מציאת האיבר ה-i בגודלו (ללא מיון)

למציאת איבר מינימלי (מקסימלי) במערך נדרשת איטרציה אחת של BubbleSort הלוקחת $O(n)$ זמן.

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 שנוכיח בקרוב).

אינטואיטיבית למצוא את האיבר ה-i נראית בעיה קשה יותר. נראה ראשית פתרון בזמן $O(n)$ בממוצע.

מציאת האיבר ה-i בגודלו (ללא מיון)

למציאת איבר מינימלי (מקסימלי) במערך נדרשת איטרציה אחת של BubbleSort הלוקחת O(n) זמן.

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

ניתן כמובן למיין, אבל כל אלגוריתם מיון למספרים כלליים דורש Ω(n log n) זמן (כפי שנוכיח בקרוב).

אינטואיטיבית למצוא את האיבר ה-i נראית בעיה קשה יותר. נראה ראשית פתרון בזמן O(n) בממוצע.

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

- בחר באקראי איבר ציר pivot .
- חלק את המערך לשני חלקים. האברים הקטנים מ-pivot יאוחסנו בחלק השמאלי של המערך, והגדולים או שווים ל-pivot בחלק הימני של המערך.

(נניח ש- q הוא מספר האיברים בחלק השמאלי של המערך)

- אם i ≤ q אז מצא רקורסיבית את האיבר ה-i בחלק השמאלי של המערך.
- אחרת, מצא רקורסיבית את האיבר ה- i-q בחלק הימני של המערך.

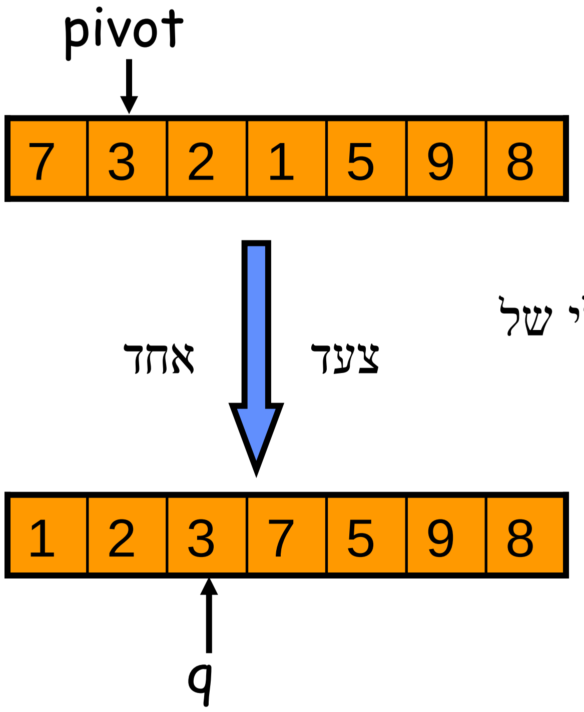
מציאת האיבר ה-i בגודלו (ללא מיון)

למציאת איבר מינימלי (מקסימלי) במערך נדרשת איטרציה אחת של BubbleSort הלוקחת O(n) זמן.

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

ניתן כמובן למיין, אבל כל אלגוריתם מיון למספרים כלליים דורש Ω(n log n) זמן (כפי שנוכיח בקרוב).

אינטואיטיבית למצוא את האיבר ה-i נראית בעיה קשה יותר. נראה ראשית פתרון בזמן O(n) בממוצע.



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

- בחר באקראי איבר ציר pivot .
- חלק את המערך לשני חלקים. האברים הקטנים מ-pivot יאוחסנו בחלק השמאלי של המערך, והגדולים או שווים ל-pivot בחלק הימני של המערך.

(נניח ש- q הוא מספר האיברים בחלק השמאלי של המערך)

- אם i ≤ q אז מצא רקורסיבית את האיבר ה-i בחלק השמאלי של המערך.
- אחרת, מצא רקורסיבית את האיבר ה- i-q בחלק הימני של המערך.

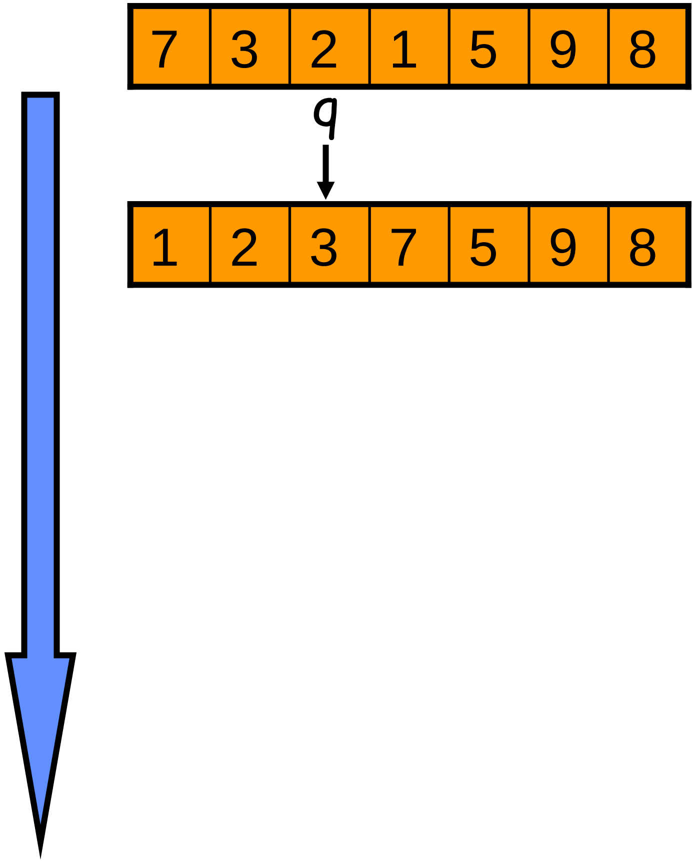
©Geiger & Itai, 2001

דוגמא מלאה

מצא את המספר הרביעי בגודלו במערך הבא:

תוצאת ההגרלה

pivot= 3



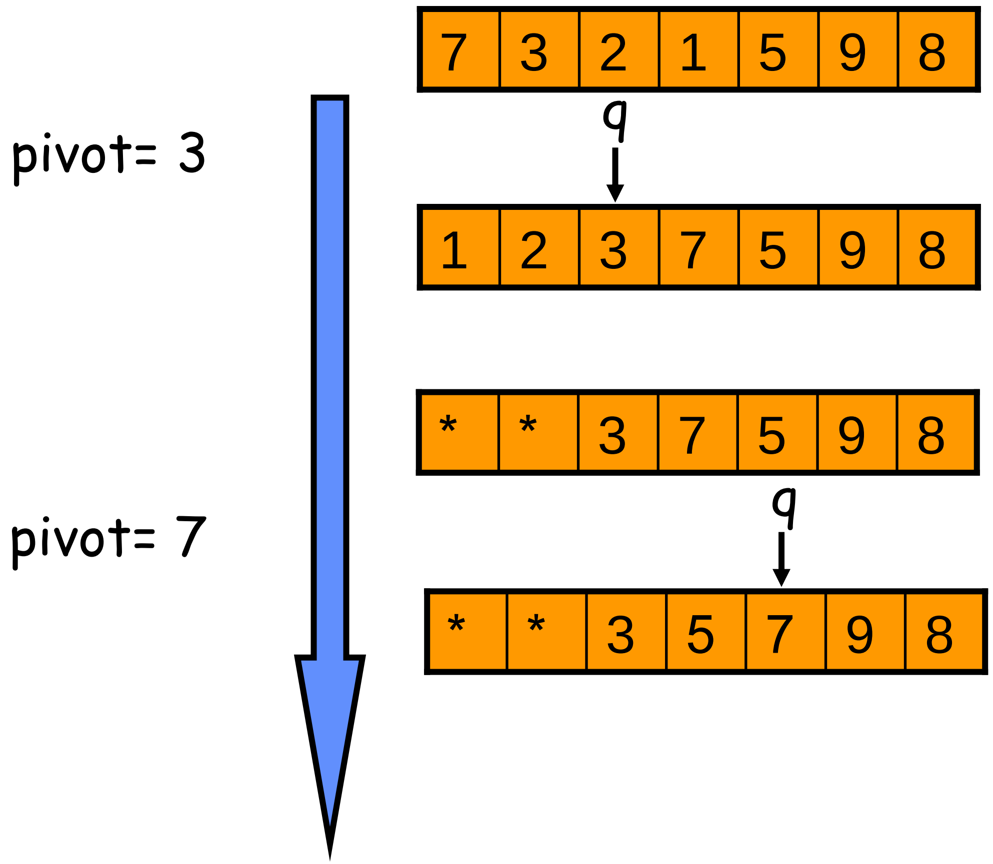
©cs,Technion

דוגמא מלאה

תוצאת ההגרלה

מצא את המספר הרביעי בגודלו במערך הבא:

מצא את המספר השני בגודלו במערך הבא:



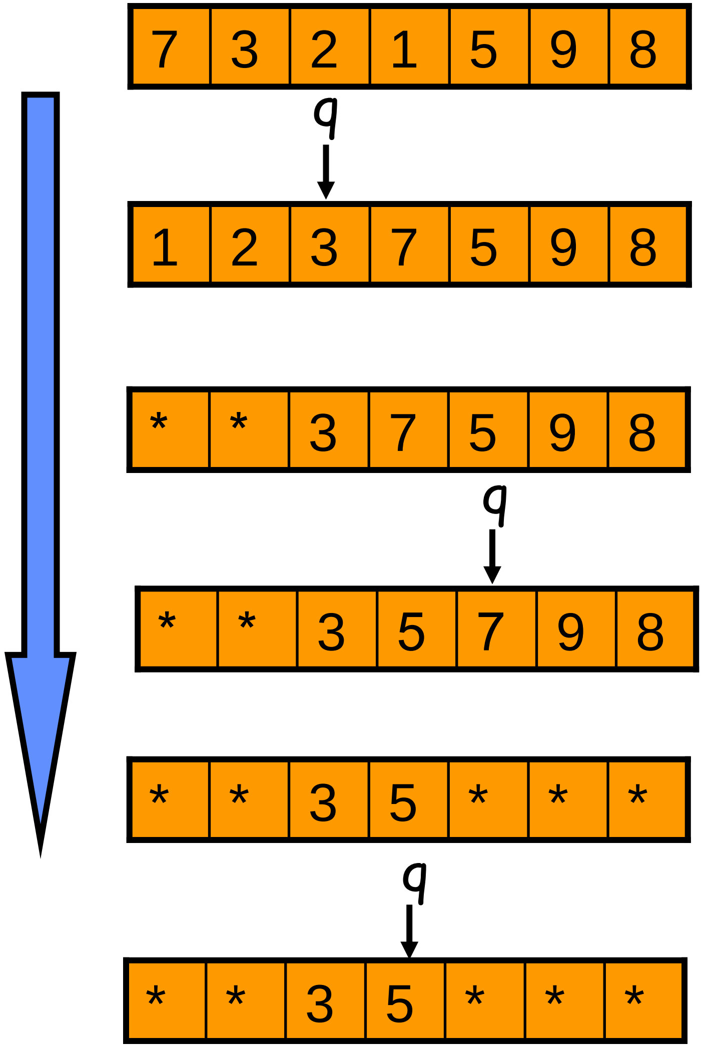
©cs,Technion

דוגמא מלאה

<u>תוצאת ההגרלה</u>

מצא את המספר <u>הרביעי</u> בגודלו במערך הבא:

pivot= 3



מצא את המספר <u>השני</u> בגודלו במערך הבא:

pivot= 7

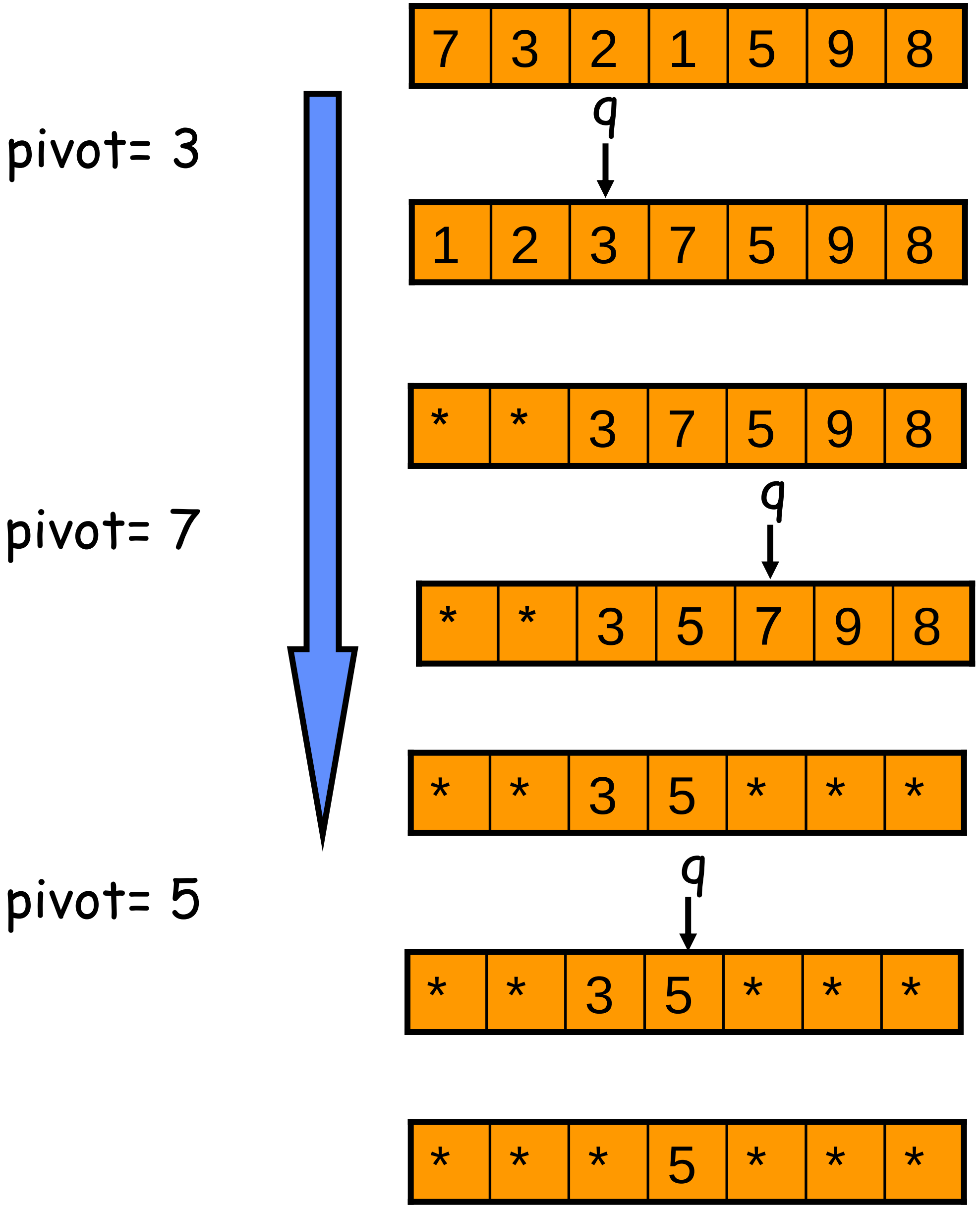
מצא את המספר <u>השני</u> בגודלו במערך הבא:

pivot= 5

©cs,Technion

דוגמא מלאה

<u>תוצאת ההגרלה</u>



מצא את המספר <u>הרביעי </u>בגודלו במערך הבא:

מצא את המספר <u>השני</u> בגודלו במערך הבא:

מצא את המספר <u>השני</u> בגודלו במערך הבא:

מצא את המספר <u>הראשון</u> בגודלו במערך הבא:

©cs,Technion

ניתוח זמנים

זמן הריצה תלוי באיבר הציר.

מקרה אופטימלי. איבר הציר הוא חציון.

זהו מקרה אופטימלי בגלל שהזמן נקבע ע"י גודל

המערך בכל קריאה (ולא ע"י דרגת האיבר אותו מחפשים).

משוואת נסיגה זו מכילה מופע אחד של T(n/2) בניגוד לניתוח האופטימלי של QuickSort

ולכן פתרונה ליניארי ולא Θ(n log n).

$$T(n) = \Theta(n) + T(n/2)$$
$$= c(n + n/2 + n/4 + ... + 1) = 2cn$$

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

מקרה גרוע. בכל שלב המערך קטן באחד בלבד.

$$T(n) = \Theta(n) + T(n-1) = c(n + (n-1) + ...1) = \Theta(n^2)$$

©cs,Technion

ניתוח זמנים במקרה הממוצע

מקרה ממוצע: איבר הציר הוא אקראי.

כאשר איבר הציר הוא הראשון

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

ניתוח זמנים במקרה הממוצע

מקרה ממוצע: איבר הציר הוא אקראי.

כאשר איבר הציר הוא הראשון

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

$$T(n) \le \frac{1}{n}\left(T(n-1) + 2\sum_{k=\lceil n/2 \rceil}^{n-1} T(k) \right) + O(n)$$

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

©cs,Technion

ניתוח זמנים במקרה הממוצע

מקרה ממוצע: איבר הציר הוא אקראי.

כאשר איבר הציר הוא הראשון

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

$$T(n) \le \frac{1}{n}\left(T(n-1) + 2\sum_{k=\lceil n/2 \rceil}^{n-1} T(k) \right) + O(n)$$

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

המחשה: n = 5

k =	1	2	3	4
n - k =	4	3	2	1
Max(n,n-k) =	4	3	3	4

כל T[k] מופיע פעמיים.

©cs,Technion

ניתוח זמנים במקרה הממוצע

מקרה ממוצע: איבר הציר הוא אקראי.

כאשר איבר הציר הוא הראשון

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

$$T(n) \le \frac{1}{n}\left(T(n-1) + 2 \sum_{k=\lceil n/2 \rceil}^{n-1} T(k) \right) + O(n)$$

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

המחשה: $n = 6$

k =	1	2	3	4	5
n - k =	5	4	3	2	1
Max(n,n-k) =	5	4	3	4	5

כל T[k] מופיע פעמיים מלבד $T[\lceil n/2 \rceil]$.

©cs,Technion

ניתוח זמנים במקרה הממוצע

<u>מקרה ממוצע</u>: איבר הציר הוא אקראי.

כאשר איבר הציר הוא הראשון

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

$$T(n) \le \frac{1}{n}\left(T(n-1) + 2\sum_{k=\lceil n/2 \rceil}^{n-1} T(k) \right) + O(n)$$

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

כיוון ש- $T(n-1)/n = O(n)$ מתקבלת המשוואה:

©cs,Technion

ניתוח זמנים במקרה הממוצע

<u>מקרה ממוצע</u>: איבר הציר הוא אקראי.

כאשר איבר הציר הוא הראשון

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

$$T(n) \le \frac{1}{n}\left(T(n-1) + 2\sum_{k=\lceil n/2 \rceil}^{n-1} T(k) \right) + O(n)$$

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

כיוון ש- $T(n-1)/n = O(n)$ מתקבלת המשוואה:

$$T(n) \le \frac{2}{n} \sum_{k=\lceil n/2 \rceil}^{n-1} T(k) + O(n)$$

פתרון משוואת הנסיגה

נראה שפתרון המשוואה

$$T(n) \le \frac{2}{n} \sum_{k=\lceil n/2 \rceil}^{n-1} T(k) + dn$$

מקיים $T(n) = O(n)$.

נניח באינדוקציה שמתקיים $T(n) \le c \cdot n$ עבור קבוע c המקיים $T(1) \le c$. לפיכך:

פתרון משוואת הנסיגה

נראה שפתרון המשוואה

$$T(n) \le \frac{2}{n} \sum_{k=\lceil n/2 \rceil}^{n-1} T(k) + dn$$

מקיים $T(n) = O(n)$.

נניח באינדוקציה שמתקיים $T(n) \le c \cdot n$ עבור קבוע c המקיים $T(1) \le c$. לפיכך:

$$T(n) \le \frac{2}{n} \sum_{k=\lceil n/2 \rceil}^{n-1} c \cdot k + dn$$

פתרון משוואת הנסיגה

נראה שפתרון המשוואה

$$T(n) \le \frac{2}{n}\sum_{k=\lceil n/2\rceil}^{n-1} T(k) + dn$$

מקיים $T(n) = O(n)$.

נניח באינדוקציה שמתקיים $T(n) \le c \cdot n$ עבור קבוע c המקיים $T(1) \le c$. לפיכך:

$$T(n) \le \frac{2}{n}\sum_{k=\lceil n/2\rceil}^{n-1} c \cdot k + dn \qquad \le \frac{2c}{n}\left(\sum_{k=1}^{n-1} k - \sum_{k=1}^{\lceil n/2\rceil -1} k\right) + dn$$

©cs,Technion

פתרון משוואת הנסיגה

נראה שפתרון המשוואה

$$T(n) \le \frac{2}{n}\sum_{k=\lceil n/2 \rceil}^{n-1} T(k) + dn$$

מקיים $T(n) = O(n)$.

נניח באינדוקציה שמתקיים $T(n) \le c \cdot n$ עבור קבוע c המקיים $T(1) \le c$. לפיכך:

$$T(n) \le \frac{2}{n}\sum_{k=\lceil n/2 \rceil}^{n-1} c \cdot k + dn \qquad \le \frac{2c}{n}\left(\sum_{k=1}^{n-1} k - \sum_{k=1}^{\lceil n/2 \rceil - 1} k\right) + dn$$

חשוב הסכומים והחלפת $\lceil n/2 \rceil$ ב- $n/2$:

$$\le \frac{2c}{n}\left(\frac{1}{2}(n-1)n - \frac{1}{2}\left(\frac{n}{2}-1\right)\left(\frac{n}{2}\right)\right) + dn$$

פתרון משוואת הנסיגה

נראה שפתרון המשוואה

$$T(n) \le \frac{2}{n} \sum_{k=\lceil n/2 \rceil}^{n-1} T(k) + dn$$

מקיים $T(n) = O(n)$.

נניח באינדוקציה שמתקיים $T(n) \le c \cdot n$ עבור קבוע c המקיים $T(1) \le c$. לפיכך:

$$T(n) \le \frac{2}{n} \sum_{k=\lceil n/2 \rceil}^{n-1} c \cdot k + dn \qquad \le \frac{2c}{n} \left(\sum_{k=1}^{n-1} k - \sum_{k=1}^{\lceil n/2 \rceil - 1} k \right) + dn$$

חשוב הסכומים והחלפת $\lceil n/2 \rceil$ ב- $n/2$:

$$\le \frac{2c}{n} \left(\frac{1}{2}(n-1)n - \frac{1}{2}\left(\frac{n}{2} - 1\right)\left(\frac{n}{2}\right) \right) + dn$$

$$\le \frac{3c}{4} n + d \cdot n \le c \cdot n$$

©cs,Technion

פתרון משוואת הנסיגה

נראה שפתרון המשוואה

$$T(n) \le \frac{2}{n}\sum_{k=\lceil n/2 \rceil}^{n-1} T(k) + dn$$

מקיים $T(n) = O(n)$.

נניח באינדוקציה שמתקיים $T(n) \le c \cdot n$ עבור קבוע c המקיים $T(1) \le c$. לפיכך:

$$T(n) \le \frac{2}{n}\sum_{k=\lceil n/2 \rceil}^{n-1} c \cdot k + dn \qquad \le \frac{2c}{n}\left(\sum_{k=1}^{n-1} k - \sum_{k=1}^{\lceil n/2 \rceil - 1} k\right) + dn$$

חשוב הסכומים והחלפת $\lceil n/2 \rceil$ ב- $n/2$:

$$\le \frac{2c}{n}\left(\frac{1}{2}(n-1)n - \frac{1}{2}\left(\frac{n}{2}-1\right)\left(\frac{n}{2}\right)\right) + dn$$

$$\le \frac{3c}{4} n + d \cdot n \le c \cdot n$$

אי השוויון האחרון מתקיים כאשר c נבחר להיות גדול מספיק כך שיתקיים $d \le c/4$.

©cs,Technion

מקרה פרטי: מציאת חציון

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

זמן הריצה, כפי שראינו הוא $O(n)$.

©cs,Technion

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. לשם דיוק, 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 ?

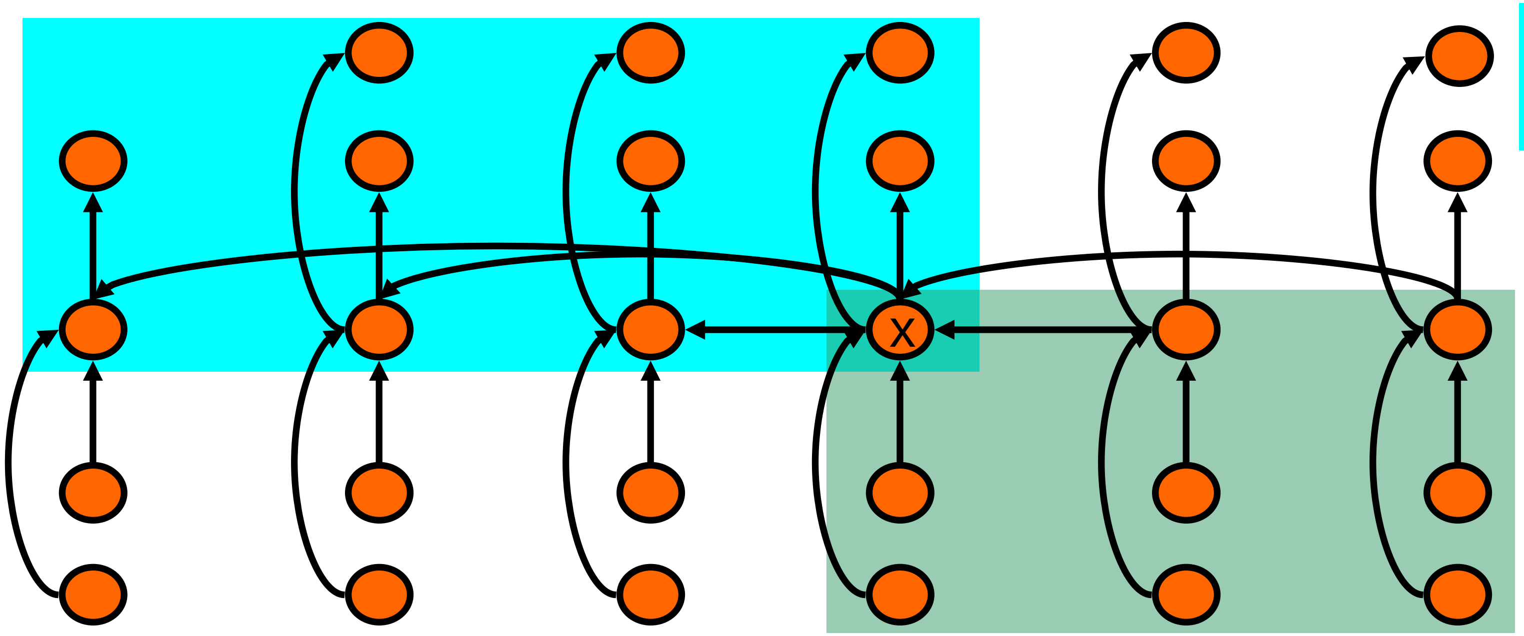
ארבעת השלבים הראשונים של האלגוריתם הבא מוצאים חלוקה עם התכונות שהגדרנו.הניתוח מראה מדוע זו הבחירה הנדרשת.

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 =Partition(A,first,last,x) (חלוקת המערך A לפי x).
5. אם $s – first \geq i$, Select(A, first, s - 1, i)
6. אחרת, Select(A, s, last, i - (s-first))

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 =Partition(A,first,last,x) (חלוקת המערך A לפי x).
5. אם s – first ≥ i , Select(A, first, s - 1, i)
6. אחרת, Select(A, s, last, i - (s-first))



האיברים בפינה השמאלית
קטנים או שווים ל-x

האיברים בפינה הימנית
גדולים או שווים ל-x

ניתוח גודל הקבוצות שנוצרו

כיון ש-x הוא חציון החציונים, וישנם $\lceil n/5 \rceil$ חציונים, מתקיים שלפחות $\lceil 1/2 \cdot \lceil n/5 \rceil \rceil$ קטנים או שווים ל-x. לשם פשטות הניתוח נניח שכל האיברים שונים.

בכל קבוצה של חציון הקטן או שווה ל-x ישנם שלושה איברים הקטנים מ- x (פרט אולי לקבוצה האחת בה אין חמישה איברים ולקבוצה בה x נמצא).

לפיכך מספר האיברים הקטנים מ-x גדול שווה ל:

$$\left(\left\lceil \frac{1}{2} \cdot \left\lceil \frac{n}{5} \right\rceil \right\rceil - 2\right) \cdot 3 \ge \frac{3}{10} n - 6$$

©cs,Technion

ניתוח גודל הקבוצות שנוצרו

כיון ש-x הוא חציון החציונים, וישנם $\lceil n / 5 \rceil$ חציונים, מתקיים שלפחות $\lceil 1/2 \cdot \lceil n / 5 \rceil \rceil$ קטנים או שווים ל-x. לשם פשטות הניתוח נניח שכל האיברים שונים.

בכל קבוצה של חציון הקטן או שווה ל-x ישנם שלושה איברים הקטנים מ- x (פרט אולי לקבוצה האחת בה אין חמישה איברים ולקבוצה בה x נמצא).

לפיכך מספר האיברים הקטנים מ-x גדול שווה ל:

$$\left(\left\lceil \frac{1}{2}\cdot\left\lceil\frac{n}{5}\right\rceil\right\rceil - 2\right)\cdot 3 \ge \frac{3}{10}n - 6$$

בצורה דומה, מספר האיברים הגדולים מ-x גדול שווה ל:

$$\left(\left\lfloor \frac{1}{2}\cdot\left\lceil\frac{n}{5}\right\rceil\right\rfloor - 1\right)\cdot 3 \ge 3\cdot\left(\frac{1}{2}\cdot\left\lceil\frac{n}{5}\right\rceil - 1 - 1\right) \ge \frac{3}{10}n - 6$$

מכאן, בכל קריאה רקורסיבית של select גודל הקלט הוא לכל היותר $\frac{7}{10}n + 6$

ניתוח זמן הריצה

נוסחת הנסיגה עבור זמן הריצה מקיימת:

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

טענה: T(n) = O(n). ההוכחה באינדוקציה.

יהי d הקבוע הנחבא בסימון O בנוסחת הנסיגה. יהי c קבוע כך שעבור n ≤ 80 מתקיים $T(n) \le c \cdot n$.

ברור שקיים קבוע כזה.

$$T(n) \le c \cdot \lceil n/5 \rceil + c(7n/10+6) + d \cdot n$$

$$\le c \cdot (n/5) + c + c(7n/10+6) + d \cdot n$$

©cs,Technion

ניתוח זמן הריצה

נוסחת הנסיגה עבור זמן הריצה מקיימת:

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

טענה: $T(n) = O(n)$. ההוכחה באינדוקציה.

יהי d הקבוע הנחבא בסימון O בנוסחת הנסיגה. יהי c קבוע כך שעבור $n \le 80$ מתקיים $T(n) \le c \cdot n$.

ברור שקיים קבוע כזה.

$$T(n) \le c \cdot \lceil n/5 \rceil + c(7n/10 + 6) + d \cdot n$$

$$\le c \cdot (n/5) + c + c(7n/10 + 6) + d \cdot n$$

$$\le 9cn/10 + 7c + d \cdot n$$

$$\le c \cdot n$$

נבחר את $c = 80\,d$ כך שיתקיים אי השוויון האחרון:

ניתוח זמן הריצה

נוסחת הנסיגה עבור זמן הריצה מקיימת:

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

טענה: T(n) = O(n). ההוכחה באינדוקציה.

יהי d הקבוע הנחבא בסימון O בנוסחת הנסיגה. יהי c קבוע כך שעבור n ≤ 80 מתקיים T(n) ≤ c·n .

ברור שקיים קבוע כזה.

$$T(n) \le c \cdot \lceil n/5 \rceil + c(7n/10 + 6) + d \cdot n$$

$$\le c \cdot (n/5) + c + c(7n/10 + 6) + d \cdot n$$

$$\le 9cn/10 + 7c + d \cdot n$$

$$\le c \cdot n$$

נבחר את c = 80 d כך שיתקיים אי השוויון האחרון:

הסבר לבחירת c:

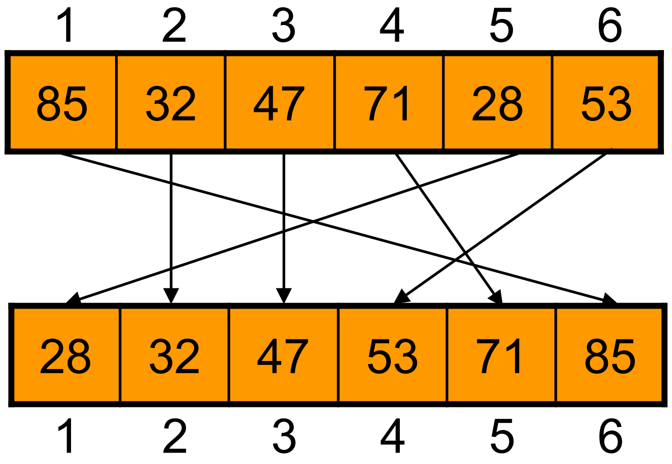
$$9cn/10 + 7c + d \cdot n = 9cn/10 + 7c + \frac{c}{80}n = \frac{73}{80}c \cdot n + 7c$$

$$= cn + \left(-7c\frac{n}{80} + 7c\right) \le cn \qquad (\text{for } n \ge 80)$$

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:



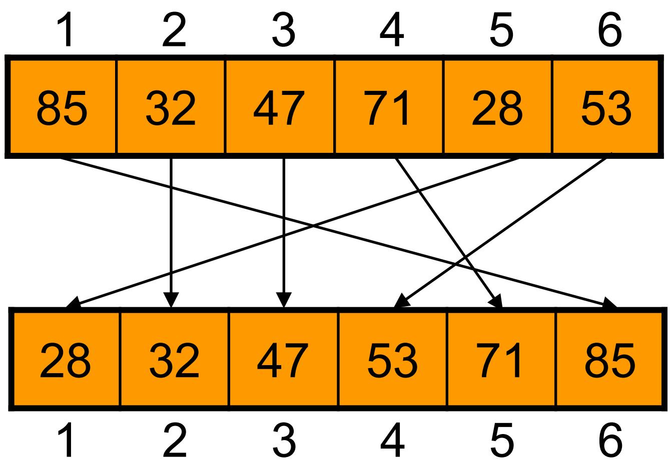
$\pi(1)=6,\ \pi(2)=2,\ \pi(3)=3,\ \pi(4)=5,\ \pi(5)=1, \pi(6)=4$

©cs,Technion

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 $\sigma=\pi^{-1}$:

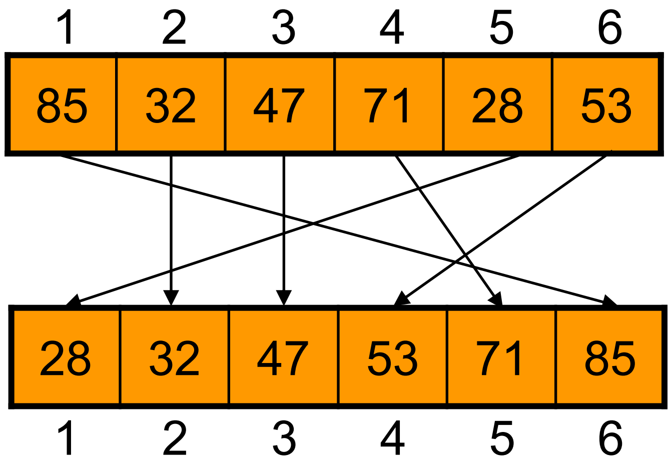
$\sigma(1)=5,\ \sigma(2)=2,\ \sigma(3)=3,\ \sigma(4)=6,\ \sigma(5)=4,\ \sigma(6)=1$

©cs,Technion

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

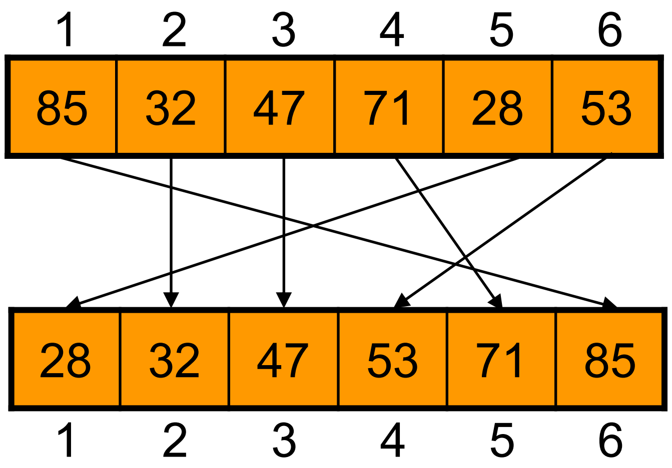
שיתקיים $A[\sigma(1)] < A[\sigma(2)] < \ldots < A[\sigma(n)]$

©cs,Technion

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

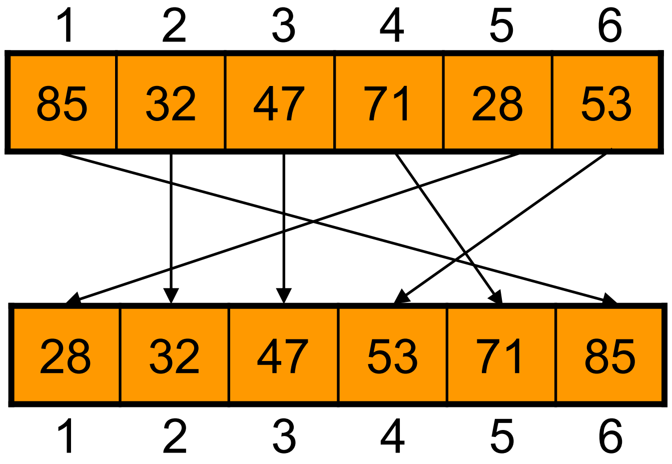
כל פרמוטציה אפשרית כפלט כיון שהאיבר ה-$A[i]$ יכול להגיע לכל מקום במערך הפלט לפי גודלו היחסי בסדרת הקלט. קיימות $n!$ פרמוטציות.

©cs,Technion

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

כל פרמוטציה אפשרית כפלט כיון שהאיבר ה-$A[i]$ יכול להגיע לכל מקום במערך הפלט לפי גודלו היחסי בסדרת הקלט. קיימות $n!$ פרמוטציות.

כמה שאלות כן/לא צריכות להישאל עד שנותרת פרמוטציה אחת אפשרית ?

©cs,Technion

חסם תחתון למיון ע"י השוואות (המשך)

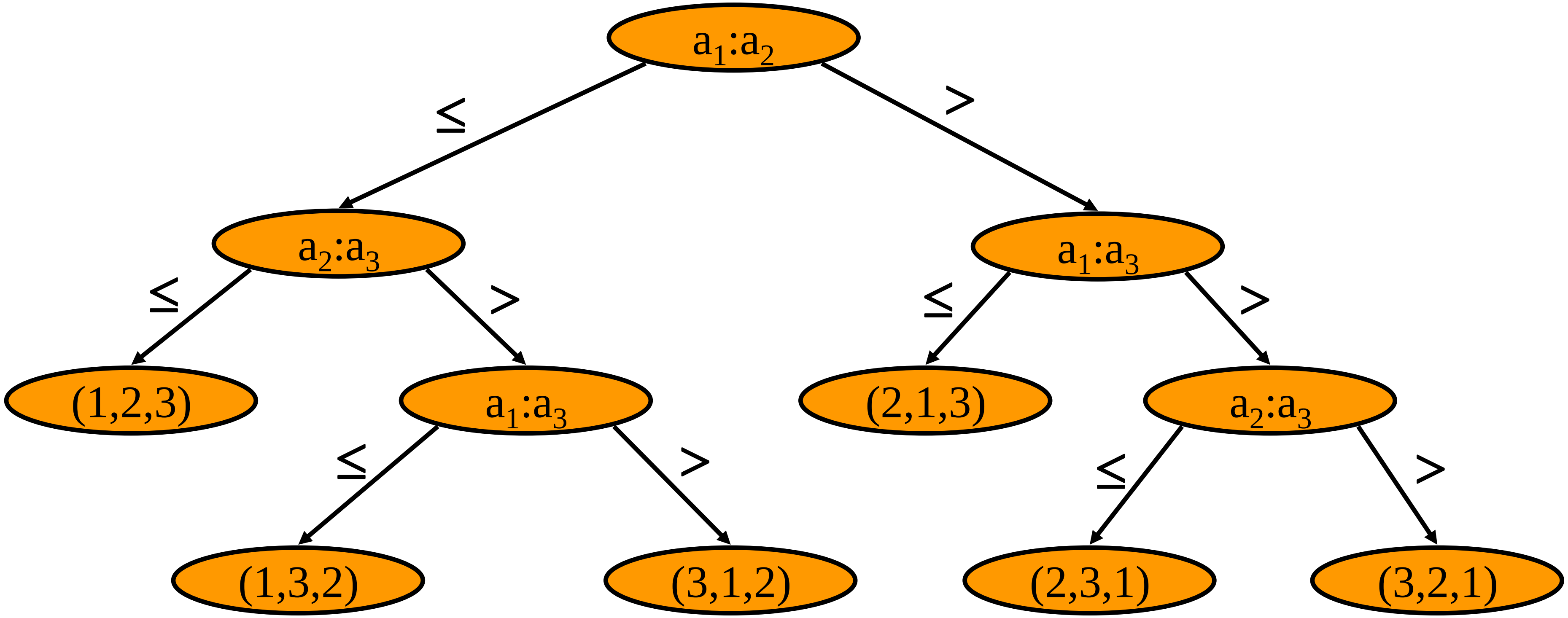
משפט: כל אלגוריתם מיון הפועל באמצעות השוואות בלבד דורש לפחות $\Omega(n \log n)$ השוואות.

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

©cs,Technion

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 $n!$.

מספר ההשוואות המקסימלי הוא גובה העץ h ומתקיים: $n \geq 9$ $h > \log_2(n!) > \frac{1}{2} n \log_2 n$

©cs,Technion

חסם תחתון למיון ע"י השוואות (המשך)

נוסחת סטרלינג לעצרת:

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

ולכן:

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

©cs,Technion

חסם תחתון למיון ע"י השוואות (המשך)

נוסחת סטרלינג לעצרת:

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

ולכן:

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

$$\log_2(n!) > n\log_2 n - n\log_2 e$$

©cs,Technion

חסם תחתון למיון ע"י השוואות (המשך)

נוסחת סטרלינג לעצרת:

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

ולכן:

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

$$\log_2(n!) > n\log_2 n - n\log_2 e$$

$$\log_2(n!) > \frac{1}{2} n\log_2 n \qquad n \geq 9$$

©cs,Technion

חסם תחתון למיון ע"י השוואות (המשך)

נוסחת סטרלינג לעצרת:

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

ולכן:

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

$$\log_2(n!) > n\log_2 n - n\log_2 e$$

$$\log_2(n!) > \frac{1}{2}n\log_2 n \qquad n \geq 9$$

מ.ש.ל.

נימוק מפורט לשורה האחרונה:

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

$$\frac{1}{2}n\log_2 n > n\log_2 e$$

$$\log_2 \sqrt{n} > \log_2 e$$

$$\log_2 \sqrt{9} > \log_2 e$$

מיון BucketSort

<u>המשימה</u>: למיין n מספרים **<u>שונים</u>** בתחום 1..k.

<u>השיטה</u>: 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...k = 6. נניח שהקלט הוא 2, 3, 1, 5.

1	1	1		1	
1	2	3	4	5	6

1	1	1			
1	2	3	4	5	6

<u>זמן</u>: איפוס המערך $\Theta(k)$, מעבר על הקלט $\Theta(n)$, איסוף $\Theta(k)$. סכ"ה $\Theta(n+k)$.

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

©cs,Technion

היכן הקסם ?

מדוע ההוכחה שנדרשים לפחות $\Omega(n \log n)$ השוואות אינה "תופסת".

מהו בדיוק תנאי המשפט שמופר במיון BucketSort.

היכן הקסם ?

מדוע ההוכחה שנדרשים לפחות $\Omega(n \log n)$ השוואות אינה "תופסת".

מהו בדיוק תנאי המשפט שמופר במיון BucketSort.

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 $k = \Theta(n)$ אזי אמנם נקבל אלגוריתם מיון ליניארי $\Theta(n)$ ואולם כל צעד מקביל ל- $\Theta(\log_2 n)$ השוואות בינריות.

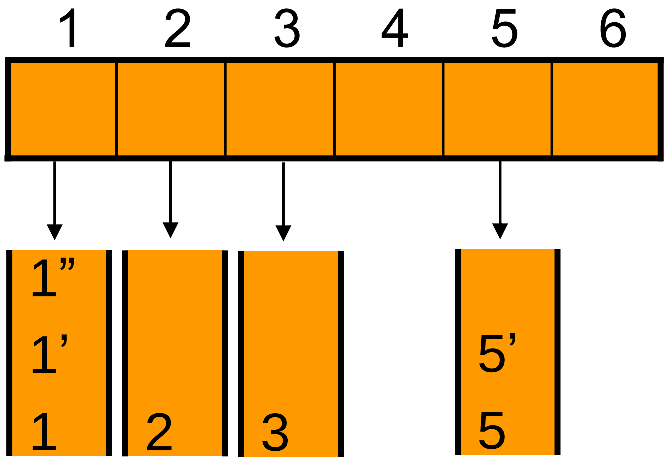
©cs,Technion

הרחבת מיון BucketSort

המשימה: למיין n מספרים שאינם בהכרח שונים מהתחום 1...k.

השיטה: נשתמש במערך של k תורים. בזמן האיסוף נשרשר את התורים.

דוגמא: נניח $n = 7$ וטווח המספרים הוא $1...k = 6$. נניח שהקלט הוא 5, 1, 1', 2, 3, 1", 5'.



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

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

מיון RadixSort

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

מיון RadixSort

אבחנה: שיטת BucketSort אינה יעילה כאשר n המספרים לקוחים מתחום "רחב מדי" $1..k$, כלומר כאשר $k \gg n$. 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. נמיין כל ספרה בנפרד תוך שימוש בשיטת מיון יציבה (לדוגמא BucketSort).

ספרות פחות משמעותיות (Least Significant Digits) ממוינות ראשונות (בניגוד אולי לאינטואיציה ראשונית).

מיון RadixSort

אבחנה: שיטת BucketSort אינה יעילה כאשר n המספרים לקוחים מתחום "רחב מדי" 1..k, כלומר כאשר k >> 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. נמיין כל ספרה בנפרד תוך שימוש בשיטת מיון יציבה (לדוגמא BucketSort).

ספרות פחות משמעותיות (Least Significant Digits) ממוינות ראשונות (בניגוד אולי לאינטואיציה ראשונית).

329
457
657
839
436
720
355

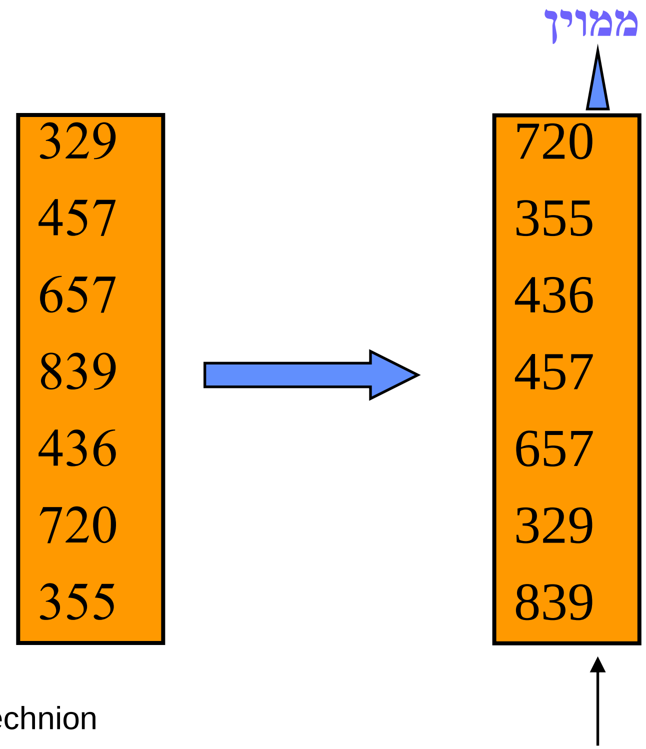
©cs,Technion

מיון RadixSort

אבחנה: שיטת BucketSort אינה יעילה כאשר n המספרים לקוחים מתחום "רחב מדי" 1..k, כלומר כאשר k ›› 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. נמיין כל ספרה בנפרד תוך שימוש בשיטת מיון יציבה (לדוגמא BucketSort).

ספרות פחות משמעותיות (Least Significant Digits) ממוינות ראשונות (בניגוד אולי לאינטואיציה ראשונית).

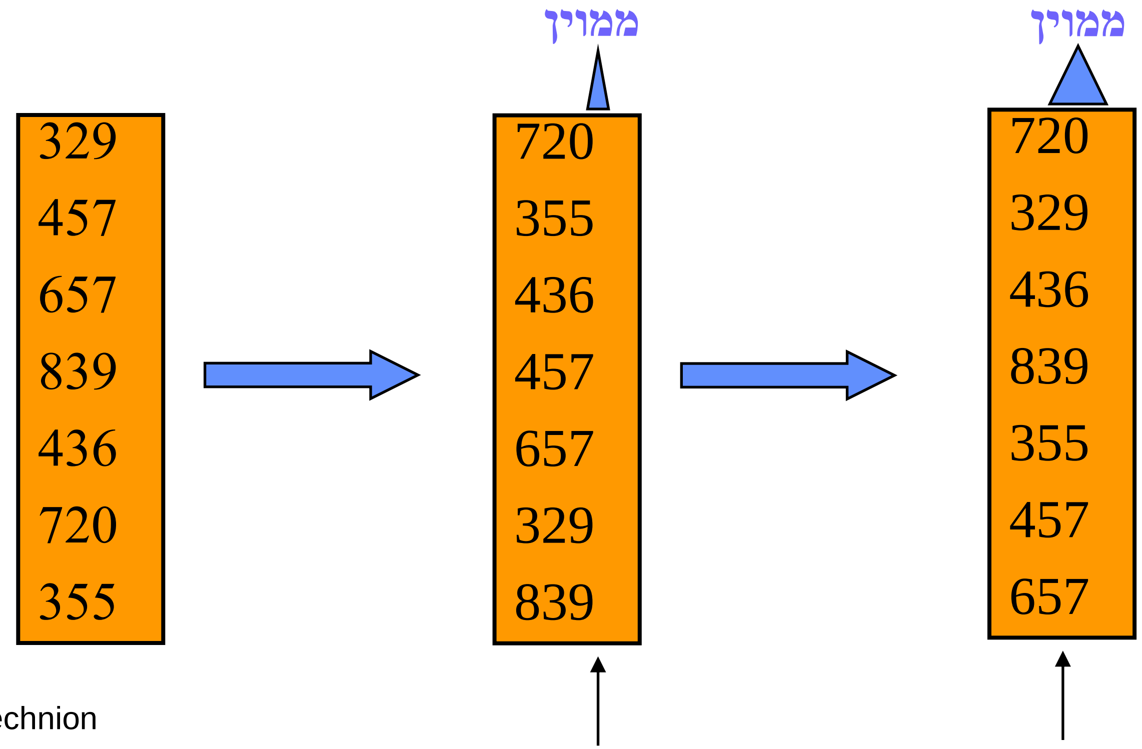


מיון RadixSort

אבחנה: שיטת BucketSort אינה יעילה כאשר n המספרים לקוחים מתחום "רחב מדי" 1..k, כלומר כאשר k >> 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. נמיין כל ספרה בנפרד תוך שימוש בשיטת מיון יציבה (לדוגמא BucketSort).

ספרות פחות משמעותיות (Least Significant Digits) ממוינות ראשונות (בניגוד אולי לאינטואיציה ראשונית).



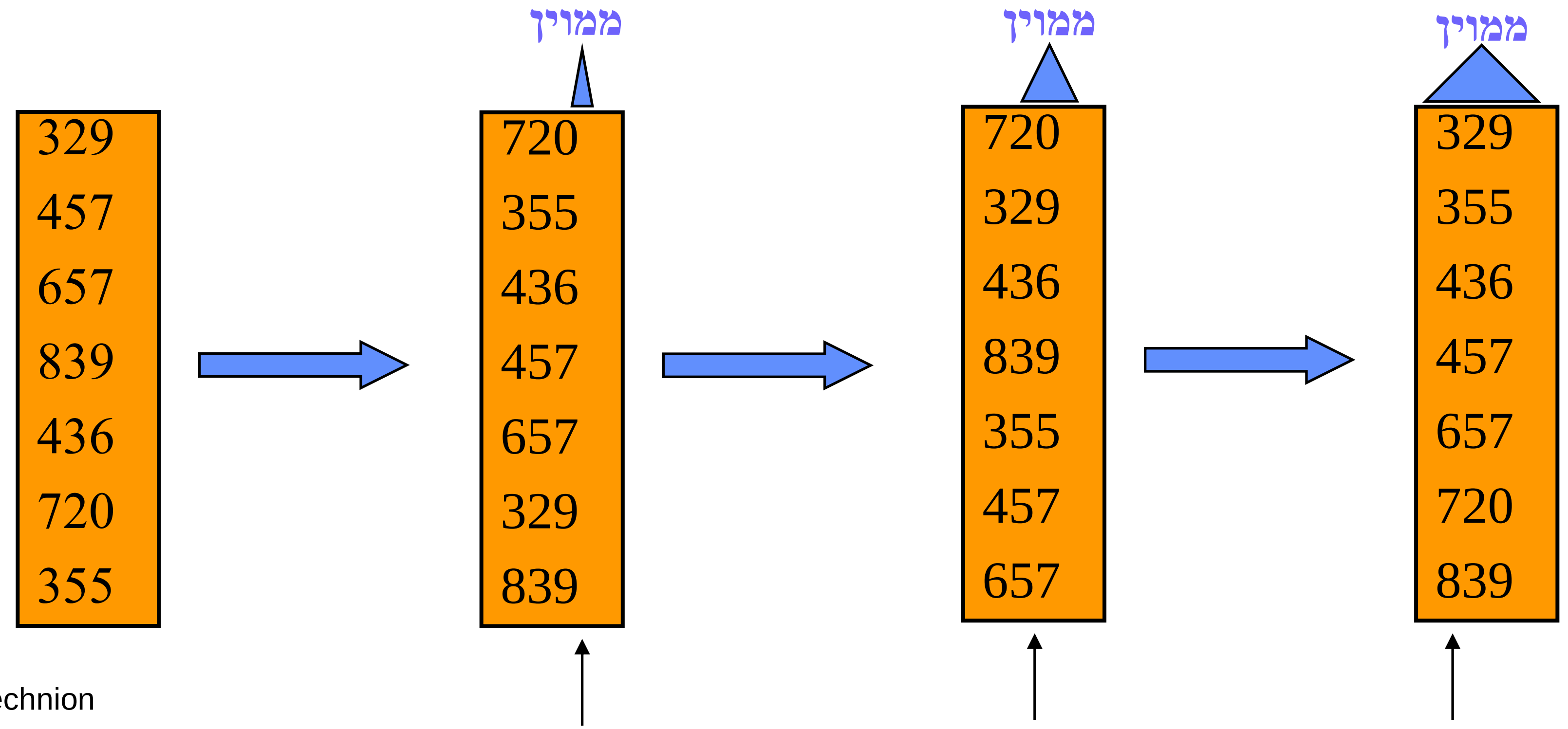
©cs,Technion

מיון RadixSort

אבחנה: שיטת BucketSort אינה יעילה כאשר n המספרים לקוחים מתחום "רחב מדי" 1..k, כלומר כאשר k >> 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. נמיין כל ספרה בנפרד תוך שימוש בשיטת מיון יציבה (לדוגמא BucketSort).

ספרות פחות משמעותיות (Least Significant Digits) ממוינות ראשונות (בניגוד אולי לאינטואיציה ראשונית).



©cs,Technion

פרוצדורת RadixSort

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

```
Radix-Sort(A,d)
for i ← 1 to d
        do use BucketSort to sort array A on digit i
```

תרגיל: הוכחת נכונות באינדוקציה על d. היכן משתמשים בעובדה 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 בסדר שכזה:

329	→	329	→	329	→	720
457		355		720		355
657		457		436		436
839		436		839		457
436		657		355		657
720		720		457		329
355		839		657		839

©cs,Technion

ניתוח זמן הריצה

לכל ספרה עשרונית נבצע BucketSort בזמן $\Theta(n+10)$.

עבור d ספרות הזמן הכולל הוא $\Theta(d(n+10))$.

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

©cs,Technion

ניתוח זמן הריצה

לכל ספרה עשרונית נבצע BucketSort בזמן $\Theta(n+10)$.

עבור d ספרות הזמן הכולל הוא $\Theta(d(n+10))$.

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

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 $\Theta(\log_r k\cdot(n+r))$.

כלומר עבור $k > n$ ובסיס- r קבוע, הזמן הנדרש הוא $\Omega(n \log n)$ כפי שדרוש במיונים עם השוואות בלבד.

©cs,Technion

Median/sort