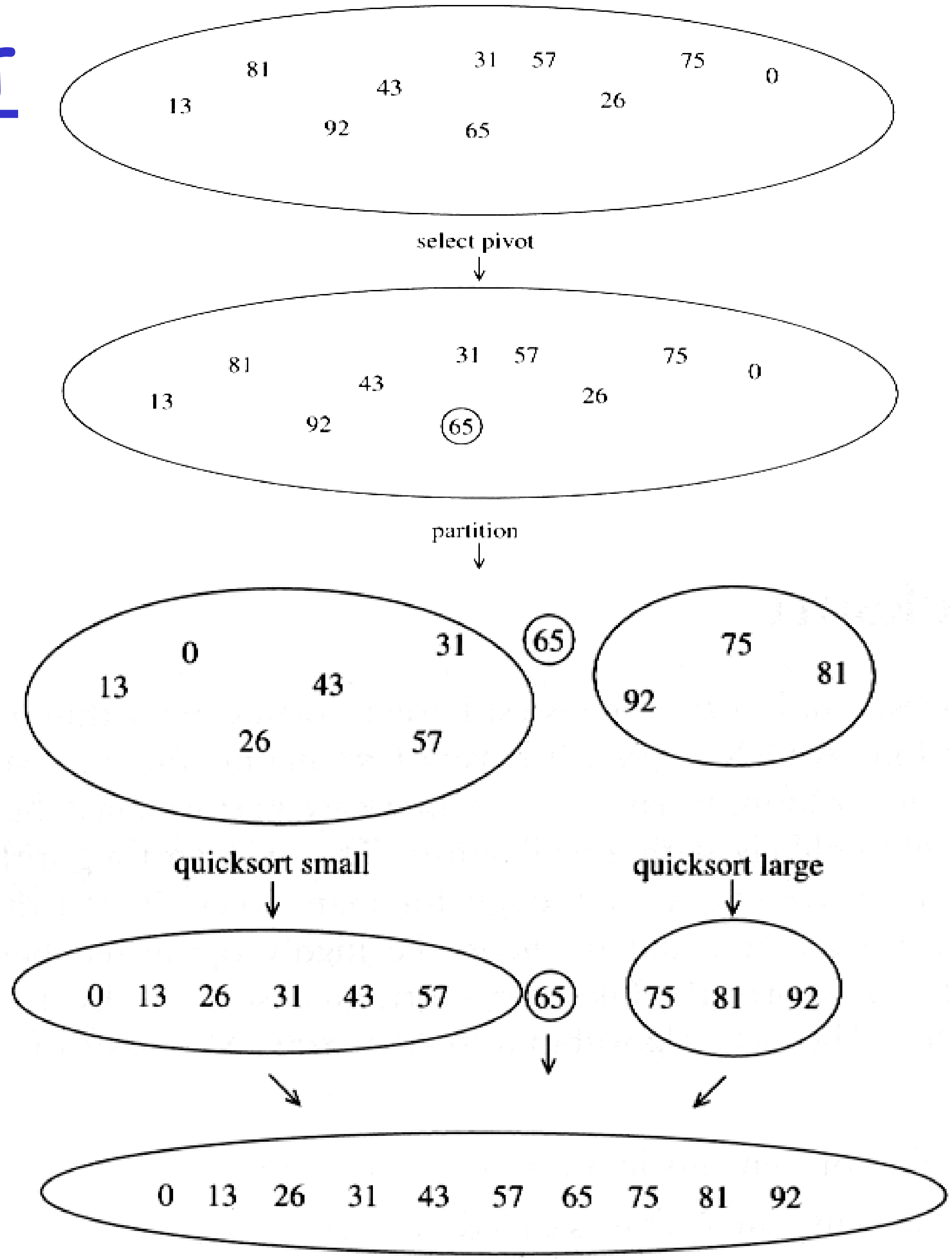
מבני נתונים ואלגוריתמים 2
תירגול: Quicksort and Order Statistics

Based on: Introduction to Algorithms
by Cormen, Leiserson, Rivest, and Stein

Quicksort Example



Quicksort - Pseudocode

Quicksort(A, p, r):

- If $p < r$, then
  - $q \leftarrow$ Partition(A, p, r)
  - Quicksort$(A, p, q - 1)$
  - Quicksort$(A, q + 1, r)$

Partition(A, p, r):

- $x \leftarrow A[r]$
- $i \leftarrow p - 1$
- For $j \leftarrow p$ to $r - 1$ do
  - If $A[j] \leq x$ then
    - $i \leftarrow i + 1$
    - $A[i] \leftrightarrow A[j]$
- $A[i + 1] \leftrightarrow A[r]$
- Return $i + 1$

קריאה התחלתית:

Quicksort$(A, 1, n)$

הערה: האיבר x שנבחר ע"י השגרה Partition נקרא פיבוט (pivot).

Worst-case partitioning

- The worst-case behavior
  - The pivot is the smallest element, all the time (always produced two subproblem: with $n-1$ elements and with 0 elements)
  - Partition is always unbalanced
  - The partitioning costs $\Theta(n)$, $T(1) = \Theta(1)$
  - $T(n) = T(n-1) + T(0) + \Theta(n) = T(n-1) + \Theta(n)$

$$
\begin{aligned}
T(N) &= T(N-1) + cN \\
T(N-1) &= T(N-2) + c(N-1) \\
T(N-2) &= T(N-3) + c(N-2) \\
&\vdots \\
T(2) &= T(1) + c(2) \\
T(N) &= T(1) + c\sum_{i=2}^{N} i = O(N^2)
\end{aligned}
$$

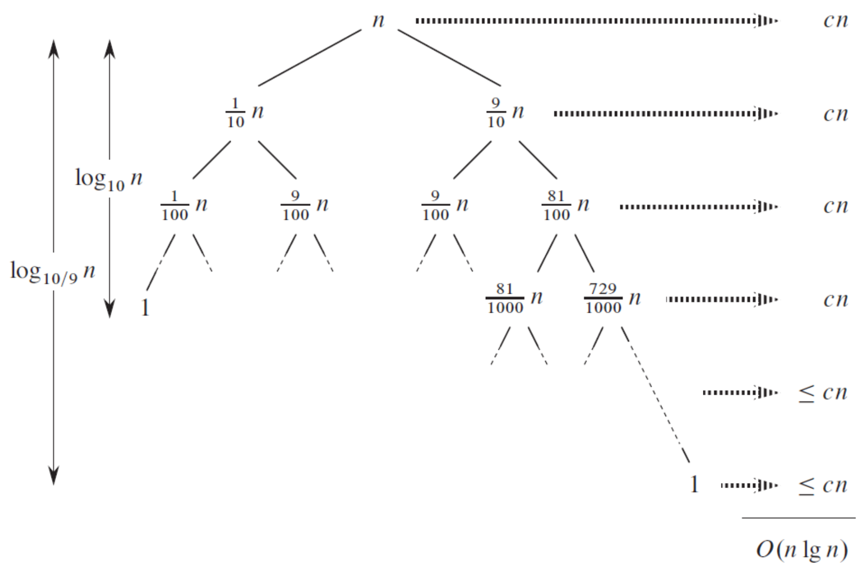
Best-case partitioning

- What will be the best case?
  - Pivot is always in the middle (median of the array)
  - Partition is perfectly balanced (each time produced two subarrays each of size no more than $\frac{n}{2}$)
  - $T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n)$

$$
\begin{aligned}
T(N) &= 2T(N/2) + cN \\
\frac{T(N)}{N} &= \frac{T(N/2)}{N/2} + c \\
\frac{T(N/2)}{N/2} &= \frac{T(N/4)}{N/4} + c \\
\frac{T(N/4)}{N/4} &= \frac{T(N/8)}{N/8} + c \\
&\vdots \\
\frac{T(2)}{2} &= \frac{T(1)}{1} + c \\
\frac{T(N)}{N} &= \frac{T(1)}{1} + c\log N \\
T(N) &= cN\log N + N = O(N\log N)
\end{aligned}
$$

Balanced partition - example

- A recursion tree for QUICKSORT in which PARTITION always produces a 9-to-1 split



- The longest simple path from the root to a leaf is n→ $(\frac{9}{10})$n → $(\frac{9}{10})^2$n → ..1
- Height of the tree h=$\log_{10/9} n$
- $(\frac{9}{10})^h$n =1 → n=$(\frac{10}{9})^h$ → $\log_{10/9} n$ = h · $\log_{10/9} \frac{10}{9}$ = h
- Total time is smaller that $cn \log_{10/9} n =? O(nlogn)$
- n $\log_{10/9} n$ < cn log n → n $\frac{\log n}{\log 10/9}$ < cn log n → $\frac{1}{\log 10/9}$ < c → c > 6.6
- Total time is $O(nlogn)$

תזכורת מההרצאה:
מציאת סטטיסטי הסדר ה- k

- נרצה לטפל המקרה הכללי של מציאת סטטיסטי הסדר ה- k.
- האלגוריתם ישתמש בטכניקת הפרד ומשול.
- האלגוריתם ישתמש בפיבוט כדי לחלק את המערך לשני חלקים, אבל בניגוד ל- Quicksort הוא ימשיך לטפל רק בחלק אחד.
- לכן נקבל זמן סיבוכיות $\Theta(n)$.

אלגוריתם למציאת סטט' הסדר ה- k

<u>אלגוריתם $\text{Select}(A,k)$:</u>

1. אם A קטן מספיק אז פתור ע"י מיון.
2. חלק את הקלט לחמישיות.
3. מצא חציון בכל חמישייה. נסמן את קבוצת החציונים ע"י B.
4. מצא את החציון של B שיסומן ע"י x, כלומר בצע $\text{Select}\left(B,\left\lceil\frac{|B|}{2}\right\rceil\right)$
5. השתמש ב- x בפיבוט: נחלק את A לשני חלקים. נסמן ב- A' את קבוצת המספרים שגדולים מ- x.
6. אם $|A \setminus A'| \le k$ אז נקרא ל- $\text{Select}(A \setminus A',k)$
אם $|A \setminus A'| > k$ אז נקרא ל- $\text{Select}(A',k-|A \setminus A'|)$

<u>הערה:</u> חסרים פרטים בבסיס הרקורסיה.

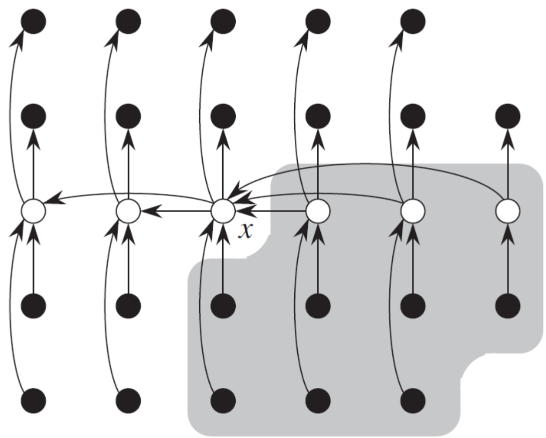
נכונות

<u>טענה:</u> האלגוריתם מוצא את סטטיסטי הסדר ה- k במערך A.

<u>הוכחה:</u>

- נסמן ב- A_i וב- k_i את הקלט לקריאה הרקורסיבית ה- i. נוכיח באינדוקציה שסטטיסטי הסדר ה- k במערך A הוא סטטיסטי הסדר ה- k_i במערך A_i.
- בסיס: ברור שזה נכון עבור A_1,k_1.
- צעד:
  - נניח שהטענה נכונה עבור A_{i-1},k_{i-1}.
  - אם $|A_{i-1} \setminus A'_{i-1}| \leq k_{i-1}$, אז סטטיסטי הסדר ה- k_{i-1} במערך A_{i-1} קטן שווה מהפיבוט x, ולכן סטטיסטי הסדר ה- k_{i-1} במערך A_{i-1} שווה לסטטיסטי הסדר ה- $k_i = k_{i-1}$ במערך $A_i = A_{i-1} \setminus A'_{i-1}$.
  - אם $|A_{i-1} \setminus A'_{i-1}| > k_{i-1}$, אז סטטיסטי הסדר ה- k_{i-1} במערך A_{i-1} גדול מהפיבוט x, ולכן סטטיסטי הסדר ה- k_{i-1} במערך A_{i-1} שווה לסטטיסטי הסדר ה- $k_i = k_{i-1} - |A_{i-1} \setminus A'_{i-1}|$ במערך $A_i = A'_{i-1}$.
- מסקנה: נכונות האלגוריתם נובעת מהנכונות של אלגוריתם המיון.

ניתוח זמן ריצה



<u>טענה:</u> $|A'| \geq \frac{3}{10}n - 6$

<u>הוכחה:</u>

- ישנם $\left\lceil \frac{n}{5} \right\rceil$ חמישיות.
- בכל הקבוצות יש חמישה אברים, מלבד אולי קבוצה אחת שמכילה פחות מ- 5 איברים (אם n לא מתחלק ב- 5).
- לפחות חצי מהחציונים גדולים או שווים לחציון החציונים x. לכן לפחות חצי מהקבוצות תורמות 3 איברים שגדולים מ- x, למעט הקבוצה של x, ואולי קבוצה שמכילה פחות מ- 5 איברים.
- סה"כ: $|A'| \geq 3\left(\frac{1}{2}\left\lceil \frac{n}{5} \right\rceil - 2\right) \geq \frac{3}{10}n - 6$

<u>טענה:</u> $|A \setminus A'| \geq \frac{3}{10}n - 6$

<u>הוכחה:</u> דומה.

<u>מסקנה:</u> $\frac{3}{10}n - 6 \leq |A'|, |A \setminus A'| \leq \frac{7}{10}n + 6$

ניתוח זמן ריצה (המשך)

- נסמן את זמן הריצה של האלגוריתם עם קלט באורך n ע"י $T(n)$.
- צעדים 2, 3, ו- 5 לוקחים זמן $O(n)$, נניח $a \cdot n$, עבור קבוע a כלשהו.
- צעד 4 לוקח זמן $T\left(\left\lceil\frac{n}{5}\right\rceil\right)$.
- צעד 6 לוקח זמן $T\left(\frac{7}{10}n+6\right)$.
- סה"כ קיבלנו:

$$T(n) \leq \begin{cases} T\left(\left\lceil\frac{n}{5}\right\rceil\right) + T\left(\frac{7}{10}n+6\right) + an, & n > 140 \\ O(1), & n \leq 140 \end{cases}$$

- נראה שזמן הריצה הוא לינארי בשיטת ההצבה
- נניח שזמן הריצה $T(n) \leq c \cdot n$, עבור קבוע c כלשהו.
- ברור שזה נכון עבור $n \leq 140$ כי אפשר להשתמש באלגוריתם מיון.
  - נסמן את הקבוע הרלוונטי ב- b.

ניתוח זמן ריצה (המשך)

- אחרת,

$$T(n) \leq T\left(\frac{n}{5}+1\right)+T\left(\frac{7}{10}n+6\right)+an \leq c\left(\frac{n}{5}+1\right)+c\left(\frac{7}{10}n+6\right)+an$$
$$\leq c\frac{9}{10}n+7c+an = cn+\left[7c+an-\frac{c}{10}n\right]$$

- נרצה שיתקיים

$$7c+an-\frac{c}{10}n \leq 0$$
$$c\left(\frac{n}{10}-7\right) \geq an$$
$$c \geq 10a\cdot\frac{n}{n-70}$$

- אם $n > 140$ נקבל

$$c \geq 20a$$

- לכן אם נשתמש ב- $c = \max\left\{20a, \frac{b}{140}\right\}$, נקבל $T(n) = O(n)$.

Exercise 9.3 from CLRS

- In the algorithm SELECT, the input elements are divided into groups of 5.

  - Will the algorithm work in linear time if they are divided into groups of 7?

  - Argue that SELECT does not run in linear time if groups of 3 are used.

Answer part 1

- Groups of 7:
- Number of elements greater than x:

$$4 * \left(\left\lceil \frac{1}{2} \left\lceil \frac{n}{7} \right\rceil \right\rceil - 2 \right) \geq \frac{4n}{14} - 8$$

- Number of elements less or equal x:

$$n - \frac{4n}{14} + 8 = \frac{10n}{14} + 8$$

- We can show by substitution that the following expression is linear:

$$T(n) \leq T(n/7) + T(5n/7 + 8) + O(n)$$

- Suppose $T(n) < cn$ for $n < k$, then, for $m \geq k$:
- $T(m) \leq T(m/7) + T(5m/7+8) + O(m) \leq c(m/7) + c(5m/7+8) + am \leq (6/7)cm + am$
- $T(m) \leq cm + (-cm/7 + am)$
- $(-cm/7 + am) < 0$ for $c > 7a$
- $T(n) = O(n)$
- The algorithm will work for every odd number greater than 5.

Answer part 2

- Groups of 3:
  - Number of elements greater than x:

$$2 * \left(\left\lceil \frac{1}{2} \left\lceil \frac{n}{3} \right\rceil \right\rceil - 2\right) \geq \frac{2n}{6} - 4$$

  - Number of elements less or equal x:

$$n - \frac{2n}{6} + 4 = \frac{4n}{6} + 4$$

  - The recurrence we get is :

$$T(n) = T\left(\left\lceil \frac{n}{3} \right\rceil\right) + T(2n/3+4) + O(n) \leq T(n/3) + T(2n/3) + O(n)$$

- We will show that $O(nlogn)$ is an upper bound for the solution to this recurrence.

Answer part 2

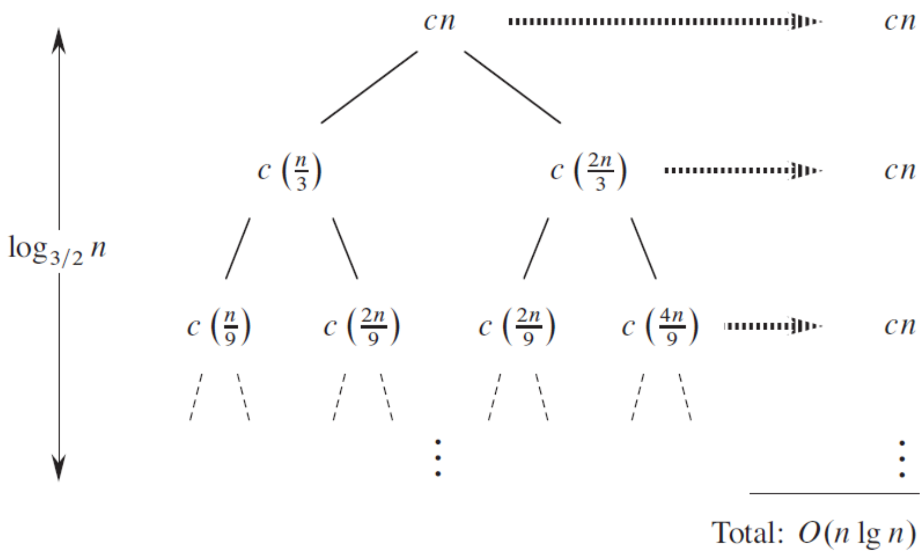
- We show that T(n) ≤ d·n·log n, where d > 0

$$
\begin{aligned}
T(n) &\le T(n/3) + T(2n/3) + cn \\
&\le d(n/3)\lg(n/3) + d(2n/3)\lg(2n/3) + cn \\
&= (d(n/3)\lg n - d(n/3)\lg 3) \\
&\qquad + (d(2n/3)\lg n - d(2n/3)\lg(3/2)) + cn \\
&= dn\lg n - d((n/3)\lg 3 + (2n/3)\lg(3/2)) + cn \\
&= dn\lg n - d((n/3)\lg 3 + (2n/3)\lg 3 - (2n/3)\lg 2) + cn \\
&= dn\lg n - dn(\lg 3 - 2/3) + cn \\
&\le dn\lg n \,,
\end{aligned}
$$

- It grows more quickly than linear: $T(n) = O(nlogn)$

Answer part 2 - recursion tree

- A recursion tree for T(n) = T(n/3) + T(2n/3) + O(n)



- The longest simple path from the root to a leaf is n→ $(\frac{2}{3})$n → $(\frac{2}{3})^2$n → ..1
- Height of the tree k=$\log_{3/2} n$
- Total time is smaller that cn $\log_{3/2} n = ?\, O(nlogn)$

- n $\log_{3/2} n$ < cn log n → n $\frac{\log n}{\log 3/2}$ < cn log n → $\frac{1}{\log 3/2}$ < c → c > 1.7

- Total time is $O(nlogn)$