מבני נתונים

תרגול 7 – ערמה

צבי מינץ – zvimints@gmail.com

תרגולים קודמים

עץ בינארי (Binary Tree) – עץ אשר לכל קודקוד יש לכל היותר 2 בנים

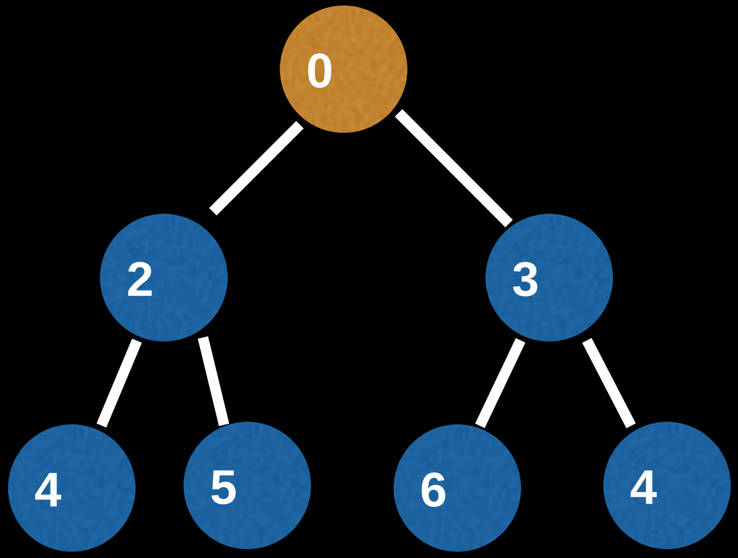
עץ בינארי שלם (Complete Binary Tree)

עץ בינארי הוא עץ בינארי שלם אם כל הרמות מלאים בעלים חוץ מהרמה האחרונה

כאשר כל העלים ברמה האחרונה הם מצד שמאל ככל היותר

```
              18
         /          \
       15             30
      /  \           /  \
    40      50     100    40
   /  \    /
  8    7  9
```

Binary Heap

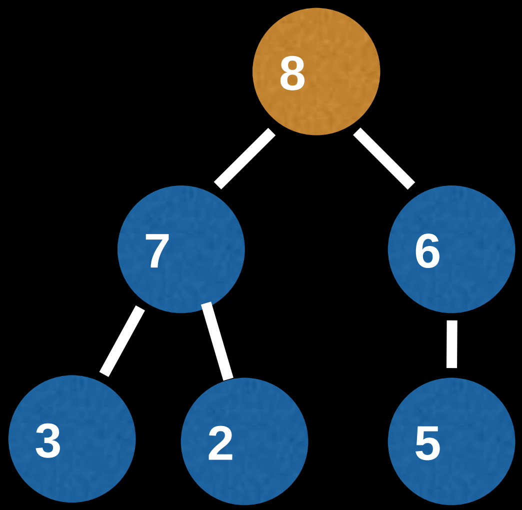


Min Heap

כל האיברים בעץ ניתנים להשוואה (Comparable) ושורש העץ הוא תמיד **המינמלי**

הערך של כל קודקוד **קטן או שווה** מהערך של בניו - **heap property**

עץ בינארי **כמעט שלם**



Max Heap

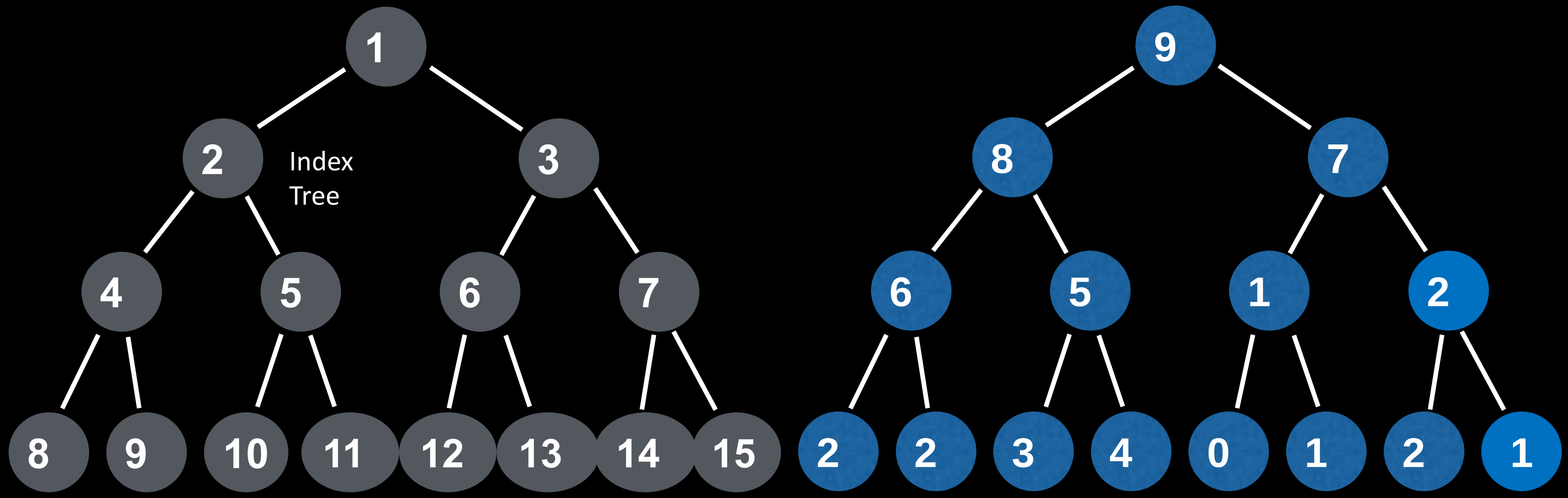
כל האיברים בעץ ניתנים להשוואה (Comparable) ושורש העץ הוא תמיד **המקסימלי**

הערך של כל קודקוד **גדול או שווה** מהערך של בניו - **heap property**

עץ בינארי **כמעט שלם**

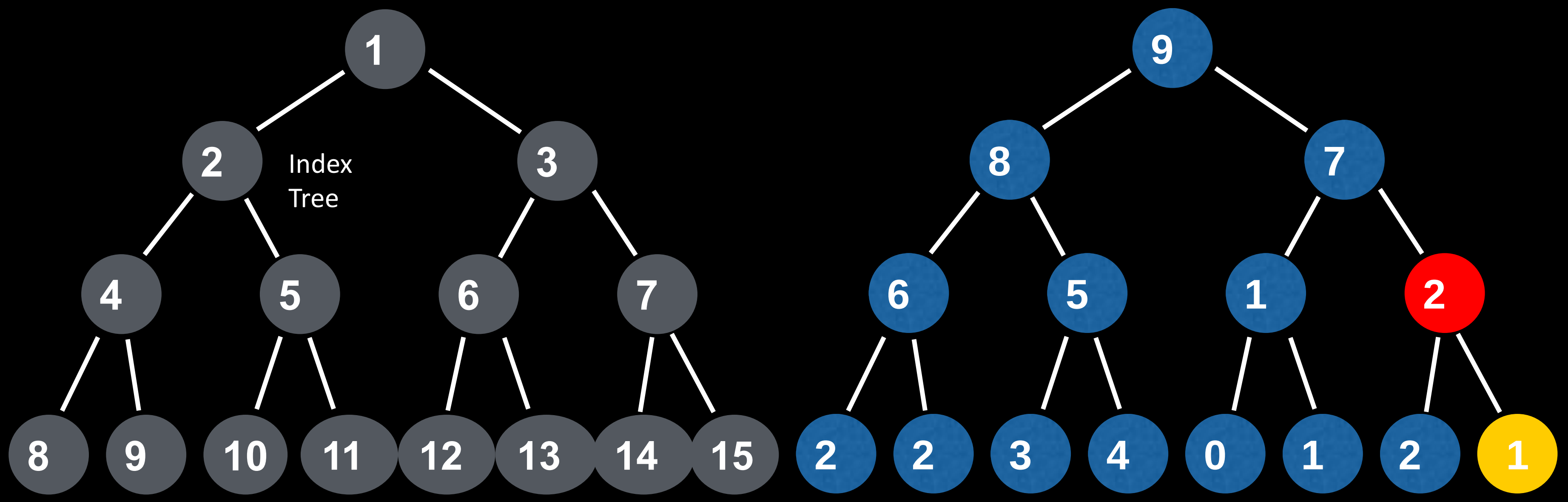
Binary Heap Representation

value	9	8	7	6	5	1	2	2	2	3	4	0	1	2	1
index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15



Binary Heap Representation

value	9	8	7	6	5	1	2	2	2	3	4	0	1	2	1
index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15



Binary Heap Representation

value	9	8	7	6	5	1	2	2	2	3	4	0	1	2	1
index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

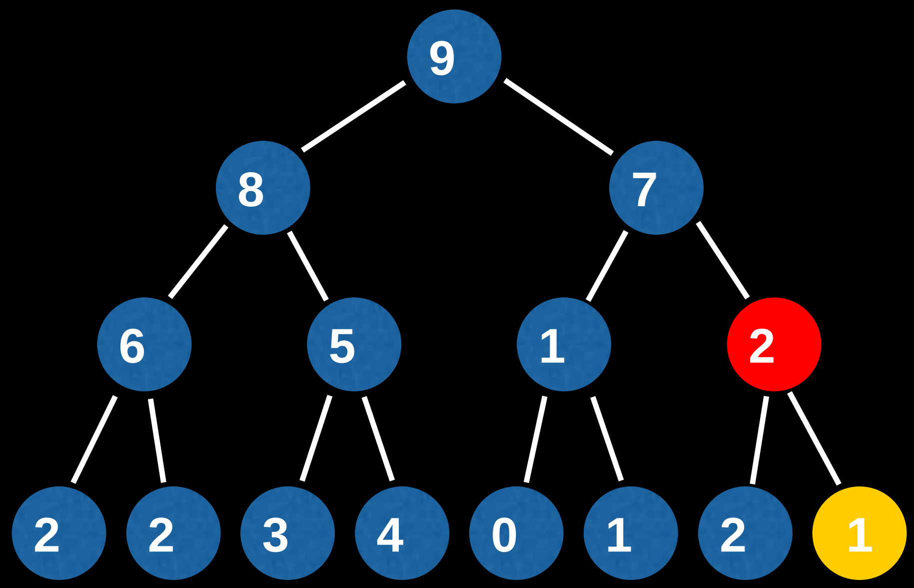
Let *i* be a node index

(one based)

Left child index: 2i

Right child index: 2i + 1

parent of i index: $\left\lfloor \frac{i}{2} \right\rfloor$



Binary Heap Representation

value	9	8	7	6	5	1	2	2	2	3	4	0	1	2	1
index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

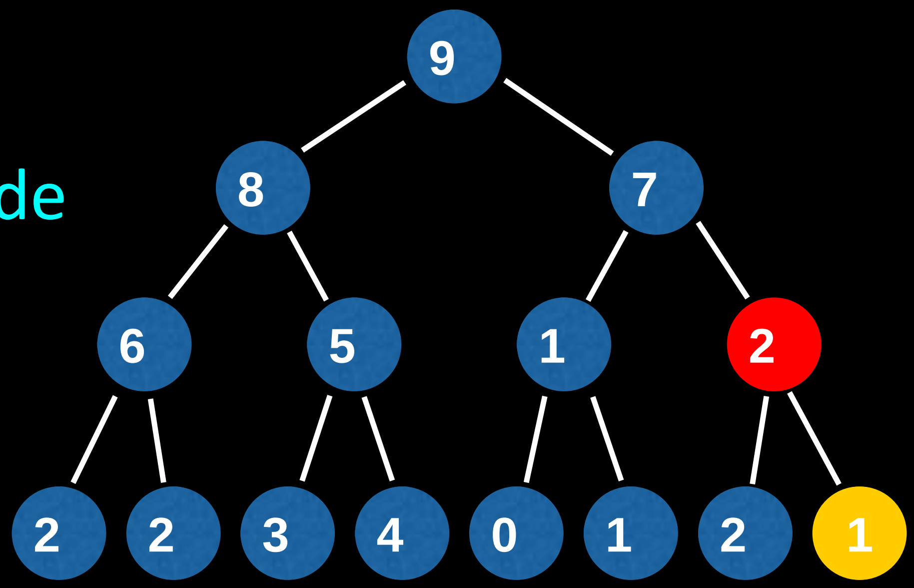
Let *i* be a node index

(zero based) → A[0] Returns the root node

Left child index: $2i + 1$

Right child index: $2i + 2$

parent of i index: $\left\lfloor \frac{i-1}{2} \right\rfloor$



Binary Heap Representation

value	9	8	7	6	5	1	2	2	2	3	4	0	1	2	1
index	1	3	3	4	5	6	7	8	9	10	11	12	13	14	15

גישה נוספת (נוחות)

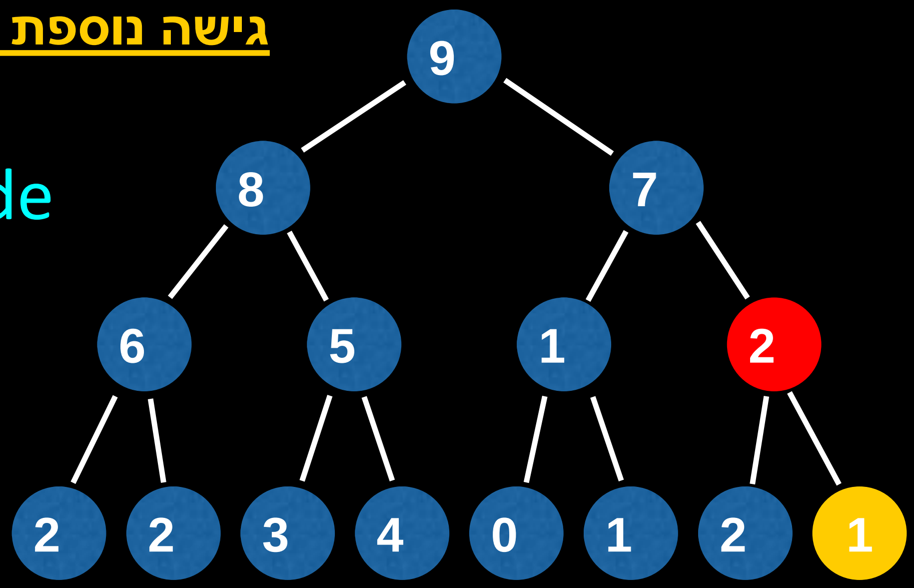
Let *i* be a node index

(one based) → A[1] Returns the root node

Left child index: $2i$

Right child index: $2i + 1$

parent of i index: $\left\lfloor \frac{i}{2} \right\rfloor$



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

```
// Traverse up and fix violated property
private void SwapUp(int index)

// A recursive function to max heapify the given
// subtree. This function assumes that the left and
// right subtrees are already heapified, we only need
// to fix the root.
public void Heapify(int i)

// Function to build a Max-Heap from the given array
public void build_heap(int[] A)

// Function to sort a given array
public void heapsort(int[] A)
```

Tim Heap

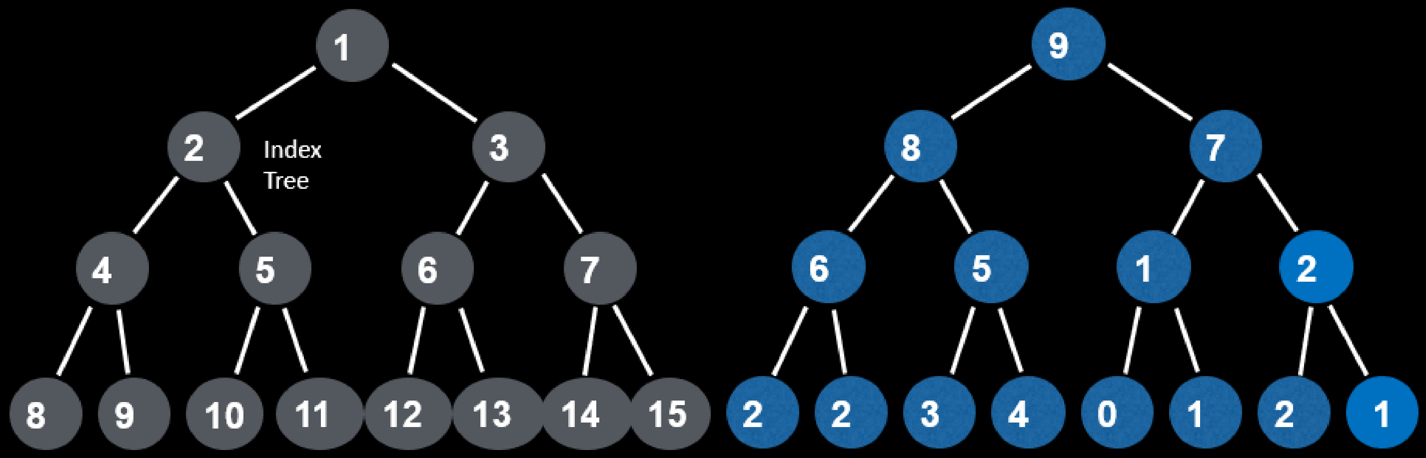
value	9	8	7	6	5	1	2	2	2	3	4	0	1	2	1
index	1	3	3	4	5	6	7	8	9	10	11	12	13	14	15

Get Max

- The

Remove

- The time to maint operations.



Remove (Not Max/Min) ?

Time Complexity of Binary Heap

Get Max or Min Element

- The Time Complexity of this operation is $O(1)$.

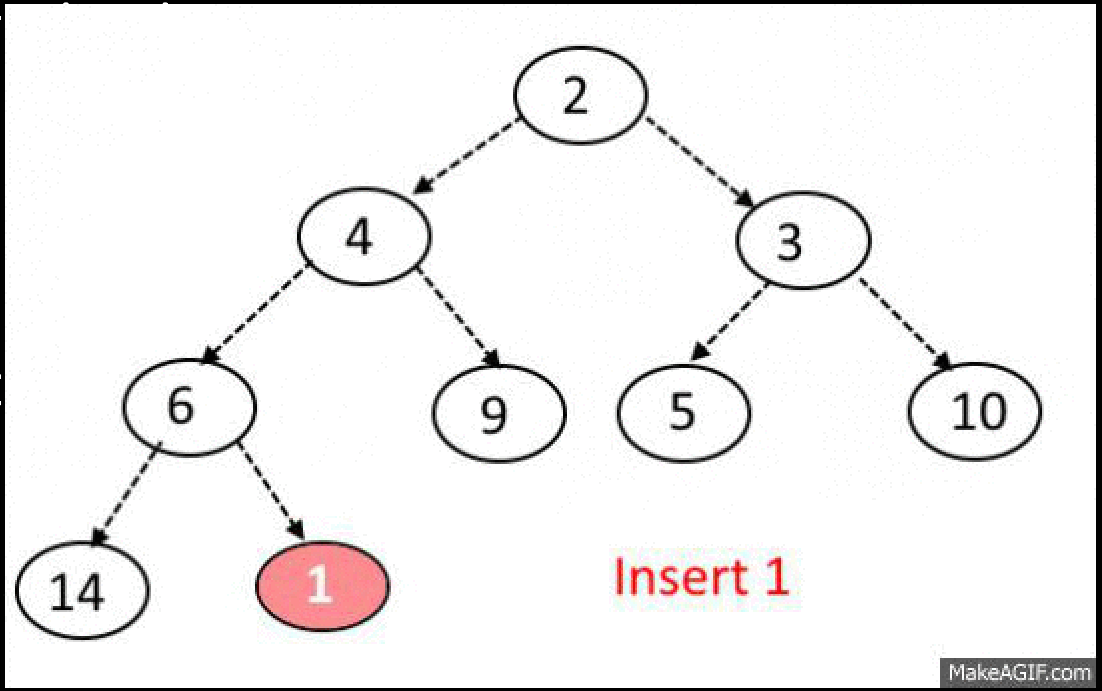
Remove Max or Min Element

- The time complexity of this operation is $O(logn)$ because we need to maintain the max/mix at their root node, wh
operations.

Remove (Not Max/Min) ?

Insert an Element

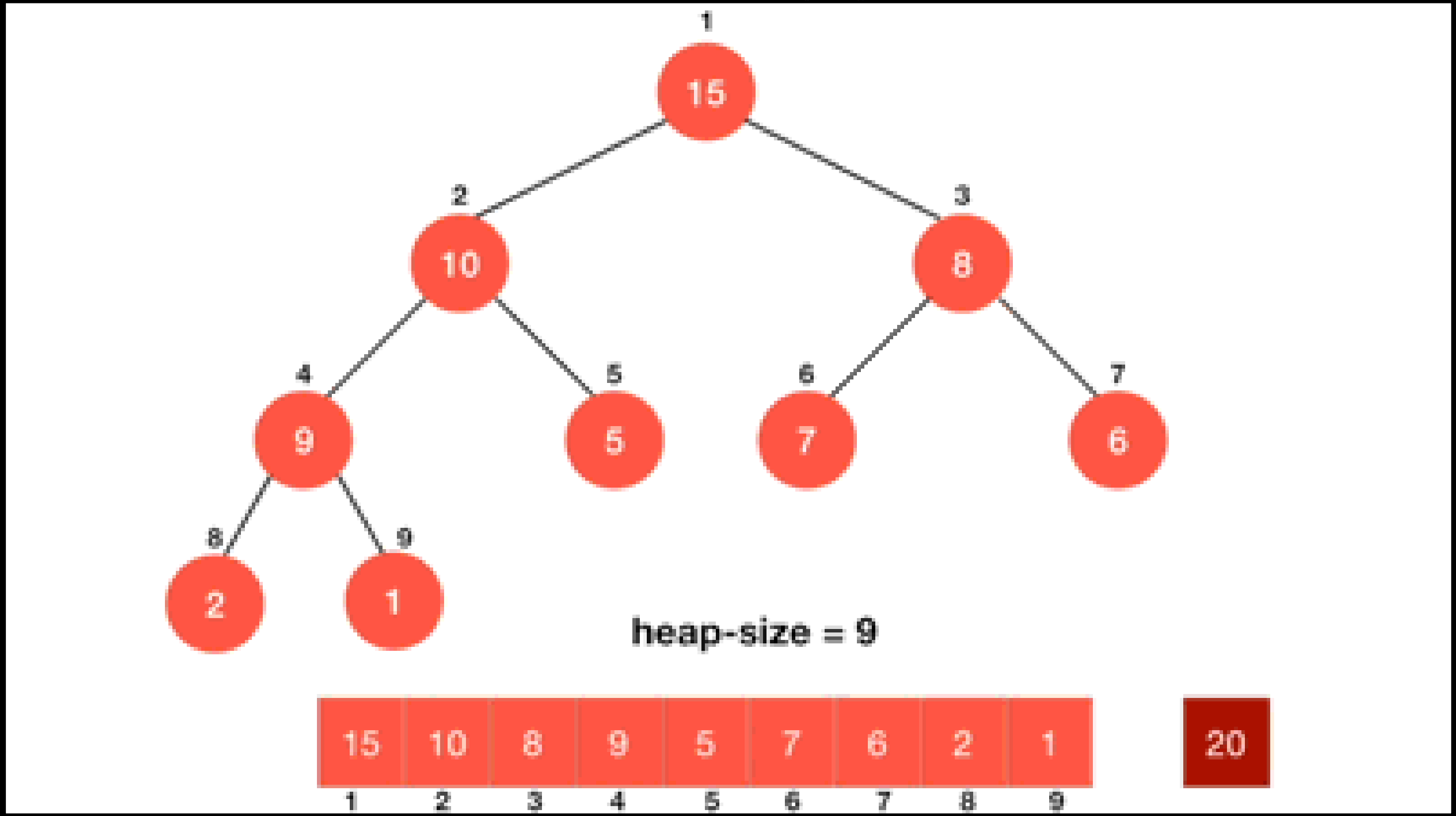
- Time Complexity of this operation is $O(logn)$ b
value at the end of the tree and traverse up to
property of min/max heap.



Time Complexity of Binary Heap

Build Heap - $O(n)$

HeapSort - $O(\log n)$ HeapSort.java



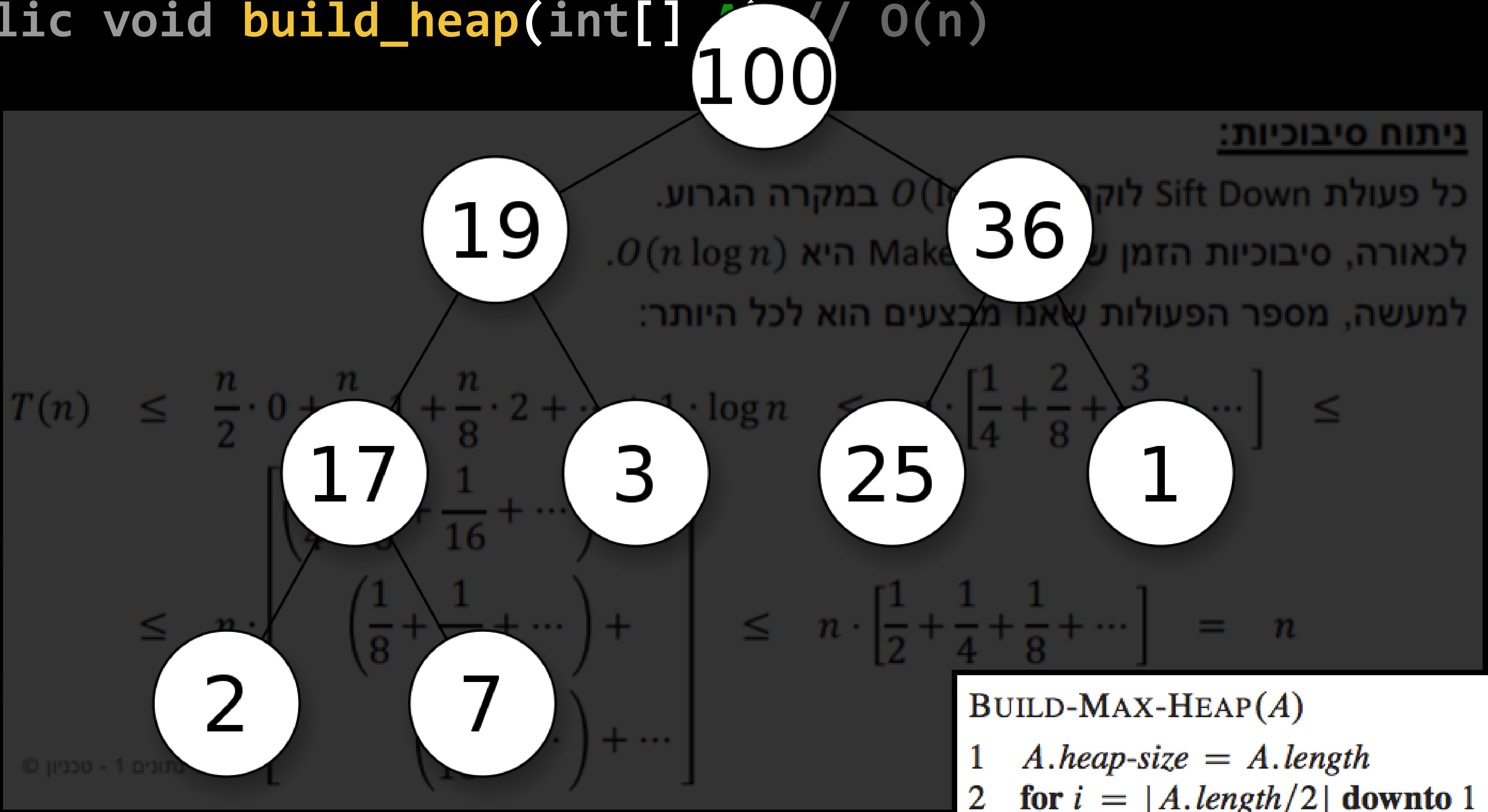
BUILD-MAX-HEAP(A) one based

```
A.heap-size = A.length
for i = ⌊A.length/2⌋ downto 1
    MAX-HEAPIFY(A, i)
```

HEAPSORT(A) one based

```
BUILD-MAX-HEAP(A)
for i = A.length downto 2
    exchange A[1] with A[i]
    A.heap-size = A.heap-size − 1
    MAX-HEAPIFY(A, 1)
```

```
public void build_heap(int[] A) // O(n)
```



BUILD-MAX-HEAP(A)

```
A.heap-size = A.length
for i = ⌊A.length/2⌋ downto 1
    MAX-HEAPIFY(A, i)
```

```
public void build_heap(int[] A) // O(n)
```

ניתוח סיבוכיות:

כל פעולת Sift Down לוקחת $O(\log n)$ במקרה הגרוע.

לכאורה, סיבוכיות הזמן של Make Heap היא $O(n \log n)$.

למעשה, מספר הפעולות שאנו מבצעים הוא לכל היותר:

$$
\begin{aligned}
T(n) \;&\le\; \frac{n}{2}\cdot 0 + \frac{n}{4}\cdot 1 + \frac{n}{8}\cdot 2 + \cdots + 1\cdot \log n \;\le\; n\cdot\left[\frac{1}{4}+\frac{2}{8}+\frac{3}{16}+\cdots\right] \;\le \\
&\le\; n\cdot\left[\begin{array}{c}\left(\frac{1}{4}+\frac{1}{8}+\frac{1}{16}+\cdots\right)+\\ \left(\frac{1}{8}+\frac{1}{16}+\cdots\right)+\\ \left(\frac{1}{16}+\cdots\right)+\cdots\end{array}\right] \;\le\; n\cdot\left[\frac{1}{2}+\frac{1}{4}+\frac{1}{8}+\cdots\right] \;=\; n
\end{aligned}
$$

תרגיל בית

one based

```
BUILD-MAX-HEAP(A)
  A.heap-size = A.length
  for i = ⌊A.length/2⌋ downto 1
      MAX-HEAPIFY(A, i)
```

מימוש MaxHeap.java

```
public class MaxHeap
{
```

שאלה 1 (20 נקודות). כתוב מחלקה MaxHeap שמהווה עץ ערמה של מספרים שלמים, ממומש על ידי מערך פנימי.

1. הגדר את המחלקה ואת השדות (המשתנים) שלה.
2. הוסף בנאי שמקבל גודל למערך הפנימי: **public MaxHeap(int capacity)**
3. הוסף פונקציה להכנסת איבר: **public boolean add(int x)**
4. הוסף פונקציה שמחזירה ומוחקת את האיבר הגדול באוסף: **public int remove()**

עבודה עצמית (One Based) (20 דקות)

```
    ...
}
```

שאלה 1
נתון מערך של מספרים שלמים: 12, 19, 10, 4, 23, 7, 45, 8, 15.
סרטט Max-Heap עבור המספרים האלה.

שאלה 2 סרטט Min-Heap עבור המספרים של שאלה 1.

Max Heap

http://btv.melezinek.cz/binary-heap.html

Min Heap

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

שאלה 3

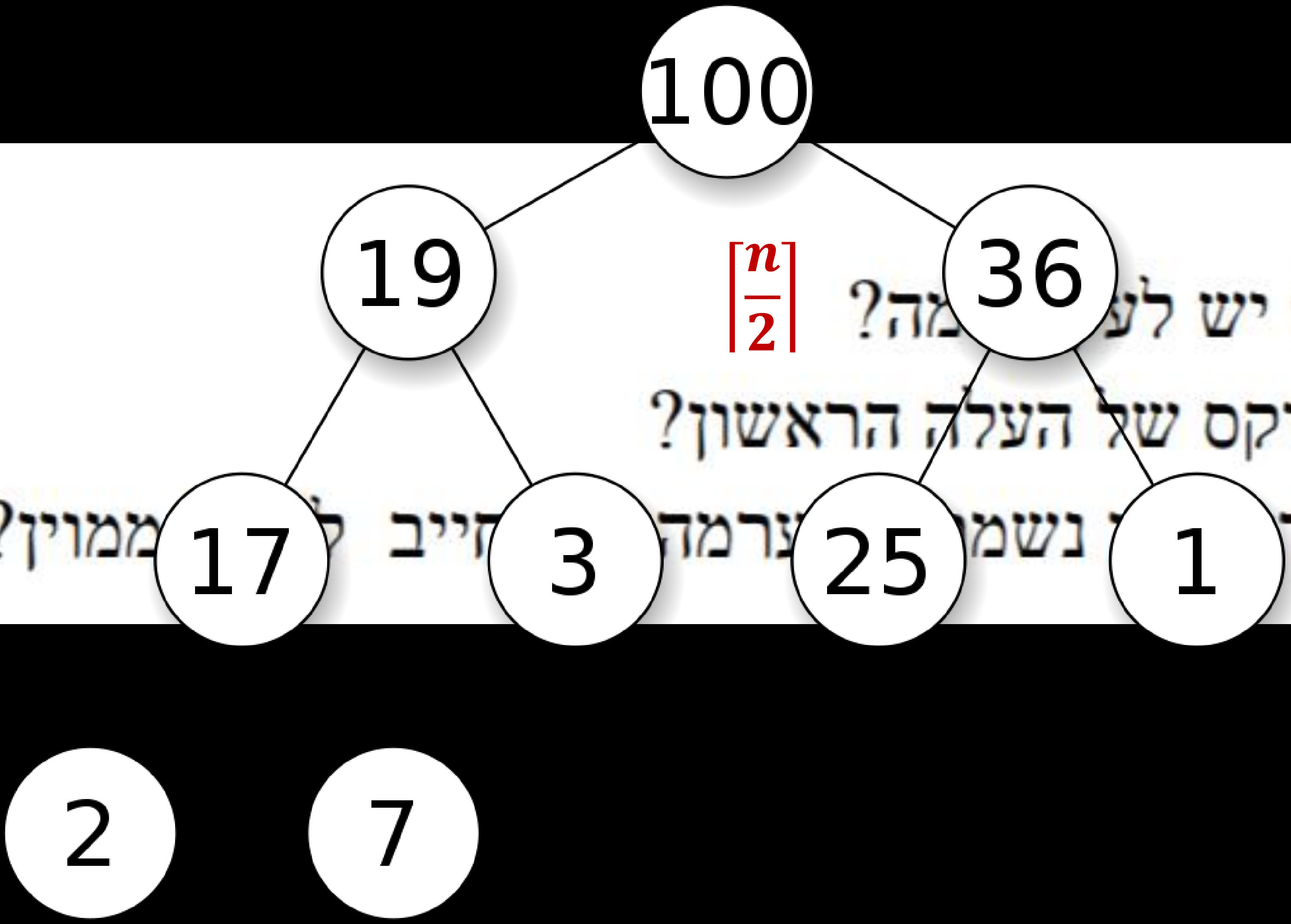
מאינדקס 1

א) כמה עלים יש לע... מה?

$\left\lceil \frac{n}{2} \right\rceil$

ב) מהו האינדקס של העלה הראשון?

ג) האם המער... נשמ... ערמה חייב ל... ממוין?



אוניברסיטת אריאל בשומרון

שאלה 3

מאינדקס 0

א) כמה עלים יש לעץ ערמה? $\left\lceil \frac{n}{2} \right\rceil$

ב) מהו האינדקס של העלה הראשון?

ג) האם המערך שבו נשמר עץ ערמה הוא חייב להיות ממוין?

zvimints@gmail.com – צבי מינץ

מאינדקס 1

שאלה 3

א) כמה עלים יש לעץ ערמה? $\left\lceil \frac{n}{2} \right\rceil$

ב) מהו האינדקס של העלה הראשון? $\left\lfloor \frac{n}{2} \right\rfloor + 1$

ג) האם המערך שבו נשמר עץ ערמה הוא חייב להיות ממוין?

אוניברסיטת
אריאל
בשומרון

שאלה 3

מאינדקס 0

א) כמה עלים יש לעץ ערמה? $\left\lceil \frac{n}{2} \right\rceil$

ב) מהו האינדקס של העלה הראשון? $\left\lfloor \frac{n}{2} \right\rfloor$

ג) האם המערך שבו נשמר עץ ערמה הוא חייב להיות ממוין?

מאינדקס 0

שאלה 3

א) כמה עלים יש לעץ ערמה? $\left\lceil \frac{n}{2} \right\rceil$

ב) מהו האינדקס של העלה הראשון? $\left\lfloor \frac{n}{2} \right\rfloor$

ג) האם המערך שבו נשמר עץ ערמה הוא חייב להיות ממוין?

מאינדקס 1

מאינדקס 0

שאלה 3

א) כמה עלים יש לעץ ערמה? $\left\lceil \frac{n}{2} \right\rceil$ $\left\lceil \frac{n}{2} \right\rceil$

ב) מהו האינדקס של העלה הראשון? $\left\lceil \frac{n}{2} \right\rceil$ $\left\lfloor \frac{n}{2} \right\rfloor$

ג) האם המערך שבו נשמר עץ ערמה הוא חייב להיות ממוין? לא לא

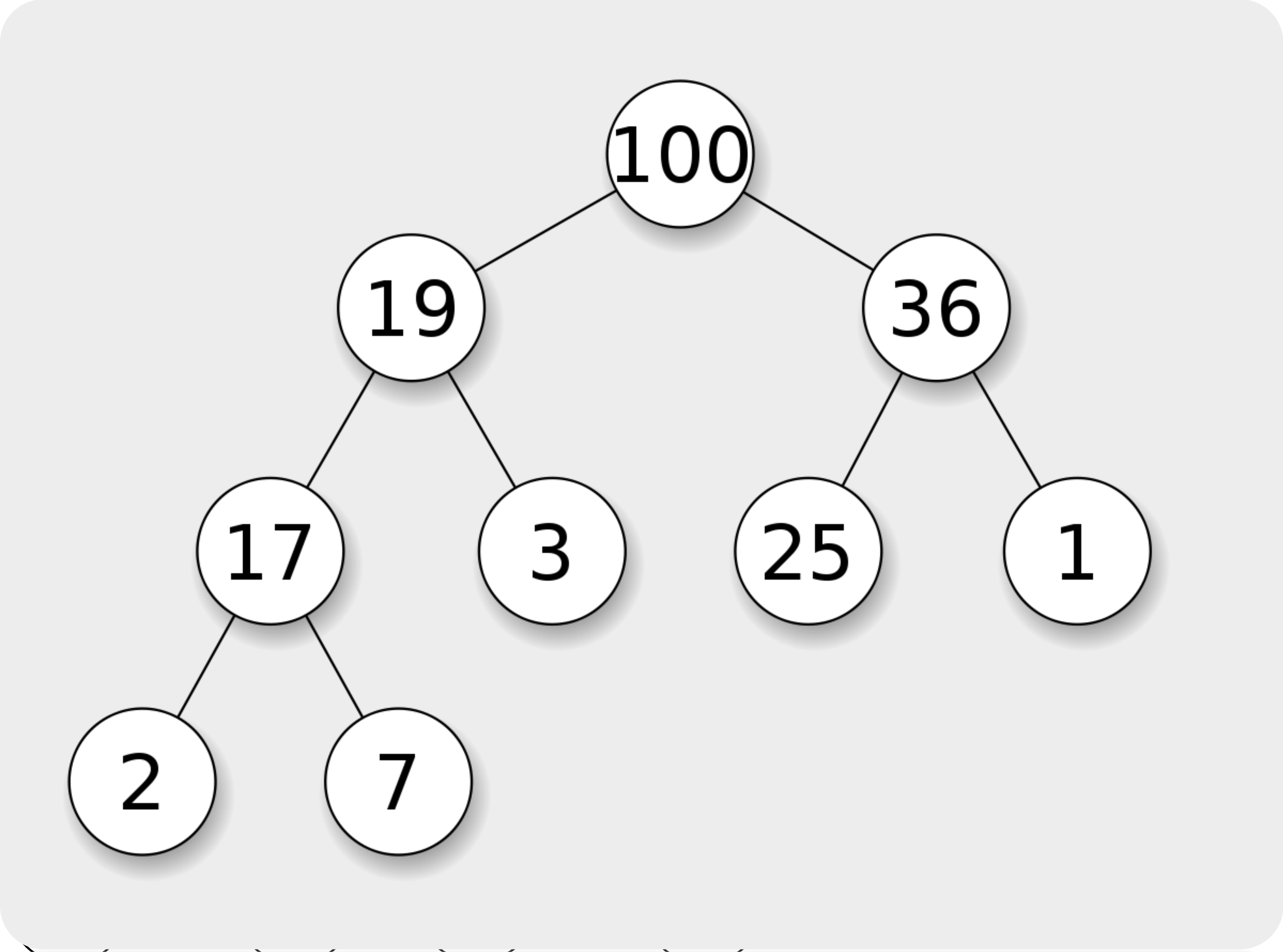
value	9	8	7	6	5	1	2	2	2	3	4	0	1	2	1
index	1	3	3	4	5	6	7	8	9	10	11	12	13	14	15

- בהינתן ערימת מקסימום H,
מה סיבוכיות הזמן למציאת 3 האיברים הגדולים בH?

רעיונות?

פתו

- לוי
הא



- מצא את זמני הריצה של מציאת האיבר ה- i הקטן ביותר בקבוצה בגודל n בשימוש במבנים הבאים:
  - מיון
  - תור עדיפות (ערימה)
- פתרון:
- מיון:
  - מיין את כל האיברים בעזרת אלגוריתם מיון בזמן $O(n\log n)$.
  - מצא את האיבר ה-i הקטן ביותר בזמן $O(1)$.
  - סה"כ $O(n\log n)$.
- תור עדיפות:
  - בנה ערימת מינימום בזמן $O(n)$.
  - בצע Extract-Min i פעמים בזמן $O(i\log n)$.
  - סה"כ זמן הריצה הוא $O(n+i\cdot\log n)$

שאלה 5 (20 נקודות)

א. המערך הבא מיצג עץ ערמה (משמאל לימין): {29 ,39 ,36 ,37 ,30 ,22 ,28 ,25 ,21 ,18 ,8}.
צייר את העץ. **תרגיל בית**

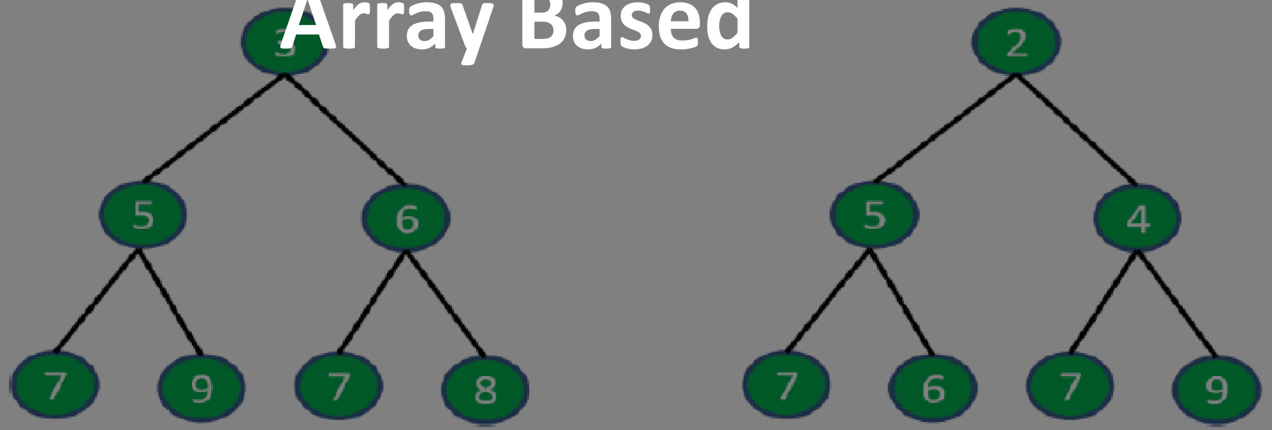
ב. צייר את העץ ואת המערך אחרי מחיקת האיבר הקטן. מהו זמן הריצה של מחיקה? **תרגיל בית**

ג. נניח ש-A,B הם שני עצי ערמה, ובמקרה העצים של שניהם שלמים וגם הגבהים שלהם זהים. כתוב אלגוריתם יעיל לאיחוד A,B לעץ ערמה חדש C, מהו זמן הריצה של האלגוריתם שכתבת? **הוכח!**

שימו לב: לצורך האלגוריתם, יש להניח שעצי-הערמה מיוצגים כעצים, ולא במערך. ניתן להניח שכל צומת מכיר את שני הבנים שלו, והאלגוריתם מקבל את השורשים של שני העצים.

דוגמה:

Array Based



עבודה עצמית

```java
// Merges max heaps a[] and b[] into arr[]
public static void mergeHeaps(int[] arr, int[] a,
                              int[] b, int n, int m) {

    for (int i = 0; i < n; i++) {
        arr[i] = a[i];
    }
    for (int i = 0; i < m; i++) {
        arr[n + i] = b[i];
    }

     n = n + m;

    // Builds a max heap of given arr[0…n-1]
    for (int i = n/2-1 ; i >= 0; i--) {
        maxHeapify(arr, n, i);
    }
}
```

BASE 0!!

$O(n+m)$

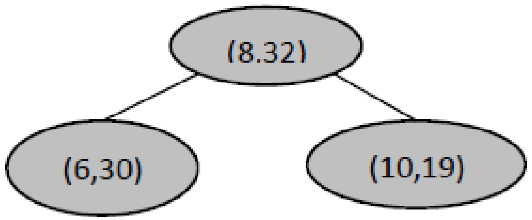
Binomial *heap*

שאלה 3 (20 נקודות)

ערימה הוא עץ בינרי בו הערך של כל צומת גדול או שווה לערך בניו. (שימו לב – לא דרשנו בהגדרה זו עץ בינרי כמעט שלם).

יהא T עץ בינרי, שבו כל צומת מכיל 2 ערכים: מפתח k ועדיפות p. העץ T נקרא **Treap** אם T הוא עץ חיפוש בינרי ביחס למפתחות, וערימה ביחס לעדיפויות.

לדוגמה:



כאשר הערך השמאלי בכל צומת הוא המפתח, והערך הימני הוא העדיפות (k,p).

א. נתונה הקבוצה הבאה של זוגות סדורים (האיבר הראשון (השמאלי) בכל זוג הוא המפתח, והאיבר השני (הימני) בכל זוג הוא העדיפות):

{(5,34),(2,13),(8,26),(6,19),(7,38),(9,14),(11,27),(10,22)}

צייר Treap עבור הקבוצה הנ"ל.

ב. בהנתן קבוצה של זוגות סדורים המורכבים ממפתח ועדיפות בה כל המפתחות שונים זה מזה, וכל העדיפויות שונות זו מזו, כתוב אלגוריתם לבניית Treap.

(2,13), (9,14), (6,19), (10,22),
(8,26), (11,27), (5,34), (7,38)

(7,38)

(5,34) (11,27)

(2,13) (8,26) (6,19)

(10,22)

(9,14)

1. Sort the nodes based on **p**.
2. Start picking the node with the largest p and decide whether it should go to the left subtree or right subtree based on **k**.

אוניברסיטת
אריאל
בשומרון

שאלה 6

כתוב פונקציה שמקבלת מערך של מספרים שלמים. הפונקציה מחזירה true אם המערך מהווה Max-Heap, אחרת היא מחזירה false.

עבודה עצמית

```java
// Function to check if given array represents Min-Heap or not
public static boolean checkMinHeap(int[] A, int i) {
// if i is a leaf node, return true as every leaf node is a heap
if (2*i + 1 >= A.length) return true;

// if i is an internal node

// recursively check if left child is heap
boolean left = (A[i] >= A[2*i + 1]) && checkMinHeap(A, 2*i + 1);

// recursively check if right child is heap (to avoid array out
// of bound, we first check if right child exists or not)
boolean right = (2*i + 2 == A.length) ||
(A[i] >= A[2*i + 2] && checkMinHeap(A, 2*i + 2));

// return true if both left and right
// child are heap
return left && right;
}
```

```java
public static void main(String[] args) {
 int[] A = {6,5,4,3,2,1,0};
 // start with index 0 (root of the heap)
 int index = 0;
 System.out.println(checkMinHeap(A, index) ?
 "YES" : "NO");
}
```

תארו אלגוריתם בזמן $O(n\log k)$ למיון k מערכים ממויינים $A_1,\ldots,A_k$ למערך ממויין אחד.

n הוא מספר האיברים הכללי וכמובן $k \le n$.

פתרון:

- נבנה **ערימה** שתשמור בכל שלב את האיברים החשודים להיות האיבר הקטן הבא.
  כלומר, נרצה לשמור את האיבר הראשון בכל אחד מהמערכים $A_1,\ldots,A_k$.
- נשתמש **בערימת מינימום** אשר כל איבר בה הוא שלשה מהצורה $(d,i,A_d[i])$
  כאשר $d\in[1,k], i\in[1,n]$, הערימה תיוצג ע"י המערך B.
- ראשית נבנה ערימה עם האיברים הראשונים במערכים $A_1,\ldots,A_k$ בזמן $O(k)$.
- בכל צעד נוציא את האיבר המינימלי בערימה ונכניס אותו למערך הפלט M.
- נוציא את האיבר שהוצאנו מהמערך המקורי שלו ונביא משם את האיבר המינימלי הבא.

d – איזה רשימה
i – איזה איבר ברשימה

```
for d←1 to k
      B[d] ← (d, 1, A_d[1])
Build-Heap(B)   /*By the order of A_d [1] */
for j=1 to n
      (d, i, x) ← Extract-Min(B)
      M[j]←x
      if i < A_d.length then Heap-Insert(B,(d, i+1, A_d[i+1]))
```

Worst case time analysis:

Build-Heap : O(k) – done 1 time
Extract-Min : O(log k) – done n times
Heap-Insert : O(log k) – done n times
Total: O(nlogk)