בהצלחה!

הרצאה 1
21.2.05

1

תכנון וניתוח אלגוריתמים

סמסטר: מה שהיה במבני נתונים.
קורס באלגוריתמיקה קומבינטורית
מבחן לא יהיה על מה שלומדים
תרגילים ממבחנים קודמים. / או יותר קשה.
לבוא לשעות קבלה / ראש החוג
לא יהיו פתרונות לתרגילים. - לפתרונות לבוא
לשעות קבלה. אילן ניומן יום ב' 10-12.
חדר שלו: או חדר ראש החוג.
תרגילים 10-15% (מגן ~~[illegible]~~ ציון לא מחשב ציון) יהיה גבול בסדיקי המבחן.
לא יהיה מבחן עזר.

- קצב הגדילה משתנה.
- תכונות לכל קלט.
- יעילות: 1 זמן -- W.C.
  2 מקום

קיים מחשב שמבצע 10^6 פעולות בשניה

סיבוכיות	1sec.	1hour	1day	1year	100 years
n	10^6	$3.6\cdot10^9$	$86\cdot10^9$	$3.0\cdot10^{12}$	$3.0\cdot10^{14}$
n^2	10^3	$6\cdot10^4$	$3\cdot10^5$	$6\cdot10^5$	$6\cdot10^7$
2^n	20	28	38	44	51
$n!$	9-10	13-[illegible]	14	16	17.5

הפרד ומשול:

מיזוג - טריוויאלית: מיון של מספרים.
מיון 1-n ומספרים (שרירותיים מתוך הקלט)
נכניס סוג המספר ה-i לתוך ~~הקלט~~ סדרה
ממוינת.

מימוש: המערך, פעולות הכנסה לוקחות $O(n)$

$$t(n) = t(n-1) + n$$
$$t(n) = O(n^2)$$

מימוש: מבנה נתונים שבו הכנסה מוצאים $O(\log n)$

$$t(n) = t(n-1) + \log n$$
$$t(n) = O(n \cdot \log n)$$

מיון 2: נחלק n מספרים ל-2 קבוצות (שרירותיות) כל אחת $\frac{n}{2}$ ונמיין כל אחת

למזג אותן יחד כשיווצר מערך שממוין

ממוינת אחת.

$$t(n) = 2t\left(\frac{n}{2}\right) + \underbrace{n}_{\text{מיזוג}}$$

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

$$t(n) = n + 2t\left(\frac{n}{2}\right)$$

$$t(n) = n + 2\left(2t\left(\frac{n}{4}\right) + \frac{n}{2}\right)$$
$$= n + n + 4t\left(\frac{n}{4}\right)$$
$$= i \cdot n + 2^i t\left(\frac{n}{2^i}\right)$$
$$= n \log n + n \cdot \underbrace{t(1)}_{1}$$

בעיית ה-ranking של איברים (בעיית חציון)

קלט: n מספרים שלמים (לא ממוינים) $x_1 \dots x_n$

פלט: מספר k ($1 \le k \le n$)

מספר ה-k-י בגודלו ברשימה.

$k=1$ - מינימום

$k=n$ - מקסימום

3

פתרון סטנדרטי: למיין ולקחת איבר ה-d
ברשימה ממוינת. $O(n\log n)$

כל מספר $\log n$ בשיטה
$O(n\log n)$

אם $n \le b$ $\quad b\log n < n\log n$

שקים סוכר $\quad$ 1kg $\quad$ = 5 שקים

משקל: סה"כ 5kg

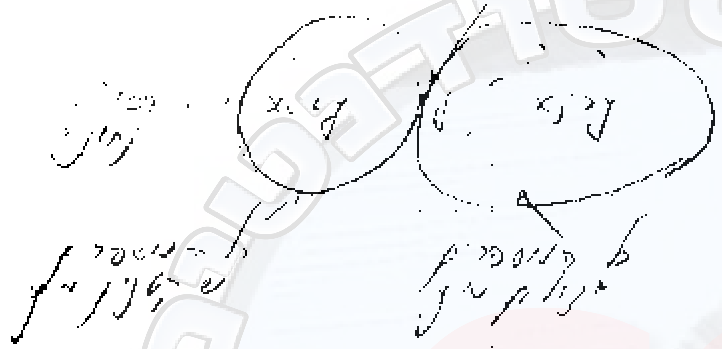
1/2

24.2.05
אלגוריתמים
הרצאה 2
(1)

- בעיית החציון

נתונים $x_1,\ldots,x_n$ מספרים
ונתון מספר k. רוצים למצוא k מספרים
שקטנים (שווים) מ-k ו- n-k גדולים
שגדולים שווים לו
לשם פשטות - מניחים כי כל המספרים שונים
מ-k.

הרעיון הוא:



נגדיר

$$S_L = \{x \mid x \le y\} \quad , \quad S_H = \{x \mid x > y\}$$

המטרה היא לא לגלות מהן הקבוצות אלא לדעת כי
$|S_L| + |S_H| = n$,אם נדע כי

$$|S_L|, |S_H| \le \frac{3}{4}n \iff |S_H|, |S_L| \ge \frac{1}{4}n$$

דוגמא:
אם רוצים למצוא מספר $\frac{n}{100}$ אז הוא יהיה בקבוצה S_L והוא יהיה איבר $\frac{n}{100}$ בקבוצה, לאותה S_L.
אם רוצים למצוא מספר $\frac{2}{3}n$ אז הוא יהיה בקבוצה S_H והוא יהיה איבר $(\frac{2}{3}-\frac{2}{5})n$ בקבוצה S_H.

בסופו של דבר אנחנו נמצאים בבעיה דומה לחפש בגודל הקבוצה ונקבל נוסחה רקורסיבית (?)
[illegible]

נראה אלגוריתם שבהינתן $(x_1, \ldots, x_n)$ מוצא y
שעבורו $|S_H|, |S_L| \le \alpha \cdot n$
עבור $\alpha < 1$ (α לא תלוי ב n).

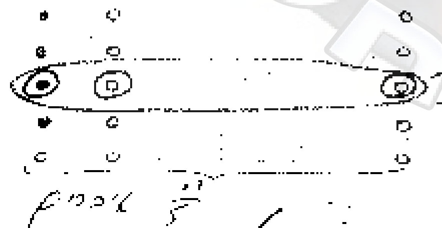
דוגמאות לא טובות של זמנים בקריטריון של אלגוריתם
- אם $|S_L|, |S_H| = n-1$
אז נקבל
$$t(n) = n + t(n-1) = O(n^2)$$
- אם $|S_L|, |S_H| \le n - \sqrt{n}$
נקבל
$$t(n) = \Omega(\sqrt{n} \cdot n)$$

אלגוריתם למציאת איבר מפצל:

נתונים לנו: $x_1, \ldots, x_n$ איברים שונים.

1. חלק את הקבוצה לחמישיות כנ"ל.
(אם לא מתחלק בחמש: נשלים במקסימום הגדול ב 4 איברים ריקים שאז נוכל לחלק ב 5 ללא שום בעיה)

$O(n)$



2. בכל חמישיה מצא את החציון של החמישיה.

3. מצא את החציון y (כלומר את האיבר ה $\frac{n}{10}$) בקבוצת חציוני החמישיות.

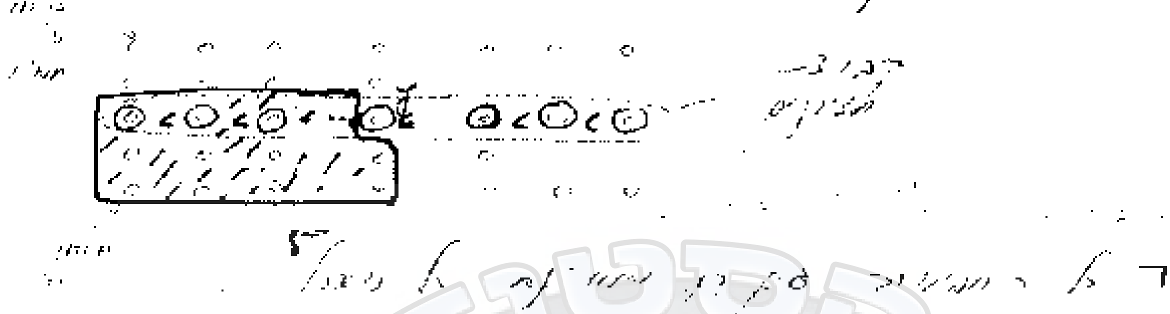
פלט: y הוא האיבר המפצל.

24.2.05
אלגוריתמים
הרצאה 2

(2)

הוכחה שהפלט y הוא איבר מפצל טוב:

- לפחות מחצית מהחציונים של החמישיות, קטנים מ-y
החמישיות בגלל שהוא החציון של החציונים
לפחות מחצית מהן קטנות מהחציון הנ"ל.



- כל החלק המסומן בכחול הוא בוודאי איברים קטנים מ-y ולכן חייבים להיות בתוך S_L ולכן

$$|S_L| \ge \frac{3n}{10} \longrightarrow |S_H| \le \frac{7}{10}n$$

- ובאופן דומה לגבי האיברים שגדולים מ-y $|S_L| \le \frac{7}{10}n$.

<u>אלגוריתם למציאת איבר ה-r בגודלו מתוך $x_1 \ldots x_n$</u>

מיון ברירה $A(\{x_1,\ldots,x_n, r\})$

0. אם קטן מחשב ישירות את האיבר ה-r באופן ישיר.

1. חלק לחמישיות, ומצא חציוני החמישיות. סמן קבוצה זו ב-B ($|B| = \frac{n}{5}$).

2. הפעל $y = A(B, \frac{n}{10})$ (מצא חציון של חציוני החמישיות).

3. בנה

$$S_L = \{x \mid x \le y\}$$

$$S_H = \{x \mid x > y\}$$

4. אם $|S_L| \ge r$ הפעל $A(S_L, r)$

אחרת הפעל $A(S_H, r-|S_L|)$

ניתוח א. זמן ריצה של האלגוריתם:

$$t(n) = n + t(\tfrac{n}{5}) + n + t(\tfrac{7}{10}n)$$

שלב 4   שלב 3   שלב 2   שלב 1

המקרה הגרוע ביותר $\frac{7}{10}n$

?

אז:

$$t(n) \le 2n + t(\tfrac{9}{10}n) = O(n)$$

$t(\frac{n}{5}) < \frac{1}{5}t(n)$

$3t(\frac{n}{3}) \le t(n)$

למה בין 5 - למשל:

$$t(n) = n + t(\tfrac{2n}{10}) + t(\tfrac{8n}{10}) \le n + t(n) = O(n\log n)$$

$$t = n + t(\tfrac{2n}{10}) + t(\tfrac{9n}{10}) = O(n^{1+\epsilon})$$

כאשר ϵ תלוי במה שכתוב ב *

תכנון דינמי

בעיה:

נתונה סדרת מספרים $a_1, \ldots, a_n$ (שליליים או חיוביים) רוצים למצוא תת-סדרה רצופה בעלת סכום גדול ביותר.

רוצים אלגוריתם ליניארי:

$a_1 \ldots a_{k-1}\ a_k \ldots a_n$

- בודקים את הפתרון הנוכחי סכום יותר מאשר שצריכים בסוף.

נגדיר:

$S_k^+ =$ ערך הפיתרון הטוב ביותר המסתיים ב a_k.

$S_k^- =$ ערך הפיתרון הטוב ביותר שמסתיים לפני a_k.

28.2.05
אלגוריתמים
הרצאה 3
1

המשך דוגמאות מתרגול הקודם:

נתונים $a_1, a_2, \dots, a_n$ (חיוביים או שליליים)
ורוצים למצוא ערך של תת סדרה $a_i, a_{i+1}, \dots, a_j$
שסכומה גדול ביותר.

רעיון:

$$a_1 \quad a_2 \; \_ \; \_ \; \_ \; \_ \; a_{n-1} \quad a_n$$
$$\underbrace{a_i \; \_ \; \_ \; \_ \; a_j}$$

- אם $n \neq j$: אזי ערך הפתרון לבעיה עם $a_1 \dots a_n$ זהה לערך הפתרון לבעיה $a_1 \dots a_{n-1}$

- אם $n = j$: ערך הפתרון הוא a_n + ערך הפתרון הטוב ביותר ל-$a_1 \dots a_{n-1}$ שמסתיים ב-a_{n-1} (או ריק)

- נגדיר לכן לכל $i = 1, \dots, n$:

$S_i^- = \{$ ערך הפתרון הטוב ביותר למספרים $a_1 \dots a_i$ שלא מכיל את a_i $\}$

$S_i^+ = \{$ ערך הפתרון הטוב ביותר למספרים $a_1 \dots a_i$ שמכיל את a_i $\}$

מסקנה 1: ערך הפתרון המבוקש הוא $\max(S_i^-, S_i^+)$

טענה:

$$\left.\begin{aligned} S_i^- &= \max\{S_{i-1}^-, S_{i-1}^+\} \\ S_i^+ &= \max\{a_i, S_{i-1}^+ + a_i\} \end{aligned}\right\} *$$

הוכחה: ההוכחה היא מתוך ההגדרה של S_i^- ו S_i^+ והיא הוכחה טריוויאלית.

אלגוריתם:

עבור $i=1$ עדכן $S_1^- = 0$, $S_1^+ = a_1$

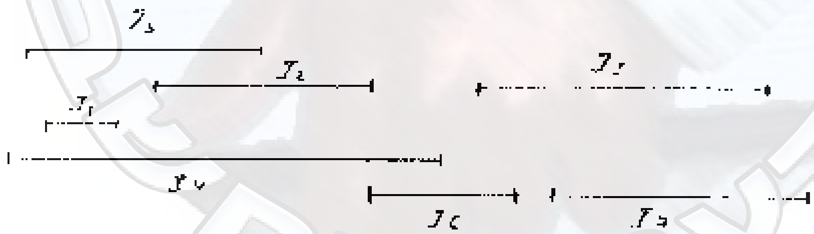
- עבור $i=2,\dots,n$ חשב S_i^-, S_i^+ לפי *
- חשב את הפלט $= \max(S_n^-, S_n^+)$

סיבוכיות: $O(n)$

מרכיבים חיוניים של תכנון דינאמי

- רעיון (לפירוק הבעיה הגדולה לבעיות קטנות יותר (לפעמים תוך כדי זה שמחשבים יותר ממה שצריך))
- פרמטר של התכנון הדינאמי
- נוסחה שמקשרת בין בעיה עם פרמטר i לבעיות עם פרמטרים קטנים.

בעיית הקטעים הזרים של הקו:



קלט: אוסף של n זוגות- $\{(a_i,b_i) \mid i=1,\dots,n\}$

רוצים למצוא את קבוצה של זוגות $(a_{i_1},b_{i_1})(a_{i_2},b_{i_2})\dots(a_{i_k},b_{i_k})$ זרים בזוגות- וכך ש k גדול ככל האפשר.

? כלומר רוצים למצוא את k ולא הקבוצות. אך אם עושים תכנון של אלגוריתם ניתן למצוא גם אותן

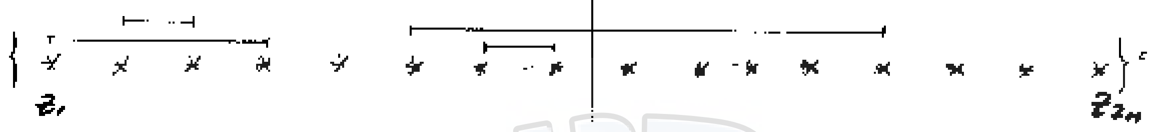
יש לנו n קטעים $\Leftarrow$ יש 2^n קבוצות.

28.2.05
אלגוריתמים
נ' גשר 3:

(2)

הערה: לשם פשטות נניח שאין שני קטעים אשר מסתיימים
או מתחילים באותה נקודה כלומר כל המספרים
$a_1, ..., a_n, b_1, ..., b_n$ כולם זרים (שונים)
זה מזה.

- נמיין את קבוצת הקצוות- $\{a_1, ..., a_n, b_1, ..., b_n\}$



קבוצה ממוינת של הקצוות

סימון: אם z_i הוא קצה שמאלי של קטע

$$L(z_i) = \begin{cases} 1 & \\ 0 & \text{אחרת} \end{cases}$$

עבור i כלשהו, נגדיר - אוסף הקטעים המוגדר ע"י i =
= כל הקטעים שמסתיימים ב-z_i או לפניו.

- נמצא לכל $i = 1, 2, ..., 2n$ את הפתרון הטוב ביותר
עבור הקטעים המוגדרים ע"י i.

- אם $L(z_i) = 1$ אז $O_i = O_{i-1}$

ללא שינוי

- אם $L(z_i) = 0$ אז
או ש $I(z_i)$ לא נמצא בפתרון האופטימלי
ואז נקבל כי $O_i = O_{i-1}$

או ש $I(z_i)$ שייך לפתרון האופטימלי.
$O_i = 1 + O_j$

כאשר $j = left(z_i)$ = הקצה השמאלי של הקטע שמסתיים ב-z_i

טענה: אם $L(z_i)=1$ אז $O_i = O_{i-1}$

אם $L(z_i)=0$ אז *

$O_i = \max(O_{i-1}, 1+O_j)$

כאשר $j = left(z_i) =$ הקצה השמאלי של הקטע שמסתיים ב z_i

הוכחה: טריוויאלית מתוך ההגדרה.

אלגוריתם: - איתחול $O_1 = 0$

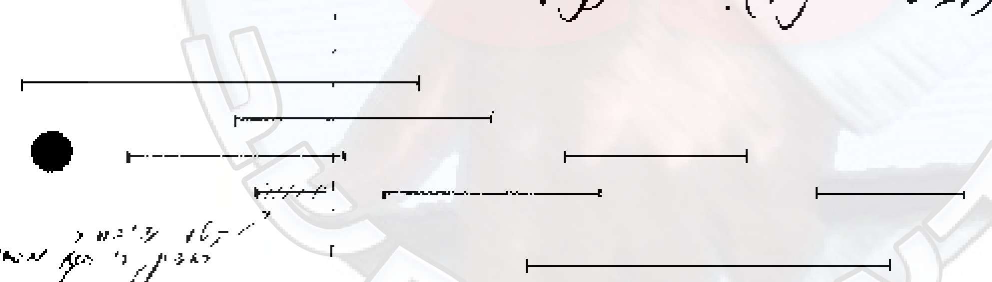
- עבור $i=2,\dots,2n$ חשב O_i לפי (*)

- פלט: O_{2n}

סיבוכיות: $O(\underline{n\log n} + n) = O(n \log n)$

מיון של הקצוות — מעבר יחיד.

פתרון (לבעיה הב'). חמדני.



הקטע שיבחר [illegible]

רעיון: נקח קטע שמסתיים הכי מוקדם מכל הקטעים.

טענה: הפתרון הטוב ביותר מכיל ללא הגבלת כלליות את הקטע שמסתיים ראשון.

הוכחה: יהי $J = \{J_{i_1}, J_{i_2}, \dots, J_{i_k}\}$ פתרון כלשהו.

נבנה פתרון שמכיל את הקטע המסתיים ראשון, J^*, עם אותו מספר קטעים כמו J.

28.2.05
אלגוריתמים
הרצאה 3

3

$$J^* = I^* \cup \{ J\in \mathcal{J} ,\ J\cap I^* = \phi \}$$

קטע שמסתיים ראשון

כל הקטעים ב-J שזרים ל-I^*!

- מדוע J^* פיתרון חוקי:

כי כל הקטעים ב-J^* נכללו ב-I^* וזרים בינם לבין עצמם באלה היחס ב-J.

- מדוע J^* מכיל לפחות k קטעים:

כי אם היו 2 קטעים ב-J שנחתכים עם I^* אז הם היו נחתכים בינהם בניגוד להגדרה של J.

מכפלת מטריצות

נתונות n מטריצות, רוצים לחשב את הצורך הטובה ביותר לבצע את המכפלה: $(((A_1 \times A_2)\times \ldots)\ldots)\times A_n$

$((A_1 \times A_2)\times(A_3 \times A_4)\ldots)\ldots \times A_n$

<u>לדוגמא:</u> רוצים להכפיל את המטריצות הבאות:

$A_{5\times 4}\times((A_{4\times 1}\times A_{1\times 5})\times A_{5\times 2})$

כאשר להכפיל 2 מטריצות $A_{d_1\times d_2}\times B_{d_2\times d_3}$

לוקח $O(d_1\cdot d_2\cdot d_3)$

- אם נכפיל בסדר הרגיל אז נקבל

$5\cdot 4\cdot 1+5\cdot 1\cdot 5+5\cdot 5\cdot 2=95$

- אם נכפיל לפי הסוגריים נקבל:

$4\cdot 1\cdot 5+4\cdot 5\cdot 2+5\cdot 4\cdot 2=100$

$A_{4\times 1}\times A_{1\times 5}$ $\quad A_{4\times 5}\cdot A_{5\times 2}$ $\quad A_{5\times 4}\times A_{4\times 2}$

- יהי O הסדר הטוב ביותר למכפלת מטריצות.

הפעולה האחרונה היא הכפלה של 2 מטריצות.

$$A \cdot B$$

$$(A_1 \times A_2 \times \dots A_i) \times (A_{i+1} \times \dots \times A_n)$$

כאשר

$A = A_1 \dots A_i$

נעשית בצורה הטובה ביותר

ו- $B = A_{i+1} \dots A_n$

נעשית בצורה הטובה ביותר.

- נניח ש $i = 1, \dots, n$, A_i מטריצות $d_i \times d_{i+1}$

- נסמן ב עלות של הכפלה מטריצות $O_{ij} = A_i \times A_{i+1} \times \dots \times A_j$

נחשב את O_{ij} לכל $i \le j$ (עבור $j = i$ עלות היא 0).

7.03.05
אלגוריתמים
הרצאה 5

נניח לנו א"ב הבא. $\Sigma=\{a,b,c\}$!

(1) רוצים למצוא כמה מילים ישנן מעל Σ, באורך n.
שלא מכילות את תת-מילים aa, bbc, bbb

נגדיר:

$x_n(aa)=$ מספר המילים (החוקיות) המסתיימות ב-aa, באורך n

$x_n(ab)=$ — " — " — " — ab — " —

$x_n(ac)=$ — " — " — " — ac — " —

$x_n(ba)=$ — " — " — ba

$x_n(bb)=$ — " — " — bb

$x_n(bc)=$ — " — " — bc

$x_n(ca)=$ — " — " — ca

$x_n(cb)=$ — " — " — cb

$x_n(cc)=$ — " — " — " — cc —

סה"כ התשובה הסופית היא

$$\sum_{\sigma_1,\sigma_2\in\Sigma} x_n(\sigma_1,\sigma_2)$$

כעת: נחשב את הערכים המוגדרים:

$$* \begin{cases} x_p(aa)=0 & \text{מתאפס כי אסור ל-0} \\ x_p(ab)=x_{p-1}(ca)+x_{p-1}(ba)+(x_{p-1}(aa)) \\ x_p(ac)=(x_{p-1}(aa))+x_{p-1}(ba)+x_{p-1}(ca) \\ \vdots \end{cases}$$

יש 9 משוואות סה"כ

$$x_p(bb) = x_{p-1}(ab) + x_{p-1}(cb)$$

<u>אלגוריתם:</u>

$x_1(\sigma_1,\sigma_2) = 3$

$x_2(\sigma_1,\sigma_2) = 2$, עבור $\sigma_1\sigma_2 \neq aa$

- עבור 3 עד p, מחשב $x_i(\sigma_1\sigma_2)$ לכל $\sigma_1\sigma_2 \in \Sigma$ לפי (*)

- סיבוכיות: $O(p)$

  אורך הקלט: $\log p$

  סוף אלגוריתם:

- נסתכל על הבעיה בצורה אחרת:

$$x_p = \begin{pmatrix} x_p(aa) \\ x_p(ab) \\ \vdots \\ x_p(cc) \end{pmatrix}, \quad x_p = \underbrace{\begin{pmatrix} 0 & 0 & \cdots & 0 \\ 1 & 1 & 1 \cdots & 0 \\ & & \vdots & \\ & & & \end{pmatrix}}_{\text{מטריצה קבועה } A} \cdot \underbrace{\begin{pmatrix} \\ \\ \\ \end{pmatrix}}_{\text{וקטור } x_{p-1}}$$

$$x_p = A \cdot x_{p-1} = A \cdot A \cdot x_{p-2} = A^{p-2} \cdot x_2$$

ניתן לחשב A^{p-2} ב $O(\lg p)$ בעזרת שיטת הכפלת מטריצות.

[illegible] נכון [illegible] כי ניתן [illegible] –
הבעיה [illegible] p [illegible]
אורך הקלט הוא באורך 2.

7.03.07
אלגוריתמים
הרצאה 5

(2)

בעיית התרמיל (Knap-Sach)

קלט: b, x_n, ..., x_1 מספרים טבעיים.

כאשר:

x_n, ..., x_1 - משקל של פריטים שונים

b - גודל תיק (או משקל, אורך ...).

הבעיה לפתור: האם יש תת-קבוצה $I \subseteq \{1, \dots, n\}$

שעבורה $\sum_{i \in I} x_i = b$

ניצור מערך מסוג bool (בוליאני) באורך $b+1$.

B_i: [0 | 1 | ... | t | ... | b]

המערך יכיל במקום ה-t' 1 אם ניתן להשיג את הסכום t.

בעיקרון יהיה לנו מערך כזה לכל i, כשהוא מתייחס רק לאיברים הראשונים עד i

טענה: עבור $i \geq 2$ $B_i(t)=1$ אמ"מ $B_{i-1}(t-x_i)=1$

אלגוריתם:

- איתחול $B(0)=1$
- עבור $i=1$ עד n עבור $l=b$ עד 0

$B(l)=1$ if $B(l-x_i)=1$

סיבוכיות: $O(n \cdot b)$.

אורך הקלט $\sim n \cdot \log b$
ולכן אלגוריתם אקספוננציאלי לאורך הקלט.

24.2.05
אלגוריתמים
תרגול 1

אורן בן צבי   שעות קבלה   [illegible]

① ק-בעיה: נתונים n מספרים. יש לפתח אלגוריתם שימצא את המקסימום.

פתרון: ניתן לסרוק את המערך פעם אחת ו-n-1 השוואות.

הוכחה: טענת האינדוקציה:
אחרי שסרקנו k איברים ועשינו k-1 השוואות,
קיבלנו מקסימום מהאיברים שסרקנו.

טענה: כל אלגוריתם A לבעיה ק (במודל ההשוואות) דורש
n-1 השוואות (לפחות)

הוכחה:
בכל השוואה לכל היותר איבר אחד יוצא מהמשחק
(לא מועמד להיות מקסימום).
בהתחלה יש n מועמדים.

[illegible]

דוגמה:
נתונה לנו נוסחת רקורסיה הבאה:

$$T(n)=T(\lceil n/2\rceil)+T(\lfloor \tfrac{n}{2}\rfloor)+\Theta(n)$$
$$T(1)=\Theta(1)$$

[illegible]

ניחוש: $T(n)=O(n\log n)$

נוכיח זאת.

הנחת האינדוקציה: $T(n)\le c\,n\lg n$

אז

$$T(n)=2T(n/2)+\Theta(n)\le 2c\,\tfrac{n}{2}\lg \tfrac{n}{2}+d\cdot n$$

כאשר d - קבוע נתון ו- c - קבוע שצריך למצוא בהוכחה.

$$T(n) \le c \cdot n \lg \tfrac{n}{2} + d \cdot n$$

$$\Downarrow$$

$$T(n) \le c \cdot n (\lg n - 1) + dn = c \cdot n\lg - c \cdot n + d \cdot n$$

$$\Downarrow$$

$$T(n) \le c n \lg n + (d - c) \cdot n$$

ועכשיו אם ניקח $d \le c$ אז נקבל

$$T(n) \le c n \log n + e$$

כאשר $e \le 0$, ולכן נקבל כי

$$T(n) \le c n \log n$$

מ.ש.ל.

- בוא נבדוק מה אם $T(n)$ ניקח ניחוש אחר $T(n) = O(n)$ ונראה האם נצליח להוכיח זאת.

הנחה: $T(n) \le cn$

אזי

$$T(n) \le 2c \cdot \frac{n}{2} + \Theta(n)$$

$$\Downarrow$$

$$T(n) \le c \cdot n + d \cdot n$$

כאשר d קבוע נתון ו-c קבוע שמחפשים.

- אין לנו נתון לגבי ערך של c כך ש נקבל בסופו של דבר כי $T(n) \le c \cdot n$, כי d נתון ולא יכול להיות אפס מספר ולכן נקבל כי $(c+d)n > cn$ ולכן זה לא אומר כי $T(n) \le c \cdot n$ ולכן ההנחה שלנו היא לא נכונה.

דוגמא: ניחוש לגבי נוסחא הבאה

$$T(n) = 2T\left(\frac{n}{2}\right) + 1$$

ניחוש: $T(n) = O(n)$

הוכחה: $T(n) \le c \cdot n$ ננסה זאת

$$T(n) \le 2 \cdot c \cdot \frac{n}{2} + 1 \Rightarrow T(n) \le c \cdot n + 1.$$

קיבלנו. אמנם נכון תנאי מקורב לכן ההנחה שלנו לא נכונה. ננסה הנחה אחרת.

הנחה 2:

נניח $T(n) \le cn - b$

$$T(n) \le 2\left[c\frac{n}{2} - b\right] + 1 \Rightarrow T(n) \le cn - 2b + 1$$

עם $c \ge 1$ נקבל

$$T(n) \le c \cdot n - b$$

דוגמא:

נתונה נוסחה הבאה:

$$T(n) = 2T(\sqrt{n}) + \log n$$

נבצע החלפת משתנים:

$$n = 2^m \Rightarrow m = \log n$$

ואז נקבל

$$T(2^m) = 2T(2^{m/2}) + m$$

נבצע החלפת פונקציה:

$$T(2^m) = S(m)$$

ואז נקבל

$$S(m) = 2S(m/2) + m$$

ואז ראינו כבר כי $S(m) = O(m \log m)$

כעת נבצע שינוי לאחור, ב"כ נקבל

$$\underline{T(n) = O(\log n \cdot \log\log n)}$$

דוגמא:

נתונה נוסחה הבאה:

$$T(n) = 3T(n/4) + n$$

$$T(1) = \Theta(1)$$

$$T(n) = 3\left[3T(n/16) + \frac{n}{4}\right] + n = 3\left[3\left[3T\left(\frac{n}{64}\right) + \frac{n}{16}\right] + \frac{n}{4}\right] + n = \dots$$

$$\dots = 3^{\log_4 n} + n\underbrace{\left(1 + \frac{3}{4} + \frac{9}{16} + \dots\right)}_{\text{סדרה הנדסית יורדת ולכן סכום הוא סופי.}} = n^{\log_4 3 < 1} + O(n) = O(n) + O(n) = O(n)$$