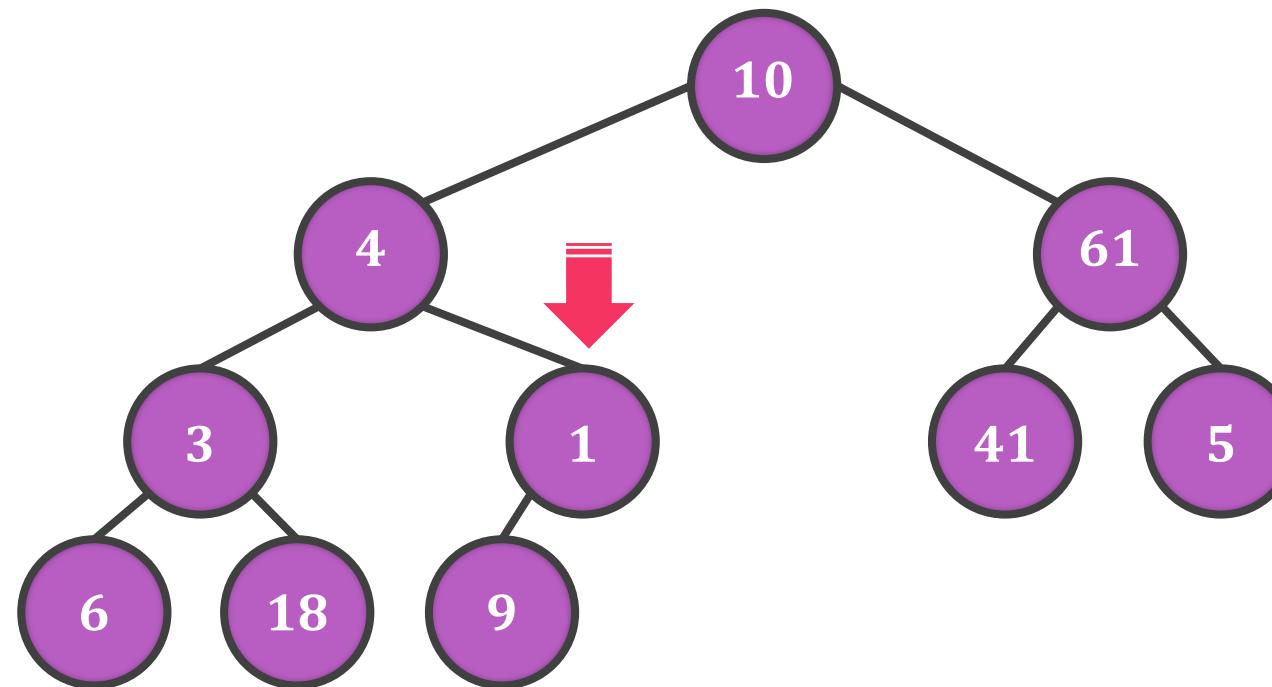
נתון: מערך A לא מוחזק



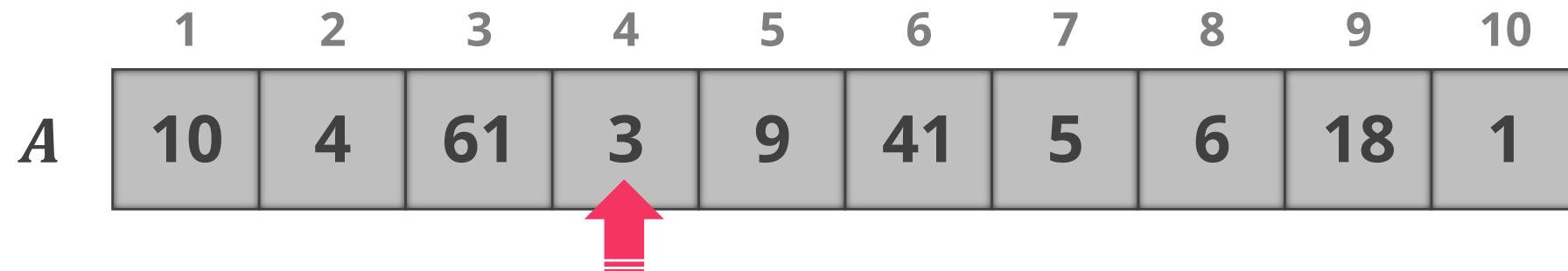
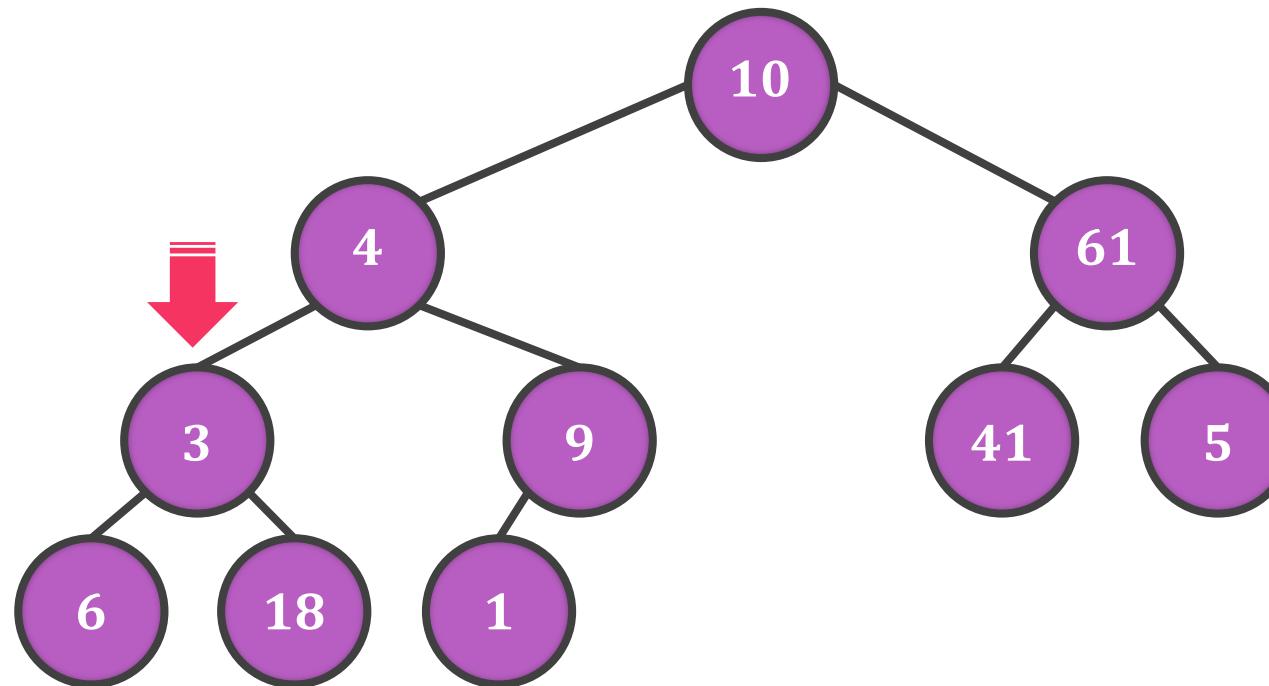
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|-----|----|---|----|---|---|----|---|---|----|----|
| A | 10 | 4 | 61 | 3 | 1 | 41 | 5 | 6 | 18 | 9 |

מטרה: לבנות עירימת קסימות כ- A

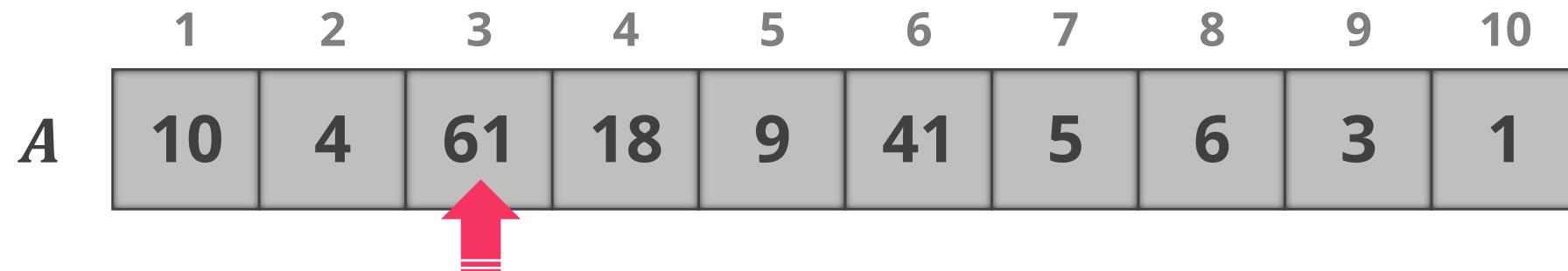
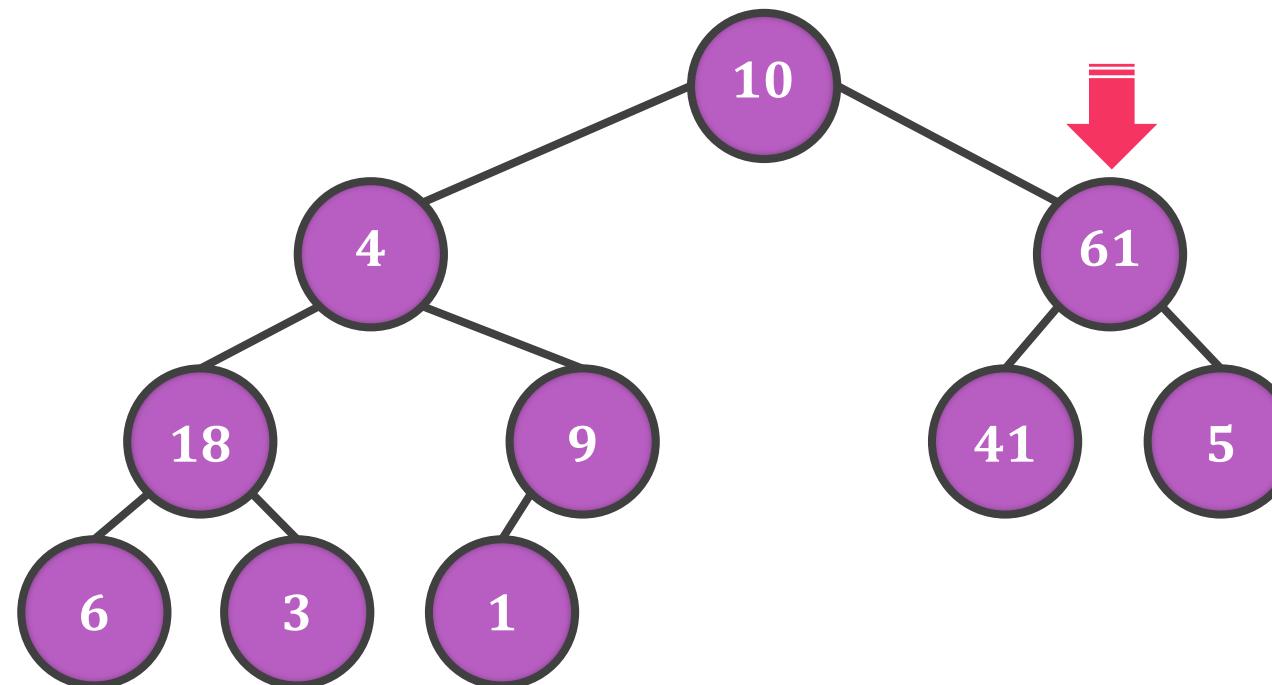
Build-Max-Heap



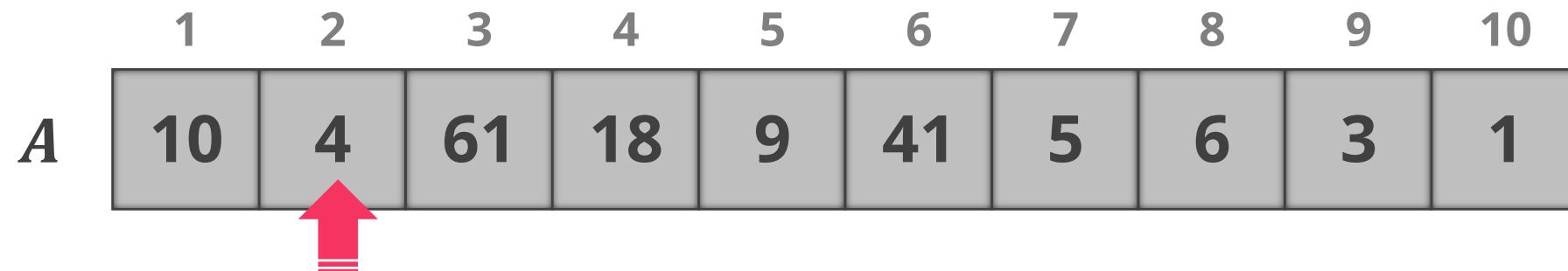
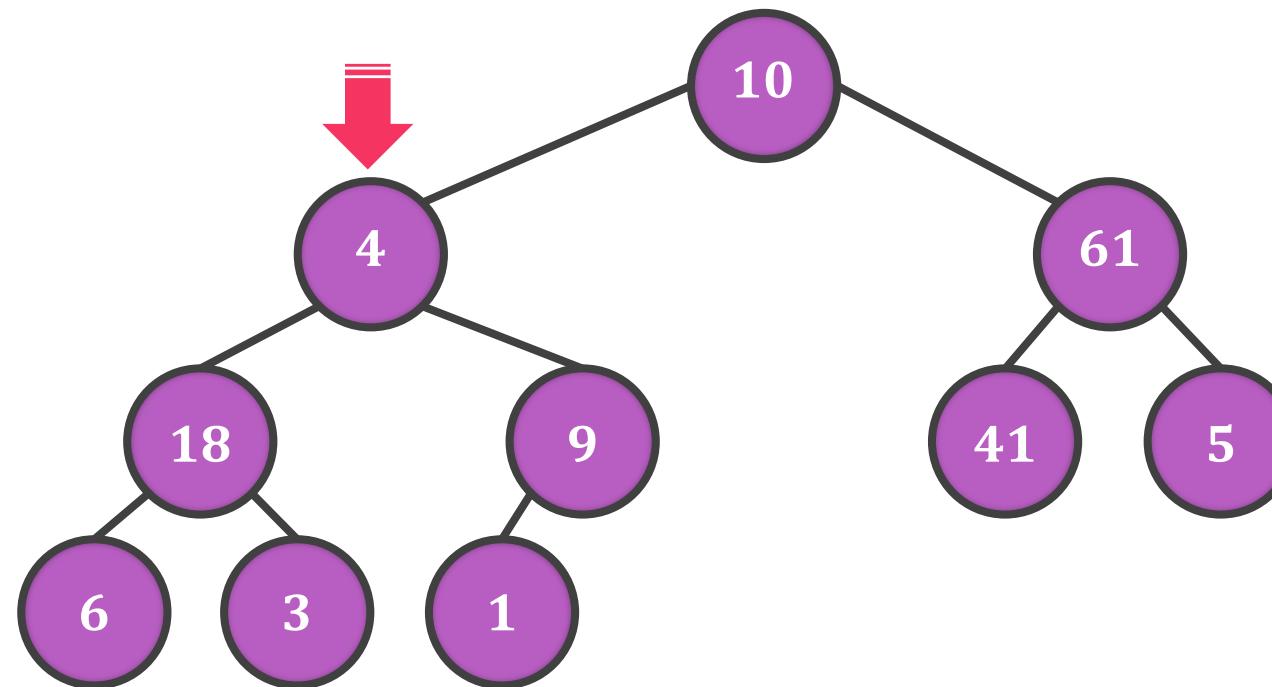
Build-Max-Heap



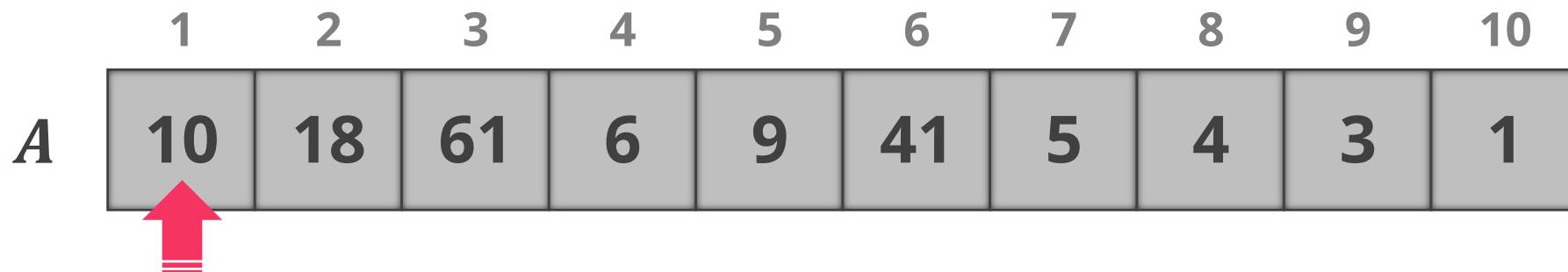
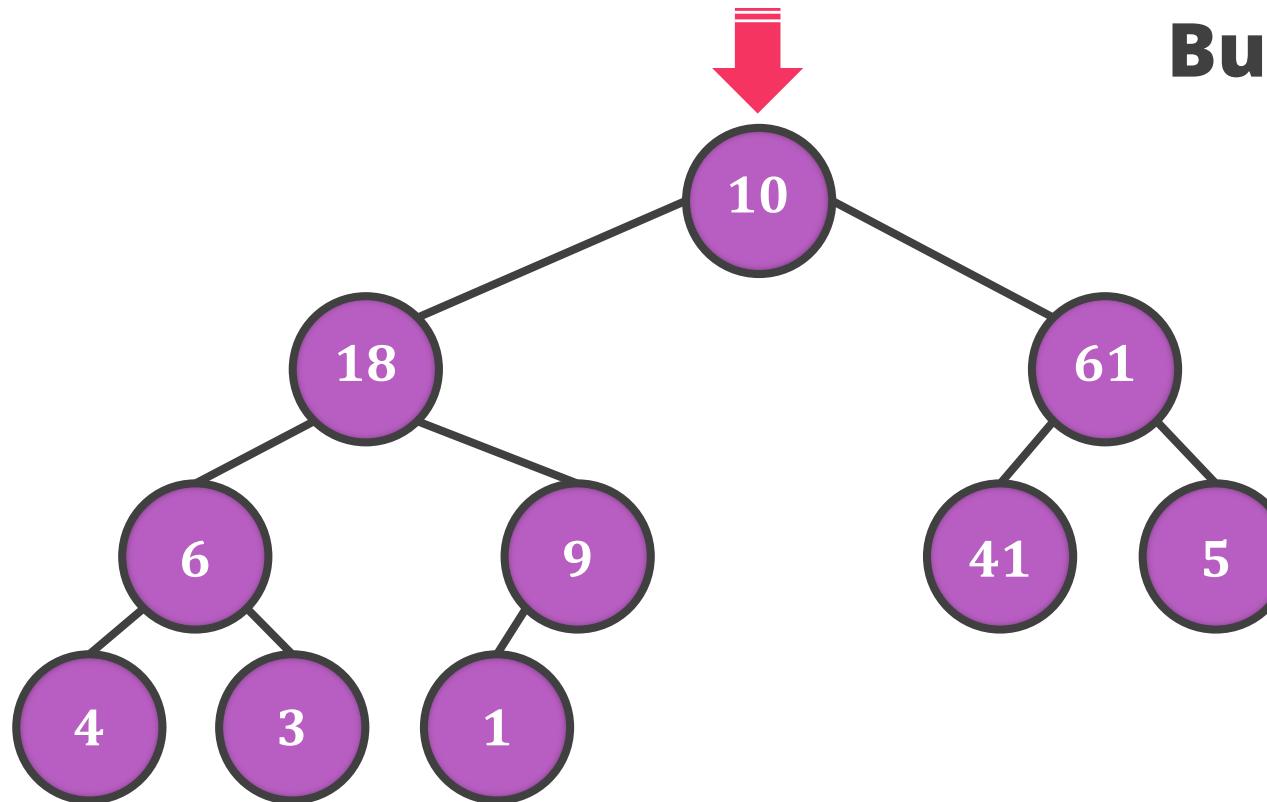
Build-Max-Heap



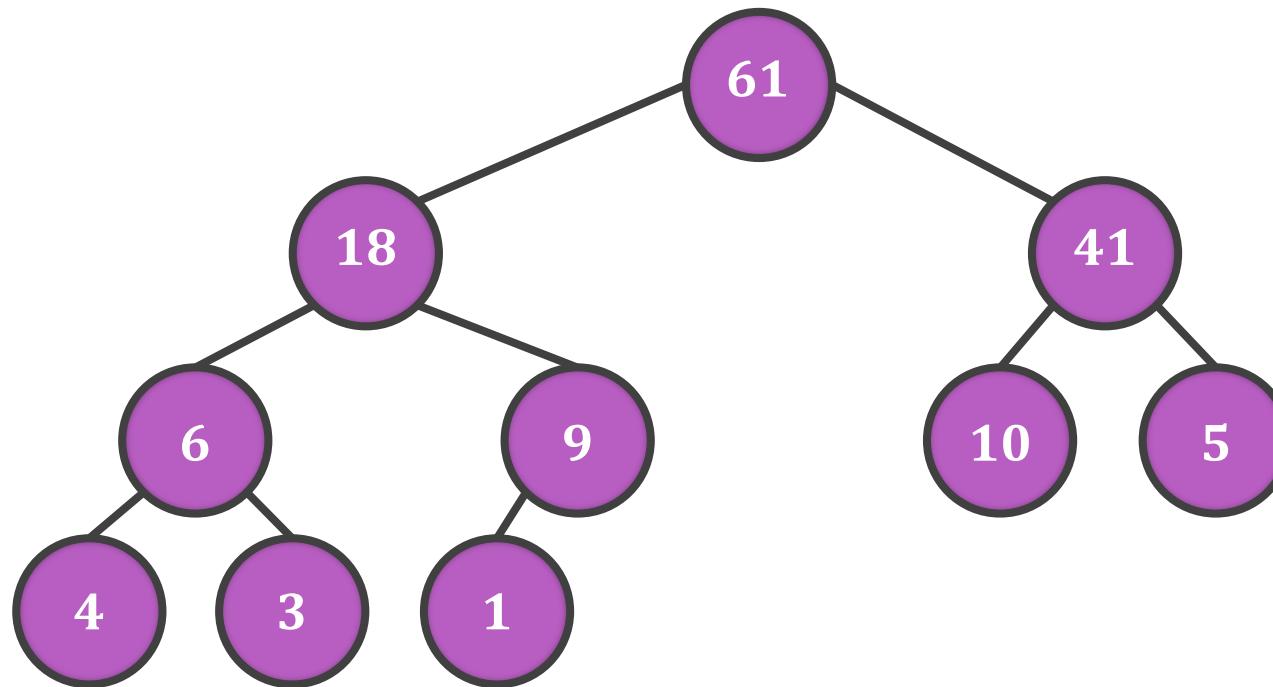
Build-Max-Heap



Build-Max-Heap



Build-Max-Heap



| | | | | | | | | | | |
|----------|----|----|----|---|---|----|---|---|---|----|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| <i>A</i> | 61 | 18 | 41 | 6 | 9 | 10 | 5 | 4 | 3 | 1 |

Build-Max-Heap

- האיברים $[n]$ הם עליים בעץ ולכן כל אחד מהם הוא עירימה בגודל 1

| | | | | | | | | | | |
|-----|----|---|----|---|---|----|---|---|----|----|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| A | 10 | 4 | 61 | 3 | 1 | 41 | 5 | 6 | 18 | 9 |

- מעבר על שאר ה指挥ים ונפעיל Max-Heapify על כל אחד מהם

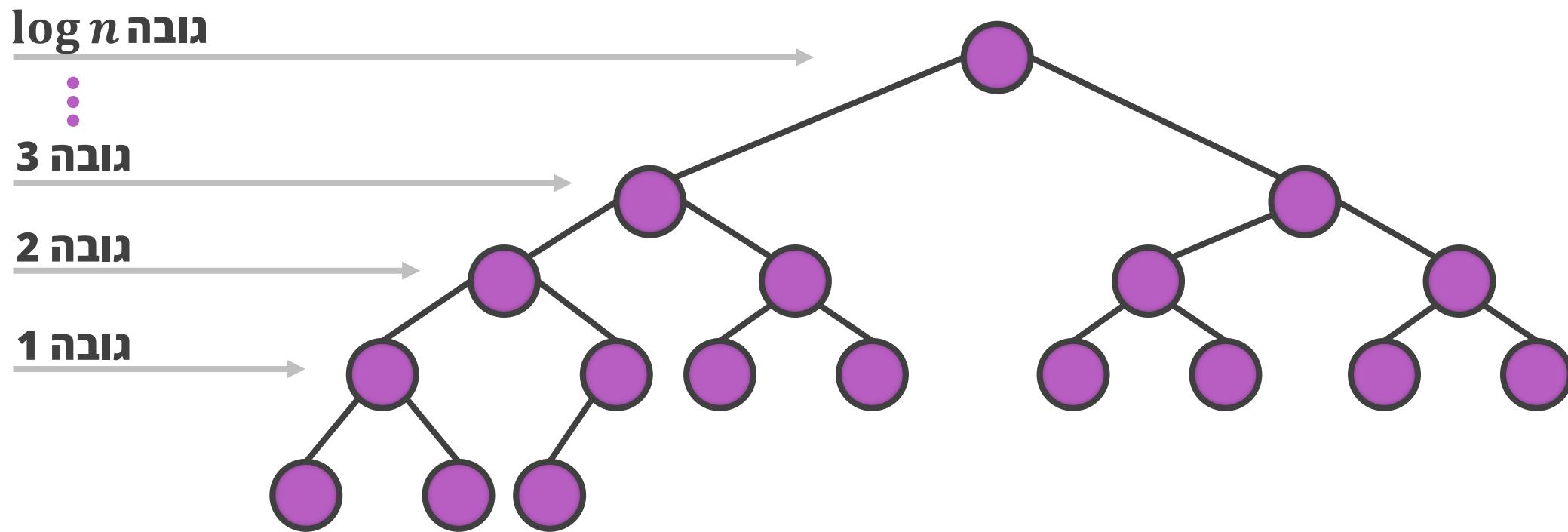
Build-Max-Heap(A)

- 1 $A.\text{heap_size} \leftarrow A.\text{length}$
- 2 $\text{for } i \leftarrow [A.\text{length}/2] \text{ downto 1}$
- 3 $\text{Max-Heapify}(A, i)$

זמן ריצה של Build-Max-Heap

- כל קרייה ל- Max-Heapify ממבצעת ב- $O(\log n)$, יشنן ($O(n)$ קריאות כאלה,
לכן זמן ריצה לכל היותר $O(n \log n)$)

חסם זה אינו חסם חזק אסימפטוטית



למה לדעתכם שווה זמן ריצה של הפועלה ?*Build-Max-Heap*

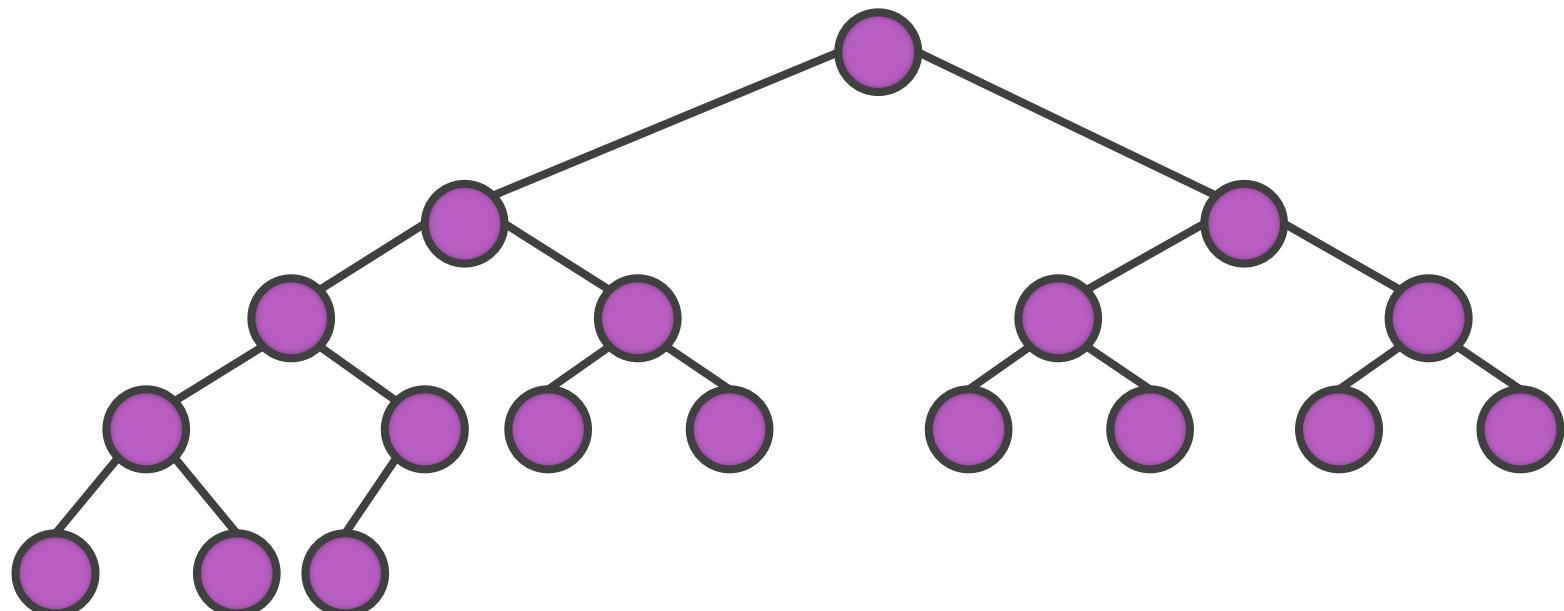


זמן ריצה של Build-Max-Heap

nocich shzon ritza shel *Build-Max-Heap* hoa $O(n)$



טענת עזר: מספר הצלחות שגובהם h בערימה בת n איברים הוא לפחות $\frac{n}{2^{h+1}}$

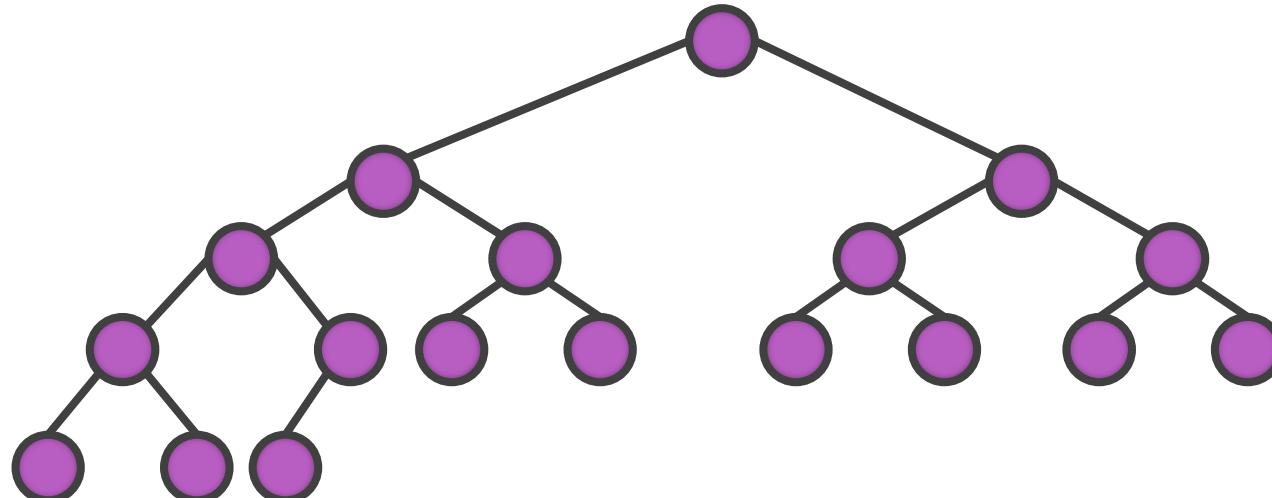


זמן ריצה של Build-Max-Heap

טענת עזר: מספר הצלחות שגובהם h בערימה בת n איברים הוא לפחות $\frac{n}{2^{h+1}}$



הוכחה: באינדוקציה על h .



- בסיס: $h = 0$

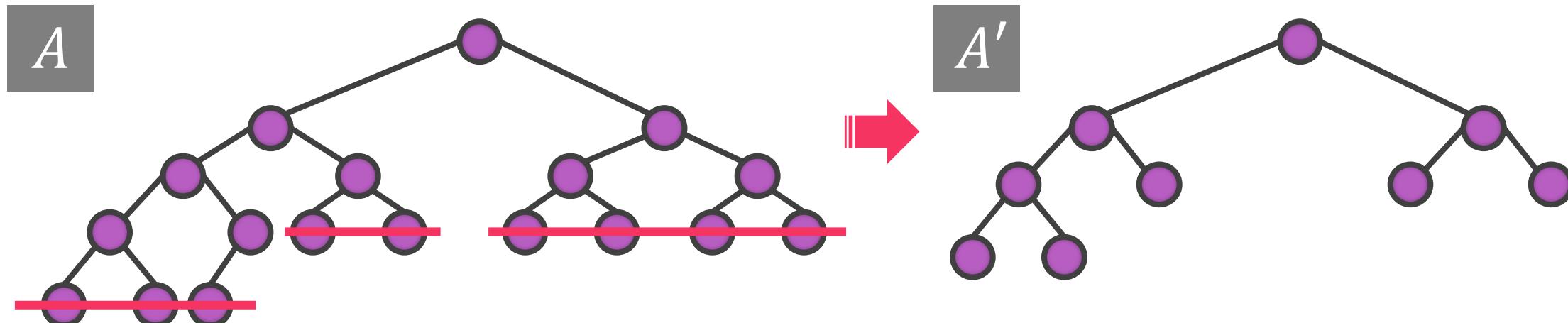
- בערימה יש $\left\lceil \frac{n}{2} \right\rceil$ עליים

זמן ריצה של Build-Max-Heap

הנחה: נכיח שהטענה נכון עבור צמתים בעלי גובה $0 \leq h$

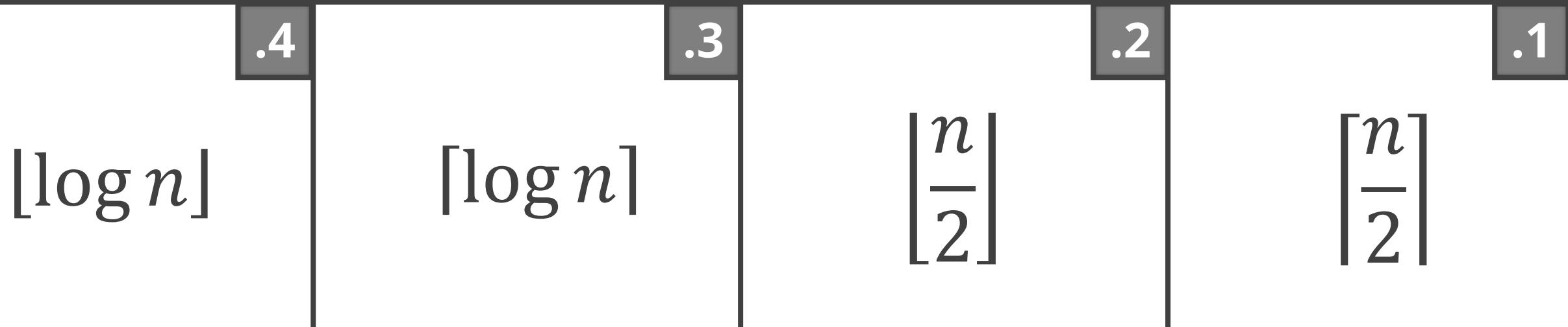
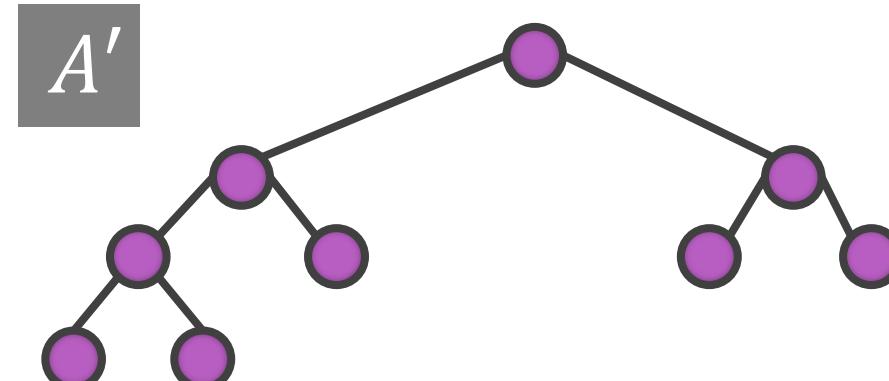
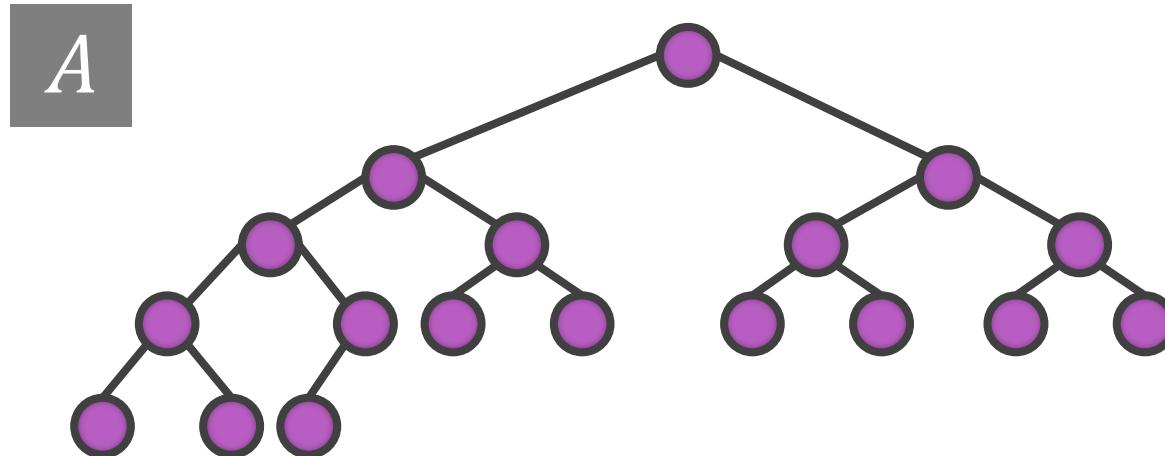


צעד: נוכיח את הטענה עבור צמתים בעלי גובה $h + 1$



צמתים בעלי גובה $h + 1$ ב- A הם צמתים בעלי גובה h ב- A'

אם מספר הצלחות בערימה A הוא n , מהו מספר הצלחות
בערימה $?A'$?





זמן ריצה של Build-Max-Heap

לפי הנחת האינדוקציה, מספר הצמתים בעלי גובה h ב- A הוא לפחות 2^h :





זמן ריצה של Build-Max-Heap

זמן ריצה הכולל של $O(n)$ *Build-Max-Heap* הוא

פעולות המוגדרות על עירימה

• **יצירת עירימה ממערך קלט בלתי ממוין** - *Build-Max-Heap*

- החזרת איבר בעל מפתח מקסימלי בעירימת מקסימום *Heap-Max*

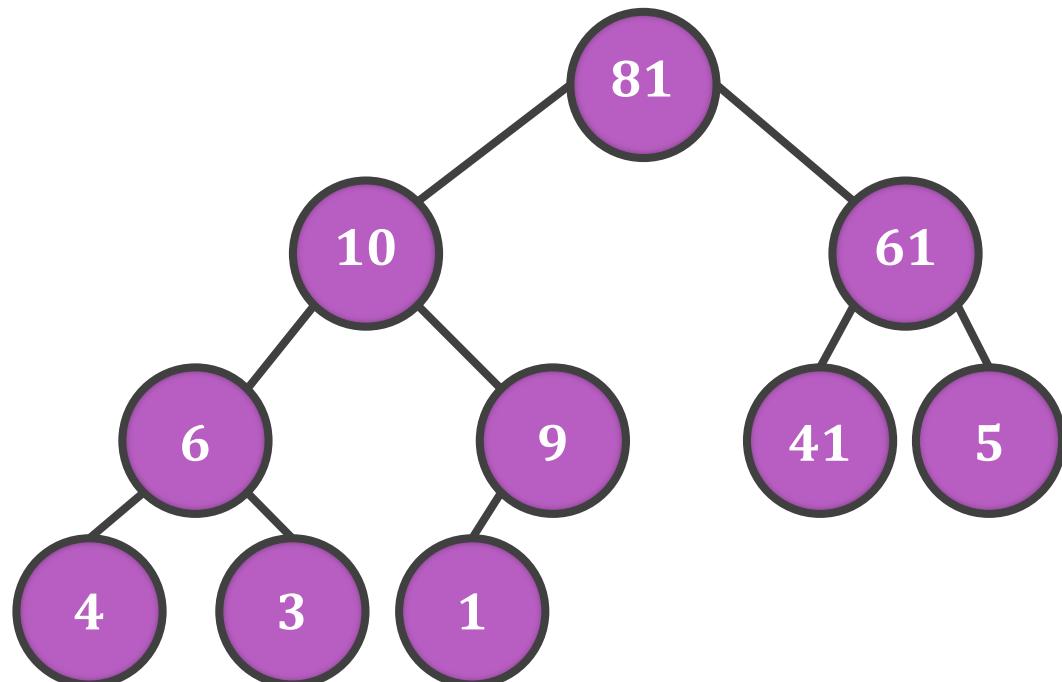
- מחיקת איבר בעל מפתח מקסימאלי בעירימת מקסימום *Heap-Extract-Max*

• **הגדלת מפתח** - *Heap-Increase-Key*

• **הכנסת איבר חדש** - *Max-Heap-Insert*

• **שמירה על תכונת העירימה** - *Max-Heapify*

Heap-Max

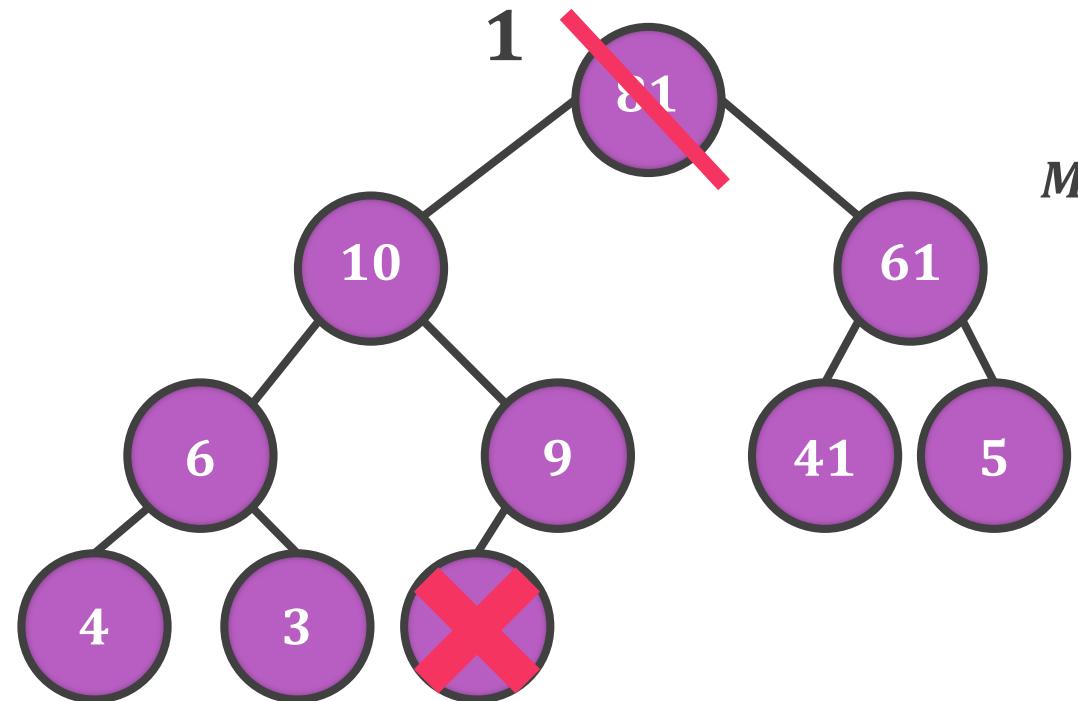


Heap – Max (A)

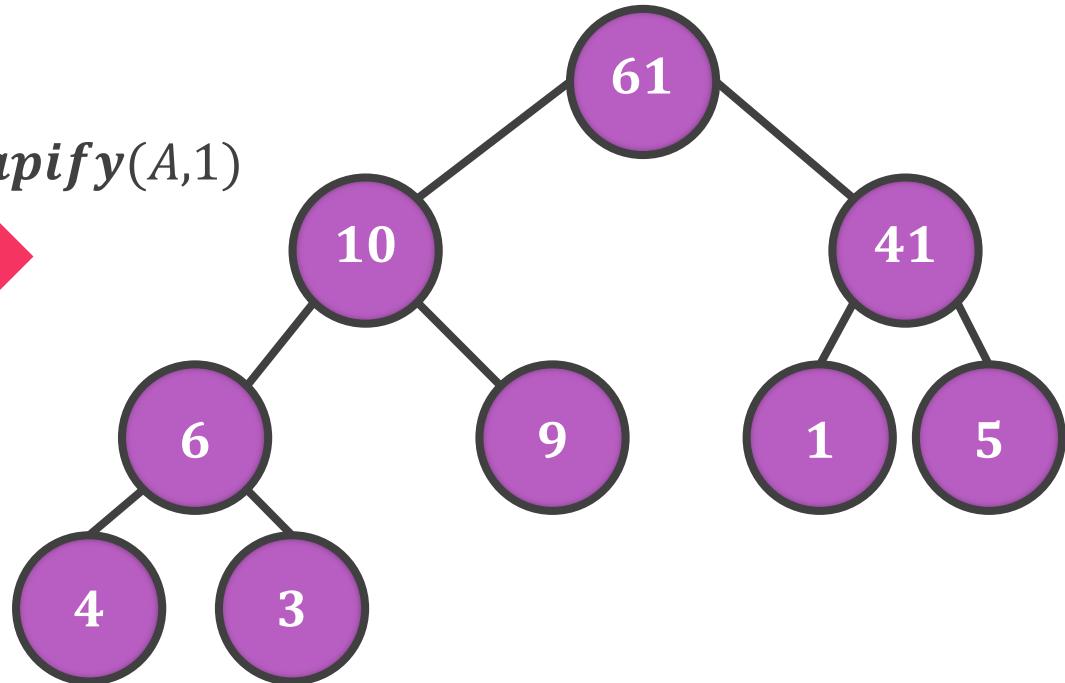
- 1 **if** $A.\text{heap_size} < 1$
- 2 **error** “*heap underflow*”
- 3 **return** $A[1]$

זמן ריצה: $O(1)$

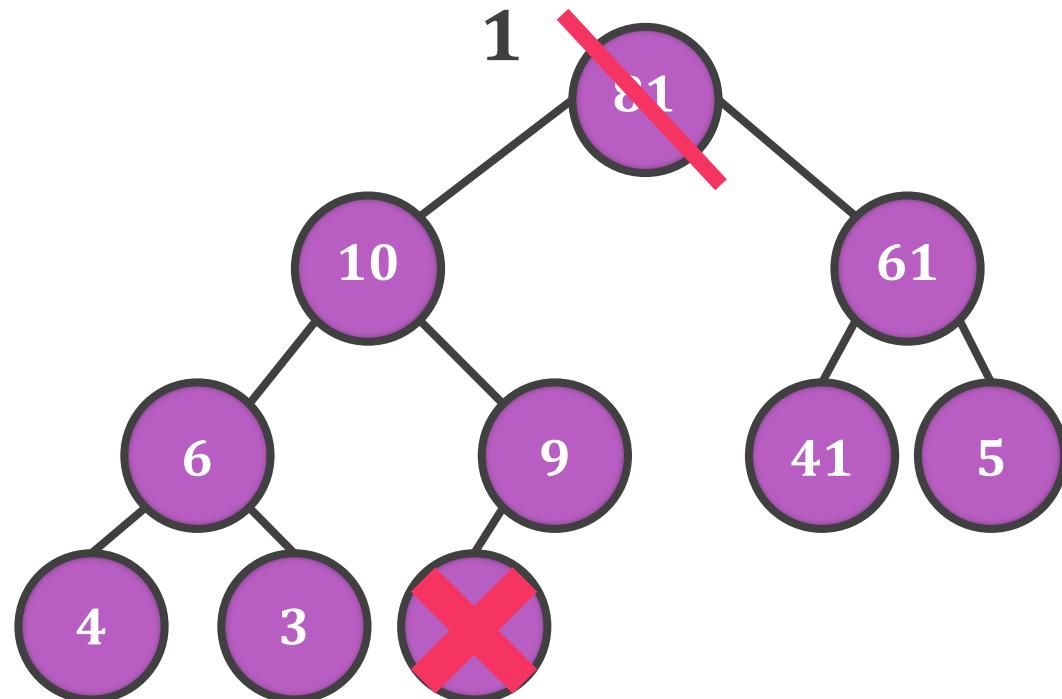
Heap-Extract-Max



Max-Heapify(A,1)



Heap-Extract-Max



Heap-Extract-Max(A)

- 1 **if** $A.\text{heap_size} < 1$
- 2 **error** “*heap underflow*”
- 3 $\text{max} \leftarrow A[1]$
- 4 $A[1] \leftarrow A[A.\text{heap_size}]$
- 5 $A.\text{heap_size} \leftarrow A.\text{heap_size} - 1$
- 6 **Max-Heapify**($A, 1$)
- 7 **return** max

זמן ריצה: $T(n) = O(\log n)$

פעולות המוגדרות על עירימה

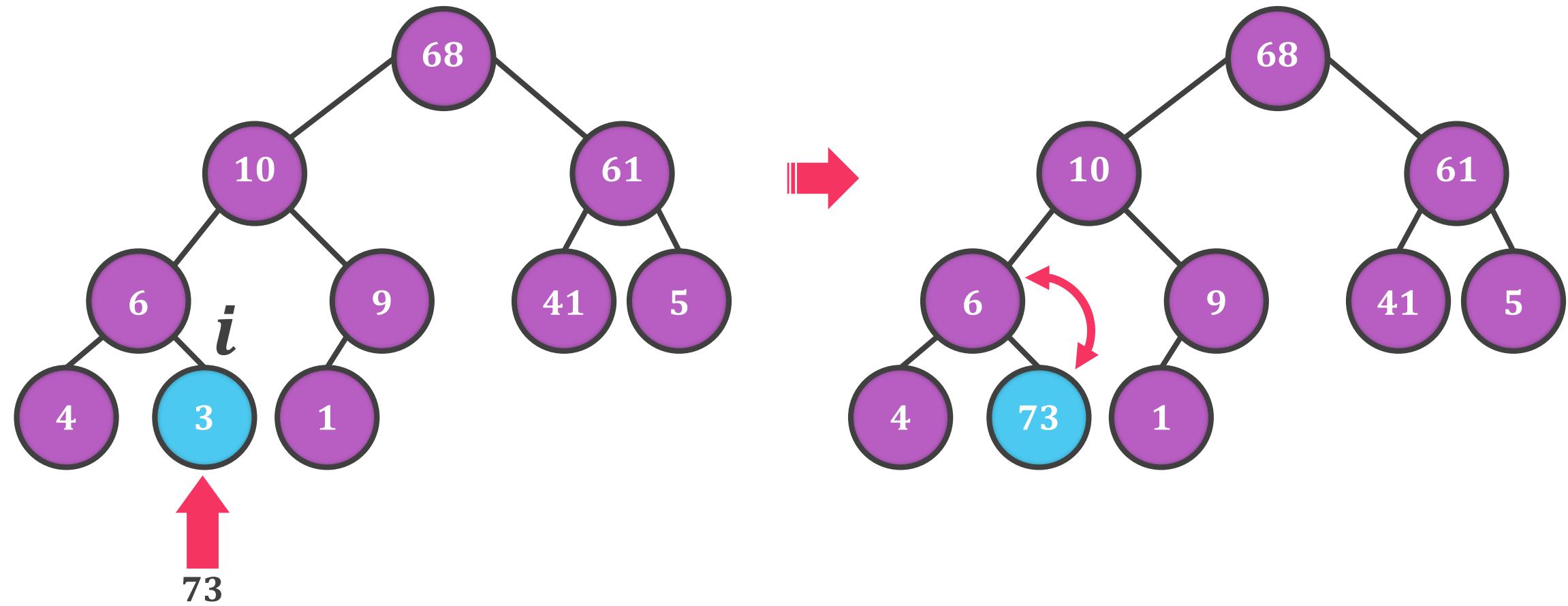
- **יצירת עירימה ממערך קלט בלתי ממוין** - *Build-Max-Heap*
- **החזרת איבר בעל מפתח מקסימלי בערימת מקסימים** - *Heap-Max*
- **מחיקת איבר בעל מפתח מקסימאלי בערימת מקסימים** - *Heap-Extract-Max*

- הגדלת מפתח - *Heap-Increase-Key*

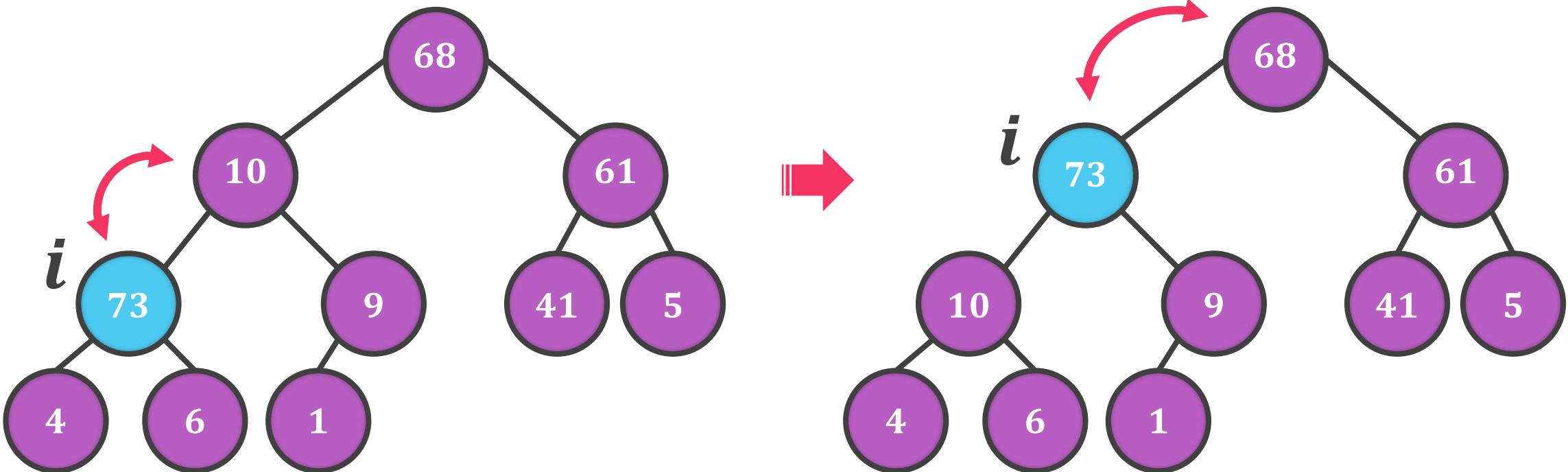
- הכנסת איבר חדש - *Max-Heap-Insert*

- שמירה על תכונת העירימה - *Max-Heapify*

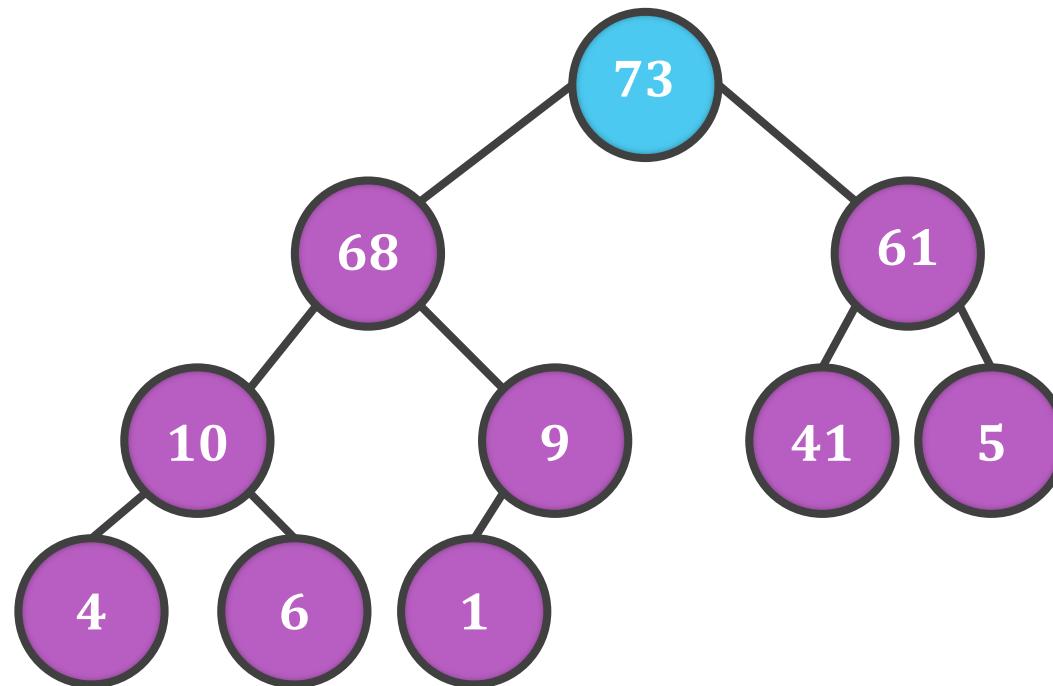
Heap-Increase-Key



Heap-Increase-Key



Heap-Increase-Key



Heap-Increase-Key

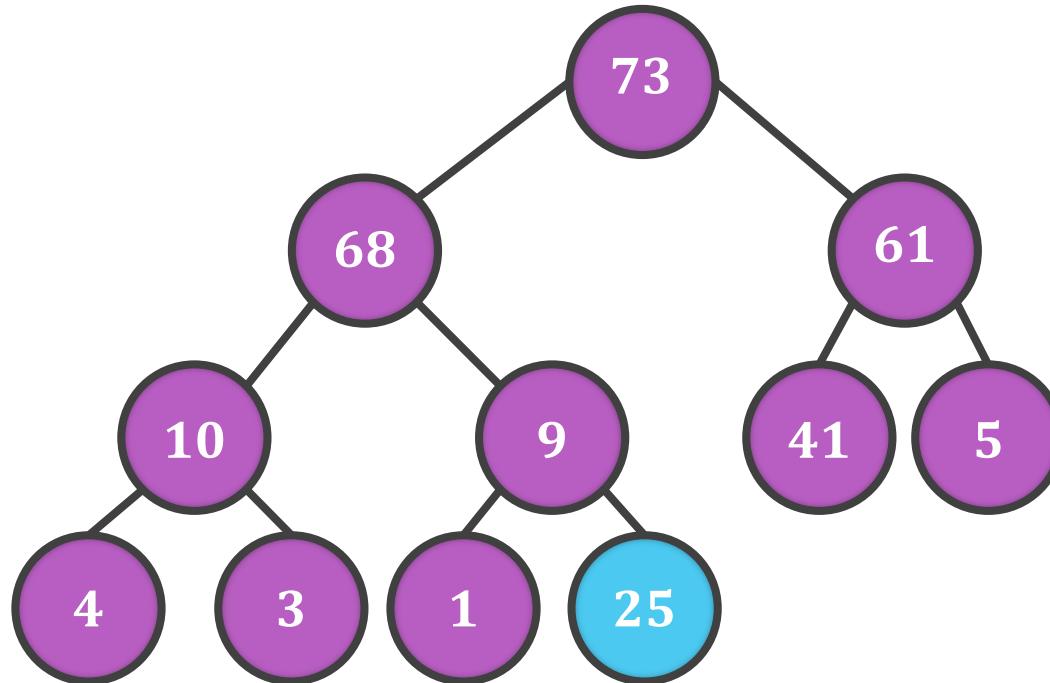
Heap-Increase-Key(A, i, key)

- 1 **if** $key < A[i]$
- 2 **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 *exchange* $A[i] \leftrightarrow A[\text{parent}(i)]$
- 6 $i \leftarrow \text{parent}(i)$

$T(n) = O(\log n)$

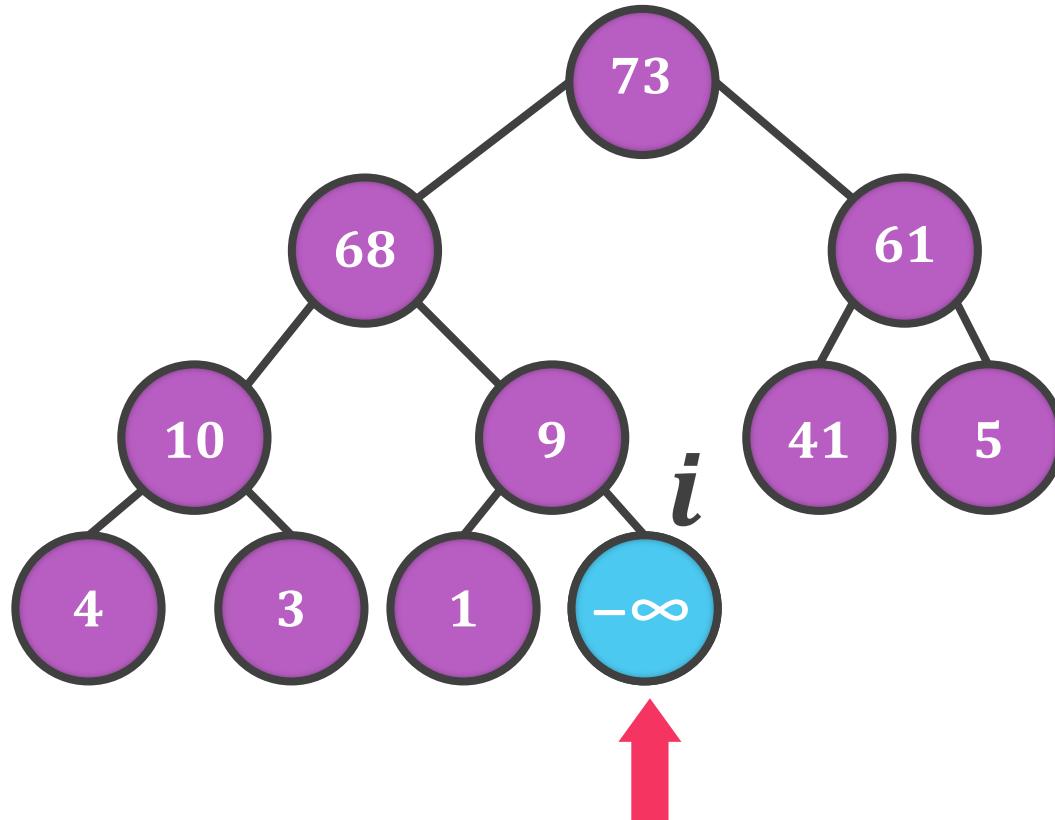
זמן ריצה:

Max-Heap-Insert



Max-Heap-Insert (A,25)

Max-Heap-Insert



Heap-Increase-Key($A, i, 25$)

Max-Heap-Insert

Max-Heap-Insert(A, key)

- 1 $A.heap_size \leftarrow A.heap_size + 1$
- 2 $A[A.heap_size] \leftarrow -\infty$
- 3 $Heap-Increase-Key(A, A.heap_size, key)$

$T(n) = O(\log n)$

זמן ריצה:

שימושים של ערימה

• מיון ערימה - *Heap – Sort*

• מימוש תור עדיפות

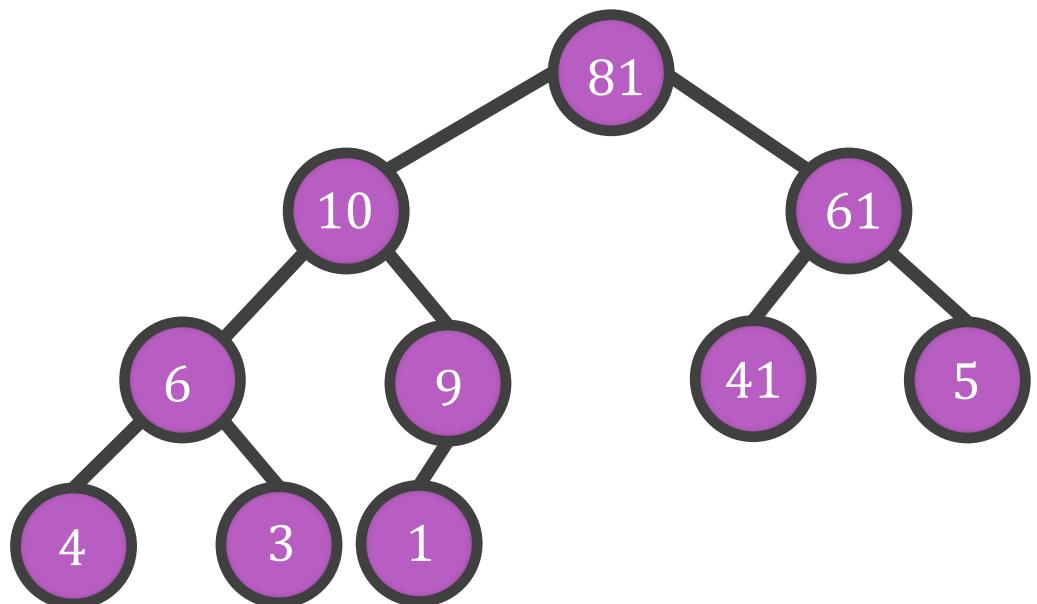
מיזן עירימה Heap-Sort

- **נתוק: מערך A לא ממוין**

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|-----|----|---|----|---|---|----|---|---|----|----|
| A | 10 | 4 | 61 | 3 | 1 | 41 | 5 | 6 | 18 | 19 |

- **המטרה: למיין את A**

מיון עריימה Heap-Sort



| | | | | | | | | | | |
|----------|----|----|----|---|---|----|---|---|---|----|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| <i>A</i> | 81 | 10 | 61 | 6 | 9 | 41 | 5 | 4 | 3 | 1 |

• **רעיון**

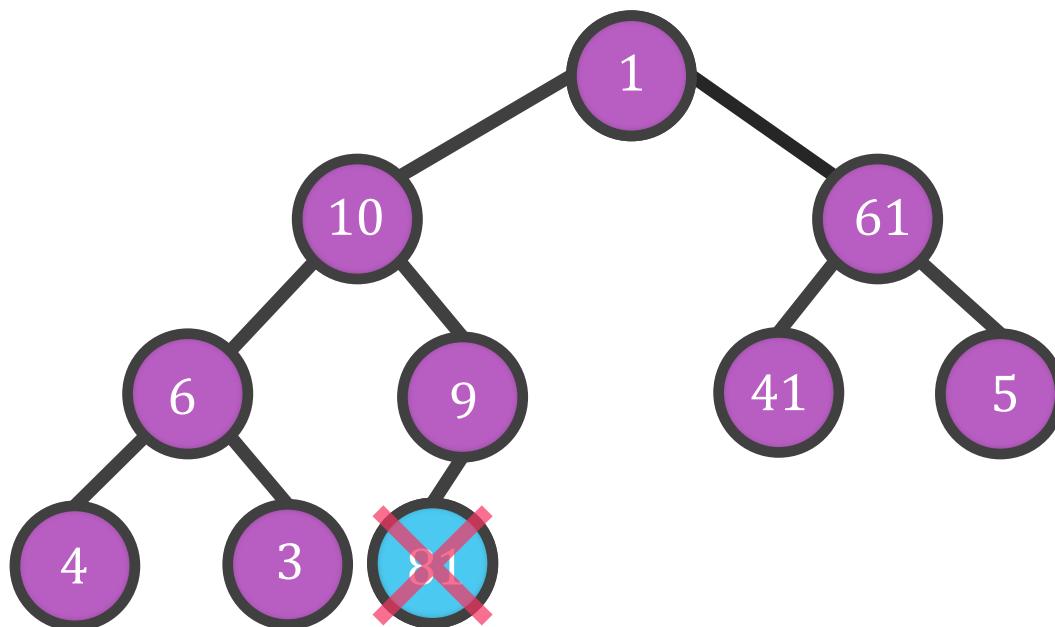
| | | | | | | | | | | |
|----------|----|---|----|---|---|----|---|---|----|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | |
| <i>A</i> | 10 | 4 | 61 | 3 | 1 | 41 | 5 | 6 | 81 | 9 |

- כל פעם נעביר את האיבר המxisimal
למקום הנכון

מיון ערימה Heap-Sort

Max – Heapify (A , 1)

כל פעם נעביר את האיבר המקסימלי למקום הנכון

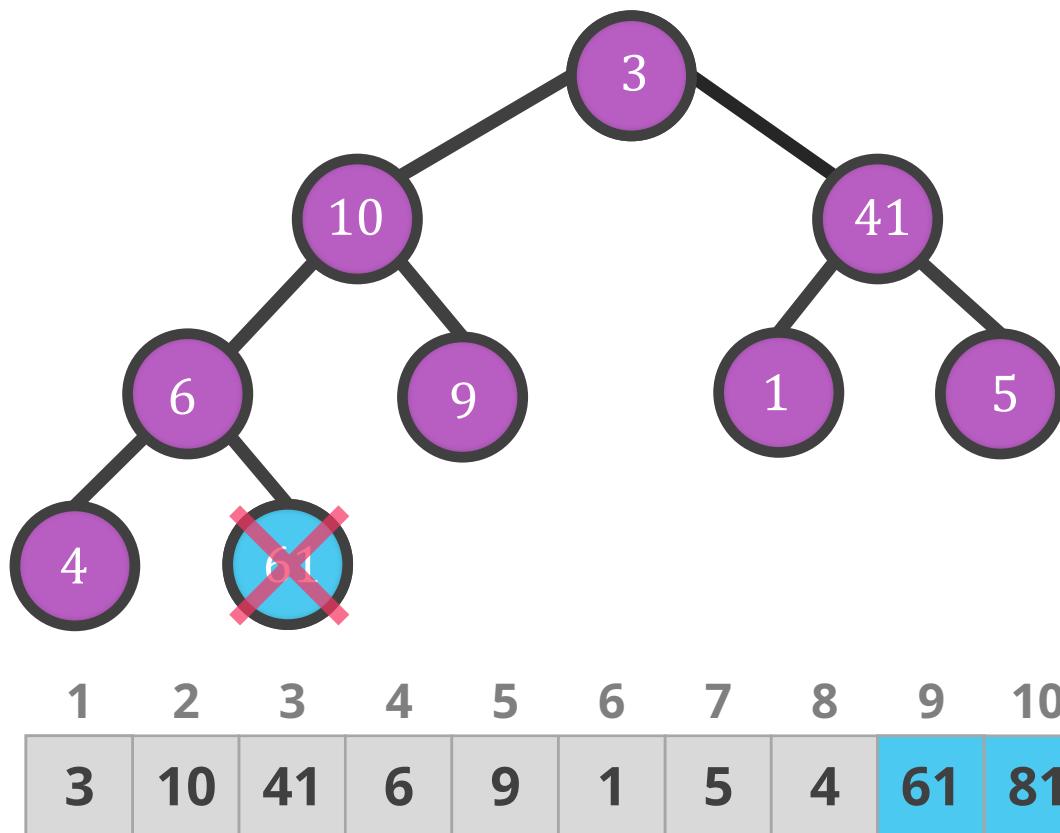


| | | | | | | | | | | |
|---|---|----|----|---|---|----|---|---|----|----|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | |
| A | 1 | 10 | 61 | 6 | 9 | 41 | 5 | 4 | 3 | 81 |

מיון ערימה Heap-Sort

Max – Heapify (A , 1)

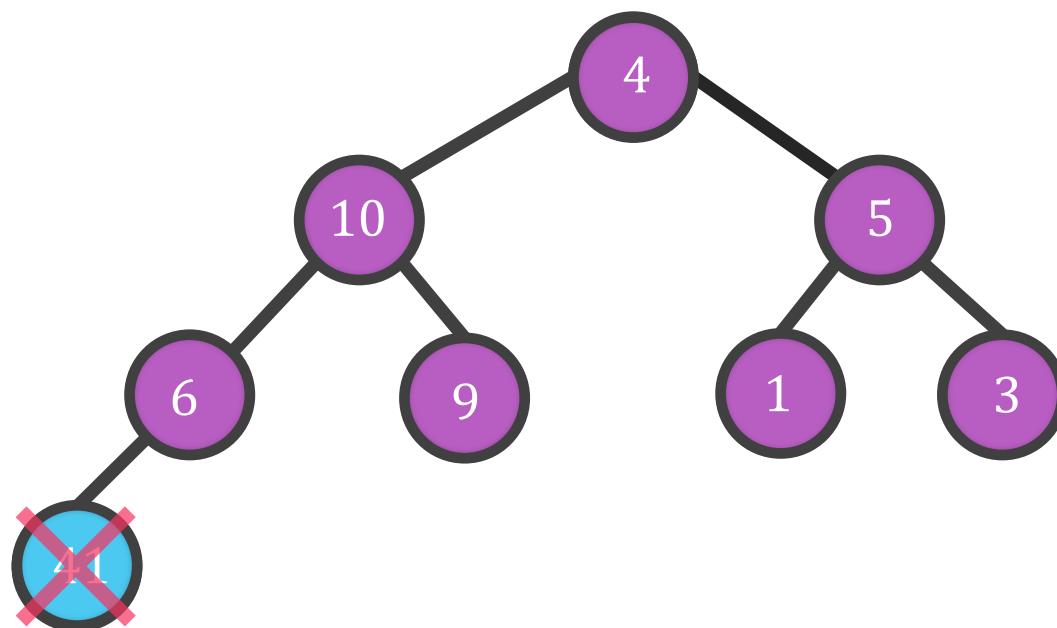
כל פעם נעביר את האיבר המקסימלי למקום הנכון



מיון ערימה Heap-Sort

Max – Heapify (A , 1)

כל פעם נעביר את האיבר המקסימלי למקום הנכון

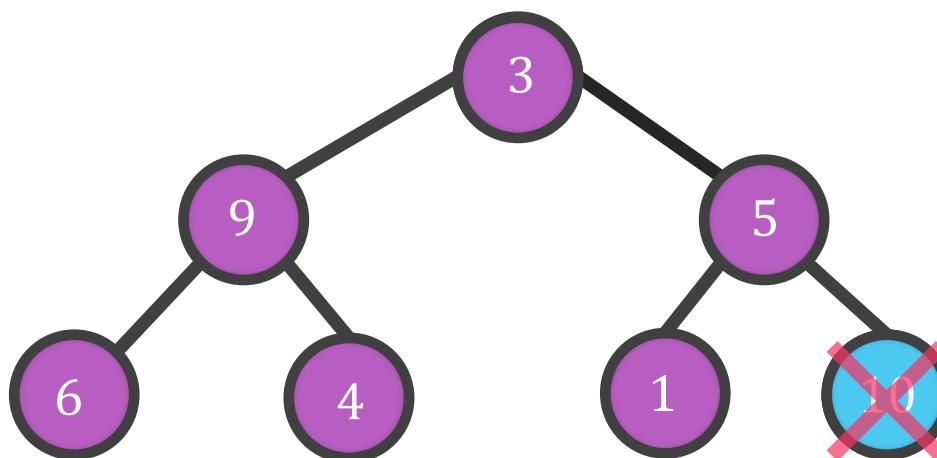


| | | | | | | | | | | |
|---|---|----|---|---|---|---|---|----|----|----|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | |
| A | 4 | 10 | 5 | 6 | 9 | 1 | 3 | 41 | 61 | 81 |

מיון ערימה Heap-Sort

Max – Heapify (A , 1)

כל פעם נעביר את האיבר המקסימלי למקום הנכון

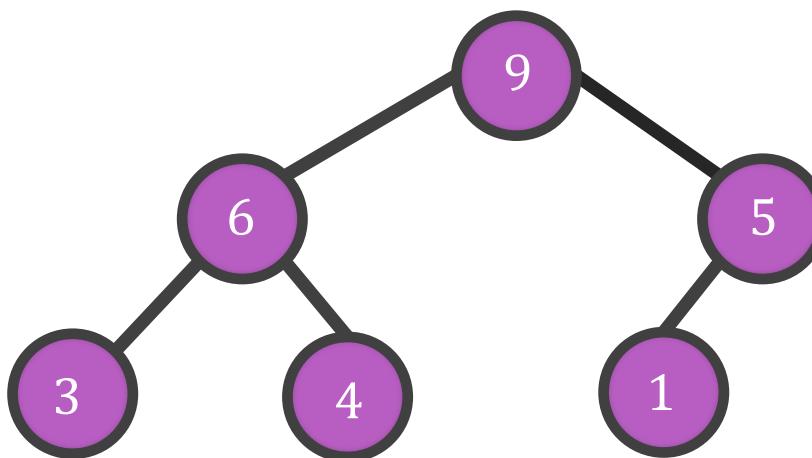


| | | | | | | | | | | |
|---|---|---|---|---|---|---|----|----|----|----|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| A | 3 | 9 | 5 | 6 | 4 | 1 | 10 | 41 | 61 | 81 |

מיון ערימה Heap-Sort

Max – Heapify (A , 1)

כל פעם נעביר את האיבר המקסימלי למקום הנכון

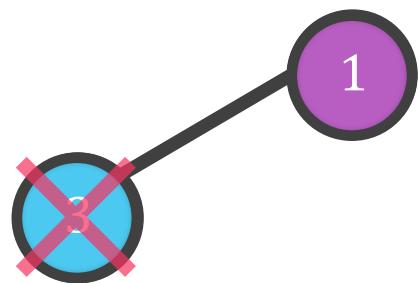


| | | | | | | | | | | |
|---|---|---|---|---|---|---|----|----|----|----|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| A | 9 | 6 | 5 | 3 | 4 | 1 | 10 | 41 | 61 | 81 |

מיון ערימה Heap-Sort

האייטרציה [האחרונה](#)

Max – Heapify (A , 1)



| | | | | | | | | | | |
|----------|---|---|---|---|---|---|----|----|----|----|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| <i>A</i> | 1 | 3 | 4 | 5 | 6 | 9 | 10 | 41 | 61 | 81 |

מיון ערימה Heap-Sort

המערך **אתחזין**

1

| | | | | | | | | | | |
|---|---|---|---|---|---|---|----|----|----|----|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| A | 1 | 3 | 4 | 5 | 6 | 9 | 10 | 41 | 61 | 81 |

Heap – Sort (A)

- 1** *Build – Max – Heap(A)*
- 2** *for i ← A.length downto 2*
- 3** *exchange A[1] ↔ A[i]*
- 4** *A.heap_size ← A.heap_size - 1*
- 5** *Max – Heapify(A, 1)*

מהו זמן ריצה של מיון עירימה במנוחים של O ?



תשובה:

$O(n \log n)$

Heap – Sort (A)

- 1 *Build – Max – Heap(A)*
- 2 *for i ← A.length downto 2*
- 3 *exchange A[1] ↔ A[i]*
- 4 *A.heap_size ← A.heap_size - 1*
- 5 *Max – Heapify(A, 1)*

זמן ריצה:

$$T(n) = O(n \log n)$$

שימושים של עירימה

• **מיון עירימה** - *Heap – Sort*

מימוש תור עדיפות

שימושים של ערים

- **שימוש תור עדיפויות**
- **תור עדיפויות** הוא מבנה נתוני מופשט לטיפול בקבוצה דינמית של איברים,
בה לכל איבר מצורף ערך הנקרא עדיפות (מפתח)
- **האיבר שנמחק ידוע מראש והוא האיבר בעל העדיפות הגבוהה ביותר**

Priority (key)

Satellite Data

+ TRIAGE



טור עדיפויות תומך בפעולות

- *MakeEmptyPriorityQueue*
- *EnqueuePriorityQueue*
- *ExtractMax*
- *Maximum*
- *IsEmptyPriorityQueue*
- *IncreaseKey*

במידה ולמימוש תור עדיפויות נשתמש במערך או בראשימה מתחומים לפני עדיפות, מה יהיה זמן הריצה של פעולה *EnqueuePriorityQueue*?



$O(\log n)$

.1

$O(n^2)$

.2

$O(n)$

.3

$O(n \log n)$

.4

**במידה ולמימוש תור עדיפות נשתמש במערך או בראשימה לא מתחייבים
לפי עדיפות, מה יהיה זמן הריצה של פועלות $ExtractMax$?**



$O(\log n)$

.1

$O(n^2)$

.2

$O(n)$

.3

$O(n \log n)$

.4

**במידה ולמימוש תור עדיפות נשתמש במערך או בראשימה לא מתחייבים
לפי עדיפות, מה יהיה זמן הריצה של פועלות $ExtractMax$?**



$O(\log n)$

.1

$O(n^2)$

.2

$O(n)$

.3

$O(n \log n)$

.4

טור עדיפות תומך בפעולות

- *MakeEmptyPriorityQueue*

- *EnqueuePriorityQueue*

 $O(\log n)$

- *ExtractMax*

 $O(\log n)$

- *Maximum*

- *IsEmptyPriorityQueue*

- *IncreaseKey*

סיכון ערימה

Heap