חומרי עזר להוראת טנ"מ תור

הוכן במסגרת קורס מורים מובילים תשס"ג

ע"י אסתי מאסטראסי ואסנת אנגלמן, ביה"ס הריאלי העברי חיפה

esti_maestracci@hotmail.co.il; osnat_angleman@walla.co.il

הקדמה

עבודה זו נועדה להוסיף חומר להוראת טיפוס הנתונים – תור, שמוזכר בחטף בספר הלימוד "עיצוב תוכנה", ומופיע פעמים רבות בבחינות הבגרות. לכן , ראינו בהרחבת הנושא מטרה שתעזור למורים רבים ללמד – תור. נושא זה נלמד בדרך כלל לאחר הכרת מבנה הנתונים "מחסנית" ודומה לו במבנה הלימוד. אפשר ללמד תור גם לפני לימוד המחסנית בגלל היותו טיפוס נתונים המוכר לתלמידים מחיי היומיום (תור לאוטובוס, תור לרופא, ...) , וקל יותר לבנות שאלות סיפוריות על תור מאשר על מחסנית. בדומה למחסנית גם תור הוא מקרה פרטי של רשימה. אשר הכנסת איבר בו היא למעשה הוספת איבר לסוף הרשימה , והוצאת איבר בו היא הוצאה מראש הרשימה.

המטרות ששמנו לנגד עינינו היו :

- היכרות עם טיפוס הנתונים תור ושימושים שונים בו.
- הכרת הפעולות המותרות על תור
- יצוג תור בסביבת העבודה
- תרגול : הכולל תרגילים ופתרונותיהם

פירטנו את מבחר האפשרויות של ייצוג תור:

- ייצוג טיפוס התור בעזרת רשימה.
- ייצוג סדרתי של תור:
  1. ייצוג תור בעזרת מערך ושני מציינים:
  2. ייצוג תור בעזרת מערך, מציין אחד ופעולת הזזה:
  3. ייצוג תור בעזרת מערך מעגלי ושני מצביעים
- בניית רשימה מקושרת עם מצביעים לראש ולזנב הרשימה
- דו-תור
- רב תור.

לכל אחד מהנושאים הללו הוספנו את היצוג והמימוש לנוחיות המורים הרוצים ללמד נושא זה . כמו כן יש מספר המלצות המצורפות לתרגילים וליצוגים הבאות מתוך הנסיון שלנו בהוראת תור .

הגדרת טיפוס הנתונים תור

תור הוא אוסף סדור של נתונים אשר מקצהו האחד, **ראש_התור**, ניתן להוציא פרטים ובקצהו האחר, **סוף_התור**, ניתן להוסיף פרטים.

(First In first Out – fifo)

הפעולות המותרות על תור(queue)

שם הפעולה	תיאור הפעולה
אתחל_תור->q	פעולה המאתחלת את התור q להיות ריק
הכנס לתור(q,x)	פעולה המכניסה את האיבר x לסוף התור q. הנחה: תור q מאותחל
הוצא_מתור(q)->x	פעולה המוציאה את האיבר x שבראש התור q הנחה: התור מאותחל ואינו ריק
תור_ריק?(q)	פעולה המחזירה 'אמת' אם התור q ריק ושקר – אחרת

ייצוג טיפוס הנתונים תור בסביבת העבודה:

א. שימוש בייצוג של רשימה.

ב. ייצוג סדרתי של תור (בעזרת מערך ומצביעים)

ג. בניית רשימה מקושרת עם מצביעים לראש ולזנב הרשימה

השוואה בין סבוכיות זמן ריצה לפעולות על תור:

הפעולה	**רשימה**	**מערך+2 מציינים***	**מערך, מציין והזזה**	**מערך מעגלי**	**רשימה מקושרת+2 מציינים**
תור_ריק?	O(1)	O(1)	O(1)	O(1)	O(1)
הכנס_לתור	O(n)	O(1)	O(1)	O(1)	O(1)
הוצר_מתור	O(1)	O(1)	O(n)	O(1)	O(1)
אתחל_תור	O(1)	O(1)	O(1)	O(1)	O(1)

- למרות היעילות הגבוהה למימוש הפעולות על תור בייצוג זה, ראה את הבעייתיות שבייצוג בהרחבה על הנושא בהמשך.

א. <u>ייצוג טיפוס התור בעזרת רשימה:</u>

אתחל_תור ←q

{פעולה המאתחלת את התור q להיות ריק}

1. **אתחל_רשימה** ←q.

הכנס_לתור(q,x)

{פעולה המכניסה את האיבר X לסוף תור q.

הנחה: התור מאותחל

P מצביע לאיבר בתור q}

1. **קודם_ברשימה**(סוף_רשימה(q),(q ←p
2. **הכנס_לרשימה**(q,p,x)

הוצא_מתור (q)←x

{פעולה המוציאה הראש התור q את האיבר x.

הנחה: התור אינו ריק

P מצביע לתור}

1. **עוקב_ברשימה**(עוגן_רשימה(q),(q ←p
2. **אחזר_מרשימה**(q,p) ←x
3. **הוצא_מרשימה**(q,p)
4. החזר את x.

תור_ריק?(q)

{פעולה המחזירה 'אמת' אם התור q הוא ריק ו'שקר' – אחרת

הנחה: התור מאותחל}

1. החזר **רשימה_ריקה**?(q)

ב. ייצוג סדרתי של תור

1. ייצוג תור בעזרת מערך ושני מציינים:

```
Const
   MaxQueue=…;{הגבלת גודל התור ע"י מערך}
Type
   QueueInfoType=….;{סוג הנתונים בתור}
    QueueType=record
         Info:array[1..maxQueue] of queuInfoType;
        Head:1..maxQueue;
        Tail:0..maxQueue;
   End;
```

הערה: בעצם הגדרת התור כמערך, אנו מגבילים את גודלו. לכן שימוש במערך לאחסון התור עלול לעורר בעיה של גלישה מתחומי המערך בעת הוספת פריטים לתור.

אתחל_תור ←q

{פעולה המאתחלת את התור q להיות ריק}

1. q.tail←0
2. q.head←1

הכנס_לתור(q,x)

{פעולה המכניסה את האיבר X לזנב תור q.

הנחה: התור מאותחל ואינו מלא}

1. q.tail←q.tail +1
2. q.info[q.tail]←x

הוצא_מתור (q)←x

{פעולה המוציאה הראש התור q את האיבר x.

הנחה: התור אינו ריק}

1. x←q.info[q.head]
2. q.head←q.head + 1
3. החזר את x.

תור_ריק?(q)

{פעולה המחזירה 'אמת' אם התור q הוא ריק ו'שקר' – אחרת

הנחה: התור מאותחל}

1. החזר q.tail<q.head

*כדאי להריץ בכיתה דוגמא של תור בעל 5 איברים. להוסיף 3 פריטים, להסיר 2 ולהראות כי ניתן להוסיף רק עוד 2 פריטים לתור, למרות שבפועל יש מקום ל- 5 איברים.

ניתן להמשיך ולהראות כי אם נוציא את 3 האיברים הללו שבתור, לא ניתן להוסיף פריטם לתור. מכאן ברור שייצוג זה אינו מניח את הדעת וכי יש לחפש ייצוג הולם יותר.

ב2: ייצוג תור בעזרת מערך, מציין אחד ופעולת הזזה:

בעזרת פתרון זה, לאחר הוצאת איבר מהתור, תהיה הזזה של כל הפריטים בתור לתחילת המערך. בדרך זו, ראש התור תמיד יצביע לתחילת המערך (q.head=1)

לשם פתרון זה, יש לשנות את הייצוג של התור לשדה info אשר יכלול את המערך ולמציין בודד לזנב התור. ולשנות את השגרות של אתחול, הוצאה ובדיקת תור ריק.

```
Const
   MaxQueue=…;{הגבלת גודל התור ע"י מערך}
Type
   QueueInfoType=….;{סוג הנתונים בתור}
    QueueType=record
        Info:array[1..maxQueue] of queuInfoType;
        Tail:0..maxQueue;
    End;
```

אתחל_תור ←q

{פעולה המאתחלת את התור q להיות ריק}

1. q.tail←0

הכנס_לתור(q,x)

{פעולה המכניסה את האיבר X לזנב תור q.

הנחה: התור מאותחל ואינו מלא}

1. q.tail←q.tail +1
2. q.info[q.tail]←x

הוצא_מתור (q)←x

{פעולה המוציאה הראש התור q את האיבר x.

הנחה: התור אינו ריק}

1. x←q.info[1]
2. עבור I←1 עד q.tail -1 בצע
   q.info[i]←q.info[i + 1]
3. q.tail ←q.tail-1
4. החזר x.

תור_ריק?(q)

{פעולה המחזירה 'אמת' אם התור q הוא ריק ו'שקר' – אחרת

הנחה: התור מאותחל}

1. החזר q.tail=0

* פתרון זה טוב יותר מקודמו, בכך שתמיד נוכל להשתמש בכל איברי המערך, אך חסרונו ביעילותו. כדי להוציא איבר עלינו להזיז את כל האיברים במערך מקום אחד קדימה. סבוכיות זמן ריצה של האלגוריתם הינו O(n).

ב3. ייצוג תור בעזרת מערך מעגלי ושני מצביעים

שאלה לתלמידים: הנח כי במקום להזיז את כל פריטי התור לתחילת המערך עם כל ביצוע של פעולת **הוצא_מתור**, נבצע את ההזזה אך ורק כאשר q.tail=maxQueue. האם שיפרנו את האלגוריתם?

תשובה: שיפרנו את זמן הריצה של האלגוריתם **במקרה הממוצע.** במקרה הממוצע פעולת הוצא_מתור תרוץ בסכוביות של O(1), אך במקרה הגרוע ביותר בו q.tail=maxQueue נשארה הסיבוכיות כפי שהיתה |O(n).

נציג כעת דרך אחרת כדי להתגבר על המצב בו לא ניתן להוסיף איברים לתור, אף כי המערך אינו מלא. בואו נתייחס אל המערך כאל מערך מעגלי שבו האיבר הראשון עוקב לאיבר האחרון (במקום maxQueue).

נבנה תור של תווים בעל 5 איברים במספר שלבים:

1. נכניס 4 פריטים לתור A,B,C,D
2. נוציא איבר מתור (x=A)
3. נוציא איבר מתור (x=B)
4. נוסיף 3 איברים לתור E,F,G במקומות הפנויים בתור

שלב מספר 1 שלב מספר 2 שלב מספר 3 שלב מספר 4

מס' איבר במערך	ערך בתור		ערך בתור		ערך בתור		ערך בתור
1	A	q.head=1					F
2	B		B	q.head=2			G
3	C		C		C	q.head=2	C
4	D	q.tail=4	D	q.tail=4	D	q.tail=4	D
5							E

מה עשינו כאן?

השתמשנו בתור מעגלי. הכנסנו ברצף איברים למערך, וכאשר q.tail=maxQueue,(כאשר הוספנו את F), הוספנו איבר למקום 1 במערך (כמובן לאחר שבדקנו שהמקום "אינו תפוס"). הבעיה בייצוג מעגלי היא הקושי להבחין מתי התור ריק. הבדיקה q.tail<q.head אינה תקפה בייצוג זה. ראה בטבלה שלב 4. בשלב 4 התור מכיל 4 איברים ולא מתקיים התנאי כי q.tail<q.head.

דרך אחת לפתרון הבעיה. Q.head יצביע על מיקומו במערך של האיבר שלפני הפריט הראשון בתור(ולא על מיקומו של הפריט הראשון עצמו).
תור ריק יוגדר ע"י q.tail=q.head.

ייצוג תור מעגלי:

```
Const
   MaxQueue=…;{הגבלת גודל התור ע"י מערך}
Type
   QueueInfoType=….;{סוג הנתונים בתור}
    QueueType=record
          Info:array[1..maxQueue] of queuInfoType;
         Head:1..maxQueue;
         Tail:1..maxQueue;
    End;
```

אתחל_תור ←q

{פעולה המאתחלת את התור q להיות ריק}

1. q.tail←maxQueue
2. q.head←maxQueue

הכנס_לתור(q,x)

{פעולה המכניסה את האיבר X לזנב תור q.
הנחה: התור מאותחל ואינו מלא}

1. אם q.tail<maxQueue אז
   q.tail←q.tail + 1
2. אחרת
   q.tail←1
3. q.info[q.tail]←x

הוצא_מתור (q)←x

{פעולה המוציאה הראש התור q את האיבר x.

הנחה: התור אינו ריק

שימו לב: קדמנו את ערכו של q.head לפני הוצאת הפריט, משום שאנו בחרנו להגדיר כי q.head מצביע על המקום שלפני הפריט הראשון בתור}

1. אם q.head<maxQueue אז

   q.head←q.head + 1

2. אחרת

   q.head←1

3. x←q.info[q.head]
4. החזר את x.

תור_ריק?(q)

{פעולה המחזירה 'אמת' אם התור q הוא ריק ו'שקר' – אחרת

הנחה: התור מאותחל}

1. החזר q.tail=q.head

הערה: בדרך מימוש זו אנו "מאבדים" מקום אחד במערך לצורך נוחות במימוש השגרות. כלומר, במערך בגודל n נוכל להכניס לתור מעגלי רק n-1 איברים.

ג. בניית רשימה מקושרת עם מצביעים לראש ולזנב הרשימה

החסרון הבולט של כל הייצוגים הסדרתיים של תור (באמצעות מערך) היא ההגבלה על מספר הפריטים בתור כתוצאה מגודלו הסופי של המערך שהוקצה לאחסון התור.

התור כמבנה נתונים מופשט, אינו מוגבל בגודלו והוא עשוי להכיל כל מספר שהוא של פריטים.

```
Unit queue;
Interface
        Uses list3;

Procedure queue_init(var q:list_type);
{this procedure init the queue q}

function queue_is_empty (q:list_type) : Boolean;
{this function return 'true' when the queue q is empty, otherwise –'false'}

procedure queue_insert(var q:queue_type; x:list_info_type)
{this procedure add the value x to the list's tail}

procedure queue_remove(var q:queue_type; var x:list_info_type);
{this procedure remove the item out of the head of list and insert it to x}

implementation

Procedure queue_init(var q:list_type);
Begin
        List_init(q);
End;

function queue_is_empty (q:list_type) : Boolean;
begin
        queue_is_empty:=list_is_empty(q);
end;
```

```
procedure queue_insert(var q:queue_type; x:list_info_type)
var
p:pos_type;
begin
        p:=list_prev(q,list_end(q));
        list_insert(q,p,x);
end;

procedure queue_remove(var q:queue_type; var x:list_info_type);
var
        p:pos_type;
begin
        p:=list_anchor(q);
        p:=list_next(q,p);
        list_retrieve(q,p,x);
        list_delete(q,p);
end;

begin
end.
```

דו-תור

ניהול שני תורים במקביל במבנה נתונים משותף.

מקובל להשתמש בו כאשר יש צורך לנהל שני תורים נפרדים בעלי ענין משותף.

לדוגמא:

א- רשימת חולים לרופא המופרד לשני תורים , תור למקרים דחופים ותור למקרים רגילים.

ב- רשימת נוסעים לעליה למטוס המופרדת לשני תורים, תור לאחמ"ים ותור לפשוטי העם.

ייצוג הדו-תור:

- כאשר מספר האיברים בתור האחד אינו תלוי במספר האיברים בתור האחר, נגדיר רשומה של דו-תור בעלת 2 שדות, שדה לתור א' ושדה לתור ב'.
- כאשר קיימת תלות בין מספר האיברים בשני התורים. לדוגמא מספר הפציינטים לרופא ביום עבודה הינו סופי(n), יתקבלו לבדיקה קודם כל המקרים הדחופים, ולאחר מכן, המקרים הפשוטים. במקרה שכזה הייצוג אפשרי ע"י מערך בגודל n, כאשר מצידו האחד של המערך מנוהל תור המקרים הדחופים ומצדו האחר – תור המקרים הרגילים.

(אפשרות ייצוג אחרת – ניהול שתי רשימות כאשר סכום האיברים בשתי הרשימות אינו עולה על n.)

מימוש:

אתחל_דו-תור(duQueue)

{פעולה המאתחלת את הדו-תור להיות ריק}

1. **אתחל_תור**(duQueue.queue1)
2. **אתחל_תור**(duQueue.queue2)

דו-תור_ריק?(duQueue)

{פעולה המקבלת דו-תור ומחזירה אמת אם הוא ריק}

החזר **תור_ריק**?(duQueue.queue1) וגם **תור_ריק**?(duQueue.queue2).

הוצא_מדו-תור(duQueue)->X

{פעולה המוציאה את X מתוך תור הדחופים (duQueue.queue1), ואם הוא ריק תוציא איבר מתוך תור הרגילים (duQueue.queue2)}

1. אם לא **תור_ריק**?(duQueur.queue1) אזי

הוצא_מתור(duQueue.queue1)->X

2. אחרת,

אם לא **תור_ריק**?(duQueur.queue2) אזי

הוצא_מתור(duQueue.queue2)->X

הכנס_לדו-תור (duQueue,x,kind)

1. אם kind = 1 אזי

הכנס_לתור(**duQueue**.queue1, x)

2.אחרת,

הכנס_לתור(duQueue.queue2, x).

שאלות:

1. בחינת בגרות, קיץ תש"ס, שאלה 15.
2. בחינת בגרות, קיץ תשנ"ו, שאלה 14.
3. שאלון בגרות לדוגמא, תשנ"ז, שאלה 3.

רב-תור

דומה במהותו לדו-תור, אלא שכאן מתנהלים מספר תורים במקביל עם קשר ענייני בניהם. לדוגמא:

- ניהול תור משתמשים במדפסת מרכזית. כל המנויים בעלי אותה הרשאה יהיה בתור אחד, האחרים בעלי הרשאה גבוהה יותר יהיו בתור אחר וכך יבנה תור לכל רמת הרשאה.
- רב-תור לרוחב פס בתקשורת בין מחשבים
- רב-תור במס הכנסה, כל תור מיועד ללקוחות בעלי אותו ענין.

בייצוג הרב-תור נבחר **רשימה בגודל n שכל אחד מאיבריה מייצג תור בודד**.

הערה: לעתים יש משמעות לתור בין התורים, כך שיוצאו מן התור קודם כל אלו בעלי ההרשאה הגבוהה ביותר ורק לאחר שהוא יתרוקן, יוצאו אלו בעלי ההרשאה הנמוכה יותר (2 הדוגמאות שלעיל) ולעתים אין תלות בין התורים, אלא בין האיברים באותו התור כדוגמא מס' 3 שלעיל.

שאלות לתרגול הרב-תור:

1. במחלקת החלב של מר חלבני יש לכל היותר 100 מוצרים שונים.

ידוע ש"אורך חיי המוצר" של מוצר חלבי הוא מוגבל.

בכל בוקר מקבל מר חלבני תוצרת טריה. הוא מעדיף לסדר את המוצרים כך:

כל מוצר שפג תוקפו – מוצא מהמדף וחוזר לספק.

בקדמת המדפים יוצבו מוצרי החלב בעלי תאריך התפוגה הקרוב ביותר, ובאחורי המדף התוצרת הטריה שהתקבלה באותו היום.

א. ייצג טיפוס נתונים לניהול סידור המדפים בצורה הכדאית ביותר מבחינתו של מר חלבני.

ב. כתוב אלגוריתם עבור הפעולות הבאות:

הוספת_מוצר_למדף(shelves, product)

הורדת_מוצר_שנקנה (shelves, code) ->X

הסרת_מוצרים_שפג_תוקפן(shelves, date)

ג. מהי סבוכיות זמן הריצה לאלגוריתמים בסעיף ב'?

ד. האם טיפוס הנתונים שבחרת לייצוג סידור המדפים מתאים גם לכדאיות הלקוח? אם כן – נמק, אם לא – בחר טיפוס נתונים אחר שיתאים לכדאיות הלקוח.

2. בחינת בגרות, קיץ תשנ"ח, שאלה 12.
3. בחינת בגרות, קיץ תשנ"ז, שאלה 14.

נציג כאן פתרון עבור שאלה מס' 1 בלבד. פתרונות לשאלות מתוך בחינות הבגרות מוצגות בספרה של נוע רגוניס ובמקורות נוספים.

הצעה לפתרון שאלה 1:

<u>ייצוג טיפוס הנתונים</u>:

נייצג את מוצרי החלב של מר חלבני ע"י טיפוס הנתונים רב-תור. כל קוד מוצר (שנסמנו בין 1..100) יהיה תור נפרד. בראש התור יהיו המוצרים בעלי תאריך התפוגה הקרוב ביותר ובסוף התור, מוצרים בעלי תאריך תפוגה הרחוק ביותר.

בגלל סווג המוצרים עפ"י קוד המוצר, קל יותר לנהל את התור בעזרת רשימה המיוצגת ע"י מערך עם 100 תאים, כך כל תא ברשימה יהיה תור, ואינדקס המערך ייצג את קוד המוצר. הנתונים בתור יהיו מסוג תאריך (dates).

```
Const
        Max=100;
Type
        Dates = record
                Year:00..99;
                Month:1..12;
                Day:1..31;
        End;
        listInfoType=record
                code: 1..100;
                date:dates;
        end;
        shelves= array[1..max] of queueType;{כפי שהגדרנו בדפים הקודמים}
```

הוספת_מוצר_למדף(shelves, product)

{פעולה המקבלת מוצר product מסוג queueInfoType ואת המדפים shelves, ומוסיפה את המוצר למדף.

הנחה: המוצרים ממינים במדף עפ"י תאריך תפוגת המוצר.}

הכנס_לתור(shelves[product.code], product.date)

<u>סבוכיות זמן</u>: O(1), זוהי פעולה בודדת כאשר התור הינו מעגלי או ברשימות מקושרות

הורדת_מוצר_שנקנה X<- (shelves, code)

{פעולה המקבלת את המדפים shelves ומוציאה את המוצר הקדמי ביותר בקוד code}

הוצא_מתור(shelves[code])->X.

<u>סבוכיות זמן</u>: O(1), זוהי פעולה בודדת כאשר התור הינו מעגלי או ברשימות מקושרות

הסרת_מוצרים_שפג_תוקפן(shelves, date)

{הסרת כל המוצרים שתאריך תפוגתן קטן מהתאריך date.

הפעולה הציץ_לתור מציגה בx את הערך של האיבר בראש התור. פעולה זו אינה ממומשת כאן, אך קלה למימוש}

1. עבור I<-1 עד max בצע
   1.1 הציץ_לתור x<-(shekves[i])
   1.2 כל עוד x.date<date בצע
      1.2.1 **הוצא_מתור** x<-(shelves[i])
      1.2.2 הציץ_לתור x<-(shelves[i])

<u>סבוכיות זמן</u>: O(n), כאשר n מייצג את מספר המוצרים בחנותו של מר חלבני. במקרה הגרוע ביותר כל המוצרים פג תוקפן.

ד. לא. הלקוח היה מעדיף כי נכניס לראש התור את המוצרים בעלי תאריך התפוגה הרחוק ביותר.(כך 'אורך חיי מוצר' שרכש יהיו ארוכים יותר.). לצורך כך היה הלקוח בוחר בטיפוס הנתונים **מחסנית.**