מבנ"ת הרצאה 12

מיונים המשך

ראינו quickSort אקראי עם קלט דטרמיניסטי תוחלת $O(n\log n)$

ניתן להראות ש quickSort דטרמיניסטי עם קלט אקראי הזמן גם כן $O(n\log n)$

חסם תחתון על ה Average case

~~ניתן~~ להראות גם שבתוחלת הממוצע של מספרים

< כל אלגוריתם הוא $\Omega(n\log n)$

הוכחה בשלילה.

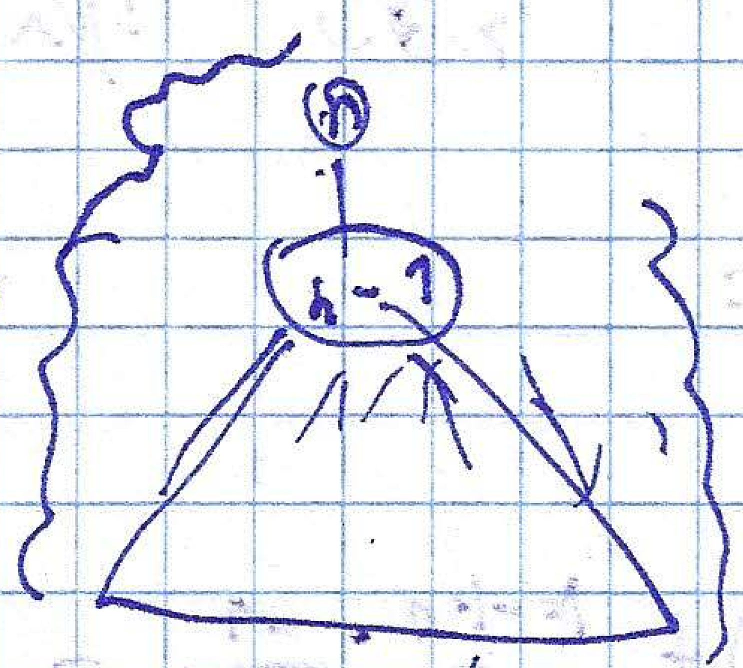
יש עץ שבו זה לא מתקיים.

יהי T העץ הקטן ביותר כך שבו לא מתקיים

אז ל T יש k עלים והמרחק הממוצע קטן מ $k\log k$.

לשורש T יש שני בנים אחרת נקצר אותו.

אחרת לא היה קטן מינימלי!



סכום ממוצע קטן מ $k\log k$

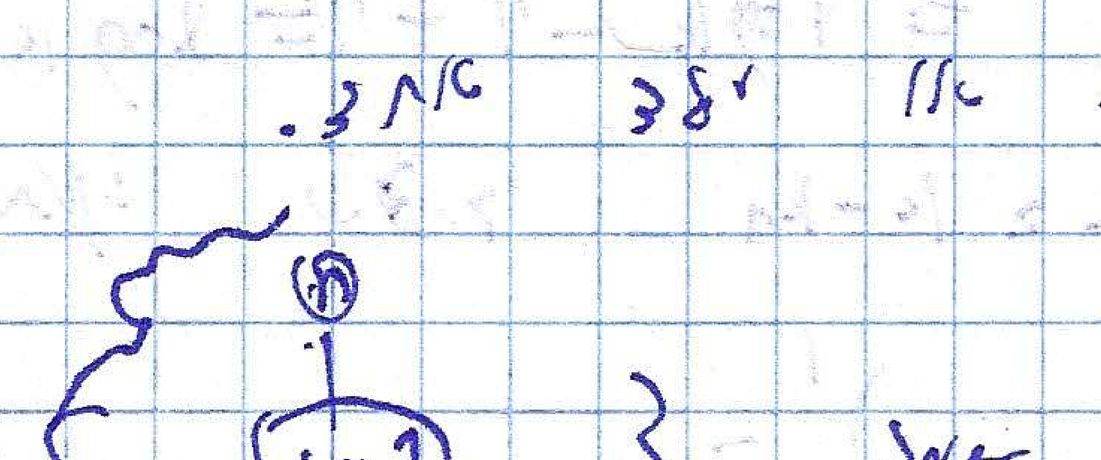
סכום ממוצע קטן מ $k\log k - n$

k עלים

סתירה למינימליות של T.

לכן לשורש של T יש שני ילדים.

מהמינימליות של T, הילדים שלו מקיימים את הטענה!



$k_1 < k$

$k - k_1 = k_2$

כלומר הסכום של n_1

גדול שווה ל $k_1\log k_1$

ורק גם ל k_2

נרשה ממוצע משוקלל

$$\frac{k_1}{k_1+k_2}\log k_1 + \frac{k_2}{k_2+k_1}\log k_2 + 1$$

חסם תחתון על עומק ממוצע של k_1 ; חסם תחתון על עומק ממוצע של k_2 עלים

לכן הזמן המשלים של T הוא

$$\frac{k_1}{k_1+k_2}\log k_1 + \frac{k_2}{k_1+k_2}\log k_2 + 1$$

נרצה להראות שהמינימום של זה מקבל את $\log k$

~~אם~~ k קבוע - • לא צריך $k_2=k-k_1$ ולפעמים,

• לא מספיק "לגזור" ולמצוא מינימום
$\underbrace{}_{\text{נקודת מינימום מקומית}}$

סלמיקים
↓
שפה שלכם
לא צריך
להשתמש.

• מספיק שהגרף: סימטרי ; או ∩ או ∪.

אם זה ∩ זה הנקודה של T רק בין 1.

אם זה ∪, נקודה $\frac{k}{2} = k_2 = k_1$

נראה לה

פונקציה מינימום ≥ $\frac{k/2}{k}\log k/2 + \frac{k/2}{k}\log k/2 + 1 = \log k/2 + 1 =$

$$= \log k - 1 + 1 = \log k$$

כמו כן נשאל: נגזור $k_2 = k - k_1$ ונציב

$$\left(\frac{k_1}{k}\log k_1 + \frac{k-k_1}{k}\log(k-k_1) + 1\right)' =$$

$$= \frac{k_1}{k}\cdot\frac{1}{k_1} + \frac{1}{k}\log k_1 + \frac{-1}{k}\log(k-k_1) + \frac{k-k_1}{k}\cdot\frac{-1}{k-k_1} =$$

$$= \frac{1}{k}\left(\log k_1 - \log(k-k_1)\right) =$$

$$= \frac{1}{k}\left(\log\frac{k_1}{k-k_1}\right) \stackrel{?}{=} 0$$

$$\log\frac{k_1}{k-k_1} = 0$$ כאשר

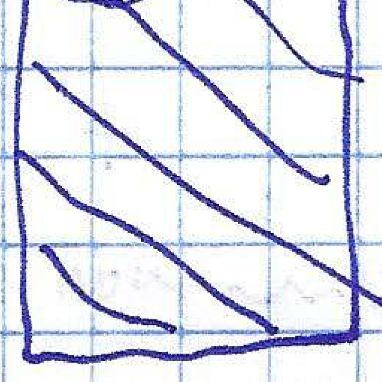
$$\log k_1 = \log(k-k_1)$$

$$k_1 = k - k_1$$

$$\boxed{k_1 = \frac{k}{2}}$$

לכן נשאר לבדוק, כמו מקודם, רק בשתי הקצוות

נכון



BIN / RADIX Sorting

האם ניתן למיין ב $\bar{O}(n\log n)$ אם צריך

משהו מיוחד על המספרים?

דוגמה 1:

אם אני יודע שהמערך $A[1...n]$ מכיל את המפתחות

$1, ..., n-1, n$ אז: מיון במערך B ב $O(n)$

$B[A[i].key] = A[i]$

ומה משמעותי כאשר אלה פוינטרים לרשומה ואני לא יכול להחזיק מערך של ~~[illegible]~~ המפתחות במקום?

אז זה קורה!

יש לנו פתרון חלקי יותר את הנתונים, דהשוואה

יש שתי אפשרויות, ~~[illegible]~~ לדוגמה, יש n אפשרויות

דוגמה 2. count sort

במערך $A[1,...,n]$ המפתחות $1....k$ $k \leq n$

כל איבר מופיע אולי יותר מפעם אחת.

מיון המערך: נבנה מערך בגודל k, המערך הזה

במקום ה i יכיל את מס' ההופעות שלו

נעבור על המערך הנתון, "נספור" כמה הופיע

ולבסוף, נבנה מערך ונשים בו את ~~[illegible]~~ המפתחות

לפי מס' ההופעות

דוגמה 3

כמו בדוגמה 1, אלא שיש מיון במקום.

כאשר אני מסתכל על איבר לא במקומו, מחליף אותו עם מי

שנמצא במקום המתאים לו, ~~[illegible]~~ ~~[illegible]~~ כאשר אני
מוסיף אותו

שנמצא במקומו, אני עובר למקום הבא במערך,

כל החלפה שמה איבר במקומו, אז לכל היותר n החלפות,

בנוסף לכך במעבר, ~~[illegible]~~ ~~[illegible]~~ אני עובר n התקדמויות

לכן היותר $2n$ פעולות $O(n)$.

שאלה (החלפה, ולא):
כאשר יש איברים
במקומם,
כי המפתחות
יכולים להיות
רשומה גדולה
ואני רק מחליף
פוינטרים.

BIN Sorting היינו מיון המתאים לתאים (~~BINS~~ (Bins))

ולפעול פרקטי למשל

פעולה יעילה פשוטה, יחידה

• הוספת מפתח ל Bin,

• חיבור שני Bins.

שי: פתרון – מימוש רשימה מקושרת

• הוספה לראש – $O(1)$

• חיבור Bins Concat $O(1)$.

סיבוכיות: m מספר הערכים האפשריים

n – מספר המפתחות

סיבוכיות הכנסה $O(n)$

$O(n+m) \Longleftarrow$

סיבוכיות ה concat $O(m)$

טוב אם מס' המפתחות לא גדול ממספר הסלים

$n > m$ $\quad O(m+n) \rightarrow O(n)$

$\Longrightarrow$ אם מס' הסלים גדול מאוד $m >> n$

$O(n) << O(n+m)$

אם $m = n^2$ אז הפתרון $O(n \log n) < O(n^2) = O(n+m)$

מיון רגיל.

Bin Sort בשלבים, (גם Radix Sort)

דוגמא: מיון של המספרים i^2 עבור $i = 0,...,10$

כאשר מיון של $0, 1, ..., 81, 100$ -

פתרון: בנה n סלים

מיין לפי הספרה הכי משמעותית (BINSort, ואחר כך הראשית)

מיין לפי הספרה הכי משמעותית, BINSort

בסוף הרשימה ממוינת

למה זה עובד?

נניח $i = 10a+b$, $j = 10c+d$

נניח $i<j$

אם $c>a$, אז כל מיון בספרה השנייה i יופיע ~~לפני~~ לפני מתקדם

אחרי זה j ... אז i יופיע לפני j כרצוי

אם $a=c$, ו $b<d$ אז במיון הראשון

i יופיע לפני j, ומאחר והמיון יציב ו ... יש סדר

כשדרש $a=c$, והמיון יציב הוא לא יזוז,

לכן i יופיע לפני j.

מתי זה עוזר?

כאשר המספרים מתפלגים כמו $1, \ldots, n^k$

המספרים באורך k.

מתי זה רע?

כאשר אין k קטן.

בעבור $k=100$ n

BIN:SORT n^k

מיון מנייה $n\log n$ - ו $n\log n < n^k$

בשורה אחת: המיון הזה יעיל רק כאשר $O(k)$

ולכן יש תחולה למינים המוגדרים.

Radix Sort

נתונים k שדות $f_1 f_2 \dots f_k$

רוצים למיין לפי סדר לקסיקוגרפי.

כלומר $(a_1, \dots, a_k) < (b_1, \dots, b_k)$ מתקיים

1) $a_1 < b_1$

2) $a_1 = b_1$ וגם $a_2 < b_2$

3) $a_1 = b_1$ $a_2 = b_2$ וגם $a_3 < b_3$

⋮

k) $a_k < b_k$ וגם $\forall i \le k-1 \ a_i = b_i$

איך נמיין?

<u>פתרון</u> נמיין BINSORT לפי ~~השדה~~ f_k,

ואז לפי f_{k-1} ... ואז לפי f_1

וסיימנו. (זה נקרא Radix)

מיונים בזמן לינאריים ע"י השוואה (השוואה של זוג):

(כמעט ממוין - למשל)

יהי נתון כמעט ממוין.

נסמן ב- I את מספר ה- "Inversions" (היפוכים)

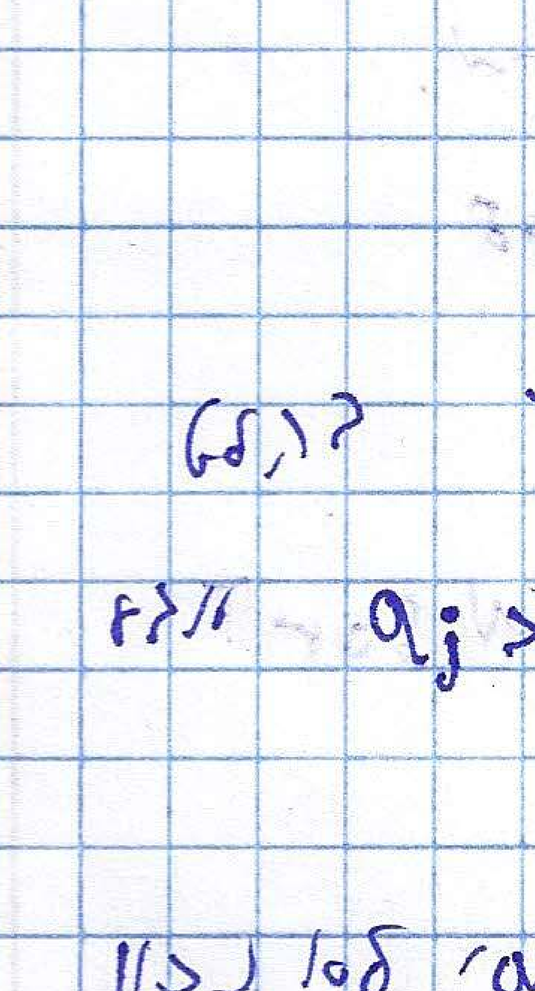
מספר הזוגות a_i, a_j כך ש- $a_j > a_i$ אבל

a_j מופיע לפני a_i

אינטואיטיבית - מספר הזוגות שמופרים בסדר מה <u>שלא נכון</u>

ככל ש- I קטן יותר המערך ממוין יותר.

מיון insertion sort בעצם:



מכניסים את במקום הנכון.

כמה פעמים נשנה את הכנסת ה- a_k?

מס' ה- היפוכים שלו עם כולם שלפניו

נסמן את זה

I_k.

בסך הכל:

$$\sum_{j=1}^{n} I_j + n = I + n$$

הכנסות כל איבר

מעבר על המערך

כאשר $I = \Omega(n^2)$ אז נקבל $\Omega(n^2)$

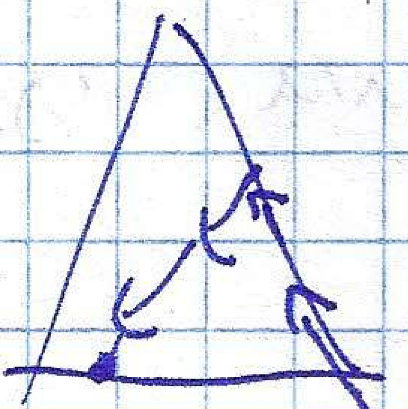
Finger red-black trees

עצים עם המיקום דרך

נתחזק את המקום בו עבדנו – הפוינטר

כמו מצביע של המקסימום, רק מבפנים.

שומרים ~~...~~ מצביע ~~...~~ רק ~~...~~

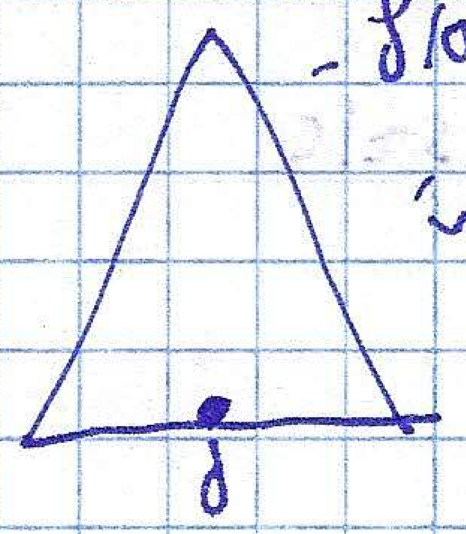


נראה ש Amortized העדכון

של התיקונים היא $O(1)$

~~נשמר~~ יהיו מעטים כלל

להכניס בעזרת שמירה של המיקום,



נניח שהמצב לשמור היה d שמוסף -

אז נעלה ~~...~~ עד ~~...~~ ויורד

כי עד של ממש $\log d$.

ה search יעלה $\log d$,

ה insert יעלה Amortized $O(1)$

נראה ש insertion sort ב FRB Tree

כל האיברים יעלו לכל היותר של הדרך.

לכל של d

$$d = O(I_k)$$

סך הכל הזמן הכולל

$$T = O(N) + \sum_{k=1}^{N} \log(I_k)$$

אותו סדר גודל כמו Insertion Sort, ~~...~~ שמתבצע על

ב"כמות מינימלית" של כולם, זה W.C. $n\log n$

$$\sum_{k=1}^{N} \log(I_k) \le \sum_{k=1}^{N} \log k = n\log n$$

נניח כי כמות ~~...~~ מינימלית: $\sum I_j = I$

ממוצע סוגי ממוצעים:

$$\sum_{i=1}^{n} \frac{a_i}{n} \ge \sqrt[n]{\prod_{i=1}^{n} a_i}$$

ניקח log - פונקציה מונוטונית:

$$\log\left(\sum_{i=1}^{n} \frac{a_i}{n}\right) \ge \log\sqrt[n]{\prod_{i=1}^{n} a_i} = \frac{\log \prod_{i=1}^{n} a_i}{n} = \frac{\sum_{i=1}^{n} \log a_i}{n}$$

לכן

$$O(n) + \sum_{k=1}^{n} \log I_k = O\left(n + n\sum_{k=1}^{n} \frac{\log I_k}{n}\right) \le O\left(n + n\log\frac{\sum I_k}{n}\right) = O\left(n + n\log\frac{I}{n}\right)$$

ניתוח Insertion Sort בזמן ממוצע,

מה זמן הריצה הממוצע? הנחה: התפלגות הזוגות $n+I$

רעיון: כל זוג מספרים, הסיכוי שהם לפי הסדר ושהם הפוכים

הוא $\frac{1}{2}$ - לכן התוחלת היא $\frac{1}{2}$, ספירת זוגות פיתוח

מכאן תוחלת מספר ההיפוכים - סכום התוחלות

$$\binom{n}{2}\cdot\frac{1}{2} \approx \frac{n^2}{4} = \Theta(n^2)$$

Selection - Order Statistic

Find the k^{th} element

- מיון המערך, ולקיחת האיבר במקום ה-k. $O(n\log n)$
- סריקה של המערך ומציאת המינימום, מחיקתו, ושוב ושוב. יעשה זאת k פעמים $O(kn)$
- בניית ערימה, $O(n)$, ועשיית k פעמים del min סך הכל $O(n+k\log n)$
- משתמשים - בניית ערימה, כל פעם שעושים del min מכניסים (insert) לערימה קטנה יותר את הבנים של מי שהוצאנו, ולוקחים את ~~ה~~ המינימום מהערימה הקטנה.
  סך הכל $O(n+k\log k)$

- Randomized selection:

בוחרים pivot, עושים partition כמו ב-quicksort ולוקחים רק צד אחד בלבד! את הצד שבו נמצא המיקום, תלוי אם k קטן או גדול מ-p, נמצא ~~...~~ מיקום ה pivot של המערך ה-k או p.

במקרה הגרוע $O(n^2)$

ניתוח
עבור מקרה ממוצע

נשתמש על זה ככה!

$p \quad q \quad r$

נסמן

$k = A[p,q]$ אם״ם $X_k = 1$

$$T(n) \le O(n) + \sum_{k=1}^{n} X_k \, T(\max(k-1, n-k))$$

לכל מקרה ערך מבין הערכים, כאשר רק אחד מהם יהיה שווה לאחד
ושאר האפסים.

$$E[T(n)] \le O(n) + \frac{1}{n} \sum_{k=1}^{n} E[T(\max(k-1, n-k))]$$

$$E[X_k] = \frac{1}{n}$$

$$E(T(n)) \le O(n) + \frac{2}{n} \left[\sum_{k=\lfloor \frac{n}{2} \rfloor}^{n-1} E(T(k)) \right]$$

נניח באינדוקציה ש כל קטנים ממנו:
עבור קבוע c כלשהו $E[T(k)] \le ck$ ונוכיח $E(T(n)) \le cn$

$$E[T(n)] \le O(n) + \frac{2}{n} \cdot \sum_{k=\lfloor n/2 \rfloor}^{n-1} E[T(k)] \le$$

$$\le a \cdot n + \frac{2}{n} \sum_{k=\lfloor n/2 \rfloor}^{n-1} ck = an + \frac{2c}{n} \left(\sum_{k=1}^{n-1} k - \sum_{k=1}^{\lfloor n/2 \rfloor - 1} k \right) =$$

$$= a \cdot n + \frac{2c}{n} \left[\frac{n(n-1)}{2} - \frac{\lfloor n/2 \rfloor (\lfloor n/2 \rfloor - 1)}{2} \right] \le$$

מעגלים כלפי מטה

לבחור את a לפי המקסימום בין
$a+c$ כך שיתקיים דרישות
c.

$$\le an + \frac{2c}{n} \left[\frac{n(n-1)}{2} - \frac{(n/2 - 2)(n/2 - 1)}{2} \right]$$

$$\le an + \frac{c}{n} \left(\frac{3n^2}{4} - \frac{n}{2} - 2 \right) \le an + \frac{3}{4} cn =$$

$$= cn - \left(\frac{cn}{4} - an \right) \overset{?}{\le} cn = O(n) \quad !!!$$

$$\boxed{c \ge 4a}$$