# מבני נתונים ואלגוריתמים – תרגול #3

*נושאים: תור קדימויות/ערימה, עצים*

<u>חזרה</u>

מבנה נתונים – אמצעי לאחסון נתונים במחשב. יש הרבה סוגים שונים, וצריך להשתמש במבנה שהכי מתאים לבעיה שלנו מבחינת שימוש בנתונים – הוספה, מחיקה וחיפוש. צריך לשים שיש מבני נתונים מופשטים (ADT – abstract data type) שמגדירים ממשק מסוים ללא מימוש. למשל, מחסנית הוא מבנה נתונים מופשט, שניתן לממש באמצעות תור או רשימה מקושרת (שהם גם מבני נתונים).

מערך – הכנסה/הוצאה- O(n), גישה – O(1), מוגבל במקום.

רשימה מקושרת - הכנסה/הוצאה- O(1), גישה – O(n), לא מוגבל במקום.

- רשימה דו-מקושרת – יש גם מצביע לאיבר הקודם.

מחסנית – (Last In First Out).

- מימוש באמצעות מערך: כל הפעולות O(1). אם נגמר המקום, צריך להכפיל את גודל המערך ולהעתיק הכל,לוקח O(n).
- מימוש באמצעות רשימה מקושרת: כל הפעולות O(1). אם רוצים לרוקן את המחסנית, צריך לשחרר את כל ההקצאות הדינמיות, לוקח O(n).

תור – (First In First Out).

- מימוש באמצעות מערך: מכיוון שכל הזמן מכניסים איברים, התור "זז" וזה מבזבז מקום. לכן נתייחס למערך כמעגל.
- מימוש באמצעות רשימה מקושרת:

## תור עדיפויות/ערימה

הרעיון – כמו תור רגיל, אך מחזיקים בנוסף ערך שמתאר את העדיפות של הנתון. לדוגמא – Scheduler של תוכניות במחשב.

מימושים נאיביים:

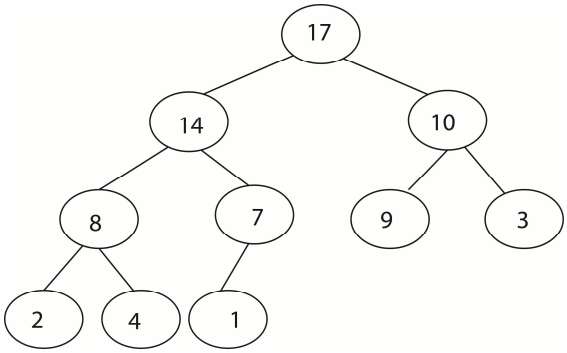
1) מכניסים את האיברים לפי הסדר (כמו תור רגיל), וכאשר מוצאים איבר מחפשים את האיבר עם הקדימות הגבוהה ביותר. הכנסה – O(1), הוצאה – O(n).
2) מכניסים את האיברים לפי הקדימויות. הכנסה – O(n), הוצאה – O(1).

מימוש נוסף: אם יש מספר קטן של קדימויות, ניתן ליצור תור עבור כל קדימות, ונוציא איבר מהתור בעל הקדימות הגבוהה ביותר.

מימוש יעיל יותר – <u>ערימה- Heap</u>.

ערימה – מערך שניתן לראות כעף בינארי כמעט מלא (=עץ בינארי שבו כל הרמות מלאות פרט אולי לרמה האחרונה, שמלאה מצד שמאל עד לנקודה מסוימת).

דוגמא לערימת מקסימום– A = [ 17 14 10 8 7 9 3 2 4 1 ]




"תכונת הערימה" חייבת להתקיים : A[Parent(i)] ≥ A[i] (ערימת מקימום).

- בהקשר של תור קדימויות – הערכים הם הקדימויות.

מציאת ערכים: בהינתן אינדקס i של צומת:

אבא - $\left\lfloor \frac{i}{2} \right\rfloor$, בן שמאלי - $2i$, בן ימני - $2i+1$.

- גובה העץ - $\log_2 n$.

מימוש:

- יותר נוח למלא את המערך מאינדקס 1.
- תריך מצביע לאיבר האחרון במערך - *end*.

פעולות בערימה:

*Heap-size*– מספר האיברים בערימה.

***Create (n):***
```
A – array of size n
end = 0;
```

***Push (x):***
```
A[end++] = x;
i = end;
while i>1 and A[i]>A[Parent(i)]
        k = A[Parent(i)];
        switch(A[i],A[k]);
        i=k;
```

***x = Pop():***
```
if end==0
        error(empty)
else
        x=A[i];
        A[1]=A[end];
        end--;
        Heapify(1);
        return x;
```

***Heapify(i):***

```
l =  Left(i);
r = Right(i);
if l≤heap-size and A[l] >A[i]
        max = l;
if r≤heap-size and A[r] >A[max]
        max = r;
if max ≠ i
        switch( A[i],A[max])
        Heapify(max);
```

תרגיל:

נתונות k רשימות ממוינות באורך n. מזגו את הרשימות לרשימה אחת ממויינת ב-$\theta(nk\log k)$.

פיתרון:

- בכל צעד רוצים למצוא את האיבר הקטן ביותר ולהכניס למערך הפלט.
- נחזיק $k$ אינדקסים (אחד לכל רשימה). כל פעם נמצא
- כל פעם נמצא את $r$ כך ש- $L_r[i_r] = \min_{s\in\{1,\dots k\}} L_s[i_s]$ - כלומר רוצים למצוא את הרשימה שהאיבר הראשון בה הכי קטן.

$L_1, \dots, L_k$ הן הרשימות , $L_r$ היא הרשימה הממוזגת ו-$i_r$ הוא האיבר הנוכחי שמוסיפים.

האלגוריתם:

```
H – heap of pairs (list index + number)
L_r - array of size nk
i_r =  i_1 = i_2 = ⋯ = i_k = 0
for j=1 to k:
        H.Push(L_j[i_j], j)
While H.IsEmpty==false
        (a,j) = H.Pop();
        L_r[i_r] = a;
        i_j + +;
        i_r + +;
        if  i_j < n
                H.Push(L_j[i_j], j)
```

O

$O(nk\log k)$

- אם היינו מוצאים את $i_r$ ע"י מעבר על כל הרשימות, הסיבוכיות הייתה $O(nk^2)$.

## עצים

נושאים:

- עצי חיפוש
- עצים מאוזני גובה: עצי 2-3 ועצי AVL

הגדרות:

עץ בינארי: עץ שבו לכל צומת יש לכל היותר 2 בנים.

עץ בינארי מלא: עץ בינארי בו לכל צומת פנימי יש בדיוק 2 בנים.

עץ בינארי שלם: עץ בינארי בו כל העלים באותו עומק.

עץ בינארי כמעט שלם: עץ שלם שהוצאו ממנו עלים מצד ימין.

תכונות של עץ בינארי שלם: ( n צמתים, L עלים וגובה h)

מספר צמתים בעומק i: $n_i = 2^i$

מספר עלים: $L = n_h = 2^h$

מספר צמתים בעץ: $n = \sum_{i=0}^{h} n_i = \sum_{i=0}^{h} 2^i = 2^{h+1} - 1$

גובה של העץ - $h = \log_2(n+1) - 1 = O(\log_2 n)$

**עצי חיפוש**

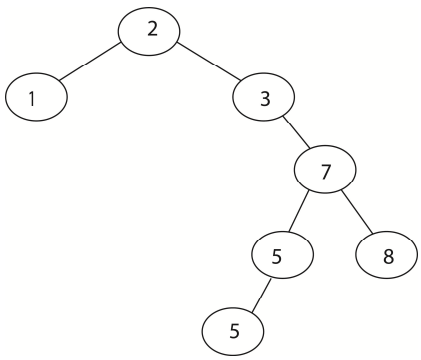
עץ בינארי שבו מתקיים הכלל הבא:

X צומת בעץ חיפוש בינארי.

אם y הוא צומת בתת-העץ <u>השמאלי</u> של x – מתקיים ש-$\boldsymbol{x \geq y}$.

אם y הוא צומת בתת-העץ <u>הימני</u> של x – מתקיים ש-$\boldsymbol{y > x}$.

דוגמא –




מעברים בעץ:

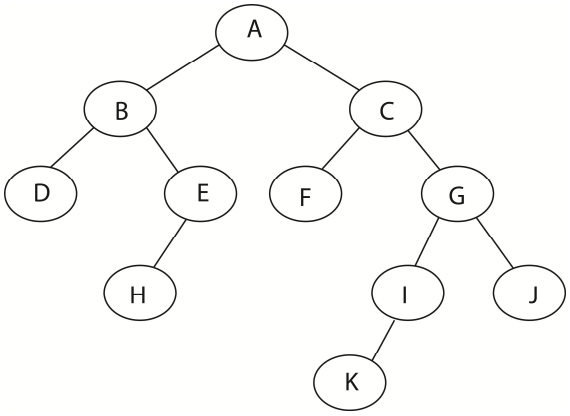
Preorder: - בקר בשורש, סייר בתת-העץ השמאלי, סייר בתת-העץ הימני.

Postorder: -סייר בתת-העץ השמאלי, סייר בתת-העץ הימני, בקר בשורש.

Inorder: - סייר בתת-העץ השמאלי, בקר בשורש, סייר בתת-העץ הימני.

<u>תרגיל:</u>

מספרו את הצמתים לפי הסדר שלהם בעץ החיפוש:



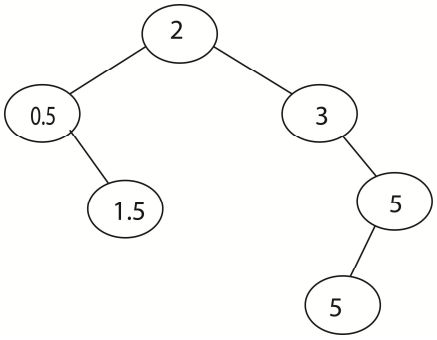

פיתרון:

מעבר inorder על העץ: D, B, H, E, A, F, C, K, I G, J

<u>פעולות בעץ חיפוש:</u>

1) חיפוש: בצורה רקורסיבית – אם הערך קטן ממה שמחפשים, לך שמלאה. אחרת – ימינה.
2) הכנסה: בצורה איטרטיבית – סורקים עד למטה (עד שמגיעים ל-null) ומוסיפים.
3) המחיקה: 3 מצבים – אין לצומת בנים – מוחקים, -לצומת יש בן יחיד – יוצרים קשת בין הבן לאב ומוחקים. – יש לצומת 2 בנים – מוחקים את הערך הקודם לו (הכי ימני בתת העץ השמאלי) ומחליפים ערכים ביניהם.

דוגמא – (בעץ הראשון):

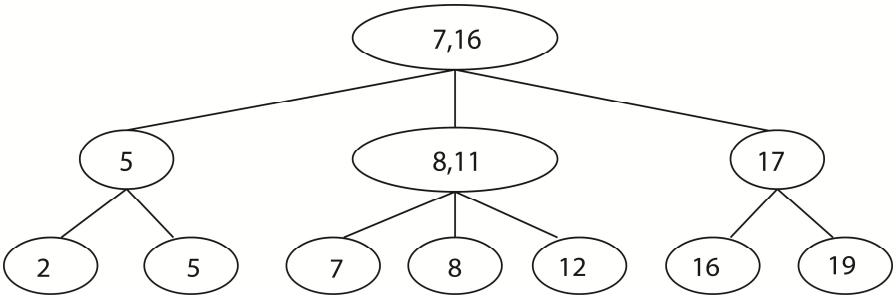



- הוסף 0.5, 1.5
- מחק 8
- מחק 7
- מחק 1

← סיבוכיות הפעולות – O(logn) (כאשר העץ מאוזן). אם לא שומרים על איזון – נקבל רשימה לינארית והחיפוש ייקח O(n).

← פיתרון – עץ מאוזן גובה – רוצים שההפרש בין עומק 2 תתי עצים יהיה לכל היותר 1.

**עצי 2-3**




- כל העלים באותה רמה

- כל הערכים בעלים
- בצמתים הפנימיים יש אינדקסים
- לכל צומת פנימי יש 2 או 3 בנים, ומספר תואם של אינדקסים.
- מספר העלים מקיים $2^h \leq L \leq 3^h$ (h- גובה העץ)
- גובה העץ: $\log_3 L \leq h \leq \log_2 L \leftarrow h = \Theta(logL)$.

בעץ 2-3 מתקיימים התנאים הבאים:

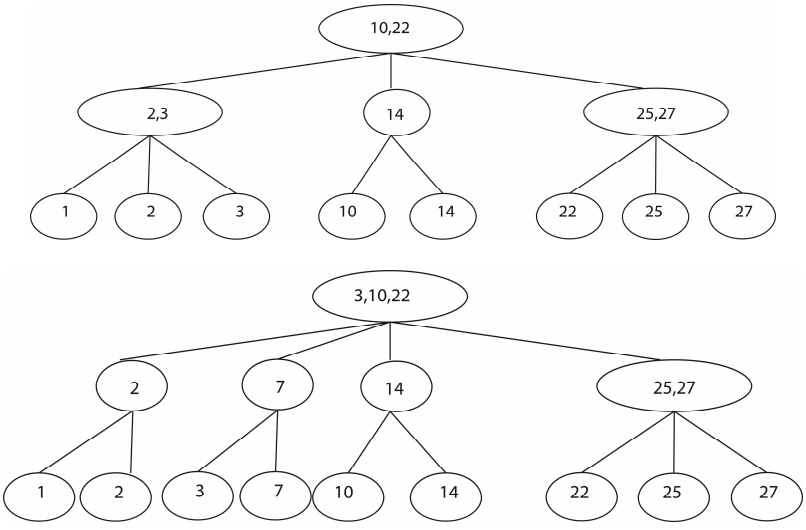
- בצומת עם 2 בנים: יש אינדקס יחיד **שגדול ממש** מהערך המקסימלי בתת העץ השמאלי **וקטן-שווה** מהערך המינימלי בתת-העץ הימני.
- בצומת עם 3 בנים: האינדקס הראשון **גדול ממש** מהערכים בתת-העץ השמאלי, **קטן-שווה** מהערכים בתת-העץ האמצעי, ואינדקס שני **גדול ממש** מהערכים בתת-העץ האמצעי **וקטן-שווה** מהערכים בתת-העץ הימני.

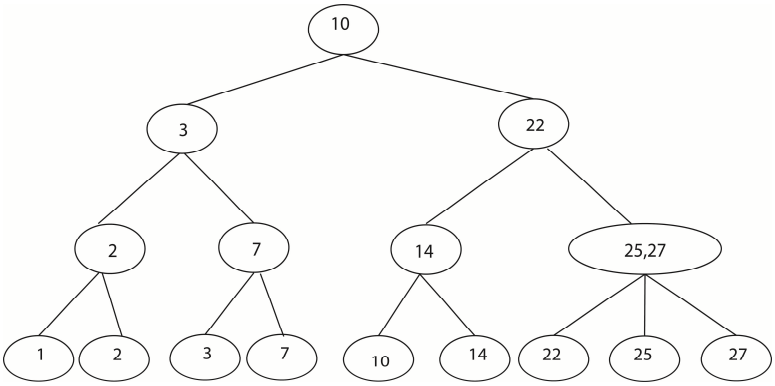
<u>פעולות בעץ 2-3:</u>

1) חיפוש - לפי הכללים הנ"ל
2) הוספה: חפש את הערך. הוסף אותו לצומת הפנימי האחרון אליו הגעת והוסף אינדקס
   2.1. אם יש 4 בנים – פצל והעלה אינדקס אמצעי
   2.2. חזור ל2.1 עבור האב
3) מחיקה: חפש את הערך. מחק את הערך ואת האינדקס משמאלו (או הכי שמאלי).
   3.1. אם נשארו 2 בנים – סיים.
   3.2. אחרת – אם לאב יש אח עם 3 בנים – השאל בן וסיים.
   3.3. אחרת – אחד את האב עם אח שלו ועדכן אינדקסים.
   3.4. חזור ל3.1 עבור האב.

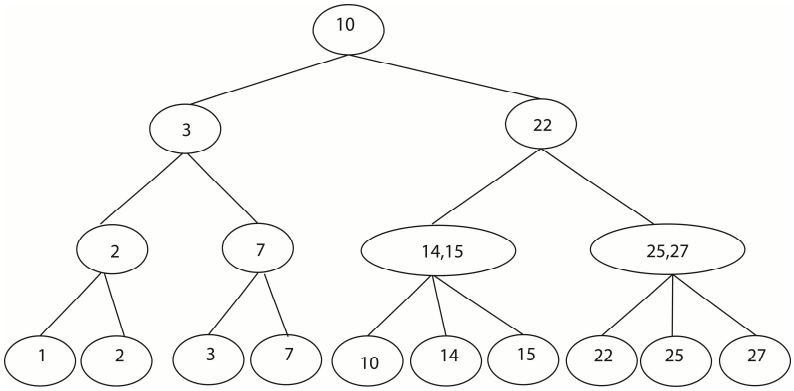
דוגמא:

הוסף 7:

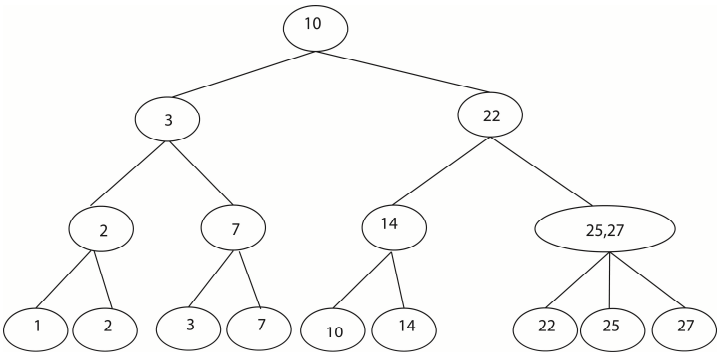




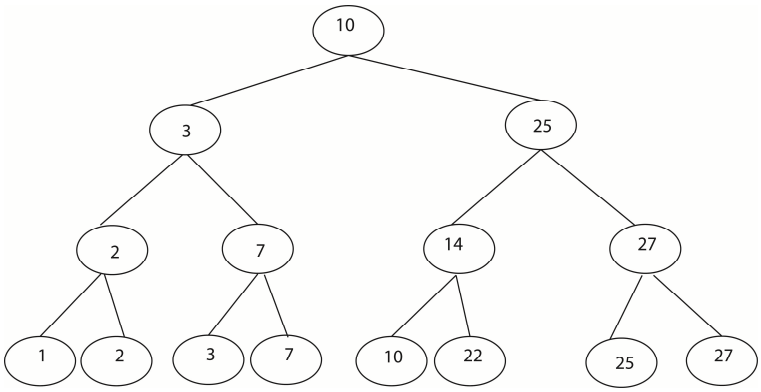

הוסף 15:



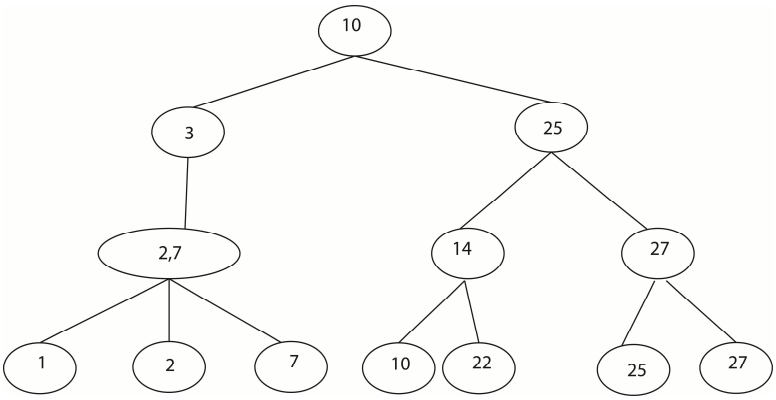

מחק 15:

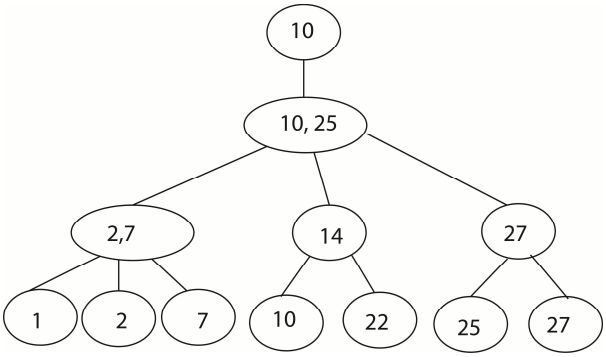


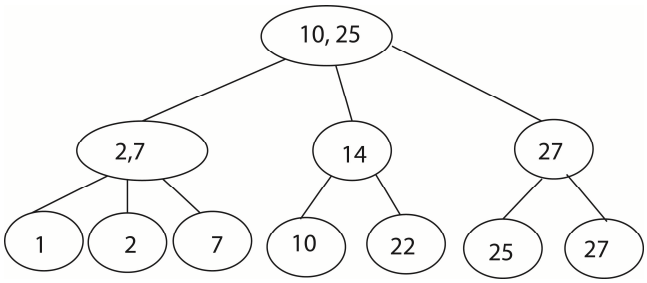

מחק 14:



מחק 3:


10
3
25
2,7
14
27
1
2
7
10
22
25
27


10
10, 25
2,7
14
27
1
2
7
10
22
25
27


10, 25
2,7
14
27
1
2
7
10
22
25
27