

## עץ חיפוש בינארי Binary-Search Tree (BST)

---

---

---

---

---

---

---

- מה נלמד?
- הגדרת עץ חיפוש בינארי
  - ייצוג של עץ חיפוש בינארי
  - פעולות על עץ חיפוש בינארי

---

---

---

---

---

---

---

### Ordered Dictionary

*Search(k)*

*Min()*

*Insert(x)*

*Max()*

*Delete(x)*

*Successor(x)*

*Predecessor(x)*

---

---

---

---

---

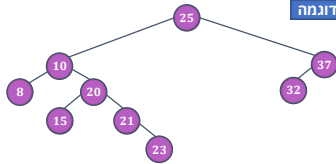
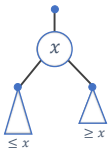
---

---



### עץ חיפוש בינארי - הגדרה

עץ חיפוש בינארי הוא עץ בינארי המקיים שנקראת **תכונת-עץ-החיפוש-הבינארי**.  
יהי  $x$  צומת בעץ חיפוש בינארי, אם  $y$  הוא צומת בתת-עץ השמאלי של  $x$   $z-1$  צומת בתת-עץ הימני של  $x$ , אז  $y.key \leq x.key \leq z.key$ .



דוגמה

---

---

---

---

---

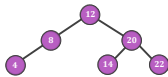
---

---

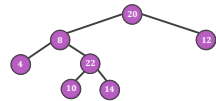
---

### אילו מהעצים הבאים הם עצי חיפוש בינאריים?

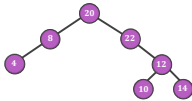
?



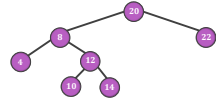
2.



1.



4.



3.

---

---

---

---

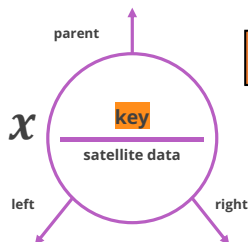
---

---

---

---

### ייצוג של עץ חיפוש בינארי



העץ נתון על ידי מצביע לשורש  $T.root$

---

---

---

---

---

---

---

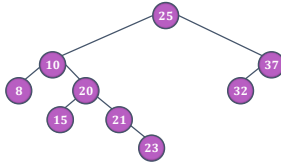
---

### סריקה תוכנית של עץ חיפוש בינארי

```

Inorder(x)
1 if (x ≠ NULL)
2   Inorder(x.left)
3   print (x.key)
4   Inorder(x.right)

```



8 10 15 20 21 23 25 32 37

---

---

---

---

---

---

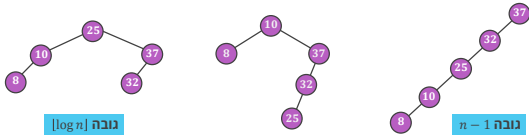
---

---

### גובה של עץ חיפוש בינארי

עבור קבוצה נתונה של מפתחות ניתן לבנות הרבה עצי חיפוש בינאריים

דוגמה: עבור קבוצת מפתחות 8,10,25,37,32



הערה: גובה של עץ חיפוש בינארי יכול לקבל כל ערך בין  $\lceil \log n \rceil$  ל- $n - 1$

---

---

---

---

---

---

---

---

### Ordered Dictionary

Search(k)

Min()

Insert(x)

Max()

Delete(x)

Successor(x)

Predecessor(x)

---

---

---

---

---

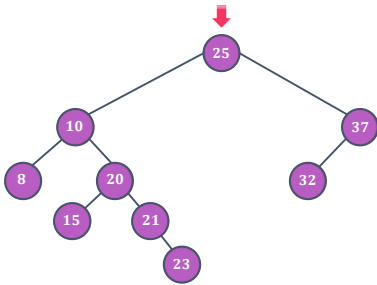
---

---

---

פעולת חיפוש

Search 21




---

---

---

---

---

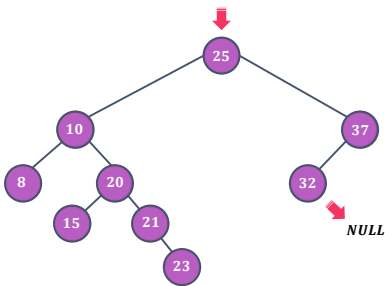
---

---

---

פעולת חיפוש

Search 35




---

---

---

---

---

---

---

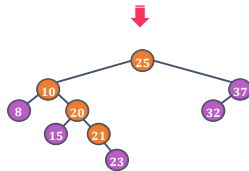
---

חיפוש פסאודו קוד

```

Search(x, k)
1 if x = NULL or k = x.key
2   return x
3 if k < x.key
4   return Search(x.left, k)
5 else return Search(x.right, k)

```

קריאה חיצונית  $Search(T.root, k)$ 


---

---

---

---

---

---

---

---

זמן ריצה במקרה הגרוע של פעולת *Search* בעץ חיפוש בינארי  
בעל  $n$  צמתים וגובה  $h$  הוא:

?

1.  $\theta(n)$
2.  $\theta(1)$
3.  $\theta(h)$
4.  $\theta(\log n)$

---

---

---

---

---

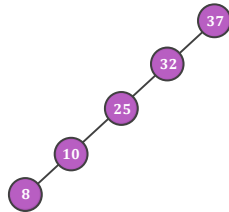
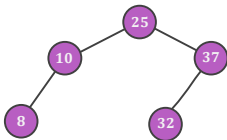
---

---

---



פעולת חיפוש  
זמן ריצה




---

---

---

---

---

---

---

---

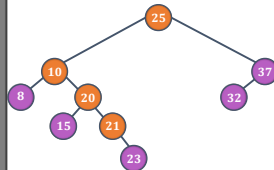


חיפוש פסאודו קוד

```

Search(x, k)
1 if x = NULL or k = x.key
2   return x
3 if k < x.key
4   return Search(x.left, k)
5 else return Search(x.right, k)

```



זמן ריצה הוא  $\theta(h)$ , כאשר  $h$  הוא גובה העץ

---

---

---

---

---

---

---

---

החפשים את 52 בעץ חיפוש בינארי.  
איזה מבין הסדרות הבאות **אינה** יכולה להיות סדרת מספרים בה נתקל  
במהלך החיפוש?

?

1. 9,10,20,34,52

2. 9,100,30,42,52

3. 9,42,32,40,52

4. 9,80,15,70,52

5. כל הסדרות אפשריות

---

---

---

---

---

---

---

#### Ordered Dictionary

Search(k)

Min()

Insert(x)

Max()

Delete(x)

Successor(x)

Predecessor(x)

---

---

---

---

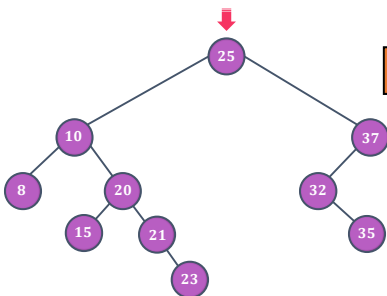
---

---

---

#### פעולת הכנסה

Insert 35




---

---

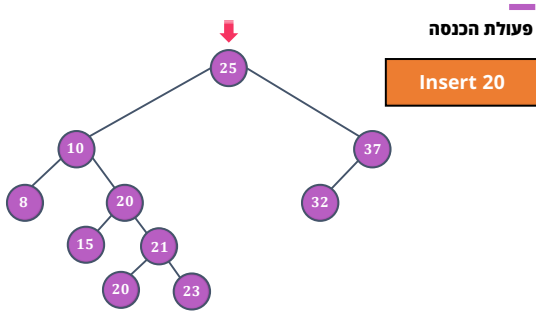
---

---

---

---

---




---

---

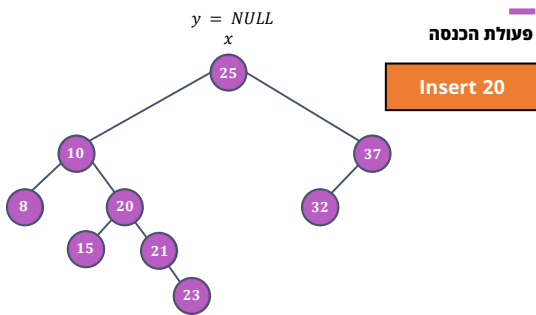
---

---

---

---

---




---

---

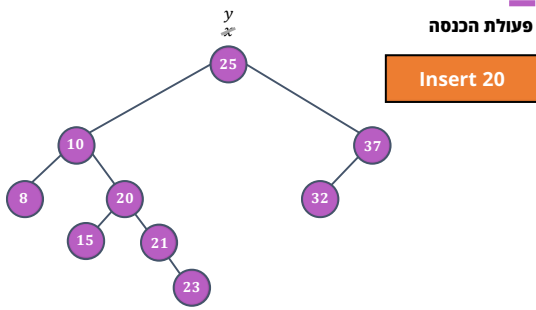
---

---

---

---

---




---

---

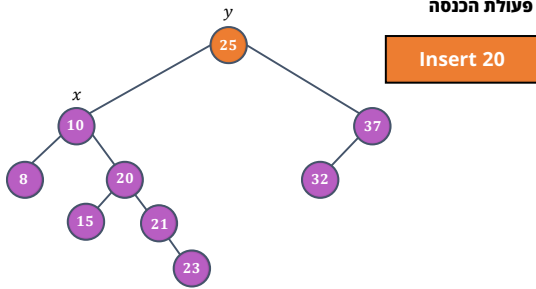
---

---

---

---

---




---

---

---

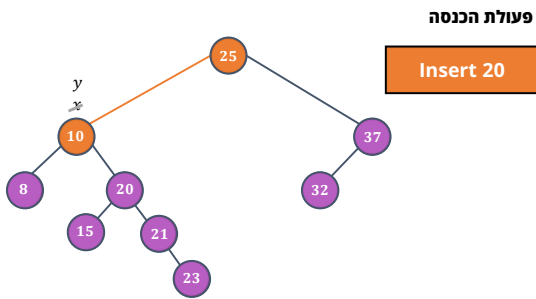
---

---

---

---

---




---

---

---

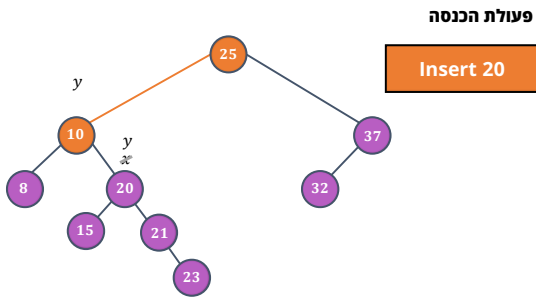
---

---

---

---

---




---

---

---

---

---

---

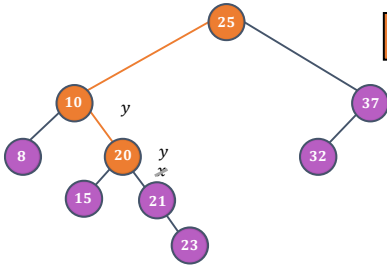
---

---



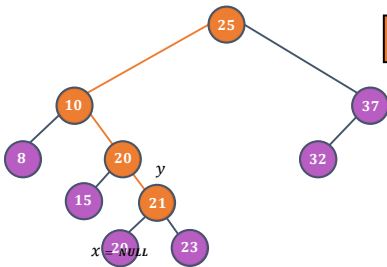
## פעולת הכנסה

Insert 20



## פעולת הכנסה

Insert 20



## Insert (T, z)

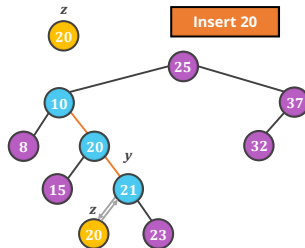
```

1 y ← NULL
2 x ← T.root
3 while (x ≠ NULL)
4     y ← x
5     if (z.key < x.key)
6         x ← x.left
7     else x ← x.right
8 z.p ← y
9 if (y = NULL) // T was empty
10    T.root ← z
11 elseif (z.key < y.key)
12    y.left ← z
13 else y.right ← z

```

## הכנסה פסאודו קוד

Insert 20



זמן ריצה במקרה הגרוע של פעולת Insert בעץ חיפוש בינארי  
בעל  $n$  צמתים וגובה  $h$  הוא:

?

1.  $\theta(n)$
2.  $\theta(1)$
3.  $\theta(h)$
4.  $\theta(\log n)$

---

---

---

---

---

---

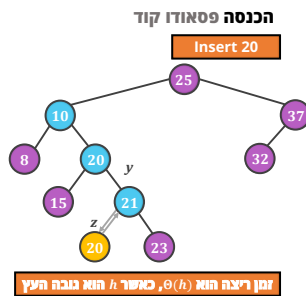
---

---

```

Insert(T, z)
1 y ← NULL
2 x ← T.root
3 while (x ≠ NULL)
4   y ← x
5   if (z.key < x.key)
6     x ← x.left
7   else x ← x.right
8 z.p ← y
9 if (y = NULL) // T was empty
10  T.root ← z
11 elseif (z.key < y.key)
12  y.left ← z
13 else y.right ← z

```




---

---

---

---

---

---

---

---

מפתחות הבאים הוכנסו לעץ חיפוש בינארי ריק מהתחלה:  
11, 2, 4, 6, 14, 13, 20 (משמאל לימין). מה הוא גובה העץ שהתקבל?

?

1. 2
2. 3
3. 4
4. 6

---

---

---

---

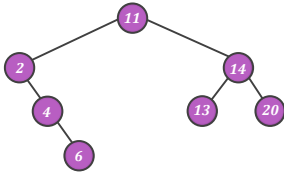
---

---

---

---

מפתחות הבאים הוכנסו לעץ חיפוש בינארי ריק מהתחלה:  
11, 2, 4, 6, 14, 13, 20 (משמאל לימין). מה הוא גובה העץ שהתקבל?



- 2 ☐ 1
- 3 ☒ 2
- 4 ☐ 3
- 6 ☐ 4

---

---

---

---

---

---

---

---

#### Ordered Dictionary

*Search(k)*

*Min()*

*Insert(x)*

*Max()*

*Delete(x)*

*Successor(x)*

*Predecessor(x)*

---

---

---

---

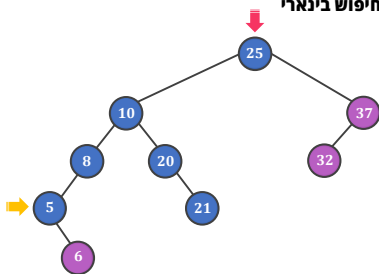
---

---

---

---

מציאת מינימום בעץ חיפוש בינארי




---

---

---

---

---

---

---

---

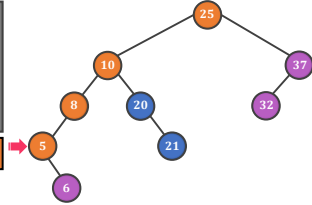
## מינימום פסאודו קוד

```

Min(x)
1 while x.left ≠ NULL
2   x ← x.left
3 return x

Min(T.root)

```




---

---

---

---

---

---

---

---

השלימו את המשפט:  
 זמן ריצה במקרה הגרוע של פעולת  $Min$  בעץ חיפוש בינארי  
 בעל  $n$  צמתים וגובה  $h$  הוא:

?

1.  $\theta(n)$
2.  $\theta(1)$
3.  $\theta(h)$
4.  $\theta(\log n)$

---

---

---

---

---

---

---

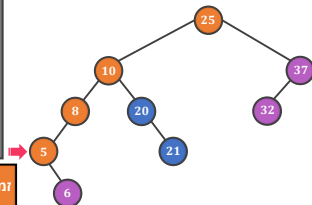
---

## מינימום פסאודו קוד

```

Min(x)
1 while x.left ≠ NULL
2   x ← x.left
3 return x

```



זמן ריצה הוא  $\theta(h)$ , כאשר  $h$  הוא גובה העץ

---

---

---

---

---

---

---

---

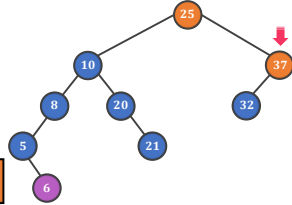
### מקסימום פסאודו קוד

```

Max
Min(x)
  1 while x.left ≠ NULL
    2   x = x.left
  3 return x

```

זמן ריצה הוא  $\Theta(h)$ , כאשר  $h$  הוא גובה העץ




---

---

---

---

---

---

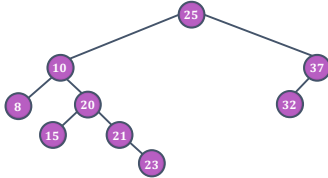
---

---



### מציאות עוקב לצומת הגדרה

אם כל המפתחות שונים זה מזה, העוקב (Successor) לצומת  $x$  הוא הצומת בעל מפתח הקטן ביותר הגדול מ- $x.key$




---

---

---

---

---

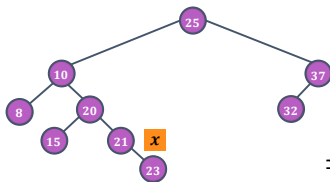
---

---

---

מיהו צומת עוקב לצומת  $x$  בעל מפתח 23 ?

?



- 21 .1
- 32 .2
- 25 .3
- אין לצומת  $x$  עוקב .4

---

---

---

---

---

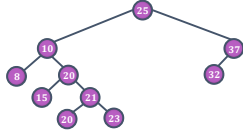
---

---

---

### מציאות עוקב לצומת הגדרה

אם כל המפתחות שונים זה מזה, העוקב (Successor) לצומת  $x$  הוא הצומת בעל מפתח הקטן ביותר הגדול מ- $x.key$ .  
 במידה והמפתחות יכולים לחזור על עצמם, העוקב לצומת  $x$  הוא צומת הבא בסדר הממוין הנקבע על ידי סריקת inorder של העץ




---

---

---

---

---

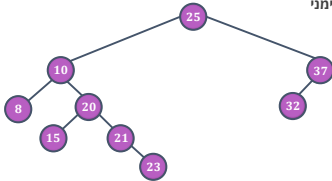
---

---

---

### מציאות עוקב לצומת

- מקרה פשוט
- לצומת  $x$  יש תת עץ ימני
- החזר מינימאלי בתת העץ הימני




---

---

---

---

---

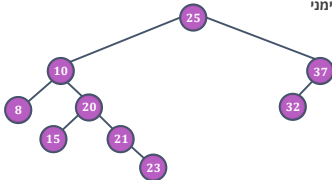
---

---

---

### מציאות עוקב לצומת

- מקרה פשוט
- לצומת  $x$  יש תת עץ ימני
- החזר מינימאלי בתת העץ הימני



- אחרת:
- יש לעלות במסלול לשרשרת עד "לפנייה ראשונה ימינה"

---

---

---

---

---

---

---

---

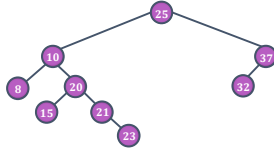
### מציאת עוקב לצומת פסאודו קוד

Successor (x)

```

1 if x.right ≠ NULL
2   return Min(x.right)
3 y ← x.parent
4 while y ≠ NULL and x = y.right
5   x ← y
6   y ← y.parent
7 return y

```




---

---

---

---

---

---

---

---

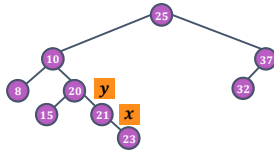
### מציאת עוקב לצומת פסאודו קוד

Successor (x)

```

1 if x.right ≠ NULL
2   return Min(x.right)
3 y ← x.parent
4 while y ≠ NULL and x = y.right
5   x ← y
6   y ← y.parent
7 return y

```




---

---

---

---

---

---

---

---

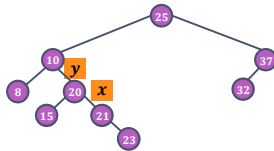
### מציאת עוקב לצומת פסאודו קוד

Successor (x)

```

1 if x.right ≠ NULL
2   return Min(x.right)
3 y ← x.parent
4 while y ≠ NULL and x = y.right
5   x ← y
6   y ← y.parent
7 return y

```




---

---

---

---

---

---

---

---

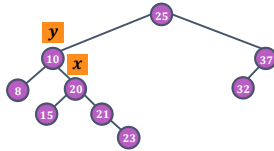
### מציאת עוקב לצומת פטאודו קוד

Successor (x)

```

1 if x.right ≠ NULL
2   return Min(x.right)
3 y ← x.parent
4 while y ≠ NULL and x = y.right
5   x ← y
6   y ← y.parent
7 return y

```




---

---

---

---

---

---

---

---

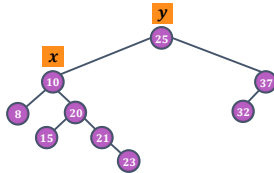
### מציאת עוקב לצומת פטאודו קוד

Successor (x)

```

1 if x.right ≠ NULL
2   return Min(x.right)
3 y ← x.parent
4 while y ≠ NULL and x = y.right
5   x ← y
6   y ← y.parent
7 return y

```




---

---

---

---

---

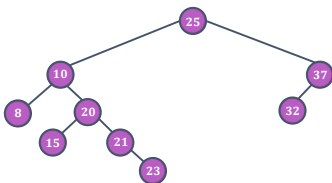
---

---

---

השלימו את המשפט:  
זמן ריצה במקרה הגרוע של פעולת Successor בעץ חיפוש בינארי בעל  $n$  צמתים וגובה  $h$  הוא:

?



- $\theta(n)$  .1  
 $\theta(1)$  .2  
 $\theta(h)$  .3  
 $\theta(\log n)$  .4

---

---

---

---

---

---

---

---



### מציאת עוקב לצומת פטאוודו קוד

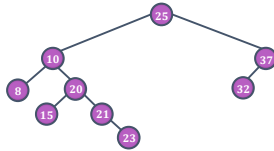
Successor (x)

```

1 if x.right ≠ NULL
2   return Min(x.right)
3 y ← x.parent
4 while y ≠ NULL and x = y.right
5   x ← y
6   y ← y.parent
7 return y

```

זמן ריצה הוא  $O(h)$ , כאשר  $h$  הוא גובה העץ




---

---

---

---

---

---

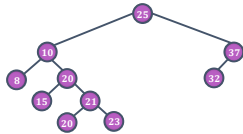
---

---



### קודם לצומת הגדרה

אם כל המפתחות שונים זה מזה, הקודם (Predecessor) לצומת  $x$  הוא הצומת בעל מפתח הגדול ביותר הקטן מ- $x.key$ .  
 במידה והמפתחות יכולים לחזור על עצמם, הקודם לצומת  $x$  הוא צומת הקודם בסדר הממוין הנקבע על ידי סריקת inorder של העץ




---

---

---

---

---

---

---

---

### סיכום

פעולות המילון  $Predecessor, Successor, Min, Max$   
 ניתנות למימוש על עץ חיפוש בינארי בגובה  $h$  בזמן  $O(h)$

---

---

---

---

---

---

---

---

### Ordered Dictionary

<code>Search(k)</code>	<code>Min()</code>
<code>Insert(x)</code>	<code>Max()</code>
<code>Delete(x)</code>	<code>Successor(x)</code>
	<code>Predecessor(x)</code>

---

---

---

---

---

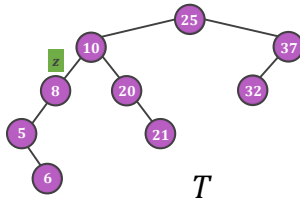
---

---

---

### מחיקה בעץ חיפוש בינארי

נתון: עץ חיפוש בינארי  $T$   
מבצע לצומת  $z$  בעץ שיש למחוק




---

---

---

---

---

---

---

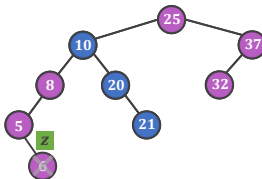
---

### מחיקה בעץ חיפוש בינארי

ישנם שלושה מקרים:

ל- $z$  אין בנים:  
מעדכנים את אביו  $z.parent$  כך שבנו יהיה  $NULL$  ולא  $z$

1




---

---

---

---

---

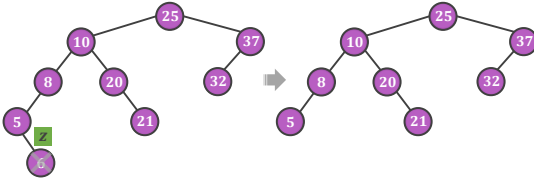
---

---

---

### מחיקה בעץ חיפוש בינארי ישנם שלושה מקרים:

1 ל- $z$  אין בנים:  
מעבירים את אביו  $z.parent$  כך שבנו יהיה NULL ולא  $z$




---

---

---

---

---

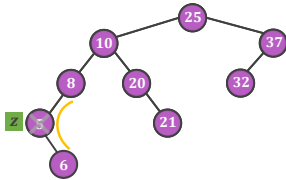
---

---

---

### מחיקה בעץ חיפוש בינארי ישנם שלושה מקרים:

2 ל- $z$  יש רק בן אחד:  
הבן של  $z$  הופך להיות בן של  $z.parent$




---

---

---

---

---

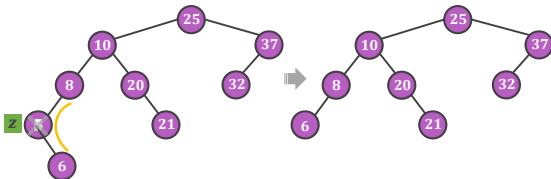
---

---

---

### מחיקה בעץ חיפוש בינארי ישנם שלושה מקרים:

2 ל- $z$  יש רק בן אחד:  
הבן של  $z$  הופך להיות בן של  $z.parent$




---

---

---

---

---

---

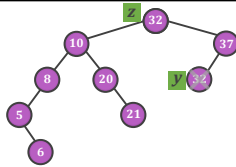
---

---

**מחיקה בעץ חיפוש בינארי**  
**ישנם שלושה מקרים:**

ל- $z$  יש שני בנים:  
לעוקב של  $z$ , נסמן  
נסיר את  $y$  (מקרה  
המפתח והנתונים

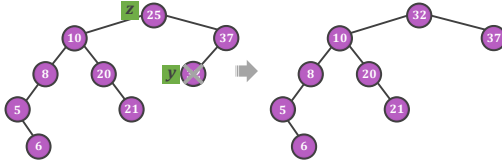
3

[illegible]

**מחיקה בעץ חיפוש בינארי**  
**ישנם שלושה מקרים:**

ל- $z$  יש שני בנים:  
לעוקב של  $z$ , נסמך  
נסיר את  $y$  (מקרה  
המפתח והנתונים

3

[illegible]

**מחיקה בעץ חיפוש בינארי**  
**ישנם שלושה מקרים**

ל- $z$  אין בנים:  
משנים את אביו  $z.parent$  כך שבנו יהיה  $NULL$  ולא  $z$

1

ל- $z$  יש רק בן אחד:  
הבן של  $z$  הופך להיות בן של  $z.parent$

2

ל- $z$  יש שני בנים:  
לעוקב של  $z$ , נסמן ב- $y$ , אין בן שמאלי  
נסיר את  $y$  (מקרה 1 או 2)  
המפתח והנתונים נלווים של  $z$  מוחלפים באלו של  $y$

3

[illegible]

השלימו את המשפט:

זמן ריצה במקרה הגרוע של פעולת *Delete* בעץ חיפוש בינארי בעל  $n$  צמתים וגובה  $h$  הוא \_\_\_\_.

?

1.  $\theta(n)$
2.  $\theta(1)$
3.  $\theta(h)$
4.  $\theta(\log n)$

---

---

---

---

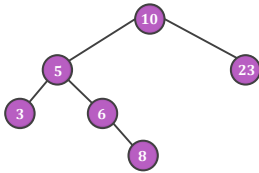
---

---

---

נתון עץ חיפוש בינארי מייד אחרי שבוצעה עליו פעולה אחת - הכנסה או הוצאה. מה יכולה להיות הפעולה שבוצעה על העץ? סמנו את כל המשובות האפשריות

?



1. *Insert*(5)
2. *Insert*(8)
3. *Insert*(10)
4. *Delete*(20)
5. *Delete*(9)

---

---

---

---

---

---

---

# סיכום

---

---

---

---

---

---

---

מה למדנו?

1

*Search(k)*

*Min()*

*Insert(x)*

*Max()*

*Delete(x)*

*Successor(x)*

*Predecessor(x)*

מה למדנו?

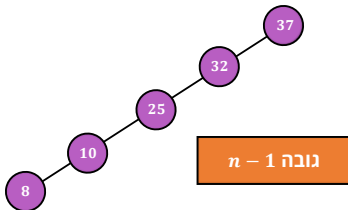
2

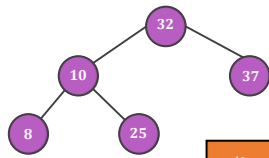


זמן ריצה של הפעולות על עץ חיפוש בינארי הוא  $O(h)$

מה למדנו?

3





גובה  $\lceil \log n \rceil$

מה למדנו?

4

---

---

---

---

---

---

---