

מבני נתונים תשפ"ב

פתרון 4 – תור ומחסנית

שאלה 1

1. א. כתבו אלגוריתם המקבל רשימה מקושרת L (חד כיוונית) ומספר זוגי n שמייצג את מספר האיברים ב-L. האלגוריתם נעזר בתור ובמחסנית, ומדפיס את איברי הרשימה בזוגות מהקצוות פנימה, כך – ראשונים יודפסו האיבר הראשון והאיבר האחרון (מופרדים ברווח ' '), אחריהם יודפסו האיבר השני והאיבר שלפני האחרון וכן הלאה. אחרי כל זוג יש להדפיס * (כאמצעי הפרדה). לדוגמא, אם הרשימה מורכבת מהאיברים 5,7,2,1,9,3,4,8, ההדפסה תראה כך:
- * 5 8 * 7 4 * 2 3 * 1 9 *
- ב. מהי סיבוכיות זמן האלגוריתם? הסבירו.

פתרון

- א. מאחר ובתור סדר היציאה של האיברים הוא FIFO (first in first out) ובמחסנית סדר היציאה הוא LIFO (last in first out), נעבור על הרשימה המקושרת ונכניס את $n/2$ האיברים הראשונים לתור, ו- $n/2$ האיברים האחרונים למחסנית. לאחר מכן נוציא איבר מהתור (מהראשון ועד האיבר ה- $n/2$), ואיבר מהמחסנית (מהאחרון ועד האיבר ה- $n/2+1$) ונדפיס אותם יחד. האלגוריתם תקין מכיוון שהוא עובר על כל האיברים ברשימה המקושרת, מסדר אותם בתור ובמחסנית לפי סדר הדפסתם, ולאחר מכן מבצע את ההדפסה כנדרש.

```
void printPairs(Node_type* L, int n)
{
    Stack_type* S;
    Queue_type* Q;
    Node_type *temp, *curr = L->head;
    int left, right;

    // adding the first n/2 elements to the queue
    for (int i = 1; i <= n/2; i++)
    {
        temp = curr->next;
        AddQueue(curr, Q);
        curr = temp;
    }

    // adding the last n/2 elements to the stack
    for (int i = 1; i <= n/2; i++)
    {
        temp = curr->next;
        push(curr, S);
        curr = temp;
    }
}
```

```

// printing
for (int i = 1; i <= n/2; i++)
{
    DeleteQueue(left, Q);
    right = pop(S);
    print(left, ' ', right, '*');
}
}

```

ב. סיבוכיות : במעבר הראשון על איברי הרשימה המקושרת מתבצעת הכנסה לתור או למחסנית, פעולות העולות $\Theta(1)$ לכל איבר שיש ברשימה, ולכן סיבוכיות הזמן של הכנסת האיברים לתור ולמחסנית היא $\Theta(n)$. לאחר מכן לולאת ההדפסה רצה $n/2$ פעמים וכל פעם מבצעת מספר קבוע של פעולות בעלות של $\Theta(1)$, שוב מדובר בעלות של $\Theta(n)$. ומכאן שסיבוכיות הזמן הכוללת של האלגוריתם היא $\Theta(n)$.

שאלה 2

2. כתבו פעולה המקבלת מחסנית שאינה ריקה, בה יכולים להיות איברים זהים. הפעולה תחזיר את המחסנית בה קיים רק איבר אחד מכל ערך.

למשל, עבור המחסנית: $\{47,6,15,3,31,3,2,15\}$ תוחזר המחסנית: $\{47,6,15,3,31,2,9\}$

פתרון

- נשתמש בשתי מחסניות עזר בשביל להשאיר במחסנית S רק איבר אחד מכל ערך. תחילה נעביר את כל האיברים ל-S2 כדי שנוכל למלא מחדש את S באיברים ייחודיים בלבד. כעת תתבצע הלולאה הבאה:
1. מוציאים איבר אחד מ-S2, שומרים את ערכו במשתנה val ומכניסים אותו ל-S.
 2. מוציאים את שאר האיברים שנותרו ב-S2, במידה וערכם שונה מערכו של val מכניסים אותם ל-S1.
 3. מעבירים את כל האיברים שנכנסו ל-S1 (כעת נותרו רק כאלה ששונים מ-val) בחזרה ל-S2 וחוזרים לבצע את שלב 1 עבור האיבר הבא.

נכונות: ניתן לראות שהאלגוריתם פועל כנדרש, כי ל-S ייכנסו רק איברים שונים זה מזה.

```
void uniqueStack(Stack_type* S)
{
    Stack_type *S1, *S2;
    int val, temp;

    // moving all nodes from S to S2
    while (NOT(empty(S)))
        push(pop(S), S2);

    // going through S2
    while (NOT(empty(S2)))
    {
        val = pop(S2);
        push(val, S); // pushing unique values only
        while (NOT(empty(S2)) // checking if other nodes in S2 are equal to val, pushing to S1 when different
        {
            temp = pop(S2);
            if (temp != val)
                push(temp, S1);
        }
        // returning all remaining nodes to S2
        while (NOT(empty(S1)))
            push(pop(S1), S2);
    }
}
```

עלות: נניח כי מספר האיברים במחסנית הנתונה הוא n . בכל שלב, אנו משווים את הערך של איבר אחד רק עם איברים שנמצאים מתחתיו במחסנית S_2 , שהרי האיברים שהיו מעליו כבר נבדקו בשלבים הקודמים ועברו ל- S (או נמחקו, במידה והיו להם כפילים) נקבל:

- **במקרה הגרוע** – כל האיברים במחסנית היו שונים זה מזה. לכן בכל שלב משווים בין val לבין כל האיברים שהיו מתחתיו. לכן, באיטרציה הראשונה יהיו $n-1$ השוואות, ובכל איטרציה נוספת מספר ההשוואות ירד ב-1. נקבל

$$\text{בסה"כ: } (n-1) + (n-2) + \dots + 1 = \sum_{i=1}^{n-1} i = O(n^2)$$

- **במקרה הטוב** – כל האיברים במחסנית היו זהים. לכן באיטרציה הראשונה נשווה את כולם לאיבר הראשון,
- ולא נעביר אף איבר למחסנית S_1 . כך לא יהיו עוד איטרציות, ולכן זמן הריצה הוא $\Theta(n)$.
- **לסיכום** – חסם תחתון $\Omega(n)$, חסם עליון $O(n^2)$.

שאלה 3

3. ברצוננו לממש שתי מחסניות בתוך מערך אחד. הגדירו את הפעולות Push, Top, Pop, IsEmpty על כל אחת מהמחסניות. (ניתן להגיש בקוד או בפסאודו-קוד). במקרה שסך כל הערכים בשתי המחסניות יחד עולה על גודל המערך, מספיק להציג התראה.

פתרון

הרעיון: בהינתן מערך A בגודל n, כל מחסנית תתחילה בקצה אחר של המערך: מחסנית S1 תתחיל ב-A[0], ומחסנית S2 תתחיל ב-A[n-1].

מכניסים איברים חדשים בכיוונים מנוגדים במערך (לכיוון אמצע המערך). צריך לשמור שהראשים של שתי המחסניות לא "יעלו" זה על זה. במידה ומגיעים למקרה כזה מודיעים על גלישה – כי זה אומר שאין יותר מקום במערך ואין אפשרות להוסיף את האיבר המבוקש.

יתרון: ניתן להכניס הרבה איברים למחסנית אחת במידה והשנייה לא מתמלאת. כלומר, לא קובעים מראש כמה איברים יוכלו להיכנס לכל אחת מהמחסניות, וההגבלה היחידה היא גודל המערך. [במלים אחרות, אם מחסנית אחת התמלאה והשנייה לא – לא תתקבל הערה על גלישה, כי עדיין יש מקום במערך.]

מימוש הפעולות:

אתחול - נאתחל כל אחת מהמחסניות:

S1->top = A[0]

S2->top = A[n-1]

– Push

ראשית, נוודא שלא נוצרת גלישה, כלומר – ששתי המחסניות לא נפגשות. במידה ולא, נכניס את האיבר החדש לראש המחסנית המתאימה:

```
void Push (Item_type x, Stack_type *S)
{
    if (S1->top+1 == S2->top)
        Error ("Array is full");

    if (S==S1)
        A[S1->top++] = x;
    else if (S==S2)
        A[S2->top--] = x;
}
```

- Pop

```
void Pop(Item_type * x , Stack_type * S)
{
    if (S==S1)
    {
```

```

        if (S1->top <= 0)
            Error ("Stack is empty");
        else
            *x = A[--S1->top];
    }
    else if (S==S2)
    {
        if (S2->top >= n)
            Error ("Stack is empty");
        else
            *x = A[++S2->top];
    }
}

```

Top – (במידה ונרצה להגדיר כפעולה)

```

Item_type *Top(Stack_type * S)
{ return S->top }

```

– Empty

```

Boolean_type Empty (Stack_type *S)
{
    if (S==S1)
        return S1->top <= 0;
    else if (S==S2)
        return S2->top >= n;
}

```

זמן – $O(1)$.

זמן לכל אחת מהפעולות – $O(1)$