מבוא למדעי המחשב – כתה יא-1

תאריך: 21.2.2026

עצים בינארים דף עבודה #3

מטרות:

- תרגול "עבודה עם מסלולים (Path)" ופעפוע מידע לאורך הדרך
- תרגול "מידע חוזר למעלה" (כמו סכומים/גובה/תכונות) אבל עם תנאים מתוחכמים יותר

הערה: בהמשך ייתכן שאוסיף (בלי נדר) תשתית של קבצי C# עבור Visual Studio שיאפשרו לכם להריץ את התרגילים.

אבל: בבגרות אין לכך את זה, ולכן חשוב קודם כל שתפתרו בעצמכם במחברת, בכתב יד, ותריצו "בראש" (בעזרת ציור) כל דוגמא. כך תפתחו את המיומנות הדורשה לבגרות.

פונקציות לחימום

ראינו בדפים קודמים פונקציות בדיקה על צומת בדידה כמו IsLeaf(), HasSingleChile.

מומלץ לכתוב פונקציות כאלו לפי הצורך (גם במתכונת וגם בבגרות). כפי שראיתם – זה מפשט מאוד את הקוד של הפונקציה העיקרית שאתם צריכים לכתוב.

להלן כמה דוגמאות לתירגול נוסף. לא לשכוח – חייבים להשתמש בפעולות שהמחלקה BinNode<T> מספקת. מכיוון שאלו פונקציות כלליות, תכתבו אותן בצורה הגנרית (הכללית).

א. כתוב פונקציה IsLeftChild(parent, child) שמחזירה true אם child הוא הבן השמאלי הישיר של parent. אני מזכיר שחתימה של פונקציה כזאת תהיה:

```
public static bool IsLeftChild<T>(BinNode<T> parent, BinNode<T> child)
```

ב. כתוב פעולה GetOnlyChild(node). אם לצומת יש בדיוק בן אחד – החזר את הבן הזה; אחרת החזר null.

ג. כתוב פעולה SameValueAsParent(node). (תזכורת: במקרה גנרי אנחנו משתמשים בפעולה Equals(...)

ד.

מסלולים ופעפוע ערכים

פעפוע ערכים בעץ הוא מעבר שבו מידע מצטבר (כשיורדים) או כשחוזרים, לאורך המסלול.

שאלו את עצמכם בכל קריאה:

- מה אני יודע כרגע?
- מה אני מעביר לבנים?
- מה אני מחזיר לאבא?

מסלולים ופעפוע מלמעלה למטה

תרגיל #20 כתוב פעולה CountGoodNodes(root). נגדיר שצומת נקרא צומת "טוב" אם הערך שלו גדול או שווה לכל הערכים שעל המסלול מהשורש אליו.

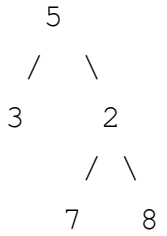
תרגיל #21 כתבו פעולה ExistsRootToLeafSum(root, target) שבודקת האם קיימת דרך מהשורש לעלה כך שסכום הערכים לאורך הדרך שווה ל- target?

תרגיל #22 כתבו פעולה CountPathsWithExactlyKNodes(root, k) שמחזירה כמה מסלולים מהשורש לעלה מכילים בדיוק k צמתים?

תרגיל #23 כתבו פעולה PrintAllRootToLeafPaths(root) שמדפיסה את כל המסלולים מהשורש לעלים, כל מסלול בשורה נפרדת. לדוגמא: 10 -> 3 -> 7

תרגיל #24 כתבו פעולה ExistsNodeEqualToPrefixSum(root). הפעולה בודקת האם קיים צומת שעבורו: value(node) == sum(values from root to parent(node))

לדוגמא: עבור העץ הבא, הצומת 7 מקיים את התנאי כי ערכו הוא 5 + 2 שזה סכום הצמתים בדרך אליו.



מידע שחוזר מלמטה למעלה

תרגיל #25 כתוב פעולה MaxRootToLeafSum(root) שמחזירה את סכום הערכים המקסימלי במסלול מהשורש לעלה.

תרגיל #26 נגדיר שעץ הוא עץ-גבהים-מאוזן אם לכל אם לכל צומת מתקיים: abs(height(left) - height(right)) <= 1. כלומר, בכל צומת בעץ ההפרש בגבהים בין תת העץ השמאלי לתת העץ הימני אינו עולה על 1. כתוב פעולה IsHeightBalanced(root) שמחזירה true/false אם העץ מקיים את התנאי שהוא עץ-גבהים-מאוזן.

תרגיל #27 כתוב פעולה FirstUnbalancedNode(root) שבודקת אם העץ הוא עץ גבהים מאוזן (בדומה לשאלה הקודמת) ומחזירה הפניה לצומת הראשון שבו התנאי נשבר במעבר preorder (זהו מעבר שבו עוברים קודם על תת העץ השמאלי, ואחר כך על תת העץ הימני, ורק אז מטפלים בשורש.) אם הכל מאוזן – החזר null.

<u>תרגיל מסכם לדף זה (</u><u>רמה מתקרבת או דומה לבגרות</u>**<u>)</u>**

תרגיל #28 נתון עץ בינארי של מספרים שלמים `BinNode<int> root`.

ידוע כי:

- בכל **צומת פנימי** הערך הוא מספר בין `0` ל- 9.
- ערך העלים הוא מספר שלם כלשהו (יכול להיות גדול מ-9).

נאמר שמסלול מהשורש לעלה יוצר מספר עשרוני כך שהספרות לאורך המסלול (בערכי הצמתים הפנימיים) יוצרות את המספר לפי סדר הירידה בעץ. לדוגמה:

אם לאורך המסלול מופיעות הספרות 3 → 1 → 4 אז המספר שנוצר הוא 314.

כתוב פעולה `ExistsMatchingLeaf`, שמקבלת מצביע לשורש עץ בינארי. הפעולה מחזירה `true` אם קיים עלה בעץ כך שהמספר שנוצר לאורך המסלול מהשורש אליו שווה לערך העלה.
אחרת תחזיר `false`.