

פרטים טכניים

- מתרגל: ליאור שפירא
- שעת קבלה: יום ג' 15-16, נא לתאם באי-מייל
- המשרד שלי: שרייבר 002 (מעבדת גרפיקה)
- שעות התרגול
- יום רביעי 17-18 ו-18-19
- אתר הקורס
- <http://www.cs.tau.ac.il/courses/Data-Structures/07b/ds07b.htm>

מבני נתונים

תרגול 1

מטרת התרגולים

- לחזור על נושאים קשים מההרצאה
- לתת דוגמאות נוספות לחומר שנלמד בכיתה
- לעבור על תרגילים קשים משיעורי הבית

שיעורי בית



- תרגילים תיאורטיים
 - ינתנו כל שבוע
 - חובת הגשה: 80%
 - 80% מהתרגילים הטובים ביותר ייחשבו כ-10% מהציון
 - התרגילים ייעשו לבד
 - אנא קראו בתשומת לב את ההוראות בכל תרגיל
 - הגשה: או בתרגול או בתא שלי (שרייבר קומה 1 מול המעלית) עד 19:00 ביום התרגול
- תרגילים מעשיים
 - ינתנו 2-4 תרגילים במהלך הסמסטר
 - חובת הגשה: 100%
 - ייחשבו כ-10% מהציון
 - התרגילים ייעשו בזוגות



פסבדו-קוד

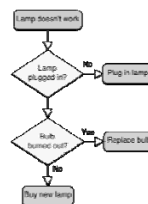
- בקורס זה נכתוב אלגוריתמים ב-psuedo-code
- זהו תיאור קומפקטי ולא רשמי של אלגוריתם במדעי המחשב
- נשמיט פרטים טכניים ולא חשובים ונשמור על העיקר

```
i ← 5
while i > 0 do
  i ← i - 1
end while
```

Assign the value 5 to i
Begin a loop, condition is i > 0
Decrease value of i by 1
End the loop

אלגוריתמים

- סט סופי של הוראות מוגדרות היטב שמטרתם השלמת משימה כלשהיא
- בדומה למתכון, אם תעקבו אחרי ההוראות (ויש לכם קצת כשרון בישול), תגיעו לתוצאה הרצויה
- דוגמה: האלגוריתם של אוקלידס למציאת GCD (מכנה משותף גדול ביותר)



Amortized Analysis

O-Notation

□ בהינתן שתי פונקציות f, g , נאמר ש- $f = O(g)$ אם קיימים c, n_0 כך ש:

$$\forall n \geq n_0, f(n) \leq c \cdot g(n)$$

- סימון אסימפטוטי (עבור n מספיק גדול)
- עד כדי קבוע
- התנהגות עבור הקלט הכי גרוע – worst case
- מחפשים תלות באורך הקלט
- כמה דוגמאות:

$$4n^2 = O(n^4)$$

$$3n = O(2^n)$$

$$\log n = O(n)$$

$$10e^n = O(e^n)$$

מונה בינארי

- מבנה נתונים המחזיק מספר אי-שלילי
- ADT תומך רק בפעולת increment
- קריאה לפעולה increment מעלה את הערך ב-1
- מימוש
- מערך אינסופי A של תאים היכולים להכיל 0 או 1 (מספר בייצוג בינארי)
- נניח שהמערך מאוחלל לאפסים

$$\dots \begin{array}{|c|c|c|c|c|c|c|c|c|} \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline \end{array} = 0$$

$$\dots \begin{array}{|c|c|c|c|c|c|c|c|c|} \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ \hline \end{array} = 1$$

$$\dots \begin{array}{|c|c|c|c|c|c|c|c|c|} \hline 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ \hline \end{array} = 13$$

מהו זמן Amortized

- זמן הריצה הממוצע לפעולה, עבור סדרת פעולות worst case
- נחשב את סיבוכיות הזמן הנדרשת לביצוע m פעולות הכי גרועות, עבור קלט הכי גרוע
- נחלק תוצאה זו ב- m ונקבל סיבוכיות "ממוצעת" לפעולה
- סיבוכיות זו נקראת Amortized
- חשוב: אין אלמנט הסתברותי בשיטה זו!

מונה בינארי

Pseudo code

```

i ← 1
while (A[i]=1) do
  A[i] ← 0
  i ← i + 1
end while
A[i] ← 1
    
```

נניח שהמס' תמיד קטנים מ- n , אזי מה סיבוכיות $w.c$ של פעולת increment?

$$O(\log n)$$

מונה בינארי

- איך תמומש פעולת ה-increment?

■ כל ה-LSB בעלי ערך 1 יהפכו לאפסים וה-0 שאחרי יהפוך ל-1

$$\dots \begin{array}{|c|c|c|c|c|c|c|c|c|} \hline 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ \hline \end{array}$$

$$\dots \begin{array}{|c|c|c|c|c|c|c|c|c|} \hline 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ \hline \end{array}$$

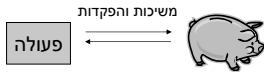
$$\dots \begin{array}{|c|c|c|c|c|c|c|c|c|} \hline 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ \hline \end{array}$$

$$\dots \begin{array}{|c|c|c|c|c|c|c|c|c|} \hline 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ \hline \end{array}$$



שיטת הבנק

- בשיטת הבנק נדרוש מחיר מכל פעולה
- Amortized Cost ייקרא זה מחיר זה
- סכום זה יכול להיות גבוה או נמוך מהעלות האמיתית
- כאשר הסכום גבוה יותר, נצבור את השארית בבנק
- כאשר הסכום נמוך יותר, נשתמש בכסף מהבנק לממן את הפעולה



$$\hat{c}_i = c_i + \text{desposit} - \text{withdraw}$$

$$\sum_{i=1}^n \hat{c}_i \geq \sum_{i=1}^n c_i$$

סך המחירים שנשלם חייבים להיות גבוהים מהעלויות של סך הפעולות

מה הסיבוכיות לביצוע m פעולות?

- נספור כמה פעמים שינינו כל ביט
- את הביט הראשון שינינו m פעמים
- את הביט השני שינינו $\lfloor \frac{m}{2} \rfloor$ פעמים
- את הביט השלישי שינינו $\lfloor \frac{m}{4} \rfloor$ פעמים
- את הביט הרביעי שינינו $\lfloor \frac{m}{8} \rfloor$ פעמים
- ...

הסכום של כל אלה הוא

$$m + \lfloor \frac{m}{2} \rfloor + \lfloor \frac{m}{4} \rfloor + \lfloor \frac{m}{8} \rfloor + \dots \leq m + \frac{m}{2} + \frac{m}{4} + \frac{m}{8} + \dots = 2m = O(m)$$

- לכן m פעולות ייקחו $O(m)$ זמן, נחלק במ ונקבל שהסיבוכיות Amortized היא $O(1)$

שיטת הפוטנציאל

- נניח "קרדיט" במערכת ע"י פונקציית פוטנציאל
- נתחיל עם מבנה נתונים D_0
- עבור פעולות $i = 1 \dots n$ נגדיר
- C_i עלות של פעולה i
- D_i מבנה הנתונים לאחר הפעולה ה- i
- Φ ממפה כל D_i למספר ממשי
- Amortized Cost: $\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1})$
- עלות כוללת:

$$\sum_{i=1}^n \hat{c}_i = \sum_{i=1}^n (c_i + \Phi(D_i) - \Phi(D_{i-1})) = \sum_{i=1}^n c_i + \Phi(D_n) - \Phi(D_0)$$

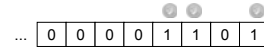
שיטת הבנק

- עבור המונה הבינארי
- המחיר להפוך ביט ל-1 יהיה 2 מטבעות (גבוה יותר)
- מטבע אחד נכנס לבנק
- המחיר להפוך ביט ל-0 יהיה 0 מטבעות (נמוך יותר)
- נצטרך מימן מהבנק

```

i ← 1
while (A[i]=1) do
  A[i] ← 0
  i ← i + 1
end while
A[i] ← 1

```



שיטת הפוטנציאל עבור המונה

- הפוטנציאל של המונה לאחר i קריאות ל-increment הוא b_i , מספר ה'1'ים במונה לאחר הפעולה ה- i .
- נניח שהפעולה ה- i הופכת t_i ביטים ל-0 אזי המחיר שלה הוא $t_i + 1$
- אם b_i שווה 0, אזי הפעולה ה- i מאפסת את כל k הביטים
- אם $b_{i-1} = t_i = k$
- אם $b_i > 0$ אז
- $b_i = b_{i-1} - t_i + 1$
- בכל מקרה
- $b_i \leq b_{i-1} - t_i + 1$
- $\Phi(D_i) - \Phi(D_{i-1}) \leq (b_{i-1} - t_i + 1) - b_{i-1} = 1 - t_i$
- $\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1}) \leq (t_i + 1) + (1 - t_i) = 2$