אוניברסיטת
אריאל
בשומרון

מבני נתונים

תרגול 3 – עצים

צבי מינץ – zvimints@gmail.com

אוניברסיטת
אריאל
בשומרון

היום

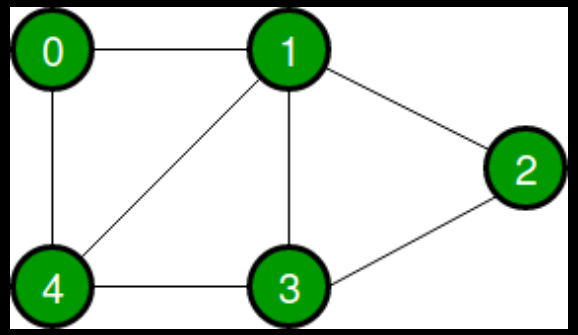
- עצים
- עצים בינאריים
- עץ חיפוש בינארי
- תכונות

- מימוש – תרגול הבא

גרף

גרף Graph (G) – אוסף של כל הקודקודים וצלעות המחברים זוגות של קודקודים

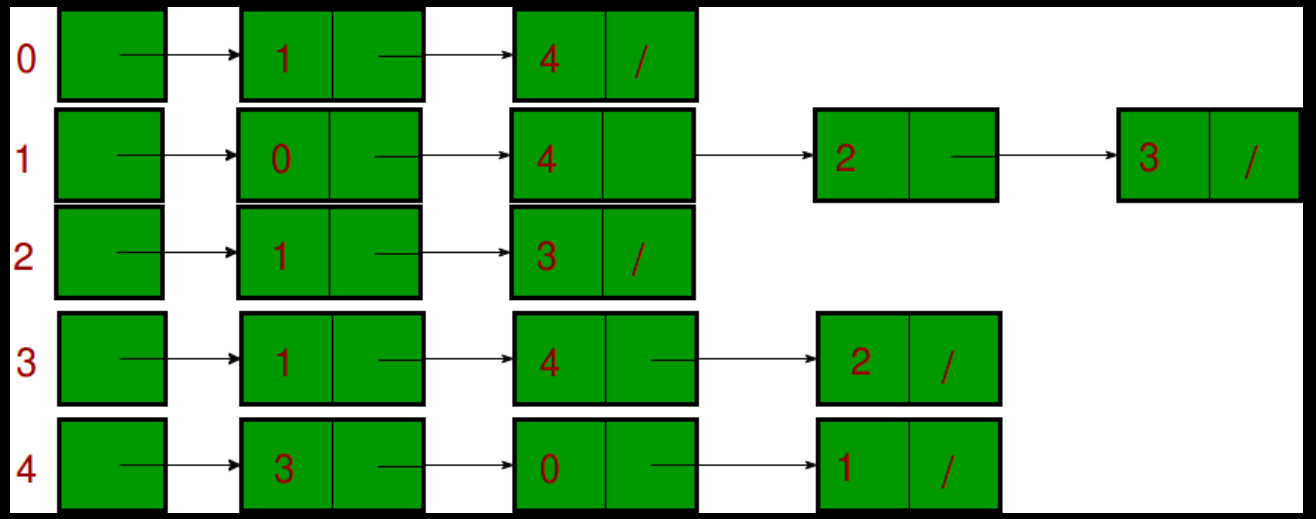
מיוצג במחשב בד"כ ע"י **Adjacency** Matrix או **Adjacency** List



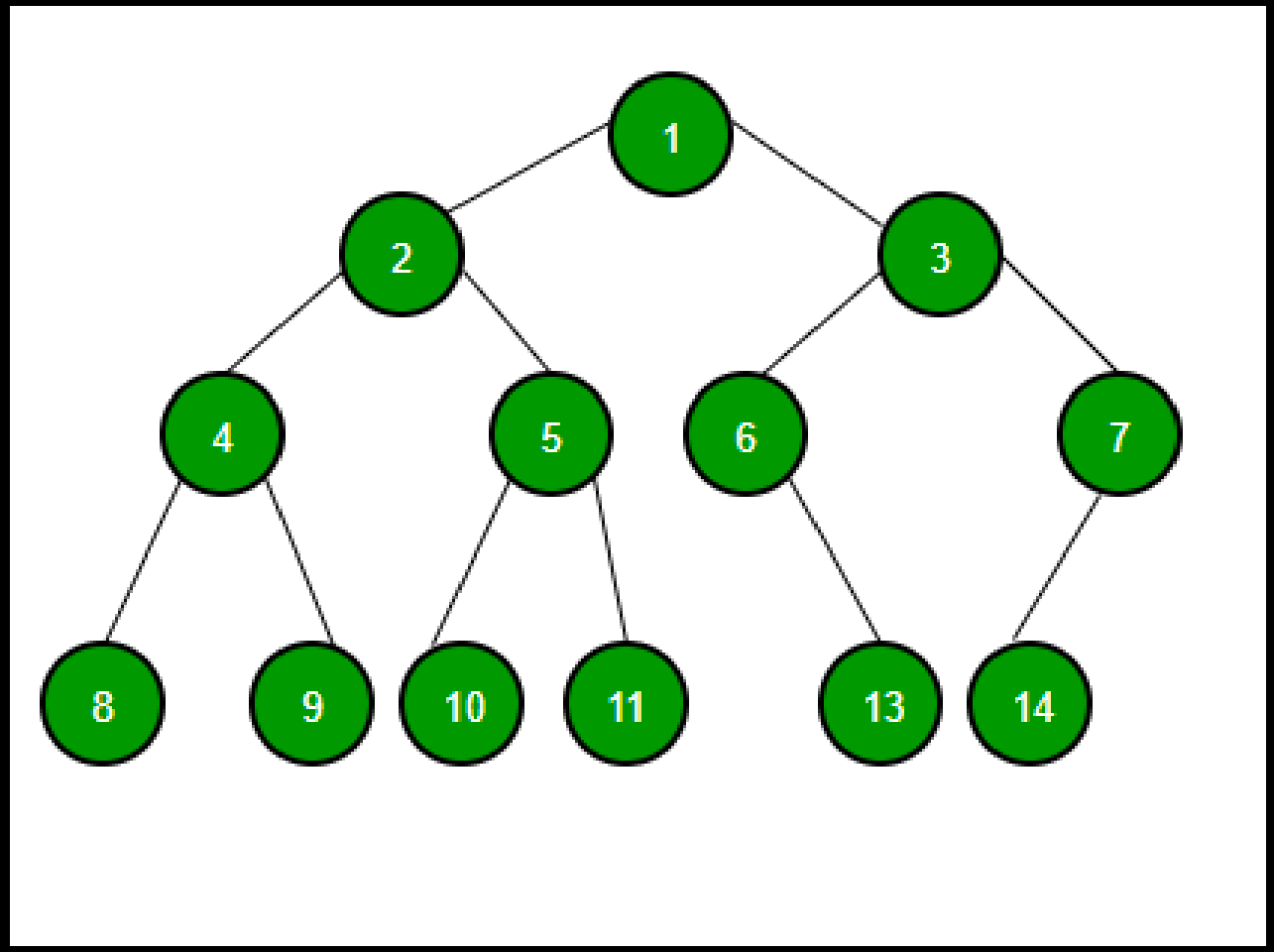
Adjacency Matrix

	0	1	2	3	4
0	0	1	0	0	1
1	1	0	1	1	1
2	0	1	0	1	0
3	0	1	1	0	1
4	1	1	0	1	0

Adjacency List



עץ

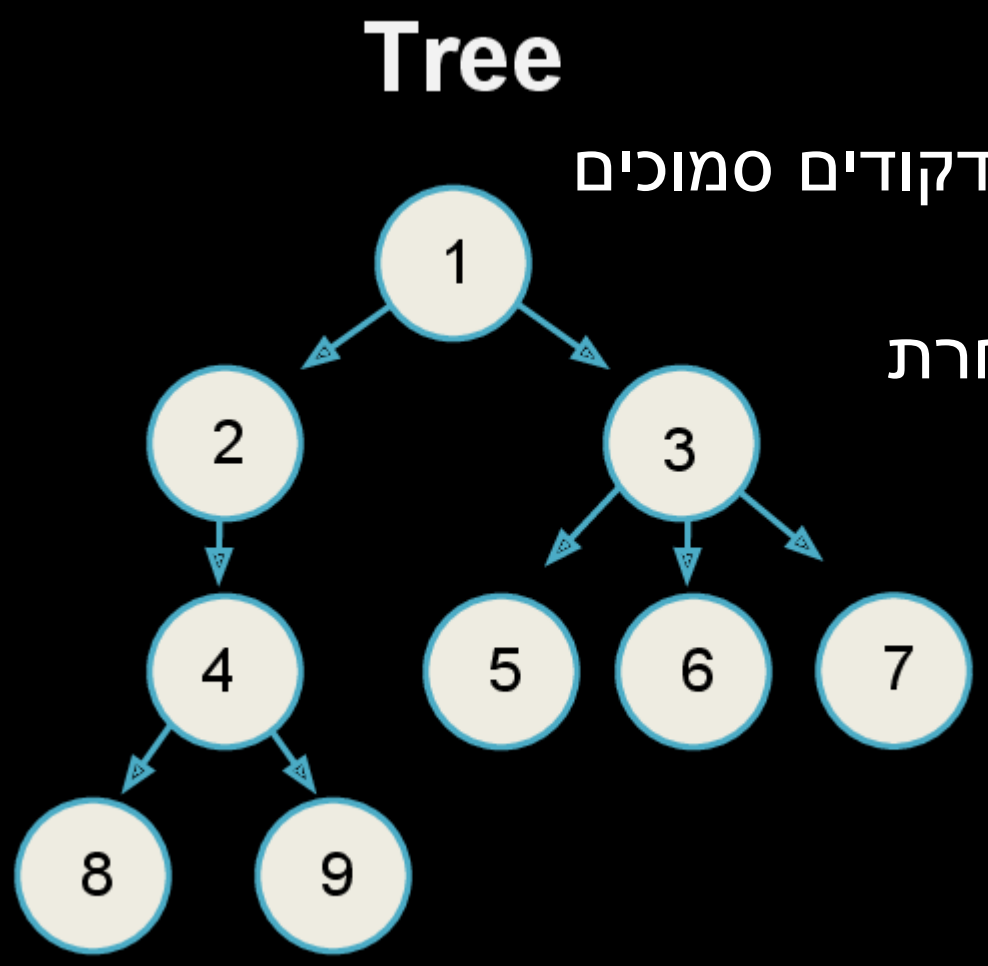


עץ Tree (T) – עץ הוא גרף **חסר מעגלים**

יהי G גרף עם n קודקודים, אז G הוא עץ אם קיימים שניים מבין התנאים הבאים:

1. G קשיר
2. G חסר מעגלים
3. G מכיל $n-1$ צלעות

הגדרות



צומת (Node) – איבר בתוך העץ

מסלול (Path) – סדרת קודקוד בגרף בה כל שני קודקודים סמוכים מחוברים ע"י צלע

אב (Parent) – צומת שיוצאת ממנה צלע לצומת אחרת

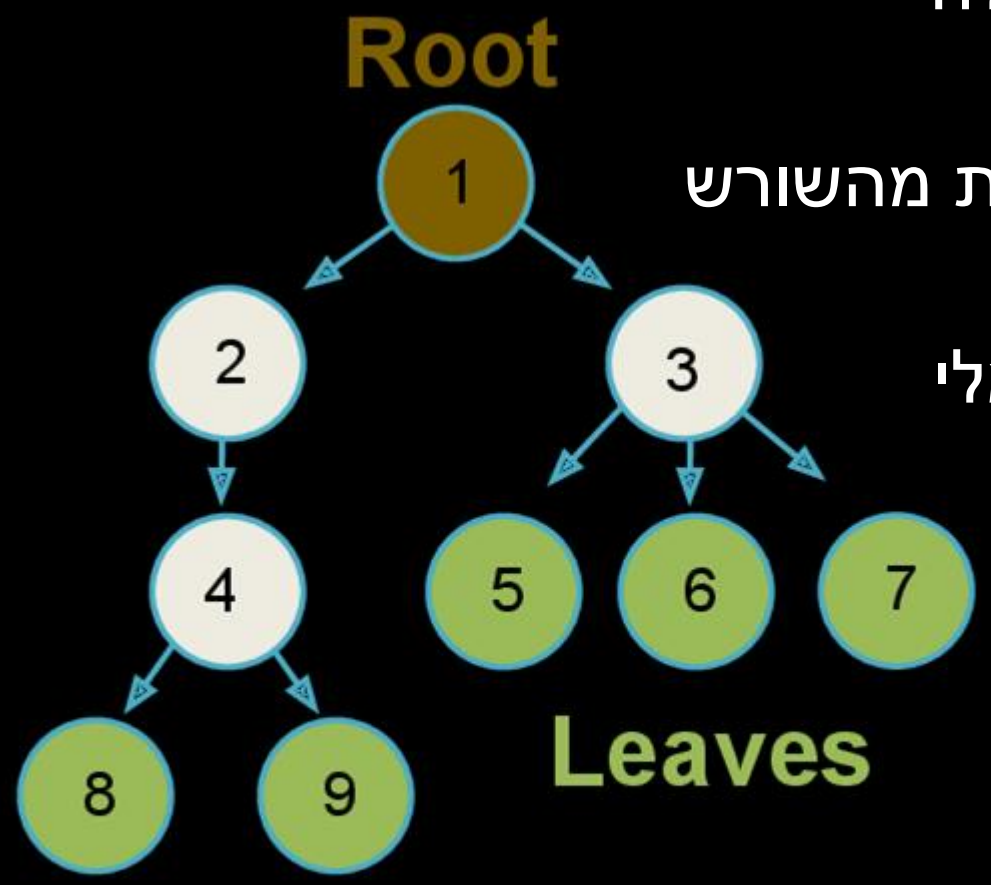
בן (Child) – צומת שיש לו אב

- Left Child
- Right Child

שורש (Root) – צומת יחיד בעץ שאין לו אב

עלה (Leaf) – צומת שאין לו בנים

הגדרות



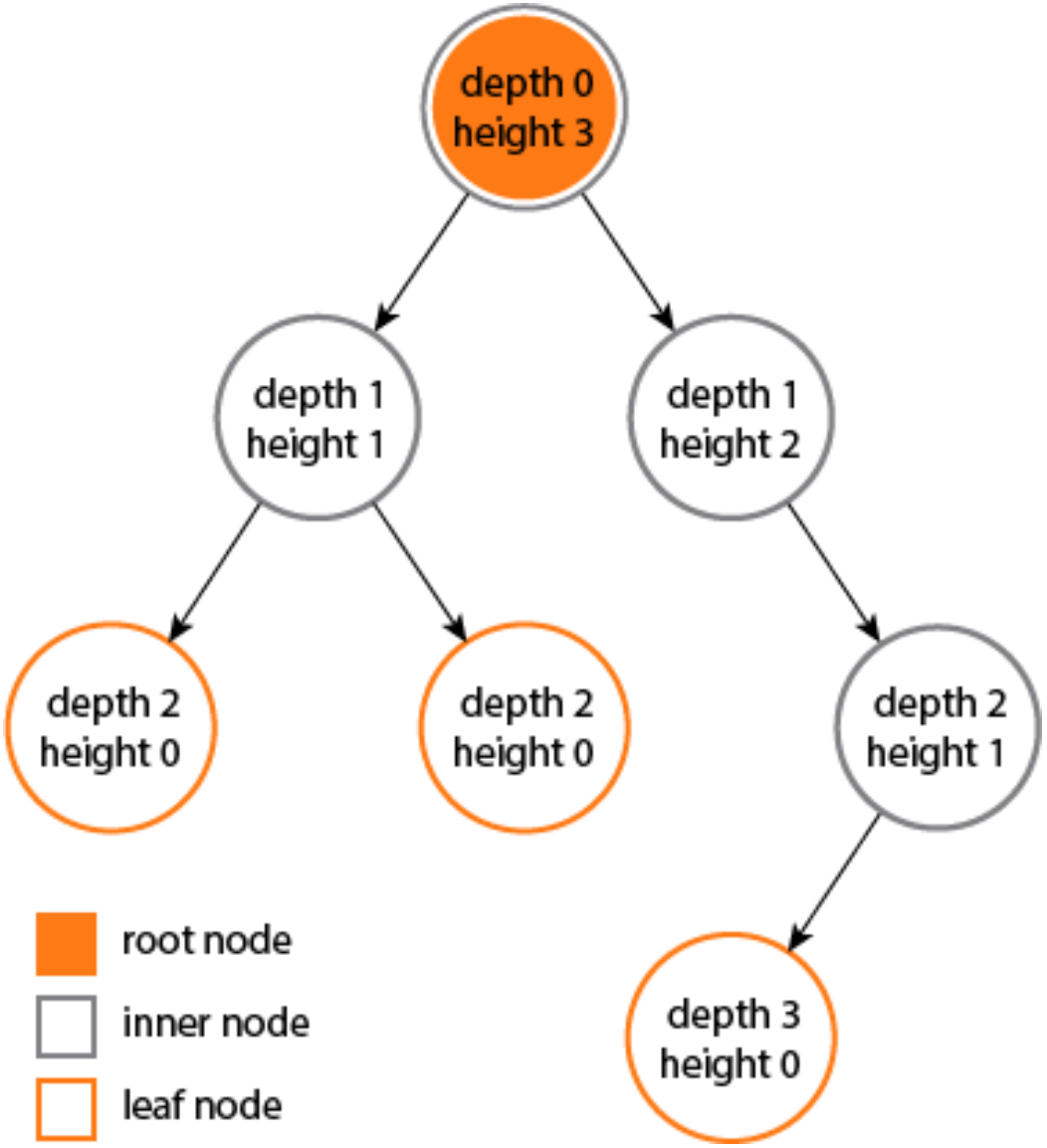
צומת פנימי (Internal Node) – צומת אשר אינו עלה

עומק צומת (Depth) – מרחק (מספר צלעות) הצומת מהשורש

גובה (Height) – המרחק (מספר צלעות) המקסימאלי של צומת מעלה (בתת עץ שלו)

גובה העץ (Tree Height) –המרחק המקסימאלי מהשורש עד עלה (נסמן בד"כ ע"י h)

אחים/שכנים (Siblings) – צומת שיש להם אותו אב


depth 0
height 3
depth 1
height 1
depth 1
height 2
depth 2
height 0
depth 2
height 0
depth 2
height 1
depth 3
height 0
root node
inner node
leaf node

שימוש בעצים

משתמשים בעצים על מנת לאחסן מידע **היררכי**.

מע"ה משתמשת בעצים בשביל תיקיות Files and Folders

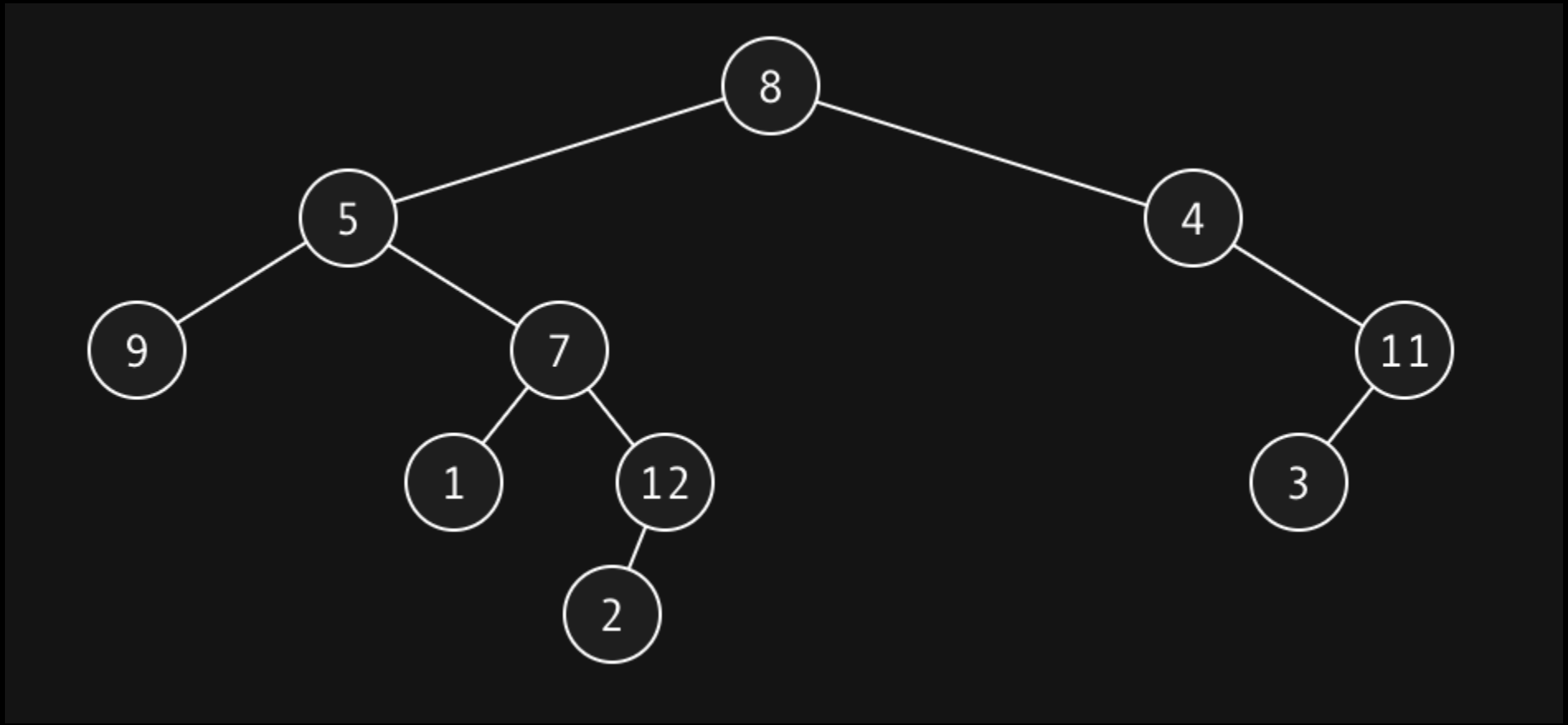
מבנה נתונים **דינאמי** – קל להוסיף ולמחוק קודקודים (מידע)

בעץ בעל **סדר** קל לבצע חיפוש, מיון ואלגורתמי סיור בעץ (Tree traversal algorithms)

...

עץ בינארי

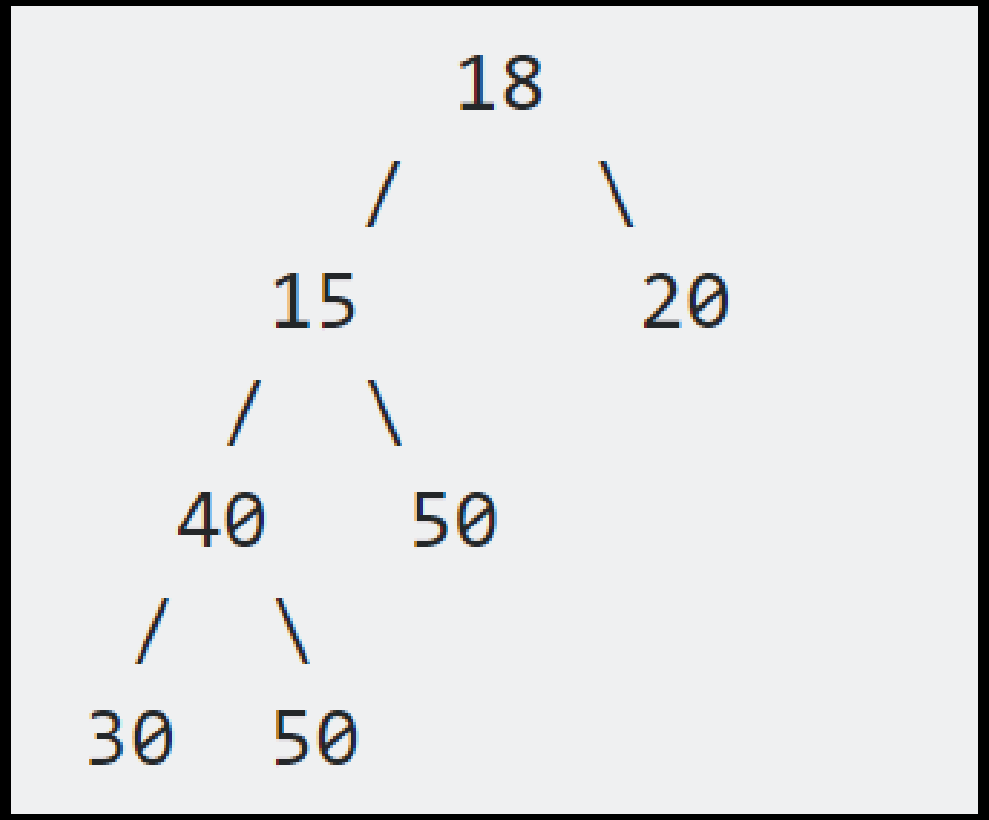
עץ בינארי (Binary Tree) – עץ אשר לכל קודקוד יש לכל **<u>היותר</u>** 2 בנים



עץ בינארי מלא

עץ בינארי מלא (Full Binary Tree)

עץ בינארי הוא מלא אם כל לכל קודקוד יש 0 או 2 ילדים



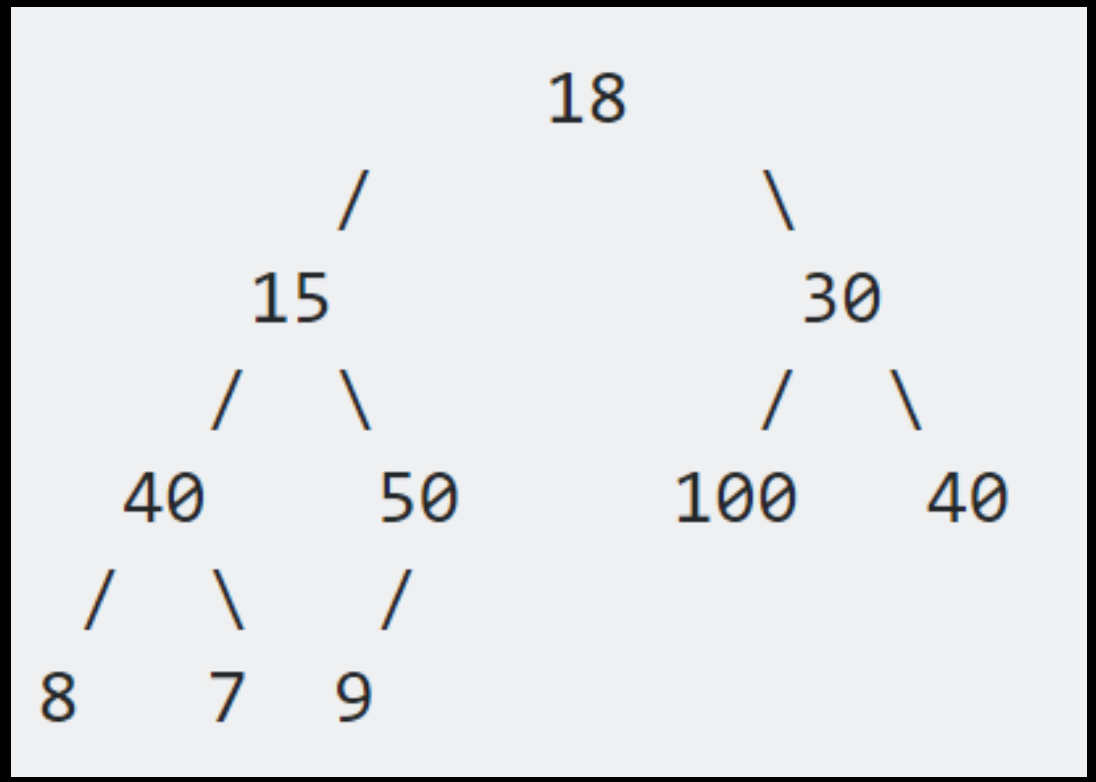
https://cs.stackexchange.com/questions/32397/is-there-a-difference-between-perfect-full-and-complete-tree

עץ בינארי שלם

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

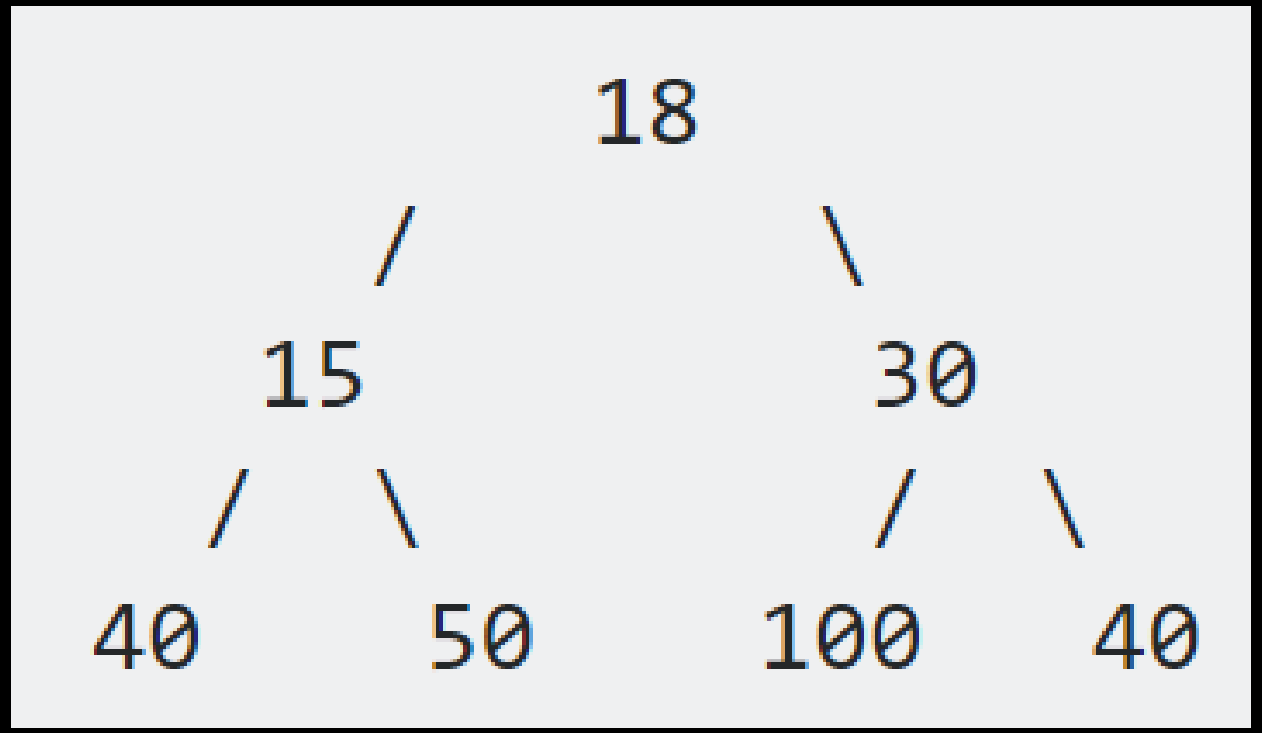


https://cs.stackexchange.com/questions/32397/is-there-a-difference-between-perfect-full-and-complete-tree

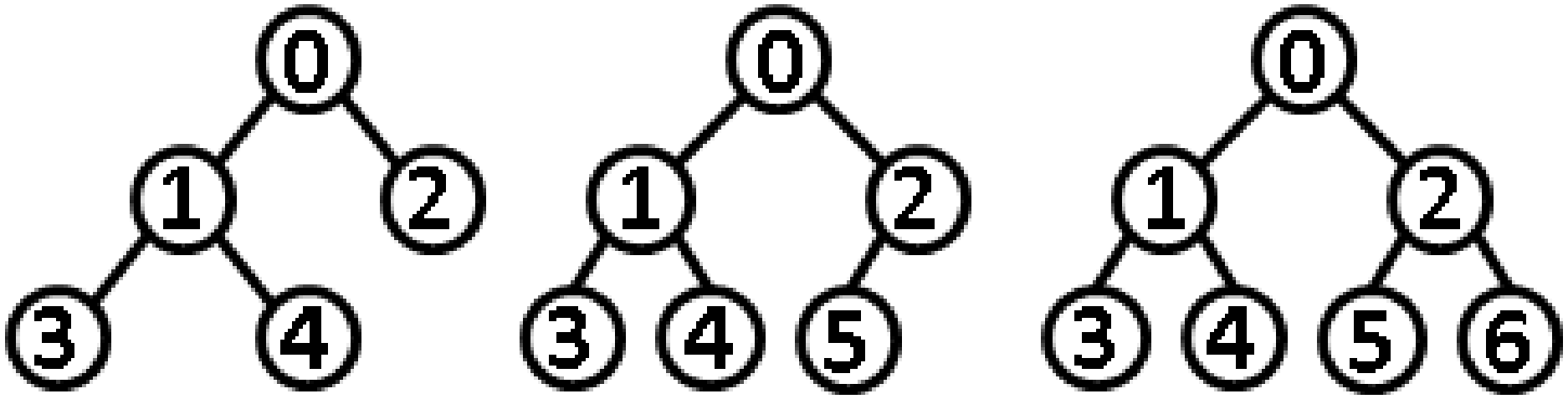
עץ בינארי מושלם

עץ בינארי מושלם (Perfect Binary Tree)

עץ בינארי מושלם הוא עץ בינארי כך שכל העלים נמצאים באותו מרחק מהשורש וכל הקודקודים הפנמיים בעלי 2 ילדים



https://cs.stackexchange.com/questions/32397/is-there-a-difference-between-perfect-full-and-complete-tree



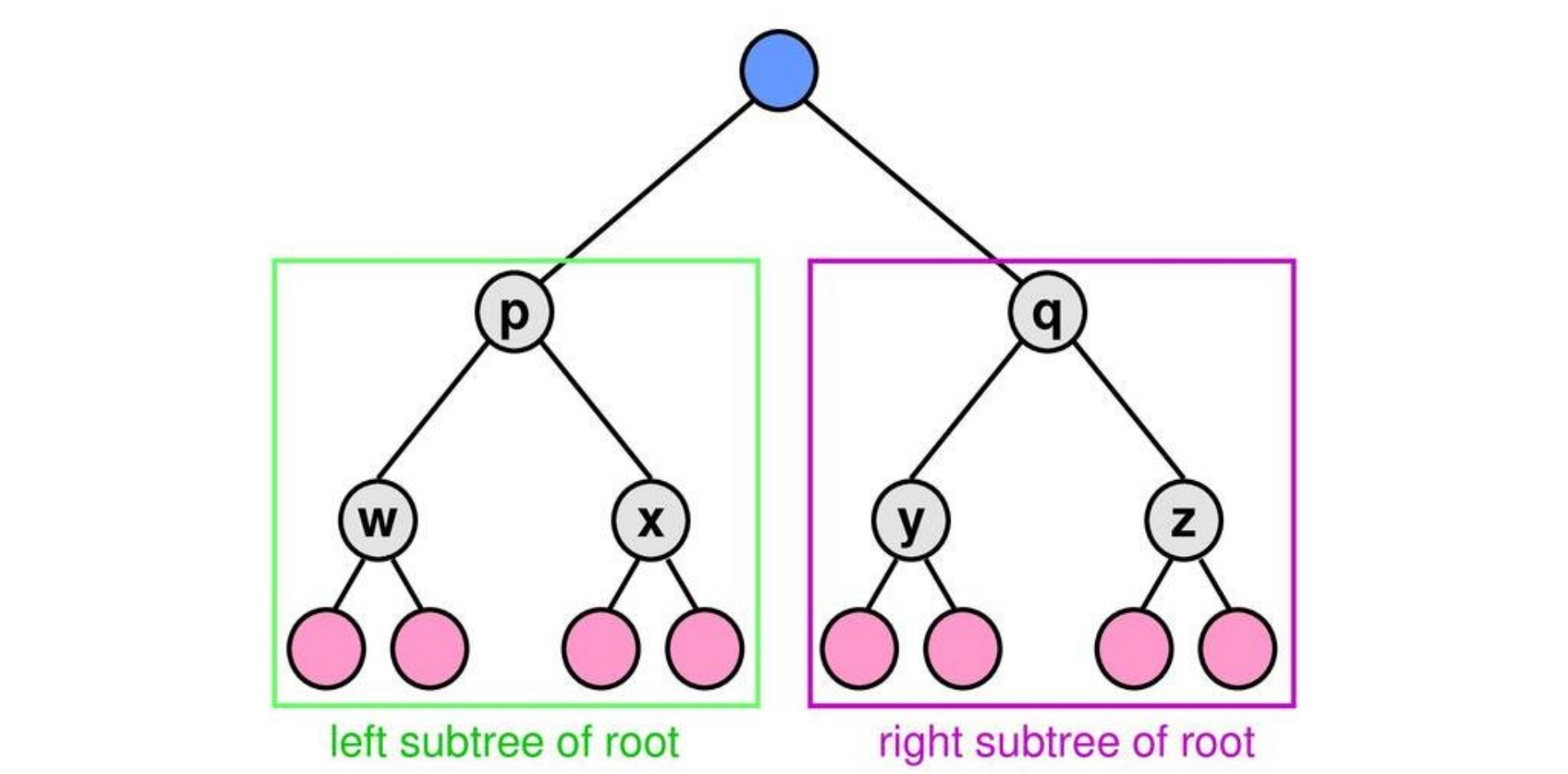
full binary tree

complete binary tree

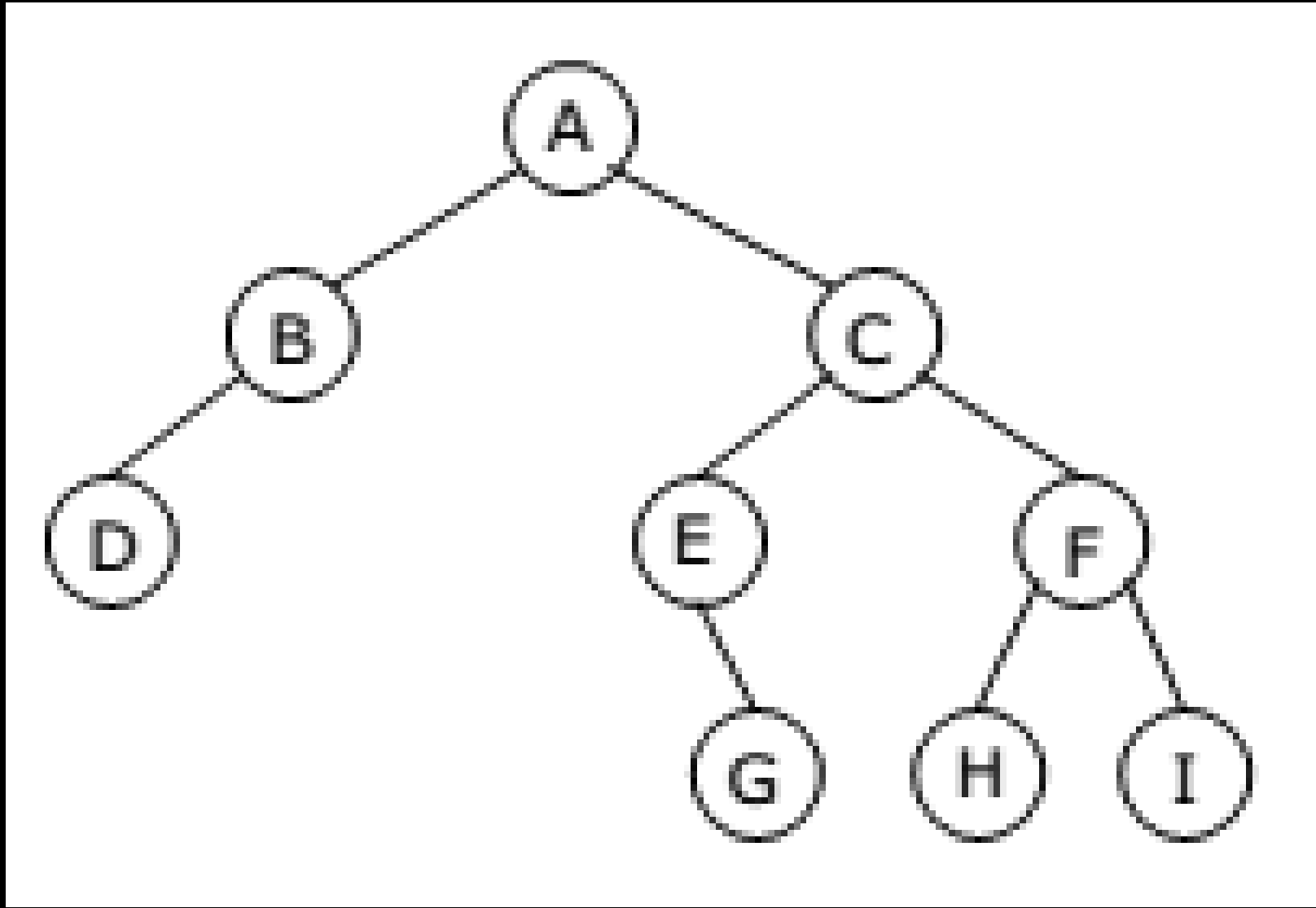
perfect binary tree

עץ בינארי

כל עץ ניתן לפרוק **לשורש**, תת-עץ **שמאלי** ותת-עץ **ימני**



שיטות מעבר על עץ בינארי tree traversal



שיטות מעבר על עץ בינארי tree traversal

- Preorder

עבור כל צומת יש להדפיס תחילה את ערכו, לאחר מכן לעבור לבנו השמאלי ולבסוף לבן הימני.

preorder tree traversal

```
public void PreOrder() // (V,L,R)
{
    PreOrder(root);
}
```

```
public void PreOrder(Node current) {
    if(current != null)
    {
        System.out.print(current.data + " " );
        PreOrder(current.left);
        PreOrder(current.right);
    }
}
```

שיטות מעבר על עץ בינארי tree traversal

- Preorder

עבור כל צומת יש להדפיס תחילה את ערכו, לאחר מכן לעבור לבנו השמאלי ולבסוף לבן הימני.

- Inorder

עבור כל צומת יש להדפיס תחילה את בן השמאלי, לאחר מכן לעבור לערכו שלו ואז הבן הימני

Inorder tree traversal

```
public void InOrder() // (L,V,R)
{
    InOrder(root);
}
```

```
public void InOrder(Node current) {
    if(current != null)
    {
        InOrder(current.left);
        System.out.print(current.data + " " );
        InOrder(current.right);
    }
}
```

שיטות מעבר על עץ בינארי tree traversal

- Preorder

עבור כל צומת יש להדפיס תחילה את ערכו, לאחר מכן לעבור לבנו השמאלי ולבסוף לבן הימני.

- Inorder

עבור כל צומת יש להדפיס תחילה את בן השמאלי, לאחר מכן לעבור לערכו שלו ואז הבן הימני

- Postorder

עבור כל צומת יש להדפיס תחילה את בנו **השמאלי**, אחר כך את בנו הימני ולבסוף

PostOrder tree traversal

```
public void PostOrder() // (L,R,V)
{
    PostOrder(root);
}
```

```
public void PostOrder(Node current) {
    if(current != null)
    {
        PostOrder(current.left);
        PostOrder(current.right);
        System.out.print(current.data + " " );
    }
}
```

שיטות מעבר על עץ בינארי tree traversal

- Preorder

עבור כל צומת יש להדפיס תחילה את ערכו, לאחר מכן לעבור לבנו השמאלי ולבסוף לבן הימני.

- Inorder

עבור כל צומת יש להדפיס תחילה את בן השמאלי, לאחר מכן לעבור לערכו שלו ואז הבן הימני

- Postorder

עבור כל צומת יש להדפיס תחילה את בנו **השמאלי**, אחר כך את בנו הימני ולבסוף

...

שיטות מעבר על עץ בינארי tree traversal

- Preorder

עבור כל צומת יש להדפיס תחילה את ערכו, לאחר מכן לעבור לבנו השמאלי ולבסוף לבן הימני.

- Inorder

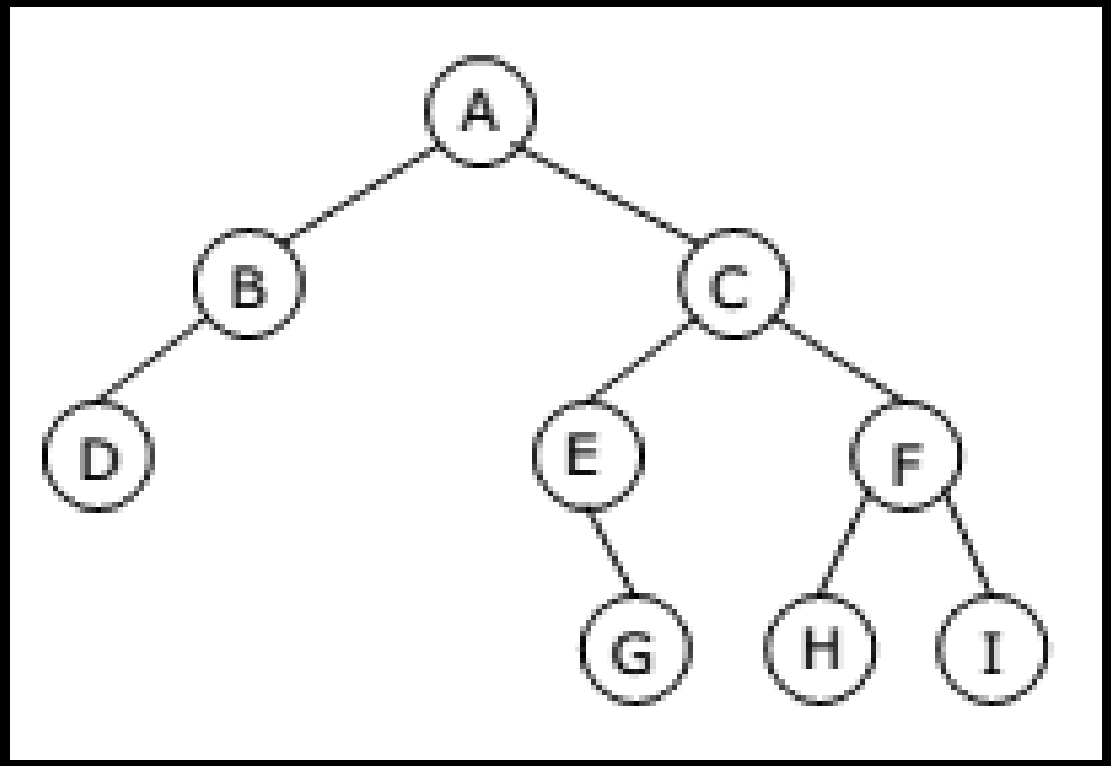
עבור כל צומת יש להדפיס תחילה את בן השמאלי, לאחר מכן לעבור לערכו שלו ואז הבן הימני

- Postorder

עבור כל צומת יש להדפיס תחילה את בנו **השמאלי**, אחר כך את בנו הימני ולבסוף

...

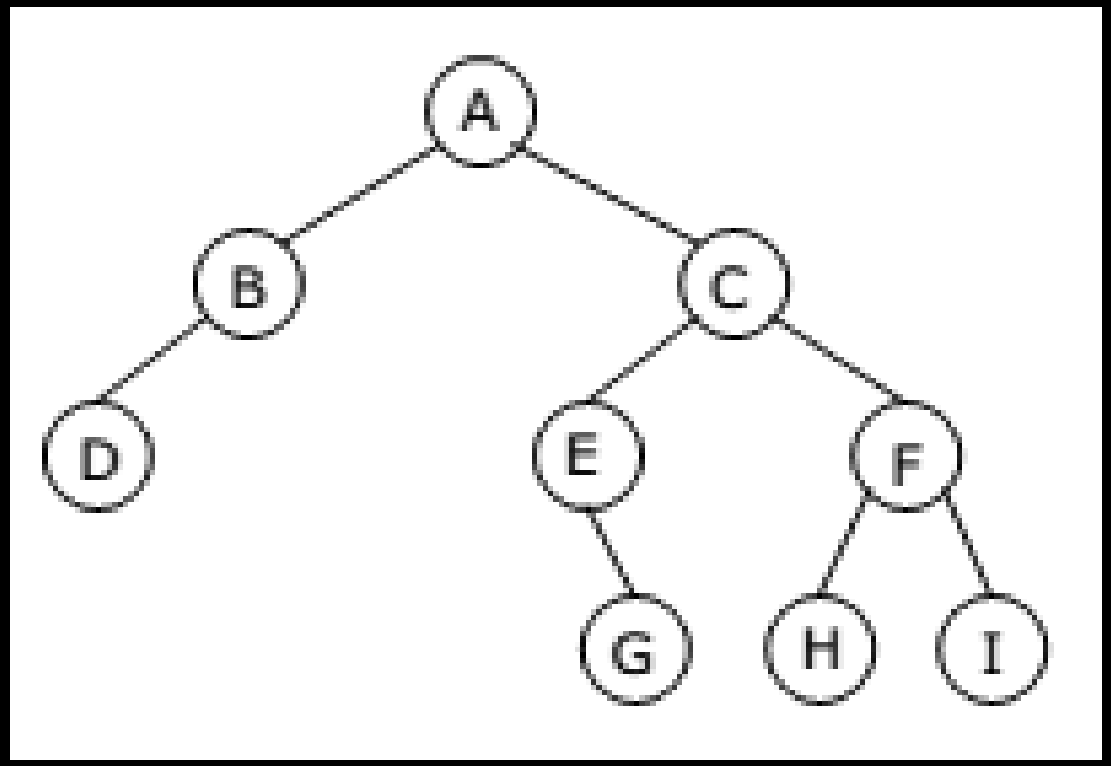
PreOrder



- Preorder traversal yields:
  A, B, D, C, E, G, F, H, I

```
public void PreOrder(Node current) {
    if(current != null)
    {
        System.out.print(current.data + " " );
        PreOrder(current.left);
        PreOrder(current.right);
    }
}
```

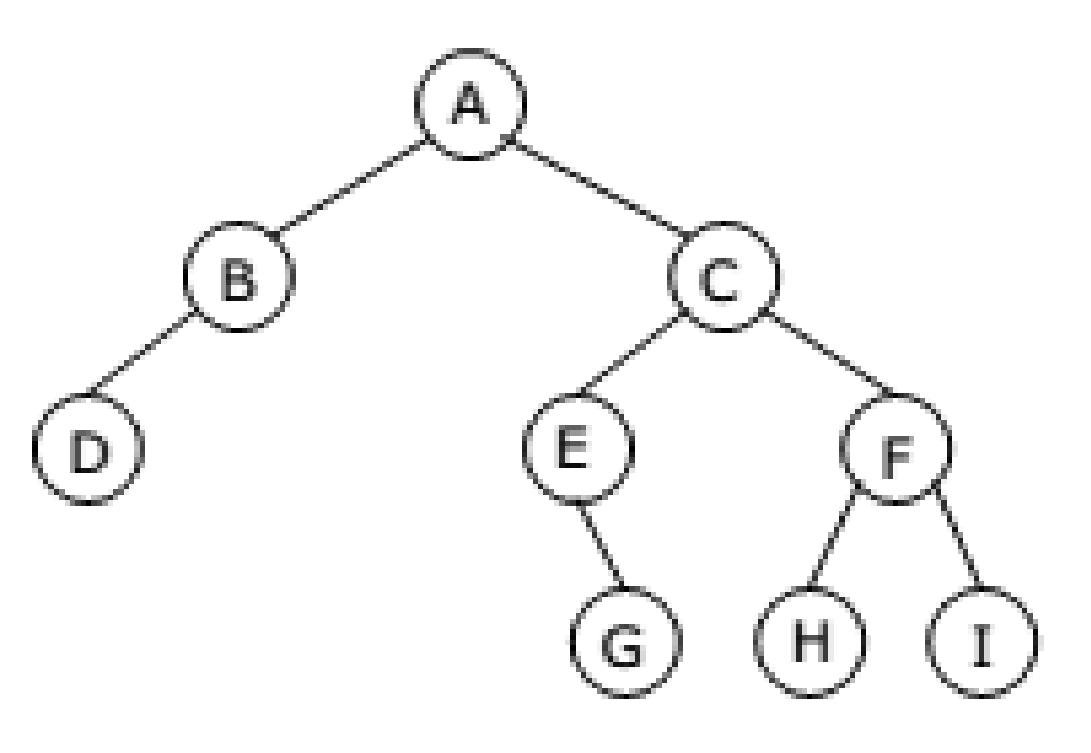
PostOrder



- Preorder traversal yields:
  A, B, D, C, E, G, F, H, I
- Postorder traversal yields:
  D, B, G, E, H, I, F, C, A

```
public void PostOrder(Node current) {
    if(current != null)
    {
        PostOrder(current.left);
        PostOrder(current.right);
        System.out.print(current.data + " " );
    }
}
```

InOrder



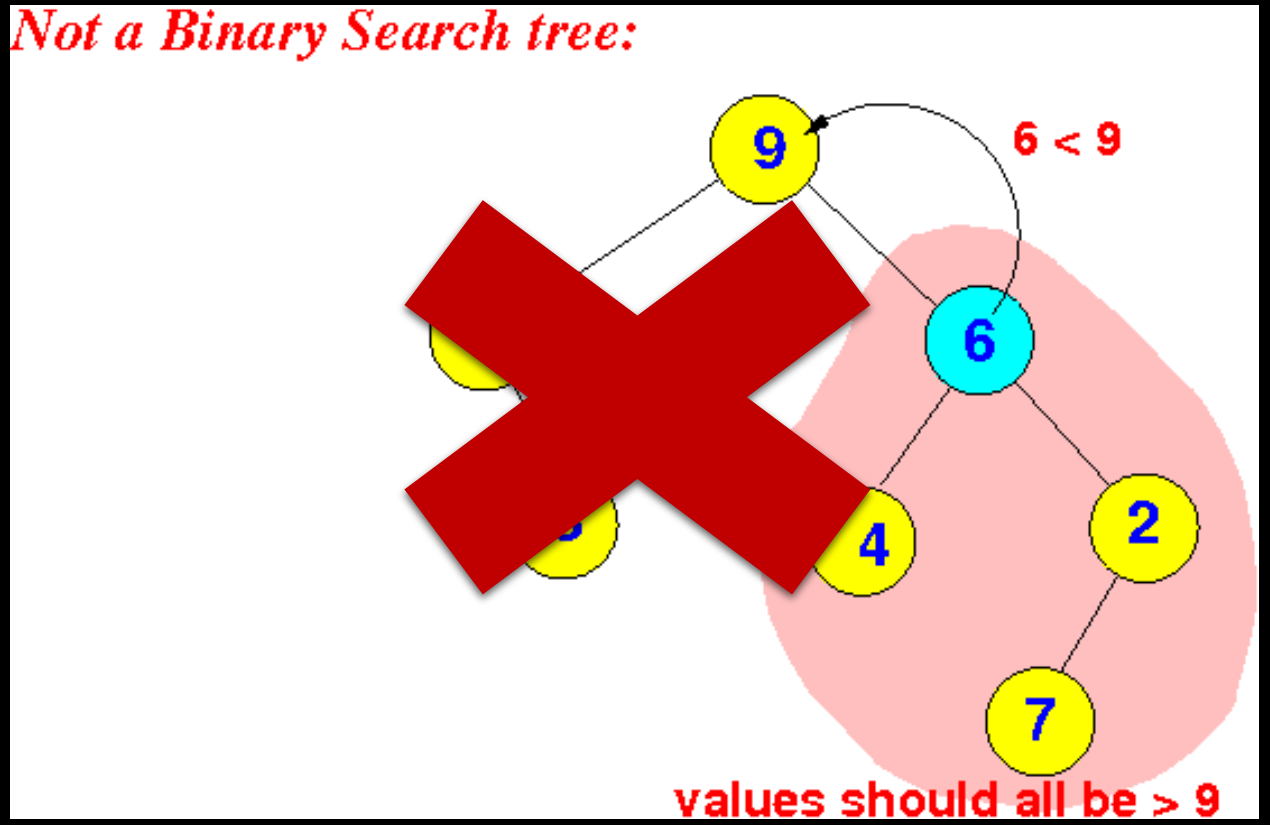
- Preorder traversal yields:
  A, B, D, C, E, G, F, H, I
- Postorder traversal yields:
  D, B, G, E, H, I, F, C, A
- Inorder traversal yields:
  D, B, A, E, G, C, H, F, I

```
public void InOrder(Node current) {
    if(current != null)
    {
        InOrder(current.left);
        System.out.print(current.data + " " );
        InOrder(current.right);
    }
}
```

עץ חיפוש בינארי Binary Search Tree (BST)

עץ חיפוש בינארי הוא עץ בינארי כך שעבור כל קודקוד x מתקיים כי :

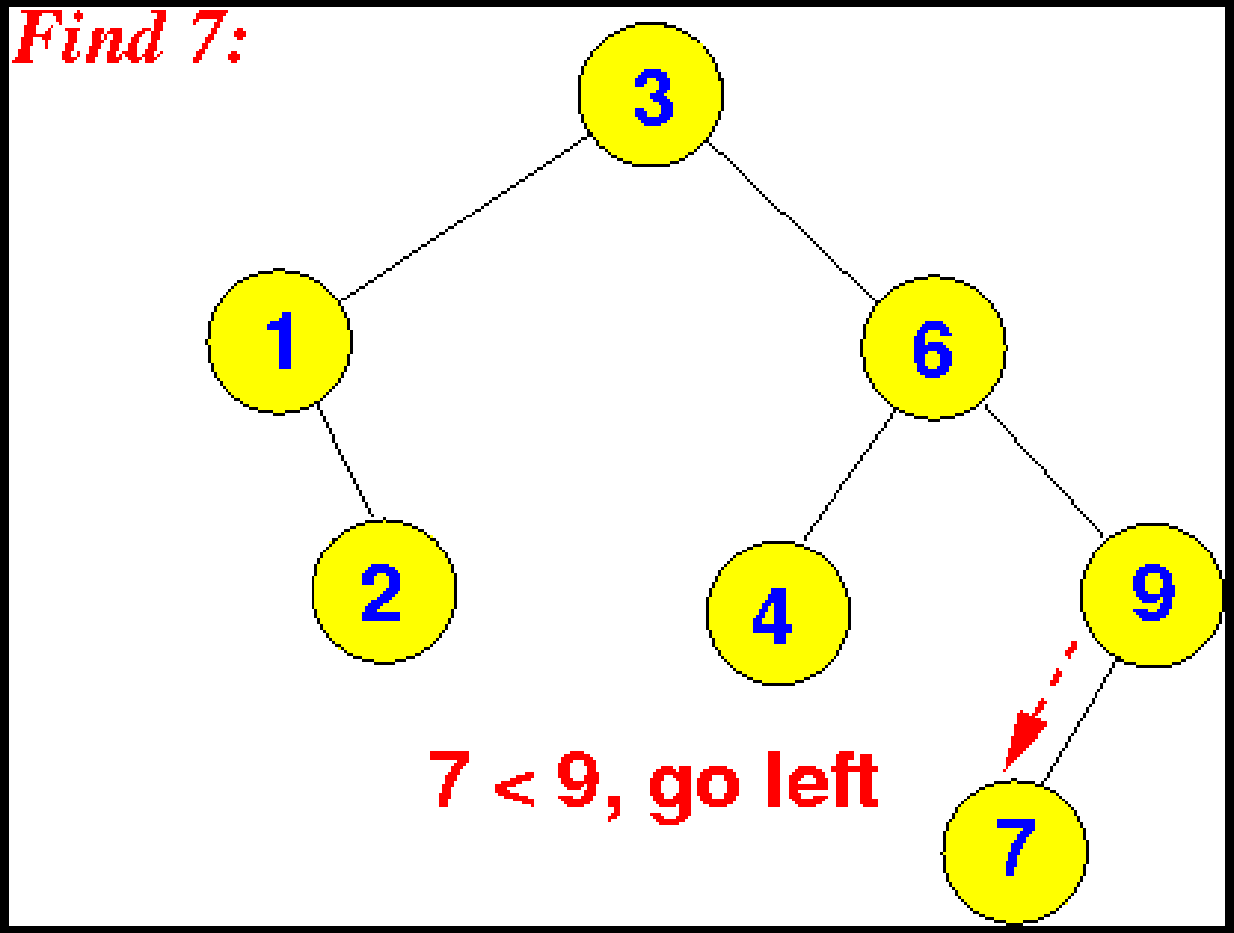
- הערכים של כל הקודקודים בתת עץ השמאלי של x קטנים ממש מ-x
- הערכים של כל הקודקודים בתת עץ הימני של x גדולים ממש מ-x



http://www.mathcs.emory.edu/~cheung/Courses/171/Syllabus/9-BinTree/BST-intro.html

חיפוש Search

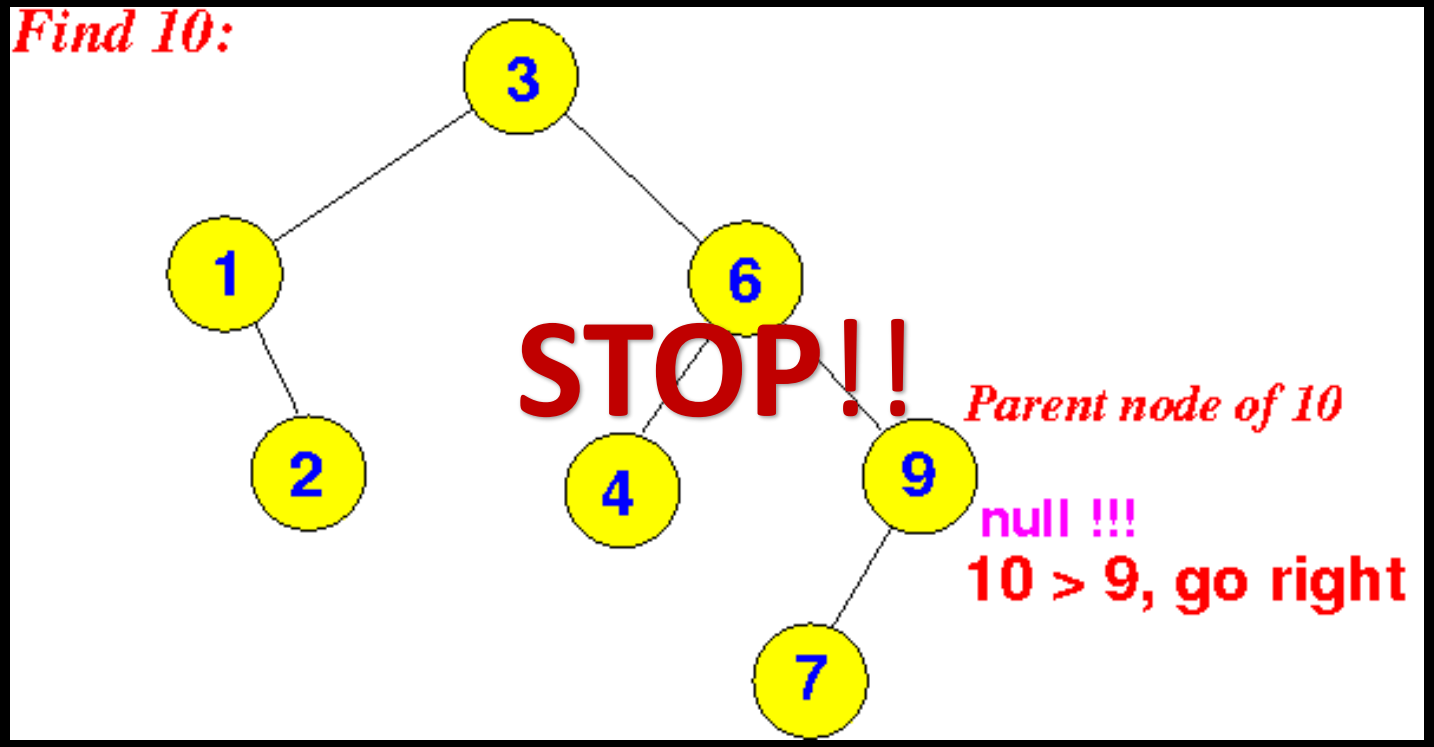
נרצה לחפש את 7 בעץ חיפוש בינארי



http://www.mathcs.emory.edu/~cheung/Courses/171/Syllabus/9-BinTree/BST-intro.html

חיפוש Search

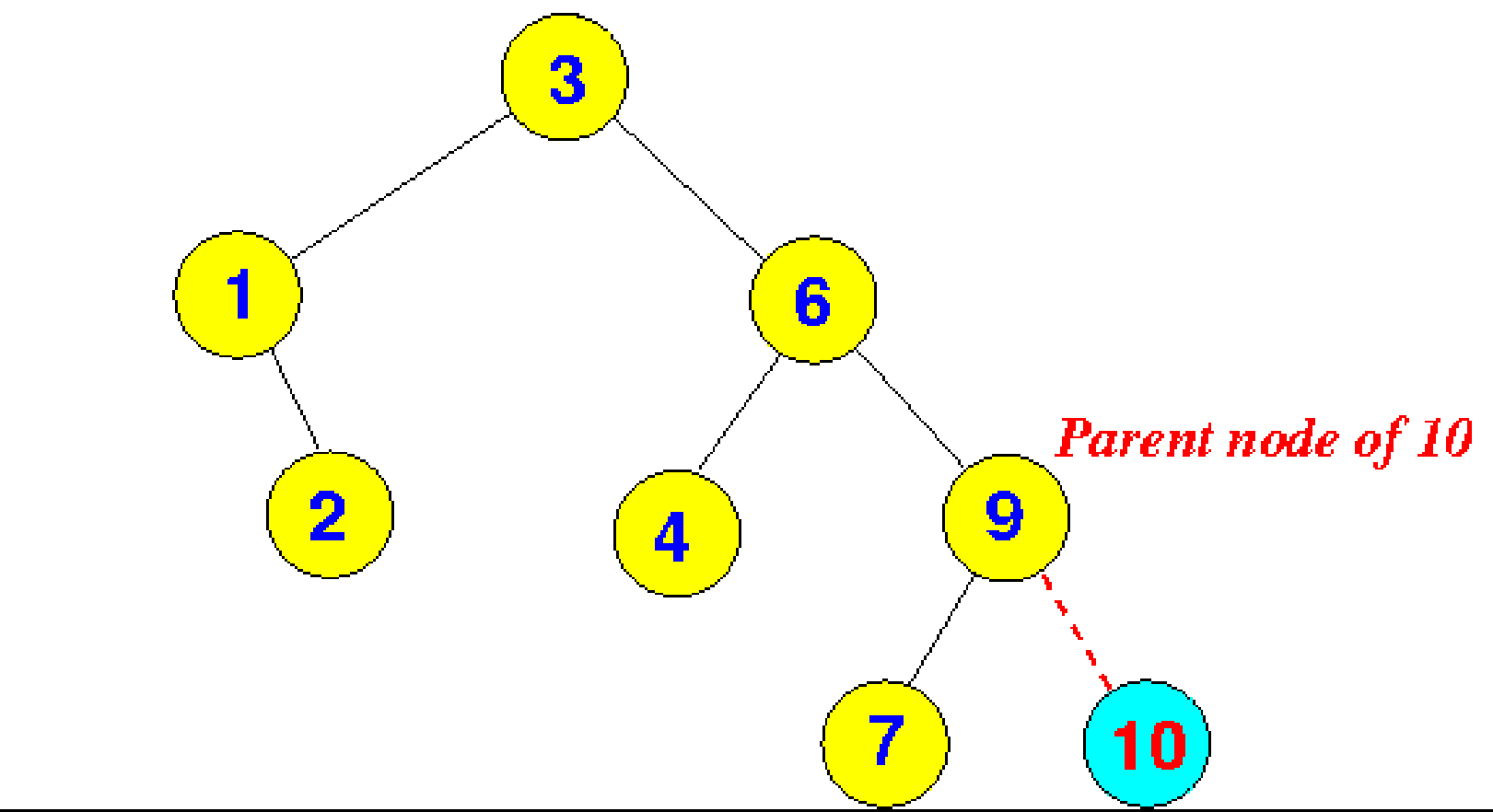
נרצה לחפש את 10 בעץ חיפוש בינארי



http://www.mathcs.emory.edu/~cheung/Courses/171/Syllabus/9-BinTree/BST-intro.html

הכנסה Insert

נרצה להכניס את 10 בעץ חיפוש בינארי

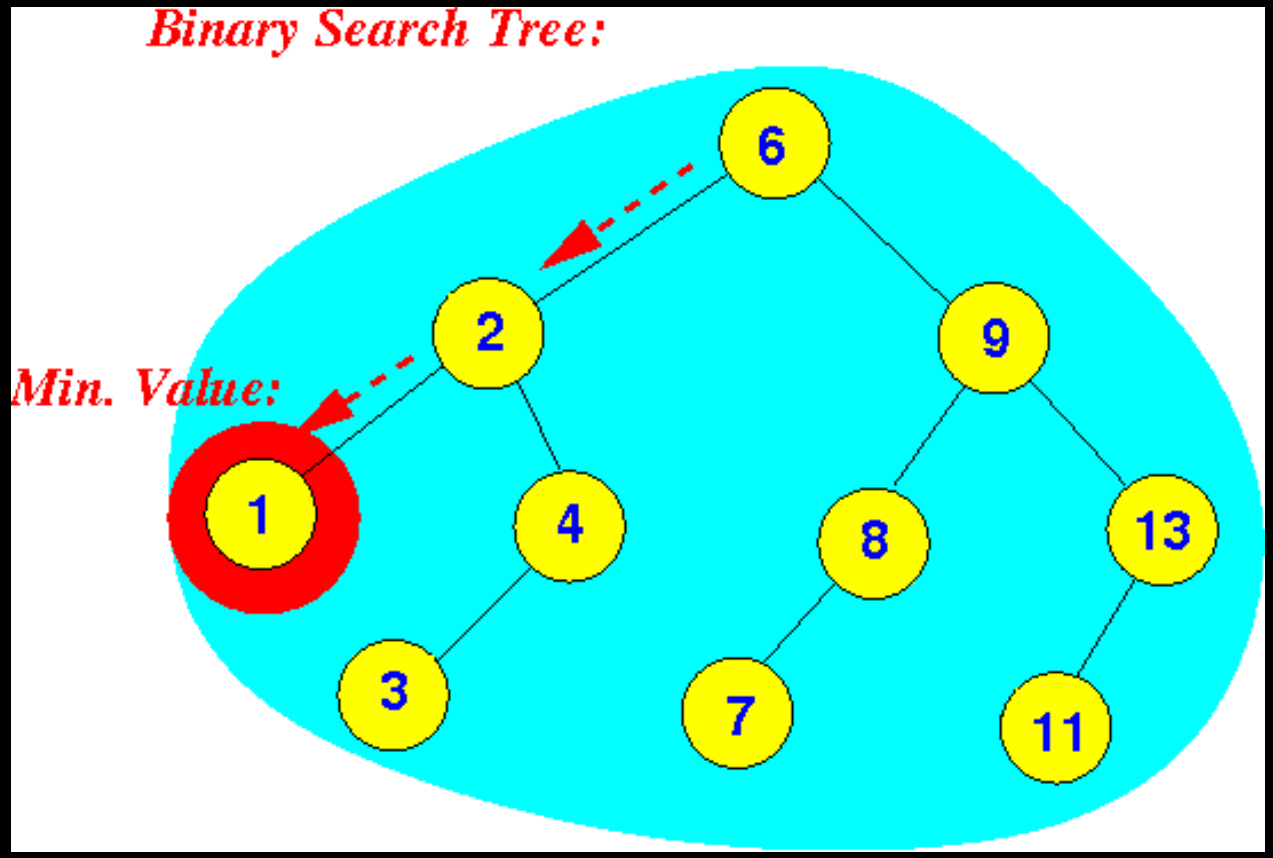


http://www.mathcs.emory.edu/~cheung/Courses/171/Syllabus/9-BinTree/BST-intro.html

הכנסה Insert

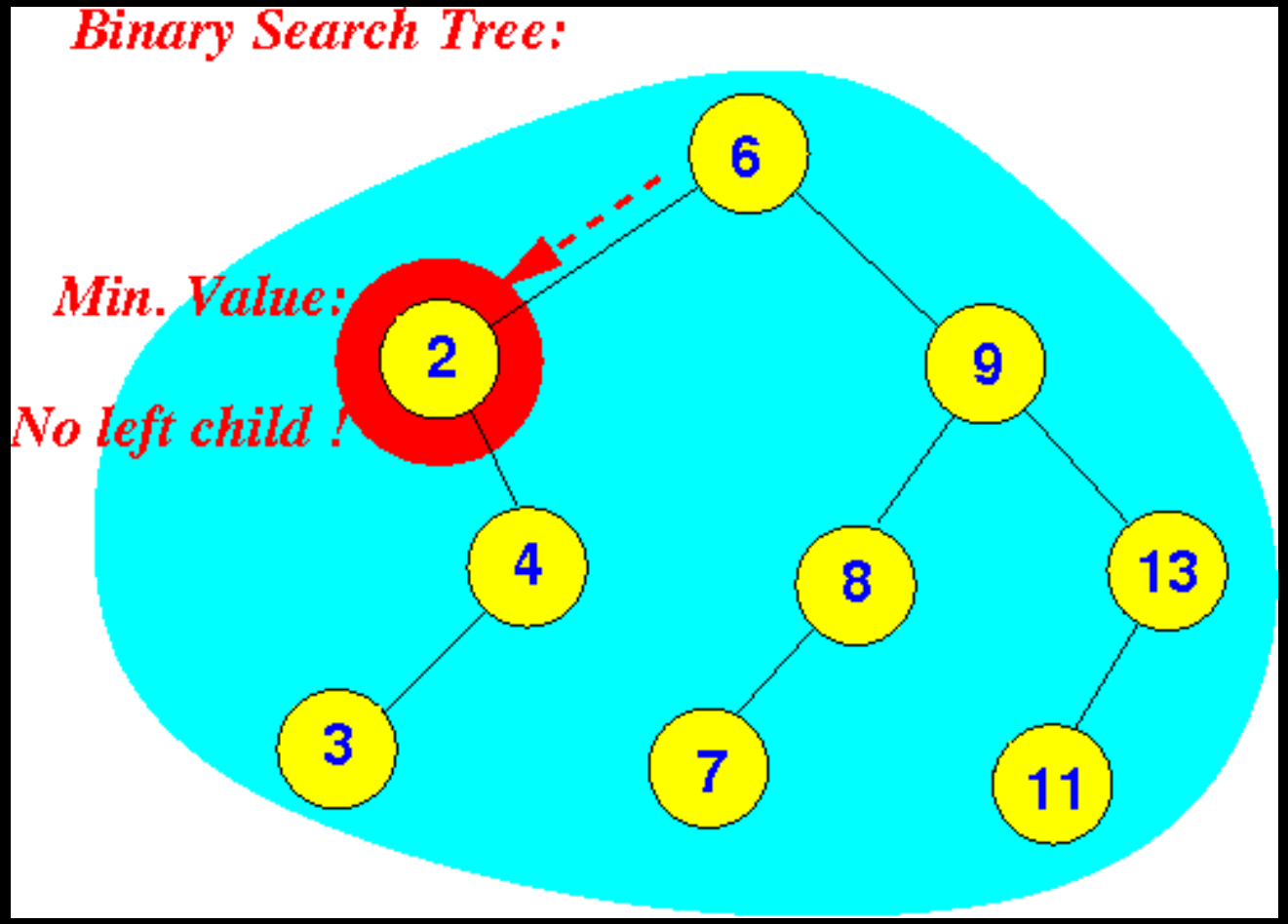
www.mathwarehouse.com

איבר מינמלי



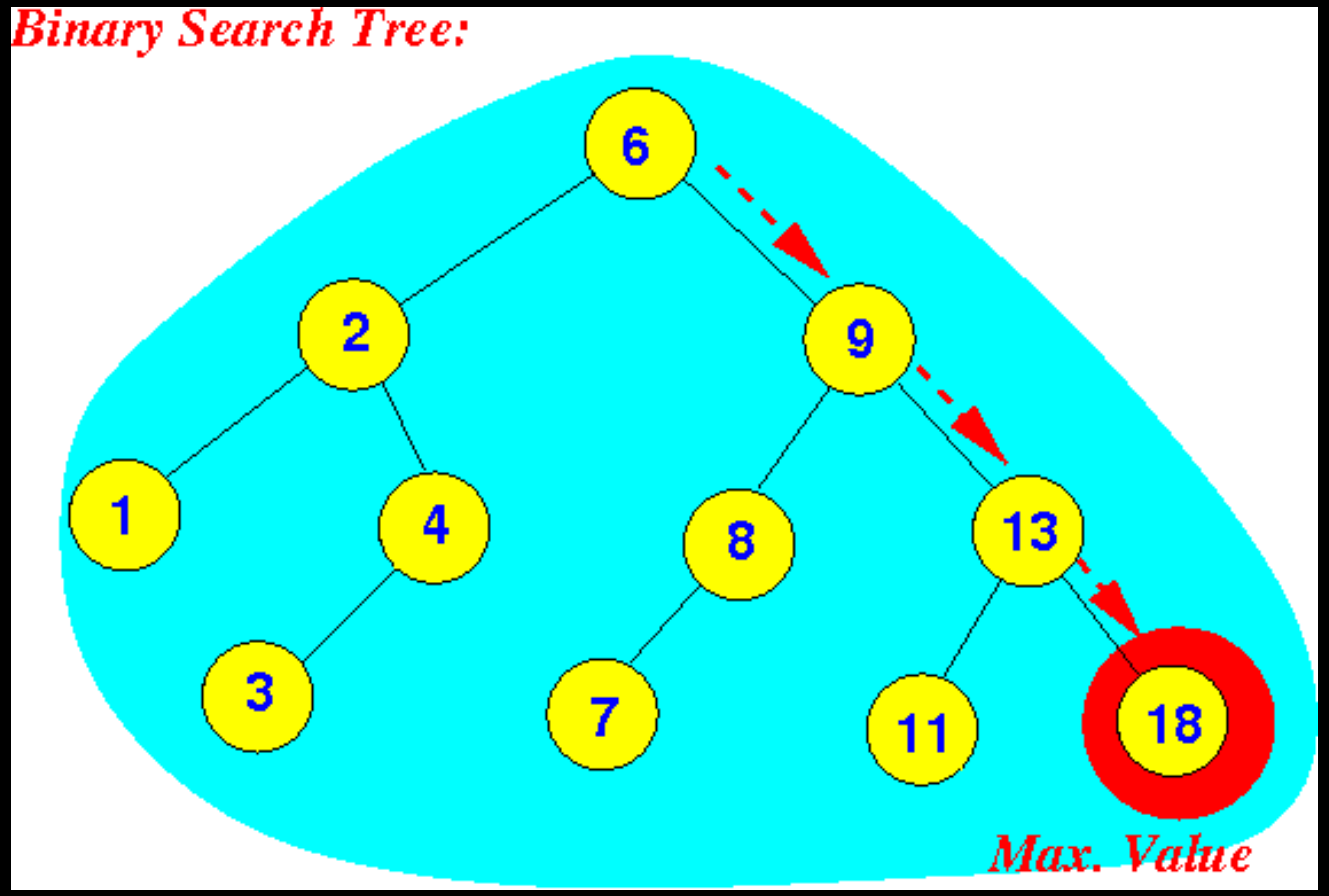
http://www.mathcs.emory.edu/~cheung/Courses/171/Syllabus/9-BinTree/BST-intro.html

איבר מינמלי



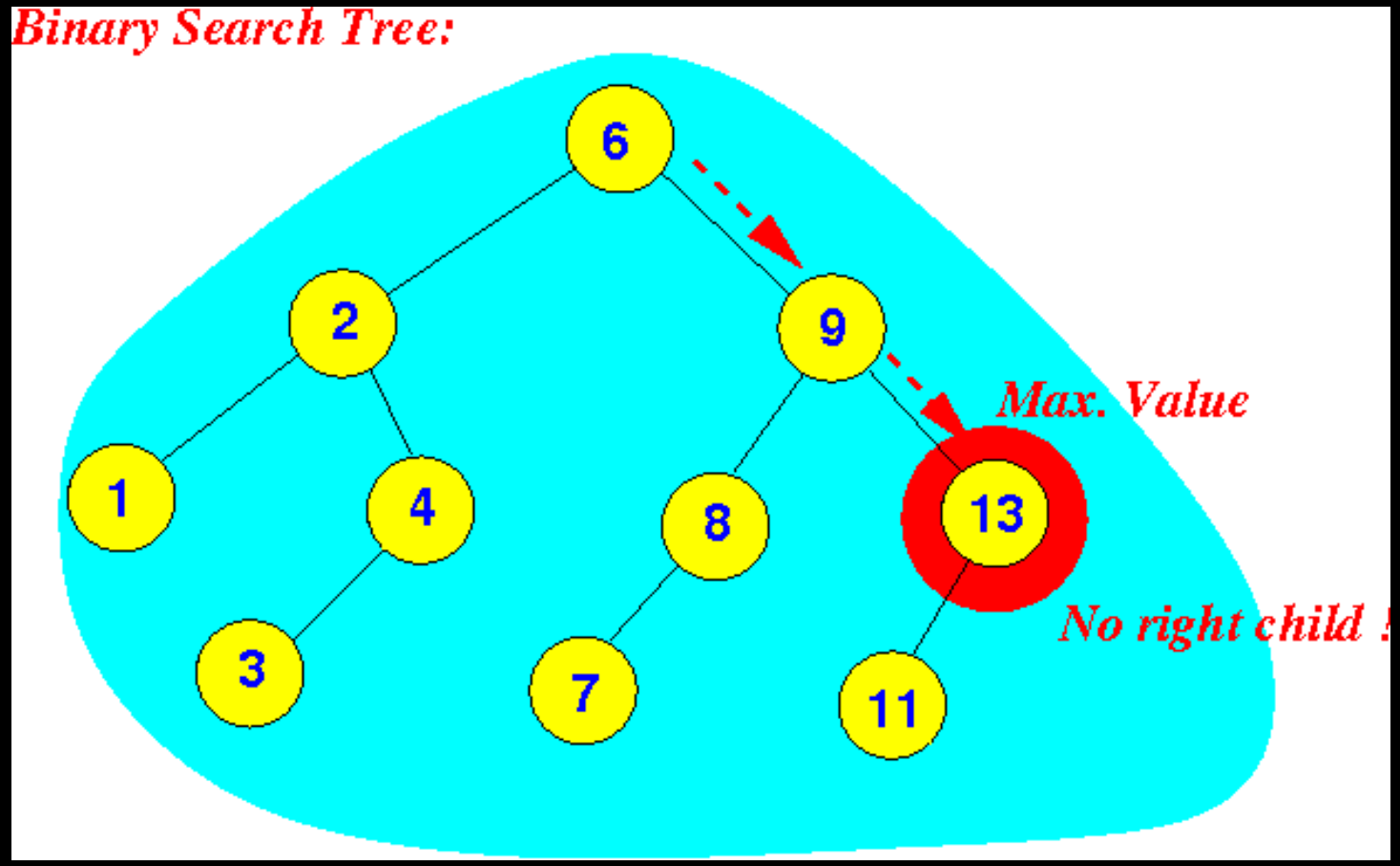
http://www.mathcs.emory.edu/~cheung/Courses/171/Syllabus/9-BinTree/BST-intro.html

איבר מקסמלי



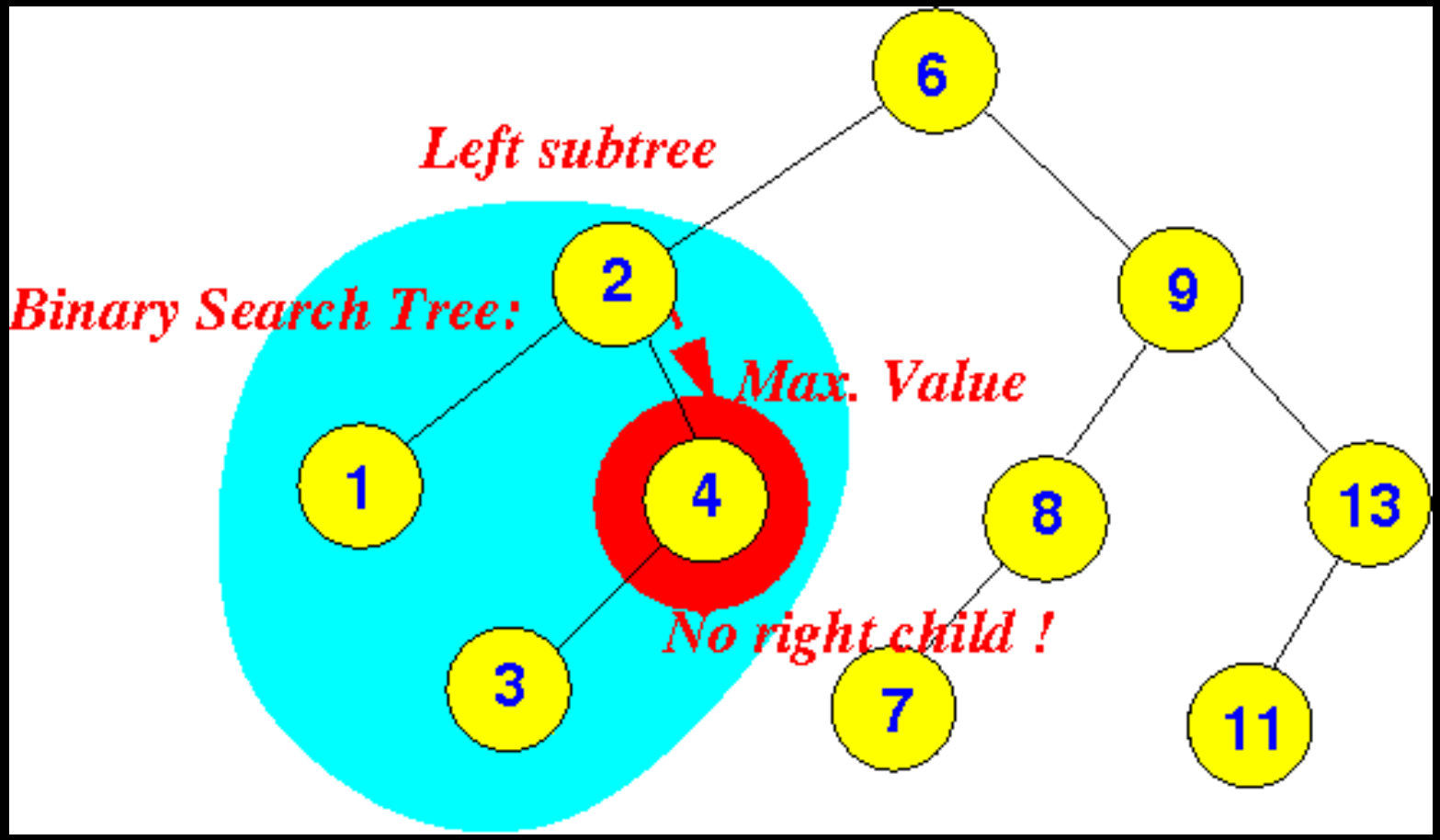
http://www.mathcs.emory.edu/~cheung/Courses/171/Syllabus/9-BinTree/BST-intro.html

איבר מקסמלי



http://www.mathcs.emory.edu/~cheung/Courses/171/Syllabus/9-BinTree/BST-intro.html

איבר מקסמלי בתת עץ



http://www.mathcs.emory.edu/~cheung/Courses/171/Syllabus/9-BinTree/BST-intro.html

מחיקה remove

מקרה 1: לקודקוד **אין** בנים

```
          50                                    50
       /      \          delete(20)          /    \
     30        70        ---------->       30      70
    /  \      /  \                          \     /  \
  20    40  60    80                        40   60    80
```

https://www.geeksforgeeks.org/binary-search-tree-set-2-delete/

מחיקה remove

מקרה 2: לקודקוד בעל בן יחיד

```
        50                                  50
     /     \          delete(30)         /    \
   30       70       ---------->      40       70
     \     /  \                              /  \
     40   60   80                           60   80
```

https://www.geeksforgeeks.org/binary-search-tree-set-2-delete/

מחיקה remove

מקרה 3: לקודקוד 2 בנים

```
        50                                  60
     /     \          delete(50)         /    \
   40       70       ---------->      40      70
            /  \                                 \
          60    80                                 80
```

https://www.geeksforgeeks.org/binary-search-tree-set-2-delete/

תכונות של עץ חיפוש בינארי

תכונות של עץ חיפוש בינארי

מספר **הכולל** של הקודקודים בעץ בינארי מושלם הינו $n = 2^{h+1} - 1$

כאשר h הוא גובה העץ

```
                18

        /   \           /   \

      40     50       100    40
```

$$n = 2^0 + 2^1 + 2^2 + \cdots 2^h = \frac{1\left(2^{h+1} - 1\right)}{2 - 1} = 2^{h+1} - 1$$

תכונות של עץ חיפוש בינארי

מספר **העלים** בעץ בינארי מושלם (Perfect) הינו 2^h

```
          18
        /    \
     15        30
    /  \      /  \
  40    50  100    40
```

תכונות של עץ חיפוש בינארי

מספר **הצמתים הפנמיים** בעץ בינארי מושלם (Perfect) הינו $2^h - 1$

```
          18
        /    \
      15      30
     /  \     /  \
   40    50 100    40
```

תכונות של עץ חיפוש בינארי

מספר **הצמתים הכולל** n בעץ בינארי שלם (Complete)

הוא בין 2^h לבין $2^{h+1} - 1$

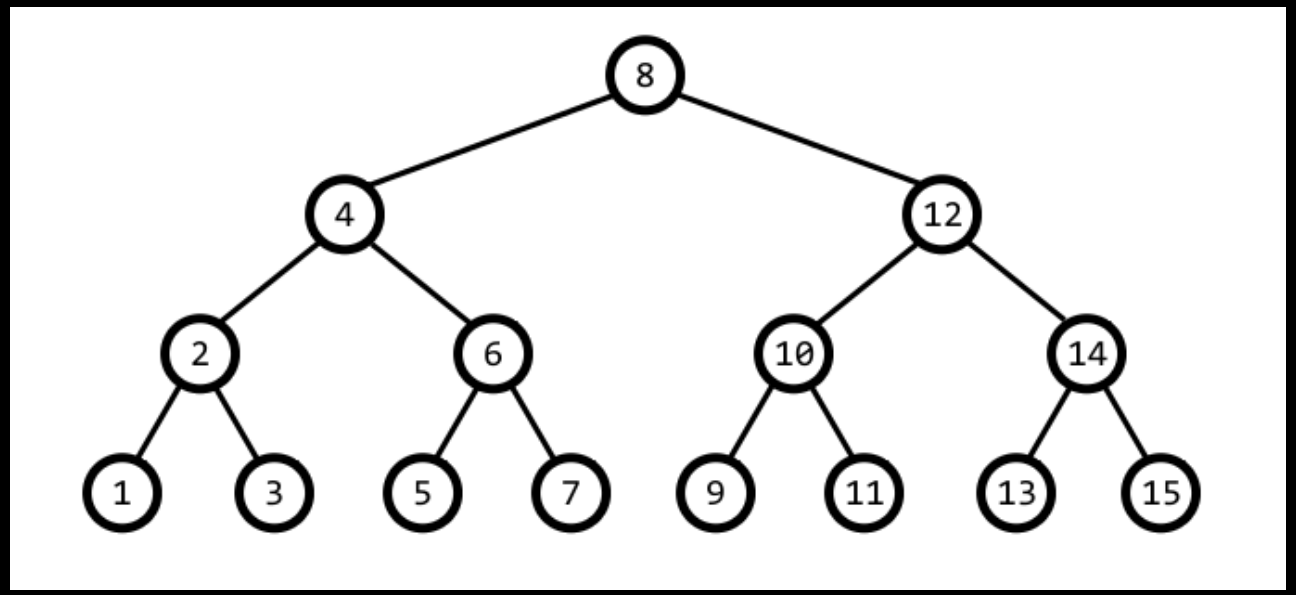
$h - 1$

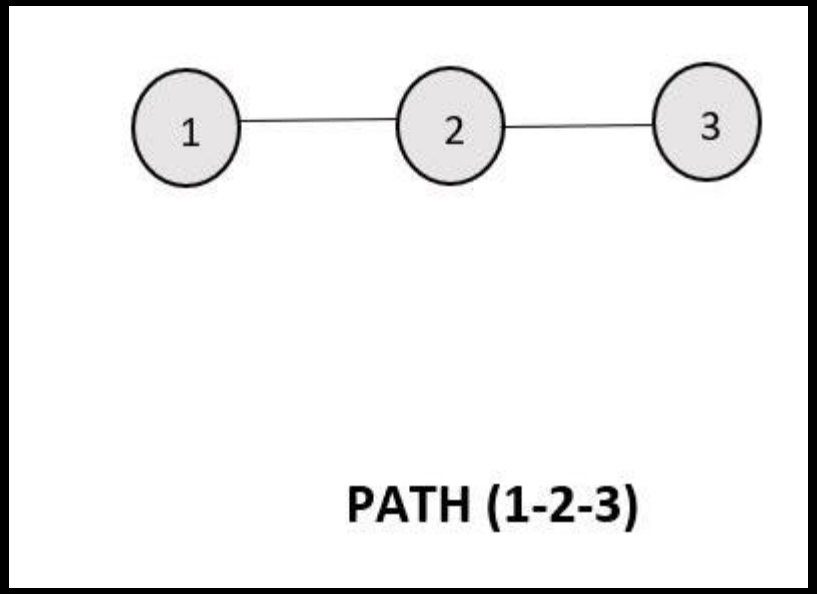
```
              18
          /         \
       15             30
      /  \           /  \
   40     50      100    40
  /  \   /
 8    7 9
```

תכונות של עץ חיפוש בינארי

גובה של עץ חיפוש בינארי הינו במקרה הטוב $O(log_2(n))$

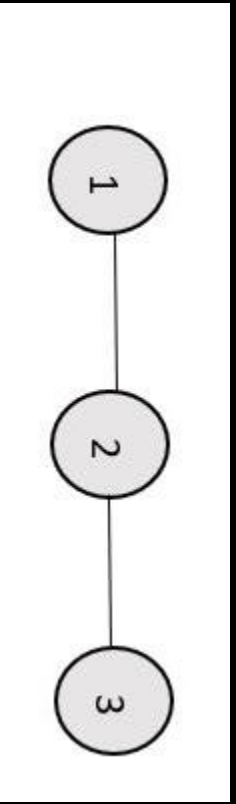
ובמקרה הרע $O(n)$





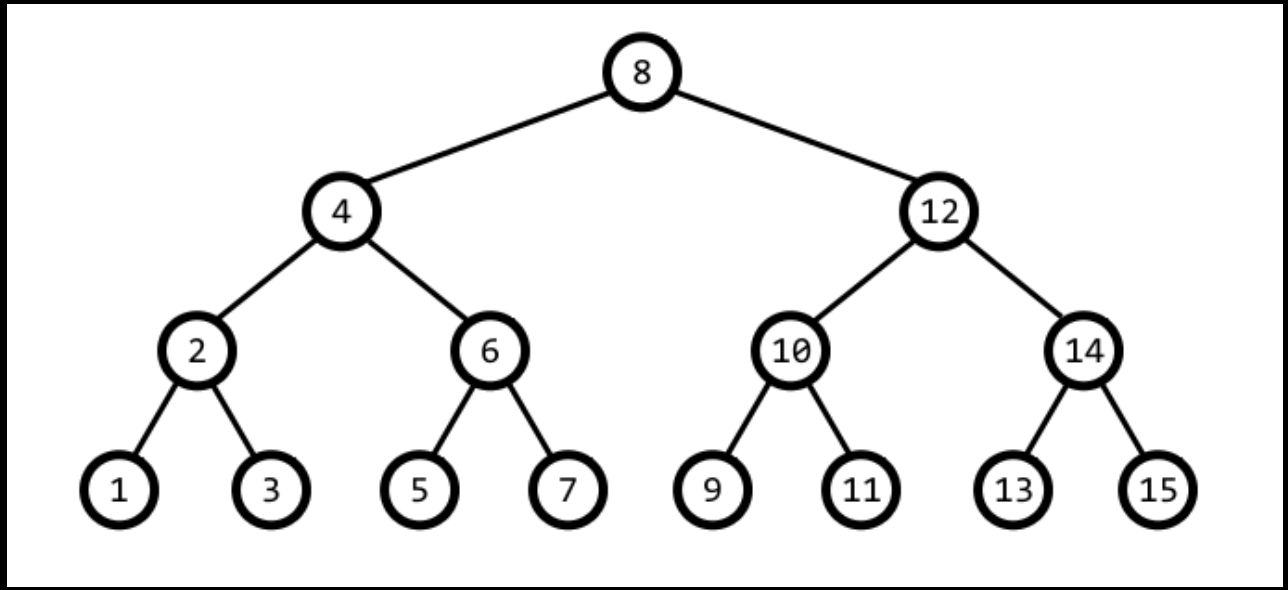
תכונות של עץ חיפוש בינארי

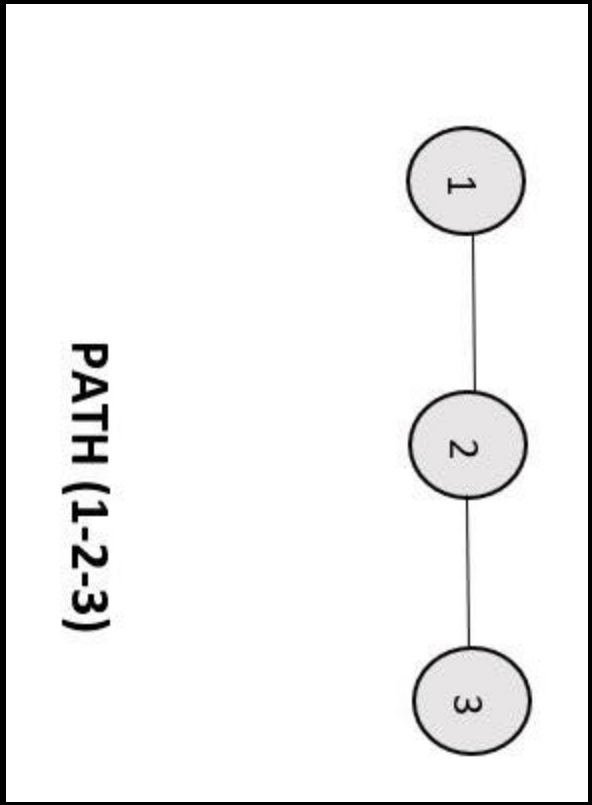
מספר הקודקודים המינמלי בעץ בינארי עם גובה h הינו $h+1$



תכונות של עץ חיפוש בינארי

כמות הצמתים הפנמיים בעץ בינארי בקטע $[h, 2^h - 1]$



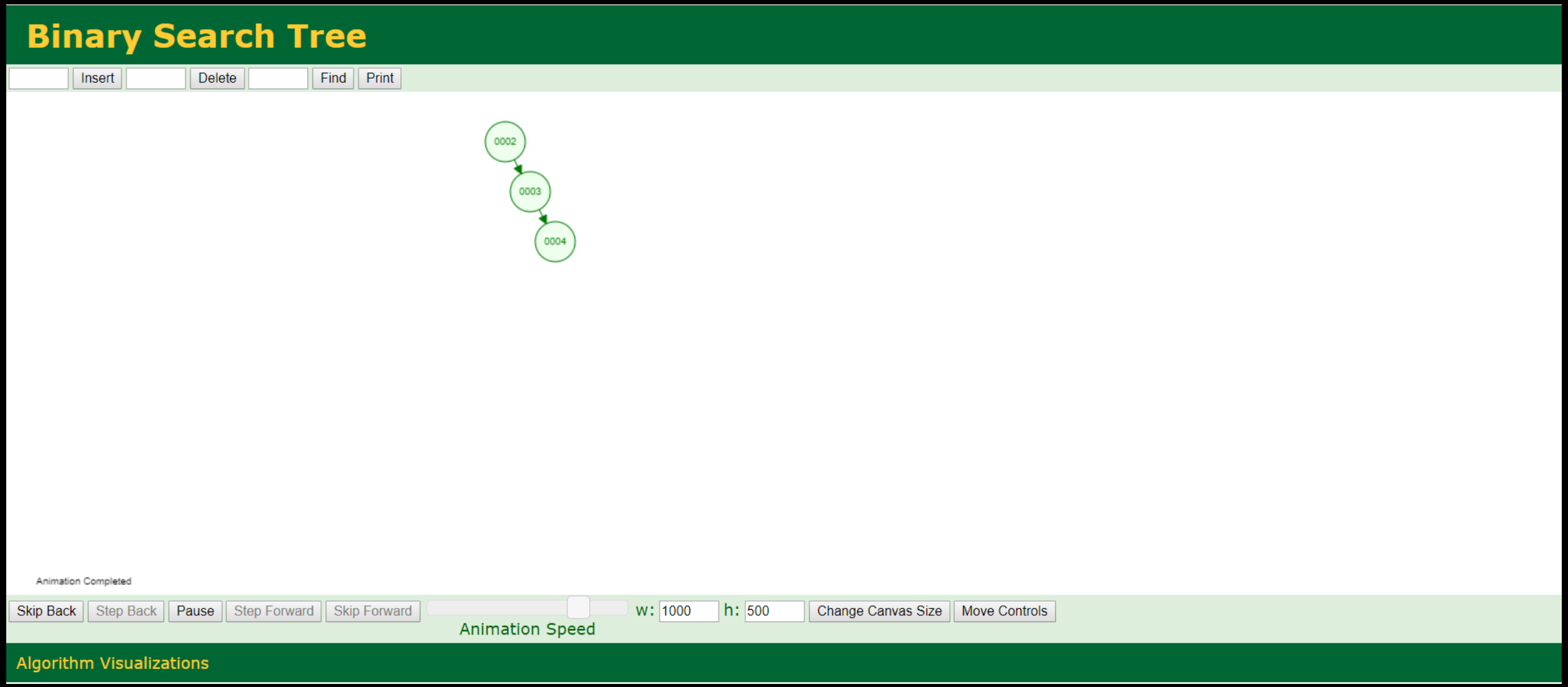


סיבוכיות BST

Data structure	Access /peek	Search	Insert /push	Delete /pop	Traverse
Linear					
Array	O(1)	O(n)	O(1)	O(n)	O(n)
Ordered array	O(1)	O(logn)	O(n)	O(n)	O(n)
Linked list	O(n)	O(n)	O(1)	O(n)	O(n)
Ordered linked list	O(n)	O(n)	O(n)	O(n)	O(n)
Matrix	O(1)	O(n^2)	O(1)	O(n^2)	O(n^2)
Stack	O(1)	O(n)	O(1)	O(1)	O(n)
Queue	O(1)	O(n)	O(1)	O(1)	O(n)
Non-Linear					
Tree (worst case)	O(n)	O(n)	O(n)	O(n)	O(n)
Tree (balanced)	O(logn)	O(logn)	O(logn)	O(logn)	O(n)

Improving a BST can be done by making it balanced - like AVL or red-black-trees.

visualization



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

צבי מינץ

מימוש BST

הרצאה + תרגול הבא

צבי מינץ

מימוש Node.java

```
/* Class containing left and right
child of current node and key value*/
public class Node {
   int data;
   Node right;
   Node left;
   public Node(int data) {
      this.data = data;
      left = right = null;
   }
}
```

מימוש Main.java

```
public class Main {
    public static void main(String[] args) {
```

```
 /----- 9
5
 \----- 4
          |        /----- 2
          \----- 1
```

```
    }
}
```

מימוש BST.java

```
public class BST
{

    ...
}
```