מטלת מנחה (ממ"ן) 15

הקורס: **20433 - מבני נתונים**

חומר הלימוד למטלה: פרקים 11, 12

מספר השאלות: 5 **משקל המטלה:** 4 נקודות

סמסטר: א2008 **מועד אחרון להגשה:** 14.12.2007

(אב)

אנא שים לב:
מלא בדייקנות את הטופס המלווה לממ"ן בהתאם לדוגמה שלפני המטלות.
העתק את מספר הקורס ומספר המטלה הרשומים לעיל.

שאלה 1 **(15 נקודות)**

נתונים שני מצביעים לשתי רשימות מקושרות. שתי הרשימות משותפות החל ממקום מסוים שאינו ידוע. נסמן ב- ℓ_1 וב- ℓ_2 את אורכן של רשימה 1 ורשימה 2, בהתאמה, מתחילתן ועד לאיבר המשותף הראשון. יהי ℓ_3 אורכה של הרשימה המקושרת המשותפת.

תאר אלגוריתם הרץ בזמן $O(\ell_1+\ell_2+\ell_3)$ המחזיר מצביע לאיבר המשותף הראשון.

פתרון

P1 מצביע לראש הרשימה הראשונה. P2 מצביע לראש הרשימה השנייה.

נספור את מספר האיברים בכל רשימה (אורך הרשימות). נסמן את ההפרש בין האורכים ב–m.

נקדם את המצביע לראש הרשימה הארוכה ביותר ב–m צמתים.

כעת נקדם את שני המצביעים P1 ו–P2 צמת אחרי צמת עד שיפגשו.

שאלה 2 (20 נקודות)

מחסנית נקראת **ממויינת** אם מתקיים אחד מבין שני התנאים הבאים:

(i) המחסנית היא ריקה.

(ii) האיבר המצוי בראש המחסנית קטן מכל שאר האיברים במחסנית, ולאחר שליפתו מתקבלת מחסנית ממויינת.

נגדיר **מיזוג מחסניות** באופן הבא:
בהינתן t מחסניות ממויינות שונות, נבנה מהן מחסנית אחת ממויינת, המכילה את כל האיברים של t המחסניות.

(10 נק') א. כתוב אלגוריתם למיזוג שתי מחסניות S_i ו- S_j לתוך מחסנית S_k.
האלגוריתם אינו רשאי להשתמש במחסניות נוספות, אך הוא רשאי להרוס את תוכן המחסניות S_i ו- S_j.
לצורך כתיבת האלגוריתם ניתן להשתמש בפעולות הבסיסיות המוגדרות על מחסנית.

(3 נק') ב. בהנחה שמחסנית S_i מכילה n_1 איברים ומחסנית S_j מכילה n_2 איברים, מהו המספר המדוייק של פעולות PUSH שיבצע אלגוריתם המיזוג שלך?

(4 נק') ג. לצורך מיזוג שלוש מחסניות משתמשים באלגוריתם מסעיף א' כשגרה.
בהנחה שגדלי המחסניות הם n_1, n_2, n_3 כך ש- $n_1 < n_2 < n_3$, באיזה סדר תבצע את המיזוג כך שהמספר הכולל של פעולות PUSH שיתבצעו יהיה מינימלי?

(3 נק') ד. הצע דרך למזג t מחסניות בסיבוכיות זמן מינימלית.
(רמז: היעזר באלגוריתם שבסעיף א').
נתח את סיבוכיות הזמן של האלגוריתם שהצעת.

א. שפוך Sk ← Si
שפוך Si ← Sj
מזג לתוך Sj   עכשיו התוצאה הרצויה נמצאת ב- Sj
שפוך Si ← Sj
שפוך Sk ← Si

אפשר גם למזג לתוך Sk ועכשיו לשפוך ל-Si ממנו ל-Sj ובחזרה ל-Sk.

ב. כל איבר מועבר ממחסנית למחסנית 4 פעמים לכן 4(n1+n2)

ג. נמזג ראשית את 2 המחסניות הקטנות יותר ולבסוף את הגדולה:

4(n1+n2)+4(n1+n2+n3)

ד. נמזג את כל המחסניות למחסנית התוצאה, נשפוך 3 פעמים עד להחזרה למחסנית התוצאה בצורה הנכונה. סה"כ 4 פעולות PUSH לכל איבר.

שאלה 3 (20 נקודות)

פתור את תרגיל 11.1-6 מספר הלימוד (עמ' 187). ממש תור בעזרת 2 מחסניות

פתרון

ניתן לשתי המחסניות שמות : לאחת נקרא head ולשניה נקרא tail (ולא בכדי...).

כל פעולת enqueue תבצע הכנסה למחסנית head, וכל פעולת dequeue, תבצע שליפה מהמחסנית tail. כמובן, שכדי שניתן יהיה לשלוף מהמחסנית tail, יש להעביר תחילה את תוכן המחסנית head לתוך המחסנית tail. מכיוון ששליפת האיברים מתוך מחסנית, מבוצעת, מהאיבר האחרון שהוכנס, ועד לראשון, הרי שהעברת האיברים מ- head ל-tail, תגרום לאיברים ב – tail, להשלף לפי סדר הכנסתם ל-head .

להלן האלגוריתמים עבור enqueue ו –dequeue, לאחריהם הדגמת הפעולות, ולבסוף ניתוח סדר הגודל של הפעולות.

<u>Enqueue (Q, x)</u>

Push (*Head*, x)

<u>Dequeue (Q)</u>

```
if empty ( Tail)
   while not empty (Head)
         push( Tail, pop(Head))
if empty ( Tail)
   return underflow
else
     return pop ( Tail)
```

שימו לב לנקודה חשובה ומעניינת :

אם נבחן את מספר הפעולות שבוצעו עבור n הכנסות ו– n הוצאות מהתור, הרי שכל איבר הוכנס פעם אחת בלבד למחסנית Head, ופעם אחת בלבד למחסנית Tail. וכן – כל איבר הוצא פעם אחת בלבד מהמחסנית Head, ופעם אחת בלבד מהמחסנית Tail.

כלומר, <u>בממוצע</u> על n הכנסות ו– n הוצאות, כל הכנסה לקחה O(1), וכל הוצאה לקחה O(1).

שאלה 4 (20 נקודות)

תהא T טבלת גיבוב בגודל m, שבה התנגשויות נפתרות באמצעות מיעון פתוח.
האלגוריתם הבא מחפש מקום פנוי בטבלה T עבור המפתח key :

```
SEARCH (T, key)
i:=h(key)
j:=0
while (T[i].k < > nilkey) and (j<m) do
      begin
         j:=j+1
         i:=(i+j) mod m
      end
   if T[i].k = nilkey
      then return i
      else error ('hash table overflow')
```

הערות :

- h היא פונקצית גיבוב שהטווח שלה הוא הקבוצה $\{0,1,...,m\text{-}1\}$
- $T[i].k$ הוא שדה המפתח במקום i בטבלה.
הערך nilkey בשדה המפתח מציין מקום פנוי.

א. הדגם את אופן פעולת האלגוריתם על טבלה T בגודל 8, שבה רק המקום $T[4]$ הוא פנוי. הנח ש- $h(key)=0$. האם האלגוריתם יצליח למצוא את המקום הפנוי בטבלה?

נדגים ע"י טבלת מעקב :

j	i
0	0
1	$(0+1)\bmod 8 = 1$
2	$((0+1)\bmod 8 + 2)\bmod 8 = 3$
3	$((0+1+2)\bmod 8 + 3)\bmod 8 = 6$
4	$((0+1+2+3)\bmod 8 + 4)\bmod 8 = 2$
5	$((0+1+2+3+4)\bmod 8 + 5)\bmod 8 = 7$
6	$((0+1+2+3+4+5)\bmod 8 + 6)\bmod 8 = 5$
7	$((0+1+2+3+4+5+6)\bmod 8 + 7)\bmod 8 = 4$

ב. מהי פונקצית הגיבוב שבה משתמש האלגוריתם?

האלגוריתם משתמש בפונקצית גיבוב ריבועית : למעשה, כאשר i הוא מספר הפעם של ביצוע החיפוש החוזר, סכמת החיפוש היא:

$$i \leftarrow h(k)+\sum_{j=0}^{i-1} j = i \leftarrow h(k)+\frac{(i-1)i}{2} = i \leftarrow h(k)+\frac{1}{2}i^2-\frac{1}{2}i \Rightarrow$$

$$\Rightarrow c_1 = -\frac{1}{2}, c_2 = \frac{1}{2}$$

ג. האם זו פונקציה טובה?

פונקצית גיבוב זו עונה על הדרישות של פונקצית גיבוב טובה, כיוון שההסתברות שמפתח כלשהו יגובב לתא מסוים שווה עבור כל 8 התאים. גם הגיבוב החוזר, עונה על הדרישה, שכאשר טבלת הגיבוב הולכת ומתמלאת, כל מיקום בה, יובא בחשבון בשלב מסוים כתא אפשרי לאחסונו של מפתח חדש.

ד. האם יש לה יתרון על-פני בדיקה לינארית? נמק את תשובתך!

באופן עקרוני, שיטת הבדיקה הריבועית טובה בהרבה מהשיטה הלינארית – כיוון שבעיית הצברים – נפתרת, אך נוצרת בעיה של צברים משניים הנובעים מהעובדה שאם מיקום הבדיקה הראשונה שווה עבור שני מפתחות, אזי סדרת הבדיקה שלהם זהות.

שאלה 5 (25 נקודות)

בסופרמרקט נמכרים n מוצרים שונים. על כל מוצר מוטבע ברקוד, שהוא מחרוזת בת 10 ספרות.
נסמן את קבוצת כל המוצרים ב-S.

תכננו מבנה נתונים, שיאפשר לבצע על קבוצת המוצרים S את הפעולות הבאות:

INSERT (S, x) : הוספת המוצר x ל-S ; תוחלת זמן ביצוע $O(1)$;

DELETE (S, x) : הוצאת המוצר x מ-S ; תוחלת זמן ביצוע $O(1)$;

PRICE (S, b) : החזרת מחירו של המוצר הנושא את הברקוד b ; תוחלת זמן ביצוע $O(1)$;

DISCOUNT (S, p) : הקטנה ב-p% של מחירם של כל המוצרים ; תוחלת זמן ביצוע $O(1)$;

תארו את מבנה הנתונים המוצע והסבירו בקצרה איך תתבצע כל פעולה.

פתרון

מבנה הנתונים הוא טבלת גיבוב (המפתח הוא x), בנוסף נחזיק טבלת גיבוב בה המפתח הוא b והמידע הנוסף הוא x. הבעיה היא כיצד ניתן להוריד את המחיר של כל המוצרים ב p% ב O(1) זמן. לשם כך נחזיק משתנה נוסף בשם Discount. המחיר האמיתי של מוצרים יהיה <u>לא</u> המחיר הרשום בטבלת הגיבוב אלא המחיר הרשום בטבלת הגיבוב כפול המשתנה Discount. יש לשים לב לכן שכאשר מכניסים איבר חדש לא מכניסים אותו כפי שהוא לטבלת הגיבוב אלא מחלקים את המחיר ב Discount (כדי שלאחר ההכפלה ב Discount יתקבל המחיר שהוכנס). בתחילת הפעולה Discount מאותחל ל 1 לאחר מכן ניתן לתת הנחה לכל המוצרים ע"י הכפלת Discount בגורם המתאים (שהוא 100/(100-p)).

לסיכום: מבנה הנתונים הוא טבלת גיבוב (לפי הברקוד) ושדה G שיחזיק את המספר בו צריך להכפיל את המחיר (הוא יאותחל ל1).

הכנסה של X: נכניס לטבלה את המוצר. את מחירו נחלק ב G.

מחיקה של X: מחיקה מטבלת גיבוב.

Price: מחפשים את המוצר בטבלה. את מחירו מכפילים ב–G.

Discount: נכפיל את G פי 1+p/100.