תאריך: 8.1.2026

מבוא למדעי המחשב – כתה יא-1

חשיבה אלגוריתמית, ומעבר ממערך לרשימה מקושרת גנרית

האם רשימה היא פאלינדרום

מבוא: החשיבה האלגוריתמית כשעוסקים ברשימות

כשאנו ניגשים לפתור בעיה מורכבת (לא רק ברשימות, לא רק במחשבים) – אנחנו שואלים את עצמנו: האם יש בעיה דומה (כלומר מאפיינים דומים) שאני יודעת להתמודד איתה יותר בקלות.

ועכשיו, נראה איך אפשר להשתמש בזה בבעיות של רשימות מקושרות.

כשאנו ניגשים לפתור בעיה מורכבת ברשימה מקושרת, כדאי קודם כל לחשוב איך היינו פותרים אותה במערך. במקרה שלנו, כיצד נבדוק אם המערך הוא פאלינדרום?

במערך, הבדיקה פשוטה: אנו מציבים שני "סמנים" (אינדקסים) – אחד בהתחלה (`lo = 0`) ואחד בסוף (`hi = arr.Length - 1`) ומתקדמים פנימה כל עוד הערכים שווים.

<u>האלגוריתם – אם זה היה מערך</u>

כלומר אם נתאר את זה באלגוריתם (או פסאודו-קוד) הפתרון יראה בערך כך:

א. `lo ← 0`
ב. `hi ← arr.Length – 1`
ג. בצע את הלולאה הבאה כל עוד `lo < hi`
  a. אם `arr[hi] ≠ arr[lo]` החזר `false` (זה כבר לא פאלינדרום).
  b. `++lo`
  c. `--hi`
ד. החזר `true`

<u>הבנת הבעייתיות מכך שזה לא מערך</u>

עכשיו נשאל את עצמנו: אז למה אי אפשר להשתמש באותו קוד או אלגוריתם בדיוק עבור רשימה מקושרת?

1) **אין גישה ישירה לפי אינדקס**: ברשימה אי אפשר לכתוב `chain[i]`. כלומר אין לנו משהו ש- "מייצר" בקלות את האיבר ה- `i` ברשימה. (**צריך פונקציה** שתחזיר את האיבר ה- `i` ברשימה)
2) **אורך לא ידוע**: לרשימה אין מאפיין `Length`. בכדי לדעת כמה איברים יש, חייבים לסרוק את כולה. או במילים אחרות "מישהו" שיגיד לנו מהו אורכה. (**צריך פונקציה** שתחזיר לנו את אורך הרשימה)

<u>האלגוריתם – עבור רשימה</u>

האלגוריתם הבא פותר את הדרישה:

א. אורך הרשימה ← len
ב. 0 ← lo
ג. hi ← len – 1
ד. בצע את הלולאה הבאה כל עוד lo < hi
  a. האיבר במקום lo ברשימה ← node1
  b. האיבר במקום hi ברשימה ← node2
  c. אם הערך של החוליות node1, node2 שונה אז החזר false .
  d. ++lo
  e. --hi
ה. החזר true

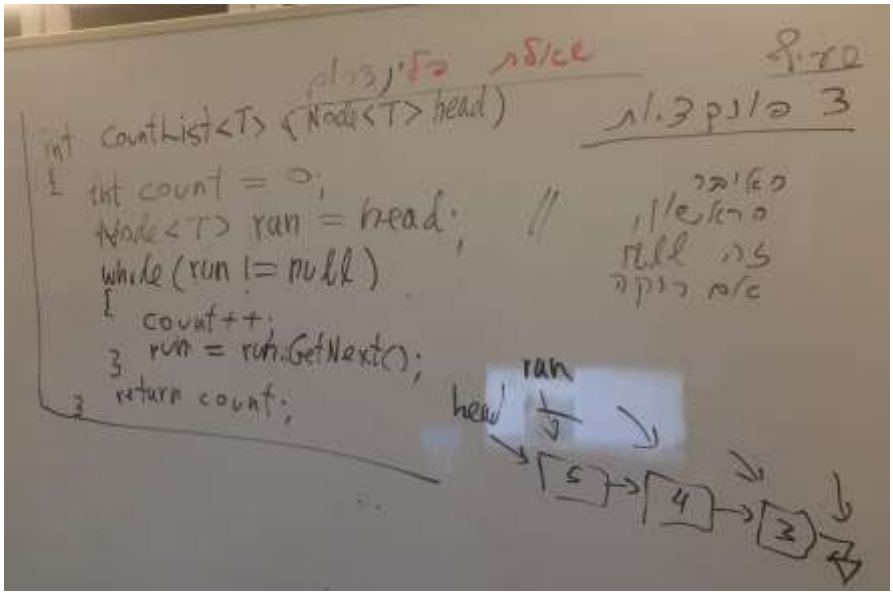
<u>סעיף ב' – כתיבת הקוד</u>

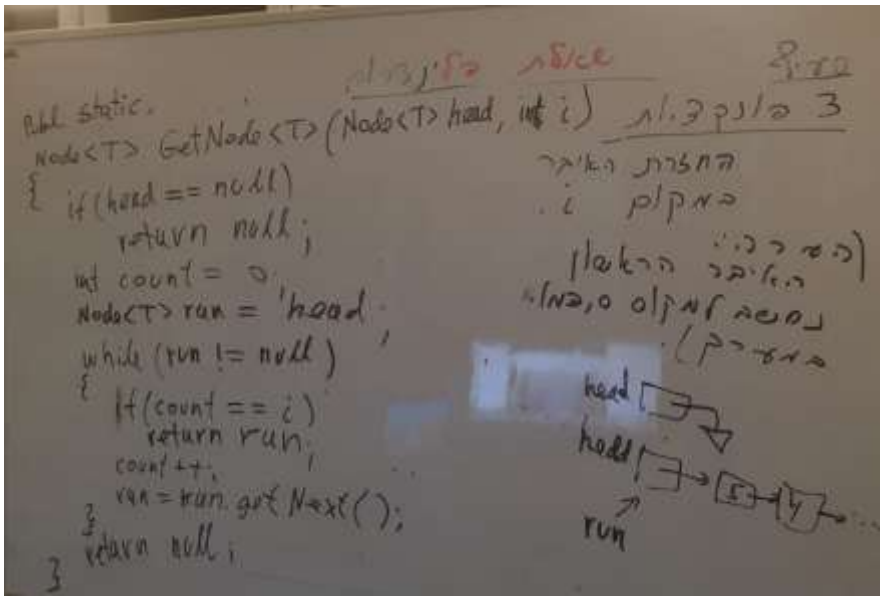
לפני שאתם קוראים את הקוד (וגם אחרי אם כבר קראתם) – התאמנו בלכתוב את הקוד הזה בכוחות עצמכם !

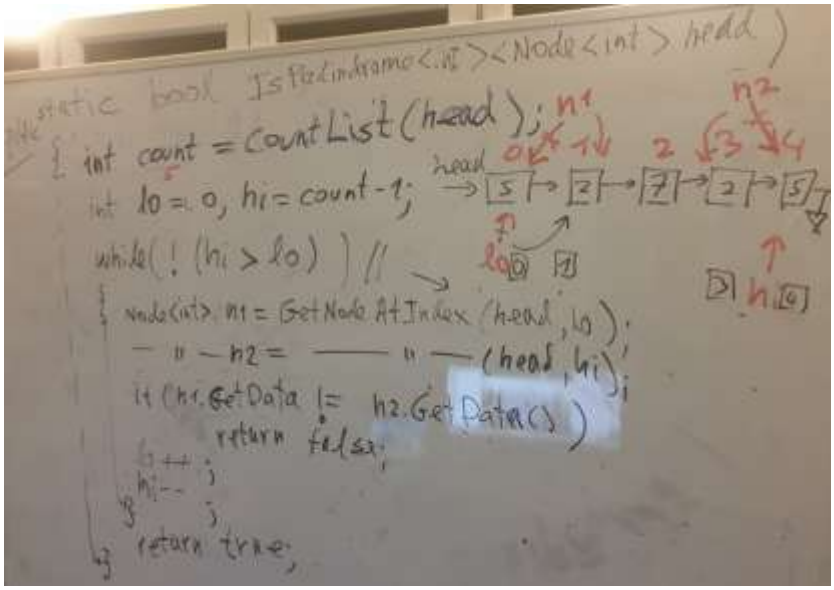
דגשים:

1. שימו לב שפונקציה גנרית כתובה עם <T> בסוף השם שלה.
2. שימו לב לשימוש ב T או ב Node<T> בתוך הפונקציה או בערך המוחזר שלה.

אני מצרף צילומי הלוח הלבן מהשיעור שהתקיים בזום. יש שם הדגמה של העבודה של הפונקציה, ואיך עוברים על הרשימה. אני מקווה שזה ברור מספיק.







```
public static Node<T> GetNode<T>(Node<T> head, int i)
{
    // בדיקה אם הרשימה ריקה
    if (head == null)
        return null;

    int count = 0;
    Node<T> run = head;

    while (run != null)
    {
        // אם הגענו למספר הצעדים המבוקש, נחזיר את החוליה
        if (count == i)
            return run;

        count++;
        run = run.GetNext();
    }

    // אם הלולאה הסתיימה ולא מצאנו (אינדקס מחוץ לטווח), נחזיר null
    return null;
}

//================================================
public static int CountList<T>(Node<T> head)
{
    int count = 0;
    Node<T> run = head; // יצירת מצביע זמני לסריקה

    while (run != null) // רצים כל עוד לא הגענו לסוף השרשרת
    {
        count++;
        run = run.GetNext(); // קידום המצביע לחוליה הבאה
    }

    return count;
}
```

```
//================================================
public static bool IsPalindrome<int>(Node<int> head)
{
    int count = CountList(head); // כמה איברים יש בשלב 1?
    int lo = 0;
    int hi = count - 1;

    // רצים מהקצוות כל עוד lo קטן מ-hi
    while (hi > lo)
    {
        // שליפת שתי החוליות שרוצים להשוות
        Node<int> n1 = GetNodeAtIndex(head, lo);
        Node<int> n2 = GetNodeAtIndex(head, hi);

        //
        //מכאן השתמשנו במספר שלם ולכן מותר להשוואת הערכים != !
        if (n1.GetValue() != n2.GetValue())
            return false;

        lo++; // התקדמות פנימה מההתחלה
        hi--; // התקדמות פנימה מהסוף
    }
    return true;
}
```