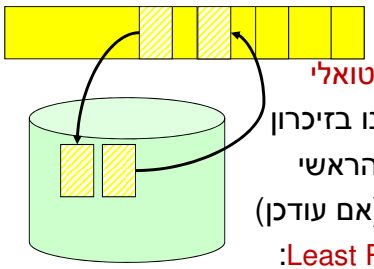
מיון בזיכרון המשני

מיון בזיכרון הוירטואלי



נמיין את הקובץ באמצעות אלגוריתם מיון בזיכרון הראשי באמצעות מנגנון הזיכרון הווירטואלי

פסיקת דפדוף (page fault): בגישה לדף שאינו בזיכרון

- מביאים את הדף מהדיסק למסגרת בזיכרון הראשי
- אם צריך, מפנים דף... כולל כתיבתו לדיסק (אם עודכן)

נניח שמפנים את הדף (LRU) Least Recently Used:
הדף שהגישה האחרונה אליו היא המוקדמת ביותר

	A	B	C	A	B	D	A	D	B	C	B
0	A	A	A	A	A	A	A	A	A	C	C
1		B	B	B	B	B	B	B	B	B	B
2			C	C	C	D	D	D	D	D	D

	A	B	C	D	A	B	C	D	A	B	C	D
0	A	A	A	D	D	D	C	C	C	B	B	B
1		B	B	B	A	A	A	D	D	D	C	C
2			C	C	C	B	B	B	A	A	A	D

מיון

נתון: יחס גדול מאוד המאוחסן בקובץ בזיכרון המשני

המשימה: למצוא אלגוריתם יעיל כך שלאחר ביצועו יתקיים:

$$x_1.key \le x_2.key \le \ldots \le x_N.key$$

- משתמשים ב M רשומות בזיכרון הראשי
- גודל היחס / קובץ (ברשומות) >> M
- גודל הזיכרון המשני אינו מוגבל

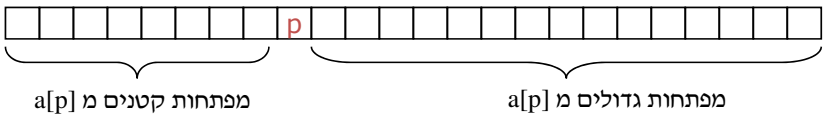
Quicksort: תזכורת

```
sort a[left...right]:
if left < right
  Partition a[left...right]
  so that
       all a[left...p-1]
             are < a[p]
       all a[p+1...right]
             are > a[p]
  Quicksort a[left...p-1]
  Quicksort a[p+1...right]
```

בוחרים אחד מהאיברים a[p] לשמש כפיבוט (ציר)

מפצלים את המערך לשני חלקים

- מפתחות גדולים מהפיבוט
- מפתחות קטנים מהפיבוט
- ממיינים כל חלק בנפרד



Quicksort: פיצול

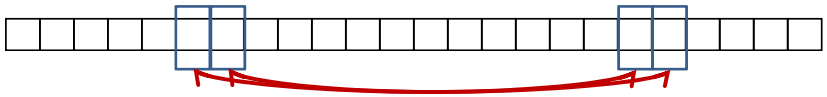
בוחרים אחד מהאיברים a[p] לשמש כפיבוט (ציר)

מפצלים את המערך לשני חלקים

- מפתחות גדולים מהפיבוט
- מפתחות קטנים מהפיבוט

ממיינים כל חלק בנפרד

- מהקצה השמאלי, חפש (ימינה) את האיבר הראשון שגדול מהפיבוט
- מהקצה הימני, חפש (שמאלה) את האיבר הראשון שקטן מהפיבוט
- החלף את שני האיברים
- המשך מאותם מקומות
- עד שנפגשים באמצע

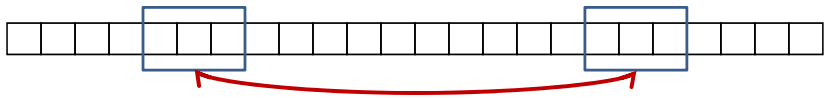


Quicksort: פיצול (דוגמה)

- בחר פיבוט — **4 3 6 9 2 4 3 1 2 1 8 9 3 5 6**
- חפש — **4 3 6 9 2 4 3 1 2 1 8 9 3 5 6**
- החלף — **4 3 3 9 2 4 3 1 2 1 8 9 6 5 6**
- חפש — **4 3 3 9 2 4 3 1 2 1 8 9 6 5 6**
- החלף — **4 3 3 1 2 4 3 1 2 9 8 9 6 5 6**
- חפש — **4 3 3 1 2 4 3 1 2 9 8 9 6 5 6**
- החלף — **4 3 3 1 2 2 3 1 4 9 8 9 6 5 6**
- חפש — **4 3 3 1 2 2 3 1 4 9 8 9 6 5 6**
- החלף עם הפיבוט — **1 3 3 1 2 2 3 4 4 9 8 9 6 5 6**

Quicksort: כמה בלוקים מביאים?

בשלב הראשון ברקורסיה: צריך שני דפים בזיכרון הראשי בכל זמן ולא חוזרים לאותו דף פעמיים במעבר.



לכן, במעבר הראשון מביאים N/b בלוקים

גם בשלבים הבאים ברקורסיה, צריך שני בלוקים בכל זמן ולא חוזרים לאותו דף פעמיים. לכן מביאים N/b בלוקים

כאשר גודל תת-המערך קטן מגודל הזיכרון, M, עובדים בזיכרון הראשי

Quicksort: סיכום

מספר שלבי הרקורסיה שבהם מביאים בלוקים הוא בממוצע $c \log_2(N/M)$, כאשר $c = 2\log_2 e \cong 2.885$ הוא הקבוע בהערכת הזמן של Quicksort

⇦מספר הבאות הבלוקים בממוצע: $(N/b)\, c \log_2(N/M)$

⇦$c \log_2(N/M)$ קריאות אקראיות של הקובץ!

⇦אולי עדיף לבצע מיון במעברים סדרתיים?

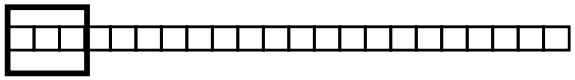
Bubble Sort: מספר הגישות לדיסק

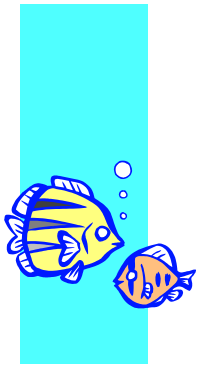
בכל מעבר קוראים את כל הקובץ

קובץ עם N רשומות, B רשומות בכל בלוק

מביאים N/B בלוקים בכל מעבר

N מעברים ⇦ N^2/B גישות לדיסק

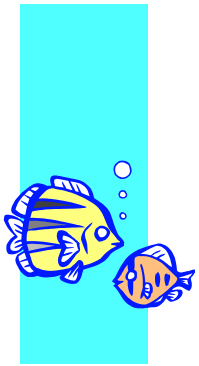




ניתן לשפר על-ידי ניצול של הזיכרון הראשי

Bubble Sort: ניצול יותר טוב של הזיכרון הראשי

מעברים סדרתיים עם M רשומות (במקום 2)



- חזור N/(M–1) פעמים:
  - קרא M–1 רשומות
  - חזור (M–1) – N פעמים:
    - קרא רשומה
    - כתוב את הרשומה בעלת המפתח הכי קטן מבין M הרשומות הנמצאות בזיכרון
  - כתוב את M–1 הרשומות שנותרו בזיכרון בסדר עולה.

Bubble Sort: מספר המעברים

מעברים סדרתיים עם M רשומות (במקום 2)

- מעבר לא מקלקל את הסדר של הרשומות עם המפתחות הכי גדולים (שנמצאות בסוף).
- לאחר המעבר ה-k, k·(M–1) הרשומות עם המפתחות הכי גדולים נמצאות במקום

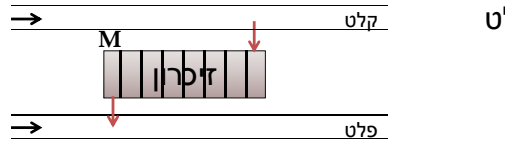
- חזור N/(M–1) פעמים:
  - קרא M–1 רשומות
  - חזור (M–1) – N פעמים:
    - קרא רשומה
    - כתוב את הרשומה בעלת המפתח הכי קטן מבין M הרשומות הנמצאות בזיכרון
  - כתוב את M–1 הרשומות שנותרו בזיכרון בסדר עולה.

צריך רק $N/M-1$ מעברים סדרתיים

מיון במעברים סדרתיים: חסם תחתון

בכל מעבר, האלגוריתם קורא מהקלט וכותב לפלט:

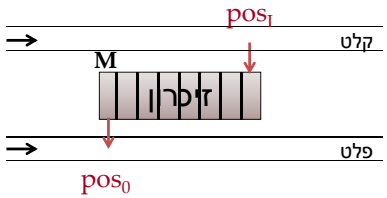
- קורא M–1 רשומות לזיכרון
- חוזר (M–1) – N פעמים:
  - קורא רשומה
  - מבצע חישוב מקומי
  - כותב רשומה
- כותב את M–1 הרשומות שנותרו בזיכרון



משפט: מיון קובץ עם N רשומות דורש לפחות $\frac{cN}{M}$ מעברים, עבור קבוע כלשהו c.

מיון במעברים סדרתיים

הנחות:
- קובץ הקלט מכיל N רשומות עם מפתחות שונים
- עוברים על הקלט סדרתית
- כותבים לקובץ הפלט סדרתית
- משתמשים ב M רשומות בזיכרון הראשי

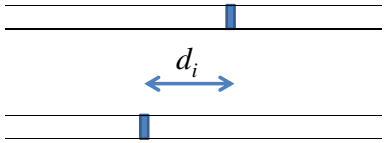


נסמן: pos_I המיקום בקלט
pos_0 המיקום בפלט

$pos_I \geq pos_0$	(כי רשומות לא נוצרות)
$pos_I < pos_0+M$	(כדי שרשומות לא יאבדו)

משפט: מיון קובץ עם N רשומות דורש לפחות $\frac{cN}{M}$ מעברים, עבור קבוע כלשהו c.

מיון במעברים: המרחק בין סדרות



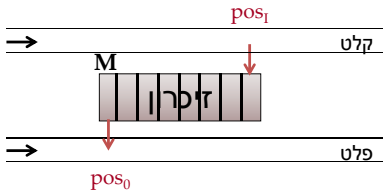
עבור שתי סדרות של אותן רשומות d_i הוא המרחק בין המיקום של הרשומה ה-i בשתי הסדרות.

המרחק הכולל בין שתי סדרות הוא $\sum d_i$

מה המרחק הכי גדול בין סדרה לא-ממוינת לסדרה הממוינת?

טענה: המרחק בין סדרת הקלט וסדרת הפלט במעבר $\geq 2N(M-1)$

חסם תחתון על מספר המעברים

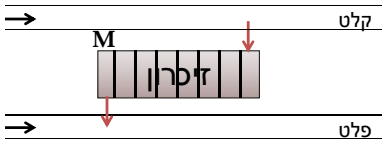


אחרי צעד של כל קריאת רשומה במעבר
יש M רשומות בזיכרון הראשי
- לכל היותר M-1 מהן זזות קדימה ומתקרבות מקום אחד ליעד
- ואחת (זו שקראנו) זזה אחורה ומתקרבת לכל היותר M-1 מקומות ליעד
- המיקום של שאר הרשומות לא משתנה

⇦ בכל צעד, השינוי הכולל במרחקים של כל הרשומות $\geq 2(M-1)$

טענה: המרחק בין סדרת הקלט וסדרת הפלט במעבר $\geq 2N(M-1)$

חסם תחתון על מספר המעברים: סיכום



טענה: המרחק בין סדרת הקלט וסדרת הפלט במעבר $\geq 2N(M-1)$

✓ המרחק המקסימלי בין סדרה בלתי ממוינת לבין הסדרה הממוינת: $\Omega(N^2)$

✓ בכל מעבר, המרחק בין תוצאת המעבר הקודם (הקלט למעבר הנוכחי) ובין תוצאת המעבר הנוכחי (הפלט של המעבר הנוכחי) $\geq 2N(M-1)$

⇦ מספר המעברים הדרושים: $\Omega(N^2)/2N(M-1) = \Omega(\frac{N}{M})$

חסם תחתון על מספר הרשומות שעוברים

אם מבצעים מעברים מלאים, זמן מעבר הוא $\Theta(N)$

⇦זמן המיון הכולל $\Theta(N) \times \Omega(\frac{N}{M}) = \Omega(\frac{N^2}{M})$

חסם תחתון על מספר הרשומות שעוברים: מקרה כללי

r_i

אם מבצעים מעברים חלקיים, ועוברים על r_i רשומות במעבר i

השינוי במעבר ה-i ≥ $2r_i(M-1)$

ומכיוון ש $\sum 2r_i\,(M-1) \geq N^2$

נובע ש $\sum r_i \geq \frac{N^2}{2(M-1)}$

⇦קיבלנו זמן פולינומי. רצוי זמן לוגריתמי...

מיון-מיזוג (Merge Sort)

הרעיון הכללי:

- מיון חלקים של הקובץ בזיכרון וכתיבת החלקים הממוינים לדיסק
- מיזוג החלקים הממוינים
  - מעבר לא סדרתי על הקובץ
  - עשוי לדרוש מספר איטרציות

הערה: זו השיטה המקובלת ביותר כיום למיון קבצים גדולים

- טובה גם ל-HDD וגם ל-SSD
- יחסית קלה לביצוע באופן מקבילי או מבוזר

מיון-מיזוג

מַעֲלֶה: סדרה עולה מקסימלית (run)

5	12	13

סדרת הקלט

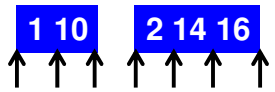
9	5	12	13	1	10	6	15

מתחלקת לארבעה מעלות

9	5	12	13	1	10	6	15

נחלק את הקלט לשתי סדרות, ואז נמזג אותן.

מיזוג של שתי סדרות



1 2 10 14 16

מיזוג: מחזור 2

פיצול: כדי לאפשר מחזור נוסף, נפצל את הסדרה לשתי סדרות

4 8 9 11 | 3 5 12 13 | 1 2 10 14 16 | 6 7 15

4 8 9 11 | 1 2 10 14 16

3 5 12 13 | 6 7 15

מיזוג:	**קלט 1**	**קלט 2**	**פלט**
	4 8 9 11	3 5 12 13	3 4 5 8 9 11 12 13
	1 2 10 14 16	6 7 15	1 2 6 7 10 14 15 16

לשלב הבא 3 4 5 8 9 11 12 13 2 6 7 10 14 15 16

מיזוג: מחזור 1

מיזוג:

סדרה 1: 4 8 11 | 3 | 2 14 16 | 7

סדרה 2: 9 | 5 12 13 | 1 10 | 6 15

	קלט 1	**קלט 2**	**פלט**
מעלה 1	4 8 11	9	4 8 9 11
מעלה 2	3	5 12 13	3 5 12 13
מעלה 3	2 14 16	1 10	1 2 10 14 16
מעלה 4	7	6 15	6 7 15

סה"כ הפלט 4 8 9 11 | 3 5 12 13 | 1 2 10 14 16 | 6 7 15

מיזוג: מחזור 3

פיצול של

3 4 5 8 9 11 12 13 1 2 6 7 10 14 15 16

לשתי סדרות:

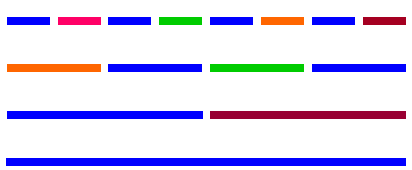
3 4 5 8 9 11 12 13

1 2 6 7 10 14 15 16

מיזוג לסדרת פלט:

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

מיזוג: מספר המחזורים



בכל מחזור מספר המעלות קטן פי 2
⇦ צריך $\log_2 N \geq$ מחזורים.

מיזוג של k סדרות

דרגת המיזוג = מספר הסדרות

עבור דרגת מיזוג k, מספר המעלות קטן פי k בכל סיבוב.

מספר הסיבובים הדרושים עם דרגת מיזוג k הוא $\log_k N = \frac{\log_2 N}{\log_2 k}$

מסקנה: הגדלת דרגת המיזוג מקטינה את מספר המחזורים בגורם כפלי.

יצירת מעלות התחלתיים (I): מיון בזיכרון ראשי

אם משתמשים ב M רשומות בזיכרון הראשי, אפשר ליצור מעלות התחלתיים באורך M, על ידי קריאת M מעלות ומיונם. (מעבר אחד)

מספר המחזורים למיון מלא: $\log_k \frac{N}{M} = \log_k N - \log_k M$

N: מספר הרשומות הכולל

k: סדר המיון

עבור $k = 10, \quad M = 1000, \quad N = 10{,}000{,}000$

$\log_k N - \log_k M = \log_{10} 10{,}000{,}000 - \log_{10} 1{,}000 = 7 - 3 = 4$

מעלות התחלתיים ארוכים מקטינים את מספר המעברים בגורם חיבורי

יצירת מעלות התחלתיים (II): Replacement / Selection

יצירת מעלות התחלתיים באורך ≤ M

- מלא את הזיכרון
- חזור עד סוף סדרת הקלט:
  - התחל מעלה חדש –
    - הוצא את המינימום לפלט.
    - קרא רשומה חדשה במקום הרשומה שיצאה.
  - כל עוד יש בזיכרון מפתחות שגדולים מהאחרון שיצא:
    - הוצא את המינימום שגדול מהאחרון שהוצא.
    - קרא רשומה חדשה במקום הרשומה שיצאה.
- הוצא את הרשומות שנותרו בסדר עולה.

Replacement / Selection: רשומות קפואות ופעילות

- רשומה היא **קפואה** אם המפתח שלה קטן מהאחרון שהוצא (לא ניתן להוציא אותה עד להתחלת מעלה חדש)
- שאר הרשומות הן **פעילות**
- בהתחלה (ובהתחלת כל מעלה) כל הרשומות פעילות
- כאשר רשומה נקראת, נקבע אם היא קפואה (עד להתחלת המעלה הבא)
  מצב הרשומות האחרות אינו משתנה

כל הרשומות קפואות ← מעלה מסתיים

כל הרשומות הופכות לפעילות ← נפתח מעלה חדש

דוגמת ריצה של Replacement / Selection

M = 3

קלט	רשומות קפואות	רשומות פעילות	פלט
42873690			
73690		248	
3690		478	2
690	3	78	24
90	36	8	247
0	36	9	2478
	036		24789
		36	247890
		6	2478903
			24789036

תוחלת מספר המעלות הצפוי

מעלה חדש מתחיל כאשר כל הרשומות קפואות
אם בזיכרון M רשומות, ו-K רשומות נכנסות קפואות

⇦ מספר המעלות בסדרת הפלט $\geq 1 + \lceil K/M \rceil$

⇦ תוחלת מספר המעלות $\cong E\left(K/M\right)$

נחשב את תוחלת מספר הרשומות הקפואות, כאשר המפתחות מוגרלים בהתפלגות אחידה: U[0,1]

פונקציית ההסתברות המצטברת: $F(t) = P(X \leq t) = t$

ופונקציית הצפיפות: $f(t) = \frac{d}{dt}F(t) = 1$

תוחלת מספר הרשומות הקפואות (המשך)

תוחלת ערך מפתחות: $E_x(x) = \int_0^1 tf(t)dt = \int_0^1 t\frac{d}{dt}F(t)dt = \int_0^1 tdt = \frac{1}{2}$

<u>הבחנה</u>: אם מחליפים רשומה עם מפתח x ברשומה עם מפתח y, הרשומה y תהיה קפואה אם ורק אם $y < x$.

ההסתברות לכך:

$P(\text{קפואה } y \mid \text{הוצאה } x) = P(y < x) = x$

עבור $x , M \ll N$ מפולג כמעט באופן אחיד, $E_x(x) \cong 1/2$.

ההסתברות שרשומה חדשה תהיה קפואה:

$P(\text{קפואה}) = E_x(P_y(\text{קפואה } y \mid x)) = E_x(x) \cong 1/2$

⇦תוחלת מספר הרשומות הקפואות:

$E(K) \cong N \cdot P(\text{קפואה } y) \cong N/2$

מספר מחזורי מיון אחרי R/S

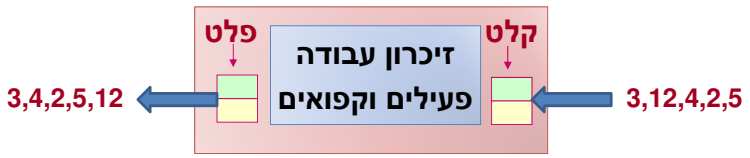
לכן, מספר המעלות הצפוי: $E\left(\frac{K}{M}\right) \cong \frac{N}{2} \times \frac{1}{M} = \frac{N}{2M}$

⇦ תוחלת האורך של מעלה: $\frac{N}{N/2M} = 2M$

⇦ מספר מחזורי המיון: $\log_k \frac{N}{2M} = \log_k \frac{N}{M} - \log_k 2$

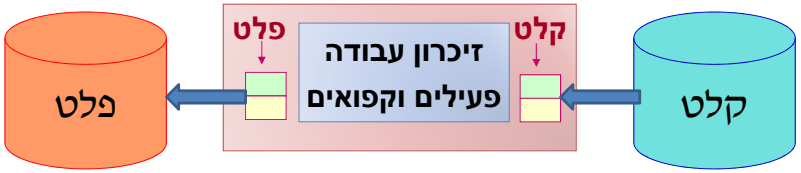
עבור קלט אקראי, שיטת R/S מקצרת את זמן המיון במחזור אחד לכל היותר, בהשוואה למיון בזיכרון הראשי (שיטה I)

דוגמה: חוצצים כפולים בשלב R / S



ניהול הזיכרון לשלב R / S

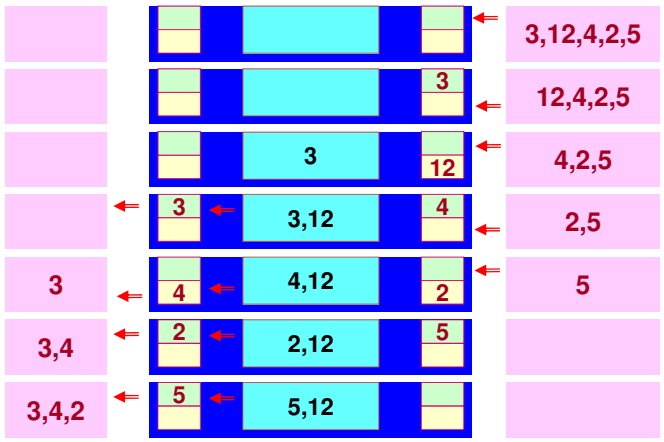
קובץ הקלט נמצא על דיסק אחד, והמעלות נכתבים לדיסק שני



רוצים כתיבה רציפה לדיסק הפלט (לאחר השלב ההתחלתי)

double buffering: מקצים שני חוצצים לקלט ושני חוצצים לפלט

דוגמה: חוצצים כפולים בשלב R / S



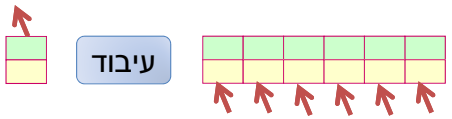
זמן R/S = זמן קריאה = זמן כתיבה.

דרגת המיזוג המקסימלית

מיזוג מדרגה k דורש 2 חוצצים לכל סדרת קלט (נראה בהמשך) ועוד 2 לפלט.

סה"כ 2(k+1) = (2k+2) חוצצים

N = גודל קובץ (בבתים)
M = גודל הזיכרון (בבתים)
b = גודל חוצץ (קבוע, בבתים)



סדר המיזוג k מקיים: $2(k+1)\cdot b \le M$

$k \le (M/2b) - ½ \cong M/(2b)$ ⇦

מספר מעברי מיזוג: נוסחה כללית

מספר המעלות לאחר מעבר R / S הוא

$r_1 = {}^{N}/_{2(M-2)} \cong {}^{N}/_{2M}$

N = גודל קובץ (בבתים)
M = גודל הזיכרון (בבתים)
b = גודל חוצץ (קבוע, בבתים)

לאחר מעבר מיזוג אחד מסדר $k \cong {}^{M}/_{2b}$, מספר המעלות הוא

$$r_2 = r_1/k = \frac{N/(2M)}{M/(2b)} = \frac{Nb}{M^2}$$

עם b = 1K, אפשר למיין

M	2 מעברי מיזוג	3 מעברי מיזוג
1MB	1GB	512 GB
100MB	10TB	512 PB

אם שני מעברים מספיקים: $r_2 \le 1$
⇦ אפשר למיין קובץ בגודל ${}^{M^2}/_{b}$

בשלושה מעברים: $r_3 = r_2/k = \frac{2Nb^2}{M^3} \le 1$
⇦ אפשר למיין קובץ בגודל $\ge {}^{M^3}/_{2b^2}$

זמן המיון: סיכום

המעבר הראשון (R/S) על הקובץ הוא סדרתי.
- יוצר בערך ${}^{N}/_{2M}$ מעלות
- זמן המעבר אינו תלוי בגודל הזיכרון הראשי

N = גודל קובץ (בבתים)
M = גודל הזיכרון (בבתים)
b = גודל חוצץ (קבוע, בבתים)

כל מעבר מיזוג דורש ${}^{N}/_{b}$ גישות אקראיות
דרגת המיזוג $k = {}^{M}/_{2b}$

מספר המעברים $\log_k \frac{N}{2M} = \frac{\log_2 N/(2M)}{\log_2 k} = \frac{\log_2 N - \log_2 M - 1}{\log_2 k}$

מספר הגישות $\frac{\log_2 N - \log_2 M - 1}{\log_2 {}^{M}/_{2b}} \times \frac{N}{b}$ (פונקציה יורדת של M)

מיון: סיכום ביניים והשוואה

	quicksort	Bubble sort	Merge sort +R/S
מספר מעברים	$c\ log_2\left(\frac{N}{M}\right)$	$\Omega(\frac{N}{M})$	$1 + log_k\left(\frac{N}{2M}\right) = 1 + \frac{log_2\left(\frac{N}{2M}\right)}{log_2 k}$
סוג המעבר	אקראי	סדרתי	אקראי

מיון מיזוג מהיר יותר מ quicksort בפקטור של $c\cdot\log_2 k \approx$ (עבור $k = 9$, בערך 9.147)