

מבני נתונים תשפ"ב

תרגיל 3 – רשימות הקשורות

1. נתונה רשימה מקווערת דו-כיוונית A.

א. (20 נק') כתבו אלגוריתם היוצר רשימה מקווערת חדשה B אשר מכילה אך ורק איברים מ-A, אשר שווים להפער האיבר העוקב והאיבר הקודם להם ב-A. (שים לב, אם ההפרש לא מוגדר כי אין לאיבר קודם או עוקב, האיבר לא נכלול ברשימה B).

ב. (10 נק') מהו סדר הגודל של מספר הפעולות המתבצעות? נמקו.

ג. (15 נק') כיצד הייתה משתנה התשובה לסעיפים א-ב במידה והרשימות הקלט והפלט היו רשימות חד-כיווניות?

ד. (15 נק') כיצד הייתה המשתנה התשובה לסעיף א במידה ולא היינו רוצחים ליצור רשימה חדשה, אלא לשנות את רשימת הקלט?

פתרונות

- אם כן, ניצור איבר חדש שיכיל את ה- $data$ של $curr$, ובכל מקרה נקדם את המצביע בצעד אחד. האלגוריתם יסיים את ריצתו כאשר $.next=NULL$.

```

Node_type* subList(Node_type* a)
{
    Node_type *curr = a;
    Node_type *headB = NULL, *currB = NULL, *newB;

    // בדיקה אם יש לפחות 3 איברים בראשימה, וקידום המצביע לאיבר השני //
    if (curr != NULL)
    {
        curr = curr ->next;
        if (curr == NULL || curr->next == NULL)
            return headB;
    }
    else
        return headB;

    // מעבר על הרשימה //
    while (curr->next != NULL)
    {
        if (curr->data == (curr->prev->data) - (curr->next->data)) // התנאי המבוקש מתקיים //
        {
            newB = MakeNode(curr->data);

```

```

        InsertBetween(newB, &headB, &currB, NULL);
    }
    curr = curr->next;
}

return headB;
}

```

הוכחת נכונות:

מאחר והרשימה זו כיוונית, ואנו מתחילה מהאיבר השני, המשתנים $curr \rightarrow next \rightarrow data$, $curr \rightarrow prev \rightarrow data$ ו- $curr \rightarrow next \rightarrow data$ מוגדרים לכל אורך הרשימה. האלגוריתם ייצור איבר חדש ויכניסו ל- B רק כאשר:

$$curr \rightarrow data == prev \rightarrow data - seq \rightarrow data$$

ולכן האלגוריתם עובד כראוי.

- ב. סדר גודל הפעולות המתבצעות הוא כסדר מספר האיברים ב- A. האלגוריתם עובד על כל האיברים ב- A ועל כל איבר מבצע מספר קבוע של פעולות (העלות של MakeNode ו- InsertBetween) והוא $\Theta(1)$ ולכון אם מספר האיברים ב- A הוא n אז סדר גודל הפעולות הוא גם כן $\Theta(n)$.

- ג. במידה והרשימה הייתה חד-כיוונית היוינו צריכים ללחזק מצביע נוסף לאיבר הקודם, $prev$, לאותל אותו לאיבר הראשון ברשימה ולערוך את השינויים הבאים:

- לבצע את הבדיקה הבאה :

$$curr \rightarrow data == (prev \rightarrow data) - (curr \rightarrow next \rightarrow data)$$

- בכל קידום של curr בסוף לולאת ה- while לקדם גם את המצביע הנוסף :

$$prev = prev \rightarrow next$$

זמן הריצה לא היה משתנה.

- ד. במידה והיינו רוצים לשנות את הרשימה הנוכחי כך שיישארו רק האיברים השווים להפרש האיבר הקודם והאיבר העוקב, היינו צריכים לבצע באלגוריתם את השינויים הבאים :

a. נctrck שני משתנים חדשים :

i. prevVal – במקומם לגשת לאיבר הקודם נשמר רק את הערך שלו, ובהתאם התנאי ישנה

באופן הבא : curr->next->data = prevVal - curr->data

ii. currA – מצביע לאיבר הנוכחי ברשימה המעודכנת של A בה יש רק איברים שמיימים את התנאי.

b. תחילת היינו שומרים את ערכו של האיבר הראשון ברשימה בתוך prevVal ומוחקים את האיבר מהרשימה. אחר כך היינו מחפשים את האיבר הראשון שמיימים את התנאי. כל עוד האיבר currA לא מקיים את התנאי מוחקים אותו מראש הרשימה. כאשר מוצאים את האיבר הראשון שמיימים את התנאי מגדירים ש- currA מצביע עליו.

c. מעבר על שאר הרשימה יתבצע באופן הבא :

i. במידה והתנאי מתקיים מגדירים ש- currA מצביע עליו.

ii. אחרת, יש למחוק את האיבר currA מהרשימה. נעשה זאת ע"י שימוש בפונקציה Delete(currA,curr) (מוחקת את הפעמטר השני מהרשימה, והפעמטר הראשון נמצא ברשימה לפני הפעמטר השני)

בכל מקרה מגדמים את הערכים prevVal ואחר לכךcurrA.

p. כאשר נגיע לסוף הרשימה נגידר ש- `currA->next=NULL` כדי שלא יצביע לאיבר האחרון ברשימה שבוודאות לא מקיים את התנאי כי אין לו איבר עוקב. זמן הרצאה לא היה משתנה, כי עדין עוברים על כל האיברים ברשימה.

- א. נתונה רשימה מקוושתת **ח' פיוונית A**.
- א. (20 נק') הצעו אלגוריתם בסיבוכיות זמן $O(n)$ ובצל דרישת זיכרון קבועה (כלומר, שאינה תלולה ב-*a*), הופכת את סדר הרשימה (האיבר הראשון יהפוך להיות האיבר האחרון, האיבר השני יהפוך להיות האיבר לפני האיבר הראשון, וכו').
- הערה** – בתרגיל כזה עליכם להראות שדרישות הסיבוכיות מתיקיימות.

פתרון:

השיטה היא להפוך את כיוון החצים כך ש**next** יהיה **prev**, כלומר – יצביע לאיבר ה"קדם" ולא לאיבר הבא, וכך בעצם להפוך את הרשימה. כל רשימה מסוימת ב-`NULL`, אז נdag שהאיבר הראשון יצביע בעצמו ל-`NULL` ו-`head->next` (הצביעו לראש הרשימה) יצביע לאיבר האחרון.

נזכיר שלושה מצביעי עזר : `prev, curr, next` :
כל עוד לא נגיע לסוף הרשימה, המשיך בהשחתת המצביעים
.head. נחזיר את `prev` כ-
המשמעותו עצמו ייראה כך :

```
Node_type* ReverseList2(Node_type* list)
{
    Node_type* prev,*curr,*next;
    prev=NULL,next=NULL;
    curr=list;
    while(curr!=NULL)
    {
        next = curr->next; // מצביע להמשך הרשימה, שלא נאבד אותה.
        curr->next = prev; // היפוך כיוון הצבעה בתוך הרשימה.
        prev = curr; // קידום המצביע לרשימה הפוכה שיצרנו אל האיבר שהוספנו כת ראש הרשימה.
        curr = next; // הכנה לאייטרציה הבאה – קידום המצביע אל האיבר הבא בראשימה המקורית.
    }
    return prev;
}
```

סיבוכיות :

מבחינת הזמן : עוברים על הרשימה בדיקת פעם אחת, ולכן אלא אלגוריתם בסיבוכיות ליניארית, כנדרש.
 מבחינת הזיכרון : אין יצירה של מבנה נתונים נוסף אלא שימוש בשלושה מצביעים בלבד = דרישת זיכרון קבועה.

ב. (20 נק') בהנחה שהנתונים בשודות המידע בראשימה הם מספרים שלמים ובהינתן מספר שלם Z כקלט נוסף, יש לכתוב אלגוריתם שמחזיר TRUE אם קיימים בראשימה זוג איברים שסכוםם, Z – FALSE אחרת. יש להראות נכונות האלגוריתם ולנתח את סיבוכיות הזמן שלו.