פרטים טכניים

- מתרגל: דן פלדמן
  - שעת קבלה: מתי שרוצים, נא לתאם באי-מייל
  - לאתר הקורס ניתן להגיע דרך:
- http://www.cs.tau.ac.il/~dannyf/
- או לרשום Danny Feldman ב-Google

1

הטלפון של המתרגל (מעדיף מיילים)

052-8550211

אפשר להתקשר מתי שרוצים.

שיעורי בית

- תרגילים תיאורטיים
  - יינתנו כל שבוע/שבועיים
  - חובת הגשה: 100%
  - הגשה: ישירות לתא הבודק שיפורסם באתר.
  - החזרה: חדר צילומים (קומה 1, שרייבר)
  - איחורים: להגיש בהקדם ולצרף פתק עם סיפור ואישורים אם יש.
  - **לא** לשלוח מייל לגבי הגשה באיחור
  - עדכונים ברשימ התפוצה של הקורס. בידקו שאתם מקבלים דואר!
- פרוייקטים
  - יינתנו 1-2 פרוייקטים במהלך הסמסטר
  - חובת הגשה: 100%
  - הפרוייקטים ייעשו ביחידים

2

* הגשה ישירות בתא של הבודק, לוקחים את התרגילים הבדוקים מחדר הצילומים.

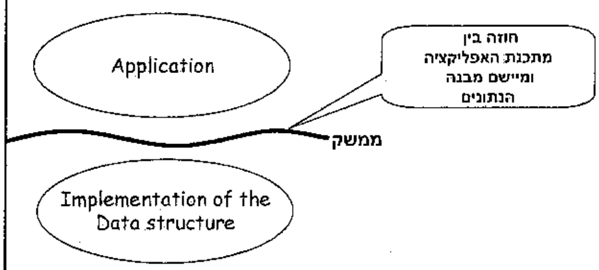
יש מבחן (אולי) תהיה מתרגילי בית.

ADT is an interface

- It defines
  - the type of the data stored
  - operations, what each operation does (not how)
  - parameters of each operation

3

ADT



4

במבחן, נתבקש לרשום אלגוריתמים
ונדון יותר עפ״י היעילות.

מידעון מבחן

פסאודו-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	Assign the value 5 to i
while i>0 do	Begin a loop, condition is i>0
i ← i-1	Decrease value of i by 1
end while	End the loop

5

Example: Stacks

- Push(x,S) : Insert element x into S
- Pop(S) : Delete the last element inserted into S
- Empty?(S): Return yes if S is empty
- Top(S): Return the last element inserted into S
- Size(S)
- Make-stack()

6

Implementation

- We will be interested in algorithms to implement the ADT..
- And their efficiency..

7

Big-O

- $T(n) = O(f(n))$ קיים c ו-n_0 כך ש:

$$T(n) \le c \cdot f(n) \quad \forall n > n_0$$

דוגמא:

$$T(n) = (n+1)^2$$

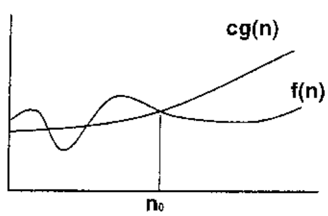
$$(n+1)^2 \le 4n^2 \Rightarrow O(n^2)$$

8

אנחנו נשתמש בכתיב O - כלומר כתיבה
עד רמה של קבועים. כלומר התעסקות
בסדרי גודל. הרעיון שעומד מאחורי
השיטה הוא שיקול כללי - לרכוש מחשב
מהיר יותר פי 2 יותר זול מאשר
להעסיק תוכניתן שירשום קוד היעיל פי 2.
כשמדובר בסדרי גודל הסיפור קצת שונה.
בדר"כ בבעיות שלנו, הפתרון הטריוויאלי יהיה
בסיבוכיות $O(n^2)$ והפתרון ה"אידיאלי" יהיה
$O(n \log n)$. למרות שבעיקרון $O(n)$ אומר
לרוץ מהר יותר מ $O(n \log n)$, הרבה פעמים
השני ירוץ הרבה יותר מהר בפועל (בהתאם
לקבועים).

Big-O

$$f(n) = O(g(n))$$



9

More examples

- $4n \in O(n^2)$
- $4n^2 \in O(n^2)$
- $2^n \notin O(n^{10})$
- $10 \in O(1)$
- $100n^3+10n \in O(n^3)$
- $\log_2(n) \in O(\log_{10}(n))$

רקורסיה

- נקבל ביטוי מהצורה $T(n) = aT(g(n)) + h(n)$
- נרצה למצוא f(n) כך ש- $T(n) = \Theta(f(n)$
- n מספר שלם
- תמיד נחשב עד תנאי עצירה מסוים
- בקורס הזה נתעניין בהתנהגות האסימפטוטית של T ולא בחישוב מדויק

תרגיל 1

- פתרו את הרקורסיה הבאה: $T(n) = 9T(n/7) + n$
- פתרון 1: Brute Force $T(n) = \Theta(f(n)$

$$T(n) = 9T(n/7) + n =$$

$$9(9T(n/49) + n/7) + n = 81T(n/49) + \frac{9n}{7} + n =$$

$$81(9T(n/7^3) + n/49) + \frac{9n}{7} + n = 9^3T(n/7^3) + 9^2\, n/7^2 + 9n/7 + n =$$

...

$$9^{k+1}T(n/7^{k+1}) + n\underbrace{\left(1 + \frac{9}{7} + \left(\frac{9}{7}\right)^2 + \dots (9/7)^k\right)}_{\infty}$$

זו שיטת חישוב נאיבית, לפעמים היא עובדת ולפעמים לא, בעקרון יש דרך אחרת לפתרון.

$\frac{n}{7^k} = 1 \Rightarrow n = 7^k \Rightarrow k = \log_7 n$

בקורס נסמן log ללא בסיס כיוון שההפרש הוא עד קבוע.

תרגיל 1

- מה עשינו לא נכון?
  - אי אפשר להמשיך עד אינסוף, צריך לעצור ב-T(1) או קבוע אחר
  - באיזה k נעצור? $\frac{n}{7^{k_0+1}} = 1 \Rightarrow 7^{k_0+1} = n \Rightarrow k_0 = \log_7 n$
  - זהו **עומק הרקורסיה** (ההתפצלות מדרגה 7)
  - ונחזור לביטוי...

$$T(n) = 9^{\log_7 n+1} \cdot T(n/7^{\log_7 n+1}) + n\left(1+\frac{9}{7}+\left(\frac{9}{7}\right)^2+\dots(9/7)^{\log_7 n}\right)$$

$$= \Theta(9^{\log_7 n} + n(9/7)^{\log_7 n}) = \Theta(n^{\log_7 9})$$

13

$$\theta(9^{\log_7 n}) = \theta(7 \cdot \tfrac{9}{7})^{\log_7 n} = \theta(7^{\log_7 n}) \cdot \theta \Leftarrow$$ [illegible]

חשוב לזכור את סכום סדרה הנדסית

$1+2+4+\dots+n = \theta(n)$

$1+3+9+27+\dots+n = \theta(n)$

סדרה חשבונית לעומת זאת:

$1+2+3+4+5+\dots+n = \theta(n^2)$

בשלב הרל"ן בקורס יהיה לזכור מפתרון של סדרה חשבונית לפתרון של סדרה הנדסית.

רקורסיה – Master Theorem

- "מתכון" לפתור רקורסיות מסוג מסוים
- עבור נוסחת רקורסיה כך ש $T(n) = aT(n/b) + f(n)$
- קיימים שלושה מקרים

1. $\exists \varepsilon > 0$: $f(n) = O(n^{\log_b a - \varepsilon}) \rightarrow T(n) = \Theta(n^{\log_b a})$
2. $f(n) = \Theta(n^{\log_b a}) \rightarrow T(n) = \Theta(n^{\log_b a} \log n)$
3. $\exists \varepsilon > 0, c < 1$: $f(n) = O(n^{\log_b a + \varepsilon})$, $af(n/b) \le cf(n)$ $\rightarrow T(n) = \Theta(f(n))$

14

שיטת "הצב" - בדר"כ עובדת (לא תמיד?)

$n^{\log_b a} = n^{\log_7 9} = n^{1+\varepsilon}$

חזרה לתרגיל 1 $T(n) = 9T(n/7) + n$

- a=9;b=7;f(n)=n
- האם אנחנו במקרה הראשון?

$\exists \varepsilon : n = O(n^{\log_7 9 - \varepsilon})$

$\log_7 9 \approx 1.2$

$\varepsilon = 0.1 \Rightarrow n \le n^{1.1} \Rightarrow$

$T(n) = \Theta(n^{\log_7 9})$

15

דוגמה נוספת

$T(n) = 2T(\frac{n}{2}) + 10n$

$a = 2, b = 2, f(n) = 10n, \log_b a = \log_2 2 = 1$

$n^{\log_b a - \varepsilon} = n^{1-\varepsilon} \Longrightarrow 10n = O(n^{1-\varepsilon})?$

$n^{\log_b a} = n \Longrightarrow 10n = \Theta(n)?$

מסקנה: זהו מקרה 2

תרגיל 3

- פתרו את הרקורסיה הבאה: $T(n) = 2T(\frac{n}{2}) + n\log^2 n$
- פתרון
  - בואו ננסה לפתוח את זה קצת...

$T(n) = n\log^2 n + n\log^2(\frac{n}{2}) + n\log^2(\frac{n}{4}) + n\log^2(\frac{n}{8}) + \ldots$

  - נמצא חסם משני הצדדים

כשן מגיעים ב $\log n$ כי זה כמו

סדרה הנדסית רק שהולכים אחורה.

כלומר נעצור ב $\frac{n}{2^{\frac{\log n}{2}}}$

$\frac{n}{2^{\frac{\log n}{2}}} = \frac{n}{2^{\frac{1}{2}\log n}} = \frac{n}{\sqrt{2^{\log n}}} = \frac{n}{\sqrt{n}} = \sqrt{n}$

$(a^{\log_a b} = b)$

תרגיל 3

- חסם תחתון
  - בואו נשמור רק את חצי האברים הגדולים ביותר
  - מיהו האבר הקטן ביותר ששמרנו?

$$\frac{n}{2^{\log n/2}} = n \cdot \sqrt{\frac{1}{2^{\log n}}} = n \cdot \sqrt{\frac{1}{n}} = \frac{n}{\sqrt{n}} = \sqrt{n}$$

  - לכן אם נחליף את n/k בשורש n נקבל...

$$T(n) \geq \frac{\log n}{2} n\log^2\sqrt{n} = \frac{\log n}{2} n(\frac{1}{2}\log n)^2 = \frac{n\log^3 n}{8}$$

$$T(n) = \Omega(n\log^3 n)$$

תרגיל 3

- חסם עליון
  - נתליף כל n/k ב-n ונקבל

$$T(n) \le n\log^2 n + n\log^2(n) + n\log^2(n) + n\log^2(n) + \dots$$
$$= \log n \cdot n\log^2(n) = n\log^3 n$$
$$T(n) = O(n\log^3 n)$$

תרגיל

- בהינתן n מספרים שלמים בטווח $1..n^c$ כיצד ניתן למיין בזמן $O(n\cdot c)$
- פתרון
  - נתחיל ממספרים בטווח $1..n^2$ האם ניתן למיין אותם ב$O(n)$?
  - אילו אלגוריתמים אנו מכירים שממיינים בזמן לינארי?
    - Counting sort – בהנתן n מספרים בטווח $0..k$ $\theta(k+n)$

$k = O(n)$

  - נכתוב כל מספר בבסיס n, כמה ספרות?
  - נמיין כמו radix sort, $O(n)$ לכל ספרה ולכן $2O(n)=O(n)$
  - עבור הבעיה המקורית כמה ספרות יהיו לנו?

הסוף

שבוע נעים