

מציאת האיבר ה- i בגודלו (ללא מיון)

למציאת איבר מינימלי (מקסימלי) במערך ניתן למשל להשתמש באיטרציה אחת של BubbleSort הלוקחת $O(n)$ זמן.

שאלה: האם ניתן למצוא את האיבר השני, השלישי, או האיבר ה- i בזמן $O(n)$?

ניתן כמובן למיין, אבל כל אלגוריתם מיון למספרים כלליים דורש $\Omega(n \log n)$ זמן (כפי שנוכח בקרוב).

אינטואיטיבית למצוא את האיבר ה- i נראית בעיה קשה יותר מאשר מציאת איבר מינימלי (מקסימלי). נראה ראשית פתרון בזמן $O(n)$ בממוצע. בהמשך השיעור נראה כיצד לעשות זאת בזמן $O(n)$ במקרה הגרוע.

מיונים ב': בחירת האיבר ה- i חסמים תחתונים למיון RadixSort ו-BucketSort

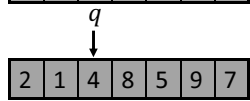
חומר קריאה לשיעור זה

Chapter 9 - Sorting in Linear Time
Chapter 10 - Medians and Order Statistics

דוגמא מלאה

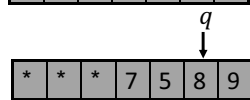
תוצאת ההגרלה

מצא את המספר החמישי בגודלו במערך הבא:



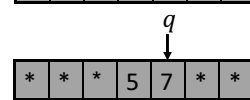
pivot=4

מצא את המספר השני בגודלו במערך הבא:



pivot=8

מצא את המספר השני בגודלו במערך הבא:



pivot=7

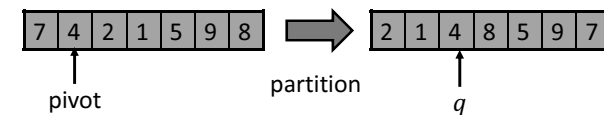
בקריאה השלישית מצאנו את האיבר המבוקש. נחזיר אותו ונסיים.

מציאת האיבר ה- i בגודלו (ללא מיון)

הפתרון משתמש ברעיון דומה ל-QuickSort. למציאת האיבר ה- i בגודלו נבצע רקורסיבית את הפעולות הבאות:

1. בחר באקראי איבר ציר pivot.
2. Partition: חלק את המערך לשני חלקים. האברים הקטנים מ-pivot יאוחסנו בחלק השמאלי של המערך, והגדולים או שווים ל-pivot בחלק הימני של המערך, כאשר ה-pivot יישמר במקום ה- q .
3. אם $q = i$, החזר את ה-pivot.
4. אם $q > i$ אז מצא רקורסיבית את האיבר ה- i בחלק השמאלי של המערך.
5. אחרת, מצא רקורסיבית את האיבר ה- $i - q$ בחלק הימני של המערך.

דוגמא:



ניתוח זמנים במקרה הממוצע

• מקרה ממוצע: איבר הציר הוא אקראי.

$$T(n) \leq \frac{1}{n} \left(\sum_{k=1}^n T(\max(k-1, n-k)) \right) + O(n)$$

$$T(n) \leq \frac{1}{n} \left(2 \sum_{k=\lfloor \frac{n}{2} \rfloor}^{n-1} T(k) \right) + O(n) \quad \text{ומכאן שמתקיים:}$$

הסכום המופיע במשוואה השנייה שווה לסכום במשוואה הראשונה עבור n זוגי. אי שוויון קורה עבור n אי-זוגי.

$n = 5$		$n = 6$	המחשה:
$k =$	1 2 3 4 5	$k =$	1 2 3 4 5 6
$k - 1 =$	0 1 2 3 4	$k - 1 =$	0 1 2 3 4 5
$n - k =$	4 3 2 1 0	$n - k =$	5 4 3 2 1 0
$Max(k - 1, n - k) =$	4 3 2 3 4	$Max(k - 1, n - k) =$	5 4 3 3 4 5

לכל k , $\lfloor \frac{n}{2} \rfloor \leq k \leq n - 1$ מופיע פעמיים. לכן $T(k)$ מופיע פעמיים. פרט ל- $T[\lfloor n/2 \rfloor]$, שמופיע רק פעם אחת.

ניתוח זמנים

Partition ניתן למימוש בזמן לינארי - $\Theta(n)$. זמן הריצה הכולל תלוי באיבר הציר. • מקרה אופטימלי: איבר הציר הוא חציון.

$$T(n) = \Theta(n) + T(n/2) \leq c \cdot (n + n/2 + n/4 + \dots + 1) \leq 2c \cdot n$$

$$T(n) = \Theta(n)$$

זהו מקרה אופטימלי מכיוון שיש לעבור על כל איברי המערך.

משוואת נסיגה זו מכילה מופע אחד של $T(n/2)$ בניגוד לניתוח האופטימלי של QuickSort ולכן פתרונה ליניארי ולא $\Theta(n \log n)$.

• מקרה גרוע: בכל שלב המערך קטן באחד בלבד.

$$T(n) = \Theta(n) + T(n-1) \leq c \cdot (n + (n-1) + \dots + 1) = \Theta(n^2)$$

מקרה פרטי: מציאת חציון

הגדרה:

בקבוצה של n איברים, החציון הוא האיבר ה- $\lfloor (n+1)/2 \rfloor$ בגודלו בקבוצה.

למציאת חציון נפעיל את האלגוריתם שפתחנו למציאת האיבר ה- $\lfloor (n+1)/2 \rfloor$ בגודלו.

זמן הריצה, כפי שראינו, הוא $O(n)$ בממוצע.

פתרון משוואת הנסיגה

$$T(n) \leq \frac{1}{n} \left(2 \sum_{k=\lfloor \frac{n}{2} \rfloor}^{n-1} T(k) \right) + d \cdot n \quad \text{נוסחת הנסיגה שקיבלנו:}$$

נראה שפתרון המשוואה מקיים $T(n) = O(n)$.

הוכחה: באינדוקציה על n . נוכיח כי קיים קבוע c כך ש- $T(n) \leq c \cdot n$ לכל n .

בסיס: נבחר c גדול דיו כך שמתקיים $T(1) \leq c$.

הנחת אינדוקציה + צעד: נניח כי מתקיים $T(k) \leq c \cdot k$ לכל $1 \leq k < n$ ונראה

כי מכאן מתקיים גם $T(n) \leq c \cdot n$. ואכן:

$$T(n) \leq \frac{2}{n} \left(c \sum_{k=\lfloor \frac{n}{2} \rfloor}^{n-1} k \right) + d \cdot n = \frac{2c}{n} \cdot \left(\sum_{k=1}^{n-1} k - \sum_{k=1}^{\lfloor \frac{n}{2} \rfloor - 1} k \right) + dn$$

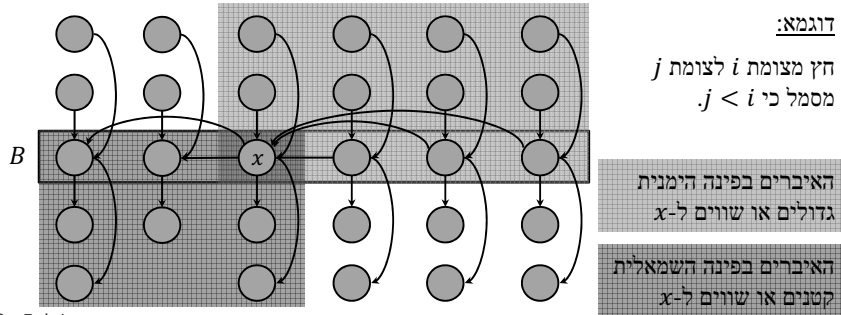
$$\leq \frac{2c}{n} \cdot \left(\frac{1}{2}(n-1)n - \frac{1}{2} \left(\frac{n}{2} - 1 \right) \frac{n}{2} \right) + dn = \frac{3cn}{4} + dn \leq cn$$

נכון לכל c לכל $4d \leq c$

מכאן ש- $T(n) \leq c \cdot n$ לכל n לכל בחירת קבוע $\max\{T(1), 4d\} \leq c$.

האלגוריתם Select(A, first, last, i)

1. חלק את מערך הקלט A לחמישיות.
2. מצא חציון של כל חמישייה. הכנס את החציונים למערך B (שגודלו בערך חמישית המערך A).
3. הפעל את Select על המערך B למציאת חציון של החציונים (נקרא לו x).
4. $s \leftarrow \text{Partition}(A, \text{first}, \text{last}, x)$ (חלוקת המערך A לפי x).
5. אם $s = i$ החזר את x.
6. אם $s - \text{first} \geq i$, $\text{Select}(A, \text{first}, s - 1, i)$
7. אחרת, $\text{Select}(A, s, \text{last}, i - (s - \text{first}))$



מציאת האיבר ה- i בגודלו (אלגוריתם דטרמיניסטי)

הזמן הדרוש לאלגוריתם שתיארנו תלוי בגודל המערך בכל קריאה רקורסיבית. אם נבטיח שבכל קריאה רקורסיבית גודל המערך קטן "בצורה משמעותית", אזי זמן הריצה במקרה הגרוע ביותר יהיה $O(n)$. הבעיה נוצרת כאשר החלוקה לשתי קבוצות אינה מאוזנת ומשאירה קבוצה אחת שבגודלה קרובה מדי לגודל הקבוצה המקורית.

נרצה שבכל שלב האלגוריתם יבצע חלוקה לשתי קבוצות הקטנות משמעותית מהקבוצה המקורית. למען הדיוק, נרצה שגודל כל קבוצה יהיה קטן מ- α פעמים גודל הקבוצה המקורית כאשר $0 < \alpha < 1$. שימו לב ש- α הוא קבוע שאינו תלוי בגודל הקבוצות. כיצד נמצא חלוקה כזו ומדוע זו החלוקה הדרושה?

ארבעת השלבים הראשונים של האלגוריתם הבא, שנקרא לו $\text{Select}()$, מוצאים חלוקה עם התכונות שהגדרנו. הניתוח מראה מדוע זו בחירה טובה.

ניתוח זמן הריצה

$$T(n) \leq T\left(\left\lceil \frac{n}{5} \right\rceil\right) + T\left(\frac{7}{10}n + 6\right) + O(n)$$

טענה: $T(n) = O(n)$

הוכחה: באינדוקציה על n . נוכיח כי קיים קבוע c כך ש- $T(n) \leq c \cdot n$ לכל n . **בסיס:** יהי c קבוע כך שעבור $n \leq 80$ מתקיים $T(n) \leq c \cdot n$ (קיים קבוע כזה). **ה.א.:** צעד: נניח כי לכל $k < n$ מתקיים $T(k) \leq c \cdot k$ ונראה כי מכאן נובע $T(n) \leq c \cdot n$. יהי d הקבוע הנחבא בסימון O בנוסחת הנסיגה.

$$\begin{aligned} T(n) &\leq c \cdot \left\lceil \frac{n}{5} \right\rceil + c \cdot \left(\frac{7n}{10} + 6\right) + d \cdot n && \text{נציב בנוסחת הנסיגה ונקבל:} \\ &\leq c \cdot \left(\frac{n}{5} + 1\right) + c \cdot \left(\frac{7n}{10} + 6\right) + d \cdot n \\ &\leq c \cdot \left(\frac{9n}{10}\right) + 7c + d \cdot n \\ &\leq c \cdot n \end{aligned}$$

כאשר עבור $d > 80$ מתקיים אי השוויון האחרון לכל $n \geq 80$:

$$\begin{aligned} c \cdot \left(\frac{9n}{10}\right) + 7c + d \cdot n &< \frac{9cn}{10} + 7c + \frac{c}{80}n = \frac{73}{80}cn + 7c \\ &= cn + \left(-7c \frac{n}{80} + 7c\right) \leq cn \end{aligned}$$

ניתוח גודל הקבוצות שנוצרו

טענה: מספר האיברים בקריאה הרקורסיבית ל- Select הוא לכל היותר $\frac{7n}{10} + 6$

הוכחה:

- כיון ש- x הוא חציון החציונים, וישנם $\lceil n/5 \rceil$ חציונים, מתקיים שלפחות $\lceil 1/2 \cdot \lceil n/5 \rceil \rceil$ חציונים הקטנים או שווים ל- x .
- בכל קבוצה של חציון הקטן או שווה ל- x ישנם שלושה איברים הקטנים/שווים מ- x (פרט אולי לקבוצה האחת בה אין חמישה איברים ולקבוצה בה x נמצא). לפיכך מספר האיברים הקטנים מ- x הוא לפחות:

$$\left(\left\lceil \frac{1}{2} \cdot \left\lceil \frac{n}{5} \right\rceil \right\rceil - 2\right) \cdot 3 \geq \frac{3}{10}n - 6$$
- בצורה דומה, מספר האיברים הגדולים/שווים מ- x הוא לפחות:

$$\left(\left\lceil \frac{1}{2} \cdot \left\lceil \frac{n}{5} \right\rceil \right\rceil - 1\right) \geq \left(\left\lceil \frac{1}{2} \cdot \left\lceil \frac{n}{5} \right\rceil \right\rceil - 1 - 1\right) \cdot 3 \geq \frac{3}{10}n - 6$$
- ומכאן שבכל קריאה רקורסיבית ניוותר עם לכל היותר $\frac{7n}{10} + 6$ איברים, כנטען.

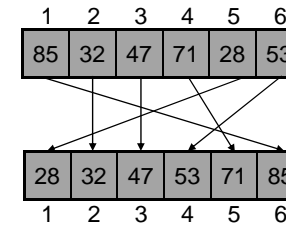
חסם תחתון למיון ע"י השוואות (המשך)

משפט: כל אלגוריתם מיון הפועל באמצעות השוואות בלבד דורש לפחות $\Omega(n \log n)$ השוואות.

הדגשה: האלגוריתם אינו מבצע, לדוגמא, פעולות אריתמטיות על סדרת הקלט.

חסם תחתון למיון ע"י השוואות

נניח ברצוננו לקבל מערך A בן n איברים שונים ולסדר את האיברים ממוינים במערך פלט. כל סדרה בת n מספרים שונים מגדירה פרמוטציה π על האינדקסים של מערך הקלט. לדוגמא:



$$\pi(1) = 6, \pi(2) = 2, \pi(3) = 3,$$

$$\pi(4) = 5, \pi(5) = 1, \pi(6) = 4$$

וכן פרמוטציה הופכית $\sigma = \pi^{-1}$:

$$\sigma(1) = 5, \sigma(2) = 2, \sigma(3) = 3,$$

$$\sigma(4) = 6, \sigma(5) = 4, \sigma(6) = 1$$

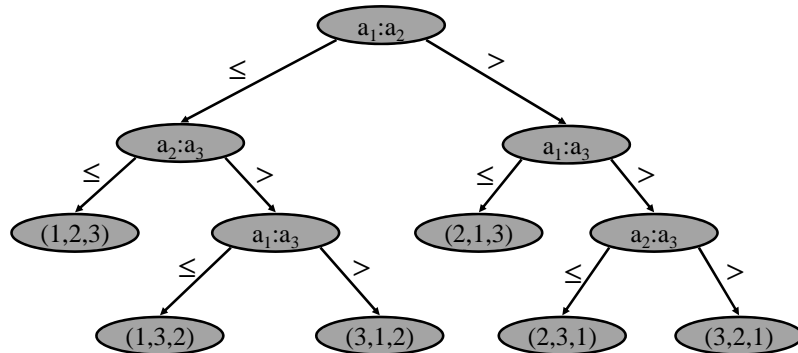
אלגוריתם מיון מקבל כקלט מערך A ומוציא כפלט פרמוטציה σ כך שיתקיים:

$$A[\sigma(1)] < A[\sigma(2)] < \dots < A[\sigma(n)]$$

כל פרמוטציה אפשרית כפלט כיון שהאיבר $A[i]$ יכול להגיע לכל מקום במערך הפלט לפי גודלו היחסי בסדרת הקלט. קיימות $n!$ פרמוטציות. שאלה: כמה שאלות כן/לא צריכות להישאל עד שנותרת פרמוטציה אחת אפשרית?

נימוק לחסם

אלגוריתם מיון המבוסס על השוואות ניתן לתיאור ע"י עץ החלטות בינרי.



בכל צומת פנימי מצוינת השוואה. בכל עלה מצוינת פרמוטציה הממיינת את הקלט. (ייצוג זה מתעלם מפעולות שאינן השוואות). מספר העלים הוא לפחות $n!$.

מספר ההשוואות המקסימלי הוא גובה העץ h ומתקיים:
 $h > \log_2(n!) = \Omega(n \log n)$

חסם תחתון למיון ע"י השוואות (המשך)

משפט: כל אלגוריתם מיון הפועל באמצעות השוואות בלבד דורש לפחות $\Omega(n \log n)$ השוואות.

הוכחה: יהי Alg אלגוריתם מיון ע"י השוואות. לכל קלט האלגוריתם מבצע סדרה של השוואות אשר בסופן נקבעת הפרמוטציה המתאימה למיון הקלט. נראה שקיימת סדרת קלט עבורה ידרשו $\Omega(n \log n)$ השוואות.

$$\pi^+ = \{\pi \mid \pi(i) < \pi(j)\}$$

לאחר השוואה אחת $A[i] : A[j]$ נחלק את

$$\pi^- = \{\pi \mid \pi(i) > \pi(j)\}$$

קבוצת הפרמוטציות לשתי קבוצות:

$$\pi^{++} = \{\pi \mid \pi(i) < \pi(j), \pi(l) < \pi(k)\}$$

לאחר השוואה נוספת $A[l] : A[k]$ נחלק

$$\pi^{+-} = \{\pi \mid \pi(i) > \pi(j), \pi(l) < \pi(k)\}$$

את קבוצת הפרמוטציות לארבע קבוצות:

$$\pi^{-+} = \{\pi \mid \pi(i) < \pi(j), \pi(l) > \pi(k)\}$$

$$\pi^{--} = \{\pi \mid \pi(i) > \pi(j), \pi(l) > \pi(k)\}$$

לאחר k השוואות יוצרו 2^k קבוצות.

נבחר קלט שקוונטיסנטני עם הקבוצה הגדולה מבין 2^k הקבוצות. קבוצה זו מכילה לפחות $n!/2^k$ פרמוטציות. כדי שאלגוריתם המיון יענה נכונה, קבוצה זו צריכה להכיל פרמוטציה בודדת (אחרת ייתכנו מספר תשובות אפשריות עבור k ההשוואות ובפרט קיימים קלטים עליהם האלגוריתם ייכשל).

מייון BucketSort

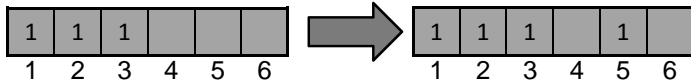
המשימה: למיין n מספרים **שונים** בתחום $1 \dots k$.
השיטה: נשתמש במערך בוליאני A באורך k .

```

1. אפס את  $A$ .
2. לכל  $x$  בקלט בצע:  $A[x] = 1$ 
3. עבור על המערך ואסוף
   for ( $i = 1; i < k; i++$ ) {
       if ( $A[i] == 1$ ) output  $i$ ;
   }

```

דוגמא: נניח $n = 4$ וטווח המספרים הוא $1 \dots k = 6$. עבור הקלט 5, 1, 3, 2:



זמן: איפוס המערך $\theta(k)$, מעבר על הקלט $\theta(n)$, איסוף $\theta(k)$. סה"כ: $\theta(n + k)$.

אם גודל הטווח מקיים $k = O(n)$ אזי נקבל אלגוריתם מיון ליניארי (בזמן $\theta(n)$).

חסם תחתון למיון ע"י השוואות (המשך)

מספר ההשוואות המקסימלי הוא גובה העץ h ומתקיים לכל $9 \leq n$

$$h > \log_2(n!) > \frac{1}{2}n \log_2 n = \Omega(n \log n)$$

הוכחה לחסם הנ"ל:

• נוסחת סטרלינג לעצרת:

$$n! = \sqrt{2\pi n} \left(\frac{n}{e}\right)^n (1 + \theta(1/n))$$

• בפרט $n! > \left(\frac{n}{e}\right)^n$

• נקבל לכל $n \geq 9$:

$$\log_2(n!) > n \log_2 n - n \log_2 e > \frac{1}{2}n \log_2 n$$

• מכיוון שמתקיים:

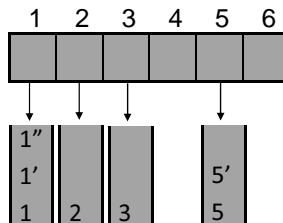
$$\frac{1}{2} \log_2 n = \log_2 \sqrt{n} \geq \log_2 \sqrt{9} > \log_2 e$$

הרחבת מייון BucketSort

המשימה: למיין n מספרים **שאינם בהכרח שונים** מהתחום $1 \dots k$.
השיטה: נשתמש במערך של k תורים. בזמן האיסוף נשרש את התורים.

דוגמא: נניח $n = 7$ וטווח המספרים הוא $1 \dots k = 6$.

עבור הקלט 5', 1'', 3, 2, 1', 1, 5:



הפלט שומר על סדר ההכנסה כאשר המפתחות שווים: 5', 5, 3, 2, 1'', 1', 1

הגדרה: שיטת מיון תקרא יציבה (Stable) כאשר מתקיים התנאי הבא:
אם בקלט $A[i] = A[j]$ וכן $i < j$, אזי יקדים את $A[j]$ בפלט.

מהדיון הנ"ל נובע כי BucketSort היא שיטת מיון יציבה.

היכן הקסם?

מדוע ההוכחה שנדרשים לפחות $\Omega(n \log n)$ השוואות אינה "תופסת"?
מהו בדיוק תנאי המשפט שמפור במיון BucketSort?

תשובה: בהוכחת המשפט כל השוואה נותנת תשובה בינרית: גדול או קטן. לעומת זאת, במיון BucketSort תוצאות "השוואה" אחת שוות ל- $\log_2 k$ השוואות בינריות.

כאשר גודל הטווח מקיים $k = O(n)$ אזי אמנם נקבל אלגוריתם מיון בזמן $\theta(n)$, אולם כל צעד מקביל ל- $\theta(\log_2 n)$ השוואות בינריות.

פרוצדורת RadixSort

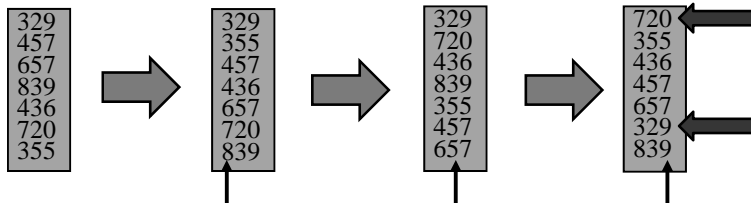
נסמן ב-1 את הספרה הפחות משמעותית (LSD) וב- d את הספרה המשמעותית ביותר (MSD).

Radix-Sort(A,d)
for $i \leftarrow 1$ to d
do use BucketSort to sort array A on digit i

תרגיל: הוכח נכונות באינדוקציה על d .

- היכן משתמשים בעבודה ששיטת המיון בתוך הלולאה יציבה?
- היכן נכשלת ההוכחה כאשר ממיינים קודם את הספרות המשמעותיות ואח"כ את הספרות הפחות משמעותיות?

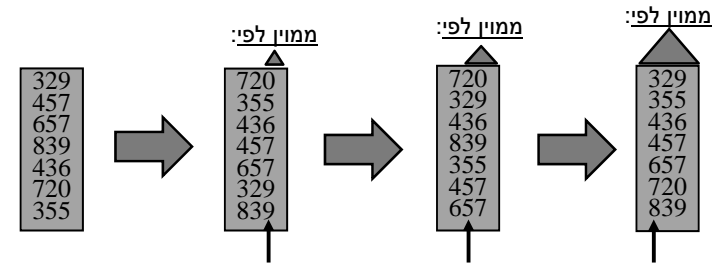
דוגמא לכישלון המיון כאשר המיון נעשה מה-MSD ל-LSD:



מיון RadixSort

אבחנה: שיטת BucketSort אינה יעילה כאשר n המספרים לקוחים מתחום "רחב" מדי " $1..k$ ", כלומר כאשר $k \ll n$. **לדוגמא:** מיון המספרים 1,1003,99999 ידרוש מערך גדול ובו 10^5 מקומות בעוד $n = 3$ (בדומה לבעיה שבגללה הצגנו את פונקציות הערבול).

פתרון: נניח שכל מפתח מורכב מ- d ספרות עשרוניות. נמייין כל ספרה בנפרד תוך שימוש בשיטת מיון יציבה (לדוגמא BucketSort). ספרות פחות משמעותיות (Least Significant Digits) ממוינות ראשונות (בניגוד אולי לאינטואיציה ראשונית).



סיכום השיעור

בשיעור זה המשכנו לדון בנושאים הקשורים למיון. בפרט, ראינו:

1. **Select** - אלגוריתם למציאת האיבר ה- i בגודלו במערך לא ממויין בזמן $O(n)$ במקרה הגרוע.
2. **חסם תחתון**, המראה כי כל אלגוריתם מיון מבוסס השוואות דורש $\Omega(n \log n)$ השוואות, ובפרט $\Omega(n \log n)$ זמן.
3. **BucketSort** - אלגוריתם מיון הממייין n מספרים שלמים בתחום $1 \dots k$ בזמן $O(n+k)$ במקרה הגרוע.
4. **RadixSort** - אלגוריתם מיון הממייין n מספרים שלמים בני d ספרות בבסיס r בזמן $O(d \cdot (n+r))$ במקרה הגרוע.

ניתוח זמן הריצה

לכל ספרה עשרונית נבצע BucketSort בזמן $\Theta(n+10)$. עבור d ספרות הזמן הכולל הוא $\Theta(d(n+10))$.

למשל: למספרים בבסיס 10 עם מספר ספרות d קבוע (שאינו תלוי במספר המספרים n), המיון מתבצע בזמן $\Theta(n)$.

באופן כללי: עבור מספרים המיוצגים בבסיס r , לכל ספרה נבצע BucketSort בזמן $\Theta(n+r)$. עבור d ספרות, הזמן הכולל הוא $\Theta(d(n+r))$.

כאשר האיבר המקסימלי הוא k , מספר הספרות בבסיס r הוא $\log_r k$ ואז הזמן הנדרש הוא $\Theta(\log_r k \cdot (n+r))$.

למשל: עבור $k = \Theta(n^c)$ עבור c קבוע ובסיס r קבוע, הזמן הנדרש הוא $\Omega(n \log n)$, כפי שדרוש במיונים עם השוואות בלבד.