תשס"ט סמסטר ב'

תרגול 1

מבני נתונים

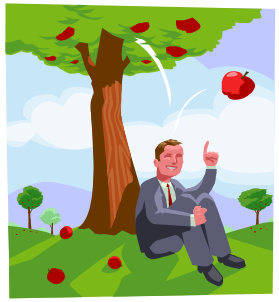
מבוסס על מצגת של
ליאור שפירא

פרטים טכניים

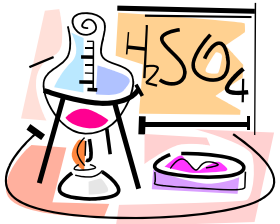
- מתרגלים: יהב נוסבאום ודני פלדמן
  - שעת קבלה: נא לתאם באי-מייל
- אתר הקורס
  - http://ds2009a.wikidot.com
  - נא להירשם לאתר הקורס, ע"י בחירה ב-Join בסרגל העליון
  - סיסמה: data
  - את כל התקשורת והשאלות יש לרכז בפורומים של האתר, פרט לעניינים אישיים-פרטיים.

מטלות

- תרגילים תיאורטיים
  - יינתנו כמעט כל שבוע, זמן הגשה – שבועיים
  - ציון בינארי 1 = עובר, 0 = נכשל
  - התרגילים מיועדים לעבודה עצמית בלבד
  - אנא קראו בתשומת לב את ההוראות בכל תרגיל
  - הגשה לתיבת עץ מס.4, קבלה בחדר 114
  - 10% מהציון הסופי



- פרויקטים תכנותיים
  - יינתנו כ-3 במהלך הסמסטר
  - התרגילים ייעשו בזוגות
  - הוראות הגשה יופיעו באתר
  - 10% מהציון הסופי



- דחיות
  - אין צורך לבקש דחייה של שבוע עבור שלוש מטלות במהלך הסמסטר
  - מעבר לשלוש דחיות, או דחייה ארוכה יותר יש לבקש ולנמק (את כל הדחיות)

מטרת הקורס

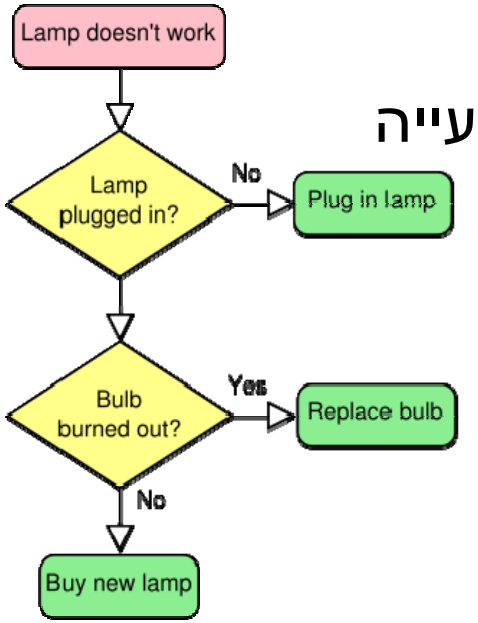
- כלים ומתודולוגיות להגדיר מבני נתונים
- אלגוריתמים שונים על מבני נתונים
- אנליזה של יעילות פעולות שונות ואלגוריתמים שונים

מבני נתונים

- Abstract Data Type
  - אוסף של ערכים, ופעולות שמוגדרות עליהם
  - ללא תלות במימוש
- Data Structure
  - ארגון מידע באופן מועיל
  - מאפשר מימוש של הקודם
  - משמש לאלגוריתמים יעילים מבחינת זמן ריצה או שימוש בזיכרון

אלגוריתמים

- סט סופי של הוראות מוגדרות היטב שמטרתם השלמת משימה כלשהיא
- בדומה למתכון, אם תעקבו אחרי ההוראות תגיעו לתוצאה הרצויה
- דוגמה: האלגוריתם של אוקלידס למציאת GCD (מכנה משותף גדול ביותר)
- אלגוריתם הוא דרך חישובית-שיטתית לפתרון בעייה



פסאדו-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

Asymptotic 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)
- אינטואיטיבית, g חוסמת מלמעלה את f, אסימפטוטית, עד כדי קבוע
- כמה דוגמאות:

$$4n^2 = O(n^4)$$
$$3n = O(2^n)$$
$$\log n = O(n)$$
$$10e^n = O(e^n)$$

מומלץ להסתכל בפרק
3 בקורמן (מהדורה 2)

עוד סימונים אסימפטוטים

סימן	משמעות אינטואיטיבית
$f=O(g)$	g חוסמת מלמעלה את f אסימפטוטית, עד כדי קבוע
$f=\Omega(g)$	g חוסמת מלמטה את f אסימפטוטית, עד כדי קבוע
$f=\Theta(g)$	g חוסמת מלמעלה ומלמטה את f אסימפטוטית, עד כדי קבוע
$f=o(g)$	f זניחה ביחס ל-g, אסימפטוטית
$f=\omega(g)$	g זניחה ביחס ל-f, אסימפטוטית

- הסימן '=' לא מציין שוויון

ניתוח אסימפטוטי

- בהינתן אלגוריתם או מבנה נתונים, ננתח את זמן הריצה שלו ואת כמות הזיכרון שהוא תופס באופן אסימפטוטי
- נחפש את התלות באורך הקלט
- בדרך כלל נתעניין בהתנהגות עבור הקלט הכי גרוע – worst case

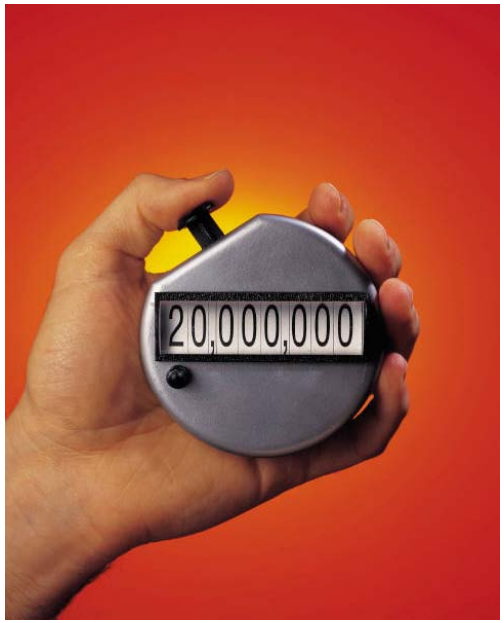
Amortized Analysis

מהו זמן Amortized

- חישוב זמן ריצה כולל תוך "התקזזות" בין פעולות
- זמן הריצה הממוצע לפעולה, עבור סדרת פעולות worst case
  - נחשב את סיבוכיות הזמן הנדרשת לביצוע הסדרה הכי גרוע של n פעולות
  - נחלק תוצאה זו ב-n ונקבל סיבוכיות "ממוצעת" לפעולה במקרה הגרוע
  - סיבוכיות זו נקראת amortized
- מה זה לא
  - זה לא זמן פעולה ממוצע
  - זה לא ניתוח הסתברותי

מונה

- מבנה נתונים המחזיק מספר אי-שלילי
- תומך רק בפעולת increment
  - קריאה לפעולה increment מעלה את הערך ב-1



מימוש מונה בינארי

- מערך אינסופי A של תאים היכולים להכיל 0 או 1
- התאים מייצגים מספר בייצוג בינארי
- נניח שהמערך מאותחל לאפסים

...	0	0	0	0	0	0	0	0	= 0
...	0	0	0	0	0	0	0	1	= 1
...	0	0	0	0	1	1	0	1	= 13

פעולת increment

- איך תמומש פעולת ה-increment?

...	0	0	0	0	1	1	0	1

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

...	0	0	0	0	1	1	**1**	**0**
...	0	0	0	0	1	1	1	**1**
...	0	0	0	**1**	**0**	**0**	**0**	**0**

מימוש increment

Pseudo code

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

נסמן ב-n את מספר הפעולות.

מהי סיבוכיות w.c של פעולת increment? $O(\log n)$

מהי סיבוכיות w.c של סדרת הפעולות? $O(n \log n)$

למה אין ל-log בסיס?

מה הסיבוכיות לביצוע n פעולות?

- נספור כמה פעמים שינינו כל ביט
  - את הביט הראשון שינינו n פעמים
  - את הביט השני שינינו $\lfloor n/2 \rfloor$ פעמים
  - את הביט השלישי שינינו $\lfloor n/4 \rfloor$ פעמים
  - את הביט הרביעי שינינו $\lfloor n/8 \rfloor$ פעמים
  - ...
- הסכום של כל אלה הוא

$$n+\lfloor n/2 \rfloor+\lfloor n/4 \rfloor+\lfloor n/8 \rfloor+\ldots \le n+\frac{n}{2}+\frac{n}{4}+\frac{n}{8}+\ldots=2n=O(n)$$

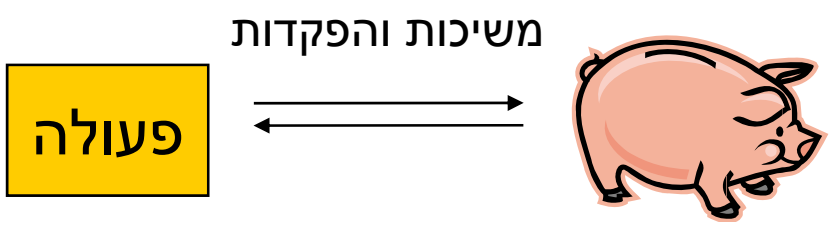
- לכן סדרת הפעולות תיקח $O(n)$ זמן, נחלק ב-n ונקבל שהסיבוכיות amortized לפעולה היא $O(1)$
  - האם קיבלנו סתירה לשקף הקודם?

שיטת הבנק

- בשיטת הבנק נדרוש מחיר מכל פעולה
- מחיר זה ייקרא Amortized Cost
  - 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
  - היתרה בבנק לעולם תהיה אי-שלילית

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

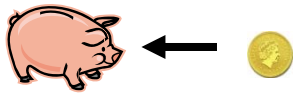
$$\sum_{i=1}^{n} \hat{c}_i \geq \sum_{i=1}^{n} c_i$$



סך המחירים שנשלם חייבים להיות גבוהים מהעלויות של סך הפעולות

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 מימון מהבנק

- הפיכת ביטים ל-1 למעשה מממנת הפיכת ביטים ל-0

…	0	0	0	0	1	1	0	1

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

שיטת הפוטנציאל

- נייצג "קרדיט" במערכת ע"י פונקצית פוטנציאל Φ
- נתחיל עם מבנה נתונים D_0
- עבור פעולות $i=1..n$ נגדיר
  - c_i עלות של הפעולה ה-i
  - D_i מבנה הנתונים לאחר הפעולה ה-i
- Φ ממפה כל D_i למספר אי-שלילי
- 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 הפוטנציאל עבור המונה

- נסמן את מצב המונה לאחר i קריאות ל-increment ב- D_i
- פונקציית הפוטנציאל תהיה מספר ה1-ים במונה
- נניח שהפעולה ה-i הופכת t_i ביטים ל-0, אז זמן הריצה שלה הוא $c_i = t_i + 1$
- השינוי בפוטנציאל הוא

$$\Phi(D_i) - \Phi(D_{i-1}) = (b_{i-1} - t_i + 1) - b_{i-1} = 1 - t_i$$

- ולכן, העלות הכוללת היא

$$\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1}) = (t_i + 1) + (1 - t_i) = 2$$