מבוא למדעי המחשב – כתה יא-1

תאריך: 22.1.2026

מבוא לעצים <T>BinNode - דף הסבר + תירגול #1

חלק א': מבוא לעצים

חלק זה הוא **תקציר (ולא תחליף) להסברים בכיתה ובספר הלימוד**.

1) מה זה עץ? זאת היא דרך לארגן את הנתונים. קצת מזכיר רשימה – יש לנו חוליה מקשרת קצת כמו ברשימה – אבל מייד נחדד את ההבדלים. החוליה המקשר מכילה משתנה מטיפוס T שהוא הנתון עצמו.
2) כלומר, בדומה לרשימה שהערכים בה הם מספרים שלמים – יכול להיות לנו עץ שהערכים בו הם מספרים שלמים. או, כל טיפוס אחר (למשל: תווים, או Students).
3) אז מה ההבדל? ברשימה מקושרת לכל צומת <Node<T הייתה הפניה לצומת הבא – NEXT. במקרה של עץ, לצומת שנקרא BinNode יש שתי הפניות – שאנחנו קוראים להם בן-שמאלי ובן-ימני.

הגדרות קצת פורמליות – (כאמור הסברים בכיתה)

- עץ יכול להיות עץ ריק (כלומר הפניה ל BinNode שהיא NULL. (מקבילה: רשימה ריקה).
- כשההפניה איננה NULL (בדומה לרשימה הצומת שאליו מפנים נקרא שורש העץ Root.
- לשורש יש שני בנים – תת-עץ שמאלי ותת-עץ ימני (כל אחד מהם הוא עץ).
- שימו לב שזאת הגדרה רקורסיבית. מדוע? כי תת-העץ השמאלי או הימני יכולים להיות בעצמם עצים ריקים (ז"א שההפניה היא NULL).

אנחנו נממש עץ באמצעות המחלקה <BinNode<T - מחלקה גנרית. בדף הבא מופיע הממשק שלה. בכל השאלות נשתמש בממשק הזה. (כמו שעשינו ברשימות או בתורים). זהו הממשק המחייב לבגרות.

הממשק של המחלקה BinNode<T>

הממשק הבא הוא המחייב לבגרות. (זה לקוח מתוך האתר של הילן קדמן).

הממשק - דגשים

- **בנאים**: יש לנו שני בנאים שונים.
(אחד מקבל רק DATA ואז הבן הימני+שמאלי הם NULL, והשני מקבל גם DATA וגם בן שמאלי + ימני).
- **פעולות Get**: לקבלת המידע, הבן השמאלי, והבן הימני
- **שאילתות בוליאניות** – האם יש בן ימני או שמאלי
- **פעולות Set:** שינוי ערכים
- **ToString** – בעיקר למטרות דיבאג.

תיאור הפעולה	חתימת הפעולה	סיבוכיות
בנאי:		
פעולה הבונה חוליה בינארית שערכה x ושתי ההפניות null	BinNode (T x)	O(1)
פעולה הבונה חוליה בינארית שערכה x, ושתי ההפניות שלה הן left ו- right בהתאמה (ערכם של הפרמטרים left ו- right יכול להיות null)	BinNode (BinNode<T> left, T x, BinNode<T> right)	O(1)
שאילתות:		
אם T מחלקה עוטפת לטיפוס בסיסי (Integer, Double, Character) יוחזר ערך החוליה, ואם הפניה לעצם, תוחזר הפנייה לעצם זה	T GetValue ()	O(1)
פעולה המחזירה הפנייה לחוליה אליה מפנה left.	BinNode <T> GetLeft ()	O(1)
פעולה המחזירה הפנייה לחוליה אליה מפנה right.	BinNode <T> GetRight ()	O(1)
פעולה המחזירה אמת אם יש חוליה משמאל, ושקר אחרת	**bool** HasLeft ()	O(1)
פעולה המחזירה אמת אם יש חוליה מימין, ושקר אחרת	**bool** HasRight ()	O(1)
פעולה המחזירה את מצב החוליה כמחרוזת. (*) סיבוכיות: אם T עצם מטיפוס פשוט ← O(1) ואם T מייצג אוסף באורך כלשהו ← O(|T|)	**string** ToString ()	O(|T|) (*)
פקודות:		
הפעולה משנה (מעדכנת) את ערך החוליה ל- x	**void** SetValue (T x)	O(1)
הפעולה משנה את ערכה של ההפניה left ל- left הפרמטר left יכול להיות גם null	**void** SetLeft (BinNode <T> left)	O(1)
הפעולה משנה את ערכה של ההפניה right ל- right הפרמטר right יכול להיות גם null	**void** SetRight (BinNode <T> right)	O(1)

חלק ב': תרגילי לכיתה, למעבדה, ולביצוע בבית

בחלק זה מופיעים תרגילים מהקל אל הכבד.

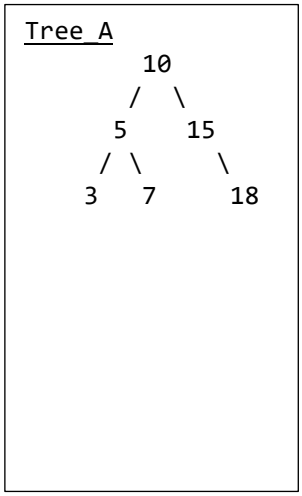
אלו תרגילים ראשונים בנושא. חשוב מאוד לבצע אותם בעצמכם לפי ההמלצות למטה. זה יפתח את החשיבה הדרושה לקראת תרגילים קשים יותר שניפגוש בהמשך (עד לרמה הנדרשת בבגרות).

המלצות כרגיל:

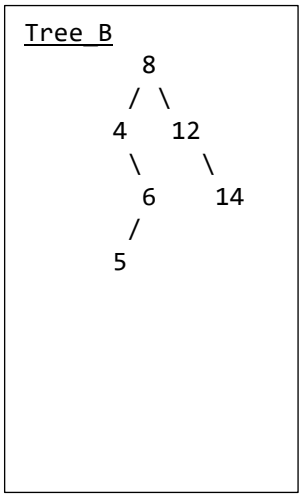
א. בצעו את התרגיל על דף נייר, תוך שימוש בציור.
ב. כתבו במחשב באמצעות תכנה כמו notepad++. התייעצו עם AI לגבי השאלה והפתרון שלכם. השתמשו בפרומפט שבאתר בכדי לכוון אותו שאתם תלמידים, ושיעזור לכם בתהליך הלמידה ולא יפתור את התרגיל במקומכם.
ג. לבסוף העתיקו את הקוד לפרוייקט שלכם ב Visual Studio, עכשיו אתם יכולים להריץ ולראות שהתרגילים השונים עובדים היטב.

דוגמאות לעצים שנשתמש בהם בכיתה ובתירגולים

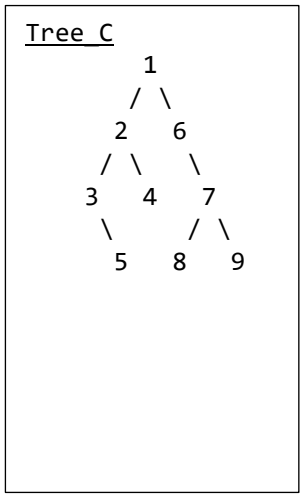
Tree_A



Tree_B



Tree_C



הקדמה:

כתוב את הפונקציות הבאות:

- כתוב פונקציה בוליאנית **`bool IsLeaf<T>(BinNode<T>)`** שמחזירה TRUE אם הצומת הוא עלה (כלומר, צומת שאין לו בנים).
- כתוב פונקציה בוליאנית (**`HasSingleChild<T>(BinNode<T>)`** bool שמחזירה TRUE אם הצומת מכיל בדיוק בן אחד, אחרת היא מחזירה FALSE.

מקבץ #1 – פונקציות מעבר על עץ

תרגיל #1 כתוב פונקציה שמחזירה את סכום הערכים בעץ.

תרגיל #2 כתוב פונקציה שמחזירה את מספר הצמתים בעץ.

תרגיל #3 כתוב פונקציה שמחזירה את מספר העלים בעץ.

תרגיל #4 כתוב פונקציה שמחזירה את העומק (גובה) של העץ: **`int Depth<T>(BinNode<T>)`**
הנחיה: עץ ריק הוא בעומק 0. עץ שהוא רק צומת בודד – העומק שלו הוא 1.

מקבץ #2 – פונקציות מעבר תוך כדי בדיקת תנאים

תרגיל #5 כתוב פונקציה שמחזירה את מספר הצמתים בעץ שיש להם בן אחד.

תרגיל #6 כתוב פונקציה שמחזירה האם עץ הוא מלא – כלומר לכל צומת פנימית (צומת שאיננה עלה) יש שני בנים.

תרגיל #7 כתוב פונקציה שמקבלת הפניה לעץ + ערך X, ומחזירה TRUE אם קיים בעץ צומת שערכו X אחרת היא מחזירה FALSE.

תרגיל #8 כתוב פונקציה שמקבלת הפניה לעץ + ערך X, ומחזירה כמה פעמים הערך X מופיע בעץ.

מקבץ #3 – פונקציות הדפסה

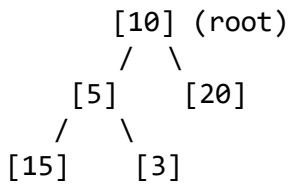
תרגיל #9 כתוב פונקציה שמדפיסה את העץ בסדר שנקרא LDR – כלומר, קודם כל היא מדפיסה את התת-עץ השמאלי (L == LEFT) ואח"כ את השורש עצמו, ולאחר מכן את התת עץ הימני (R).

תרגיל #10 כתוב פונקציה שמדפיסה את העץ בסדר שנקרא DLR – כלומר, קודם כל היא מדפיסה שורש העץ, לאחר מכן את תת-העץ השמאלי (L == LEFT) ואז את תת-העץ הימני (R).

תרגיל #11 כתוב פונקציה שמדפיסה את העץ בסדר שנקרא LRD – כלומר, קודם כל היא מדפיסה את תת=העץ השמאלי, לאחר מכן, את תת-העץ הימני, ולבסוף את שורש העץ.

שאלת אתגר (בינתיים)

כתוב פונקציה שמחזירה האם קיים צומת בעץ ששווה לסכום האבות שלו. לדוגמא בעץ הבא הצומת [15] מקיים את התכונה:



שאלת חשיבה – לא לביצוע (בינתיים)

כתוב פונקציה שבודקת אם בעץ יש שני צמתים בעלי אותו ערך, ומחזירה TRUE אם כן או FALSE אם זה לא מתקיים. אין להשתמש במבנה נתונים נוסף פרט לעץ.

מישהו הציע את הפתרון הבא:
נעבור על הצמתים בעץ ולכל צומת נבדוק אם הערך מופיע באחד מתתי העצים שלו.

א. האם הפתרון הזה נכון?
ב. אם כן – מדוע.
ג. אם לא – נסו לחשוב על דוגמה נגדית.
ד. האם אתם יכולים לתכנת את ההצעה הזאת (כנראה שכן)
ה. האם אתם יכולים לחשוב על פתרון אחר?