

מבנה נתונים תשפ"ב

פתרון 4 – תור ומחסנית

שאלה 1

- . א. כתבו אלגוריתם המקבל רשימה מקוشرת 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. לאחר ובתור סדר היציאה של האיברים הוא FIFO (first in first out) ובמחסנית סדר היציאה הוא LIFO (last in, out), נעבור על הרשימה המקוشرת ונכנס את $n/2$ האיברים הראשונים לתור, $-1/2$ מהאיברים האחרונים
למחסנית. לאחר מכן נוציא איבר מהתור (מהראשון ועד האיבר ה- $-2/n$), ואיבר מהמחסנית (מהאחרון ועד האיבר
ה- $-1+2/n$) ונדפיס אותם יחד. האלגוריתם תקין מכיוון שהוא עבר על כל האיברים ברשימה המקוشرת, מסדר
אוטם בתור ומחסנית לפי סדר הדפסתם, ולאחר מכן מבצע את החזרה 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 הראשוני 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 $(n)\Theta$. לאחר מכן לולאת החדפסה רצה $n/2$ פעמים וכל פעם מבצעת מסטר קבוע של פעולות בעלות של $(1)\Theta$, שוב מדובר בעלות של $(n)\Theta$. ומהן שסיבוכיות הזמן הכוללת של האלגוריתם היא $(n)\Theta$.

שאלה 2

2. כתבו פעולה המקבלת מהסנית שאינה ריקה, בה יכולים להיות איברים זרים. הפעולה תחזיר את המהסנית בה קיים רק איבר אחד מכל ערך.

למשל, עבור המהסנית: $\{47, 6, \underline{15}, 3, 31, 2, 9\}$ תחזיר המהסנית: $\{47, 6, \underline{15}, 3, 31, 3, 2, \underline{15}\}$

פתרון

- נשתמש בשתי מהסניות עזר בשביל להשאיר בmahsniot S רק איבר אחד מכל ערך. תחילת נעביר את כל האיברים ל- S_2 כדי שנוכל למלא מחדש את S באיברים ייחודיים בלבד. בעת תבצע הלולאה הבאה:
1. מוצאים איבר אחד מ- S_2 , שומרים את ערכו במשתנה val ומכניסים אותו ל- S .
 2. מוצאים את שאר האיברים שנותרו ב- S_2 , מידעה ורעם שונה מערכו של val מכניסים אותו ל- S_1 .
 3. מעבירים את כל האיברים שנכנסו ל- S_1 (עתה נותרו רק-Calha שווים מ- val) בחזרה ל- S_2 וחוזרים לבצע את שלב 1 עבור האיבר הבא.

נקודות: 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 משוים את הערך של איבר אחד רק עם איברים שנמצאים מתחתיו במחסנית S_2 , שהרי האיברים שהיו מעליו כבר נבדקו בשלבים הקודמים ועbero ל- S (או נמחקו, במידה והיו להם כפילים) קיבל:

- **במקרה הגורע** – כל האיברים במחסנית היו שונים זה מזה. לכן בכל שלב משוים בין av לבין כל האיברים שהיו מתחתיו. לכן, באיטרציה הראשונה יהיו $1-n$ השוואות, ובכל איטרציה נוספת מספר ההשואות ירד ב-1. קיבל

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

• **במקרה הטוב** – כל האיברים במחסנית היו זהים. לכן באיטרציה הראשונה נשווה את כלם לאיבר הראשון,

• ולא נעביר אף איבר למחסנית S_1 . כך לא יהיו עד איטרציות, ולכן זמן הריצה הוא $O(1)$.

לסיכום – חסם תחתון $O(n^2)$, חסם עליון $O(n)$.

שאלה 3

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

פתרון

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

מכניסים איברים חדשים בכיוונים מנוגדים במערך (לכיוון אמצע המערך). צריך לשמר שהראשים של שתי המחסניות לא "יעלו" זה על זה. במידה ומגיעים במקרה כזו מודיעים על גישה – כי זה אומר שאין יותר מקום במערך ואין אפשרות להוסיף את האיבר הבא.

יתרונו : ניתן להכניס הרבה איברים למחסנית אחת במידה והשניה לא מתמלאת. כמובן, לא קובעים מראש כמה איברים יכולים להיכנס לכל אחת מהמחסניות, והגבולה היחידה היא גודל המערך. [במלים אחרות, אם מחסנית אחת התמלאה והשניה לא – לא תתקבל הערתה על גישה, כי עדין יש מקום במערך].

IMPLEMENTATION:

Initialization: – נאותל כל אחת מהמחסניות :

S1->top = A[0]

S2->top = A[n-1]

– Push

ראשית, נזדאת לא נוצרת גישה, כמובן – שתי 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];
    }
}

```

(במידה ונרצה להגדיר כפעולה – Top)

```

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;
}

```

. $O(1)$ – זמן

זמן לכל אחת מהפעולות – $O(1)$