הקצאת חוצצים

לא מביאים רשומות בודדות, אלא חוצצים שלמים עם b רשומות כל אחד

רוצים לנהל את החוצצים באופן שמבטיח פעולה **רציפה**:

- כאשר מתמלא חוצץ קלט, יש חוצץ ריק שאליו אפשר להמשיך לקרוא
- כאשר גומרים לכתוב לדיסק חוצץ פלט, חוצץ הפלט הבא כבר מוכן

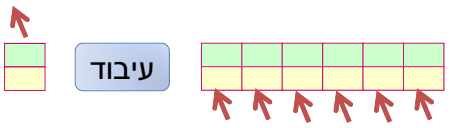
"תקורת" החוצצים קובעת את סדר המיזוג, ולכן רוצים להקטין אותה

צריך לפחות k+1 חוצצים: אחד לכל מעלה קלט, ואחד לפלט

מודל להקצאת חוצצים

בכל מחזור, האלגוריתם:

- קורא חוצץ קלט
- מעביר b רשומות מהקלט לפלט
- כותב חוצץ פלט



מקרי קצה:

- במחזור הראשון אין פלט
- כאשר נגמר הקלט של מעלה, מפסיקים לקרוא אותו
- אחרי המחזור האחרון, כותבים חוצץ פלט

הנחות:

- דיסק קלט אחד, ודיסק פלט אחד
- ממזגים k מעלות קלט למעלה פלט
- 2k+2 חוצצים, כ"א עם b רשומות

k+1 חוצצים לא מספיקים לפעולה רציפה

b = 3 , k = 2

מעלה 1:	1, 3, 5,	9, 20, 21,	22, 23, 24,	25, ...
מעלה 2:	2, 4, 6,	7, 8, 10,	11, 12, 13,	14, ...

מעלה 1:	מעלה 2:	פלט:
1 , 3 , 5	2, 4, 6	
5 , -- , --	--, 4, 6	1, 2, 3
- , -- , --	--, 6	4,5???-

הקלט הדרוש עדיין בדרך

נתקענו!

הקצאה קבועה לא מאפשרת פעולה רציפה

- הקצה שני חוצצים לכל מַעֲלֶה
- מלא את חוצצי הקלט באופן מעגלי

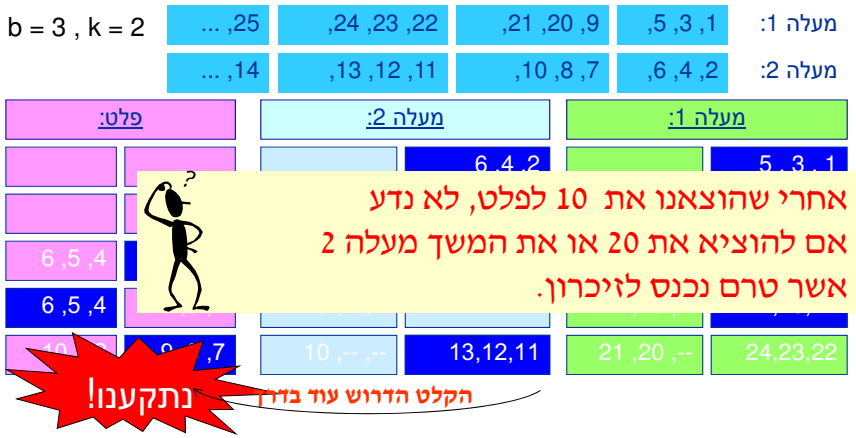
b = 3 , k = 2

מעלה 1:	1, 3, 5,	9, 20, 21,	22, 23, 24,	25, ...
מעלה 2:	2, 4, 6,	7, 8, 10,	11, 12, 13,	14, ...

מעלה 1:	מעלה 2:	פלט:	
1 , 3 , 5	2, 4, 6		
5 , -- , --	--, 4, 6	1, 2, 3	
9, 20, 21	10, 8, 7	1, 2, 3	4, 5, 6
--, 20, 21	--, --, 10	7, 8, 9	4, 5, 6
--, 20, 21	--, --, 10	7, 8, 9	?,?,10
22,23,24	11,12,13	7, 8, 9	?,?,10

הקצאה קבועה לא מאפשרת פעולה רציפה

- הקצה שני חוצצים לכל מַעֲלֶה
- מלא את חוצצי הקלט באופן מעגלי



עקרון החיזוי (forecasting)

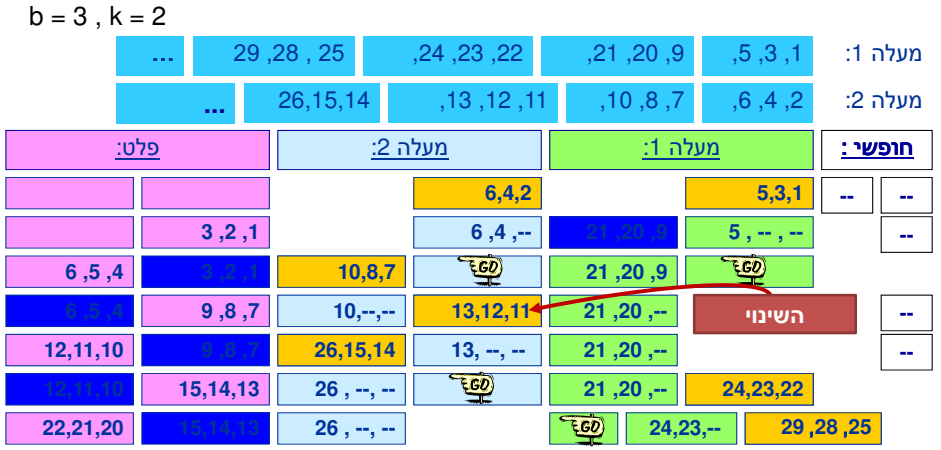
הקצאה דינמית: חוצץ יכול להשתייך למעלות שונים במהלך האלגוריתם

כאשר חוצץ מתפנה, מקצים אותו למעלֶה שעתיד להיגמר ראשון

- אם $last_i$ המפתח האחרון בין רשומות מעלה i בזיכרון
  (המפתח המקסימלי במעלה i שנמצא בזיכרון)
- נקצה חוצץ למעלֶה m המקיים $last_m = min_i \{last_i\}$

מספר החוצצים המוקצים למעלה משתנה במהלך האלגוריתם
(עד כדי הקצאת k+1 חוצצים למעלה אחד וחוצץ אחד לכל מעלה אחר)

דוגמה להקצאה ע"פ עקרון החיזוי



משפט החיזוי

עם 2(k+1) חוצצים, עקרון החיזוי מבטיח פעולה רציפה

מוכיחים באינדוקציה לכל מחזור r:

א. בתחילת המחזור: יש חוצץ קלט ריק

ב. תוך כדי המחזור: כשחוצץ נגמר, וצריך את המפתח הבא של אותו המעלה, החוצץ הבא של אותו מעלה כבר נמצא בזיכרון

ג. בסוף כל מחזור יש b(k+1) רשומות בזיכרון:
kb רשומות קלט שטרם עיבדנו (אלא אם נגמר מעלה);
b רשומות בחוצץ הפלט

הבסיס (מחזור k+1) פשוט

משפט החיזוי: צעד האינדוקציה

תחילת המחזור

נניח כי במחזור r:

א. בתחילת המחזור: יש חוצץ קלט ריק

ב. תוך כדי המחזור: כשחוצץ נגמר, וצריך את המפתח הבא של אותו המעלה, החוצץ הבא של אותו מעלה כבר נמצא בזיכרון

ג. בסוף כל מחזור יש b(k+1) רשומות בזיכרון
kb רשומות קלט שטרם עיבדנו (אלא אם נגמר מעלה);
b רשומות בחוצץ הפלט

אם כל $2k$ חוצצי הקלט לא ריקים בתחילת מחזור r+1
לכל מעלה יש לכל היותר חוצץ קלט אחד לא מלא
- k *מהם* מלאים לגמרי, עם kb רשומות
- לפחות רשומה אחת בכל-אחד מ- k חוצצי הקלט הנותרים

לפחות k+kb רשומות בחוצצי הקלט, סתירה ל-ג

משפט החיזוי: צעד האינדוקציה

ניהול חשבונות

נניח כי במחזור r:

א. בתחילת המחזור: יש חוצץ קלט ריק

ב. תוך כדי המחזור: כשחוצץ נגמר, וצריך את המפתח הבא של אותו המעלה, החוצץ הבא של אותו מעלה כבר נמצא בזיכרון

ג. בסוף כל מחזור יש b(k+1) רשומות בזיכרון
kb רשומות קלט שטרם עיבדנו (אלא אם נגמר מעלה);
b רשומות בחוצץ הפלט

נניח כי החוצץ של מעלה i הסתיים והחוצץ הבא לא מוכן
אם לכל מעלה יש חוצץ אחד בלבד
- לכל שאר המעלות יש לכל היותר b(k-1) רשומות
- מתחילת המחזור עברו < b רשומות לפלט

היו לכל היותר kb > (k–1)b +(b–1) רשומות בחוצצי הקלט סתירה ל-ג

משפט החיזוי: צעד האינדוקציה

הטיעון המרכזי

נניח כי במחזור r:

א. בתחילת המחזור: יש חוצץ קלט ריק

ב. תוך כדי המחזור: כשחוצץ נגמר, וצריך את המפתח הבא של אותו המעלה, החוצץ הבא של אותו מעלה כבר נמצא בזיכרון

ג. בסוף כל מחזור יש b(k+1) רשומות בזיכרון
kb רשומות קלט שטרם עיבדנו (אלא אם נגמר מעלה);
b רשומות בחוצץ הפלט

נניח כי החוצץ של מעלה i הסתיים והחוצץ הבא לא מוכן
אם יש מעלה t עם יותר מחוצץ קלט אחד
יהי a המפתח האחרון של מעלה i
ו-b המפתח האחרון של החוצץ הראשון של מעלה t
$a<b$, כי a כבר יצא לפלט ו-b עדיין בזיכרון

למעלה t הוקצה חוצץ לפני מעלה i סתירה לעיקרון החיזוי

משפט החיזוי: צעד האינדוקציה

הכנה למחזור הבא

נניח כי במחזור r:

א. בתחילת המחזור: יש חוצץ קלט ריק

ב. תוך כדי המחזור: כשחוצץ נגמר, וצריך את המפתח הבא של אותו המעלה, החוצץ הבא של אותו מעלה כבר נמצא בזיכרון

ג. בסוף כל מחזור יש b(k+1) רשומות בזיכרון
kb רשומות קלט שטרם עיבדנו (אלא אם נגמר מעלה)
b רשומות בחוצץ הפלט

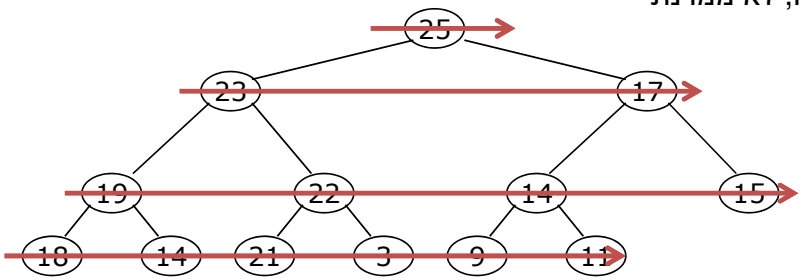
בגלל א' וב' המחזור ה-r עבר ללא תקלות, ובמהלכו:
b רשומות עברו מחוצצי הקלט לחוצצי הפלט
b רשומות נקראו מהקלט
b רשומות נכתבו לפלט

דוגמה נוספת: Heap Sort

- בונים ערימה (heap), שבה המפתח של צומת גדול מהמפתחות של ילדיו
- מוציאים מהערימה את האיבר בשורש (עם המפתח המקסימלי), ושמים במקום הפנוי הגדול ביותר במערך
  מתקבל מבנה של 'כמעט ערימה' הקטן בגודלו באיבר אחד
- את האיבר האחרון במערך שמים במקום השורש שהוצא.
- מבצעים פעולת Heapify אשר מתקנת את הערימה
  פעולה זו מחזורית מהשורש ומטה
- חוזרים שוב על תהליך הוצאת השורש עד אשר הערימה מכילה איבר בודד והמערך מכיל את כל המפתחות, ממוינים

דוגמה: מצב התחלתי של ערימה

ערימה, לא ממוינת



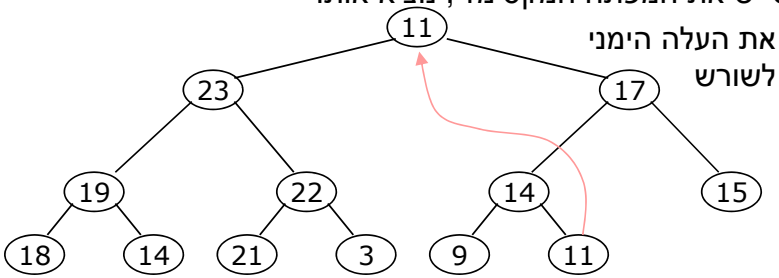
25	23	17	19	22	14	15	18	14	21	3	9	11

- הילד השמאלי של צומת i נמצא במקום $2*i+1$
- הילד הימני של צומת i נמצא במקום $2*i+2$

דוגמה: הוצאת השורש

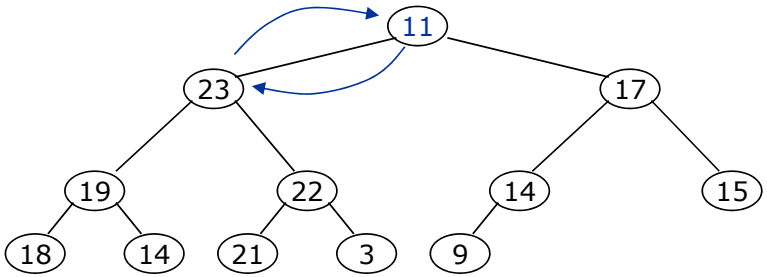
לשורש יש את המפתח המקסימלי, נוציא אותו

נעביר את העלה הימני ביותר לשורש



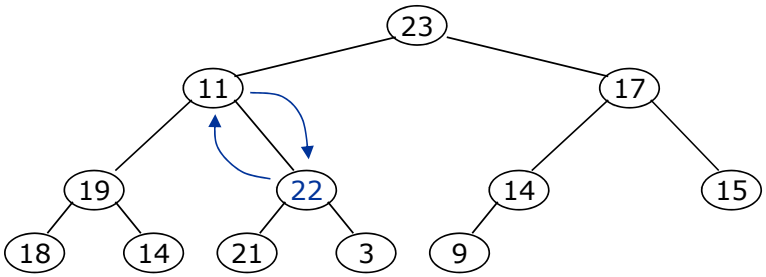
11	23	17	19	22	14	15	18	14	21	3	9	11

דוגמה: תיקון הערימה



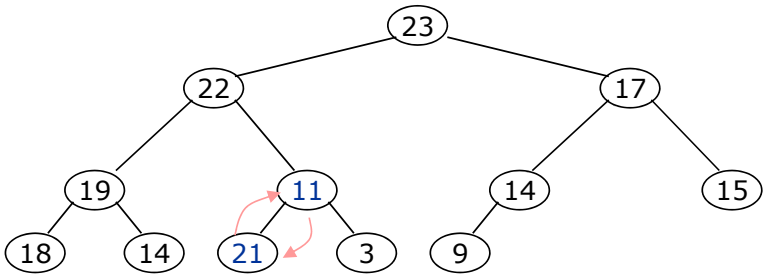
11	11	17	19	22	14	15	18	14	21	3	9

דוגמה: תיקון הערימה



23	11	17	19	11	14	15	18	14	21	3	9

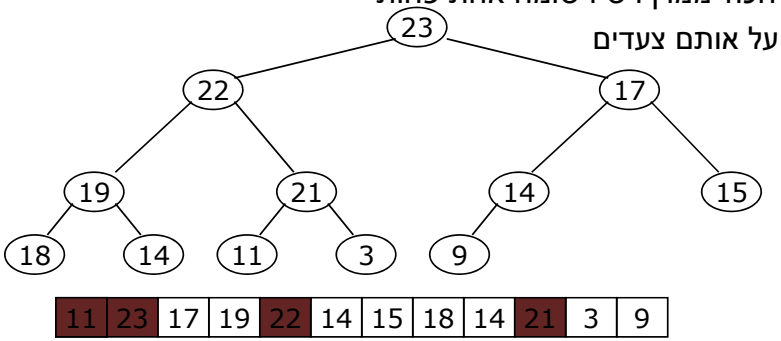
דוגמה: תיקון הערימה



23	22	17	19	11	14	15	18	14	11	3	9

דוגמה: המשך המיון

עכשיו הכול ממוין ויש רשומה אחת פחות
נחזור על אותם צעדים

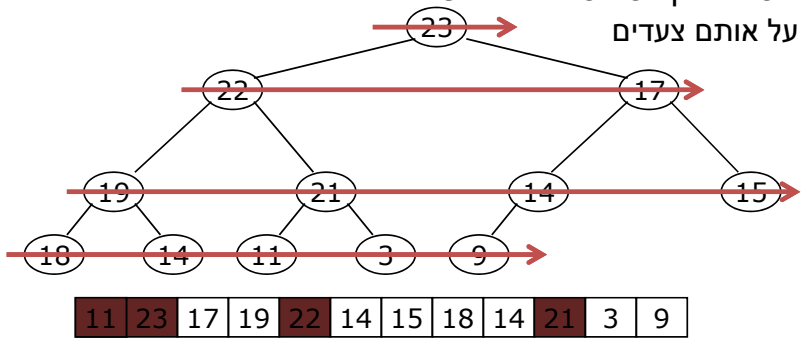


11	23	17	19	22	14	15	18	14	21	3	9

הגישות למערך מאוד לא רציפות!

כמה בלוקים מביאים?

עכשיו הכול ממוין ויש רשומה אחת פחות
נחזור על אותם צעדים



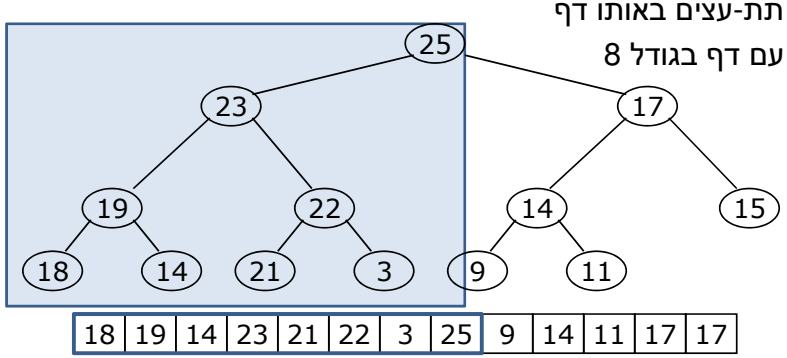
11	23	17	19	22	14	15	18	14	21	3	9

הגישות למערך מאוד לא רציפות!
בגלל אופן המיפוי של העץ למערך

N גישות ⇦ $\log_2 N$ בלוקים

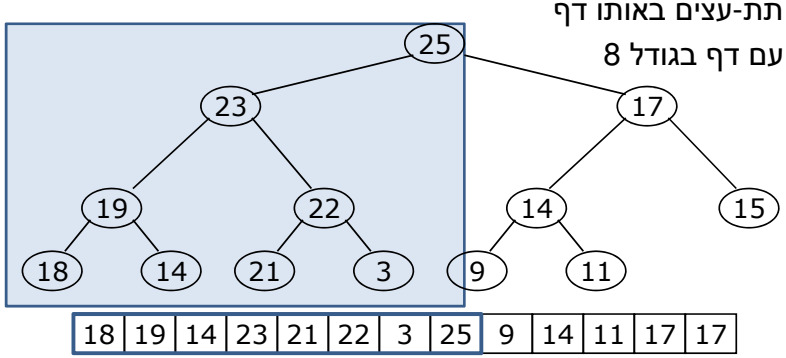
Heap sort: מיפוי אחר

לשמור תת-עצים באותו דף
למשל, עם דף בגודל 8



Heap sort: מיפוי אחר

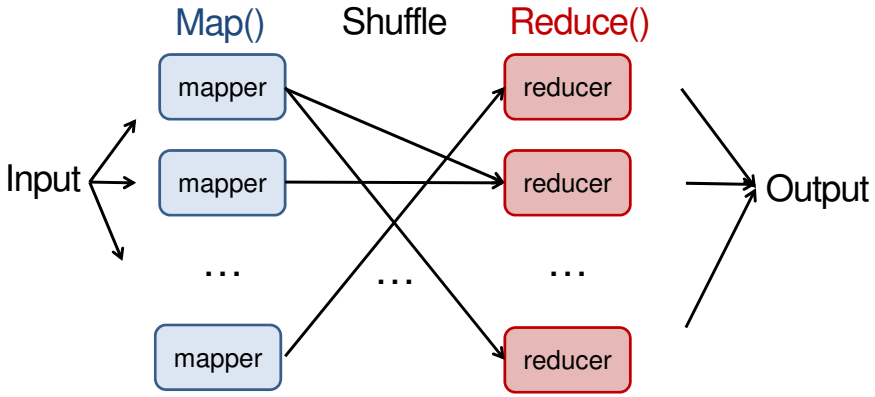
לשמור תת-עצים באותו דף
למשל, עם דף בגודל 8



סדרת הגישות הקודמת גורמת להבאת בלוק אחד
ובכללי, אם בכל מעבר מביאים כל בלוק ≥ פעם אחת ⇦ $Mog_2\ N/b$ בלוקים

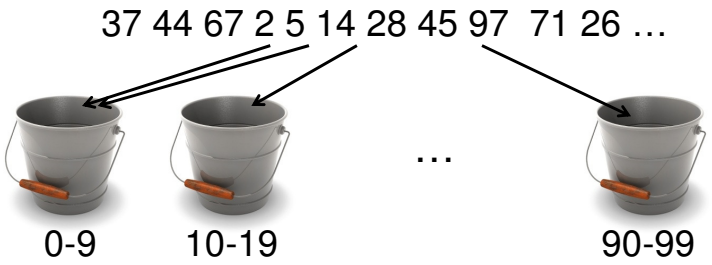
MapReduce מיון בעזרת

- תזכורת: שאילתת MapReduce מוגדרת בשני שלבים הניתנים למיקבול



תזכורת: מיון בדליים (Bucket Sort)

- הנחה: תחום המפתחות למיון וההתפלגות ידועים
- מגדירים תת תחום לכל "דלי" וכותבים אליו את כל הערכים המתאימים



- ממיינים **ביעילות** כל דלי בנפרד
- מוציאים לפלט את תוכן הדליים לפי הסדר

מיון בעזרת MapReduce

- ננצל את העובדה שמובטח שביצוע ה- Reduce הוא על קלט ממויין
  - כלומר, המערכת דואגת למיין את תוצאות ה- Map

- Map() ו- Reduce() הן פונקציות הזהות (identity)

- Map (k1, value) → (value, NULL)
- Reduce (value, NULL) → list(value)

- "אוספים" את תוצאות ה-Reduce() לפי הסדר
- ואיפה החוכמה?

Partition/Shuffle

- נניח ש-R מכונות מבצעות Reduce
- דוגמים N ערכים מתוך הקלט
- ממיינים בזיכרון: $\{Sample_0,\ldots,Sample_{N-1}\}$
- בוחרים R-1 ערכים מתוך הדגימות כך ש- $Reducer_i$ מקבל את הערכים
- $Sample'_{i-1} < value \leq Sample'_i$

- הקלט של כל Reducer ממויין במכונה נפרדת
- אוספים את הקלט הסופי לפי הסדר

דוגמא

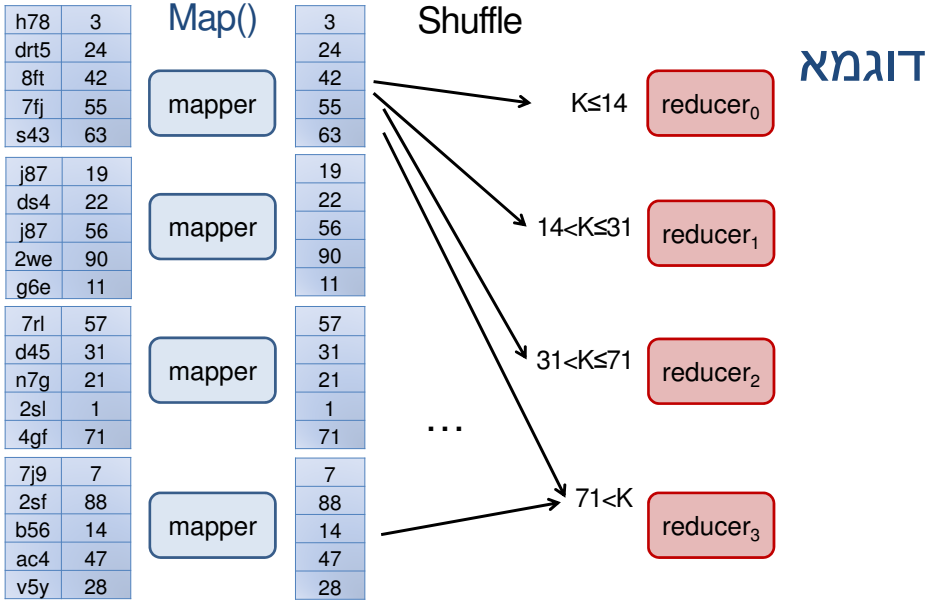
Key	Value
h78	3
drt5	24
8ft	42
7fj	55
s43	63
j87	19
...	
v5y	28

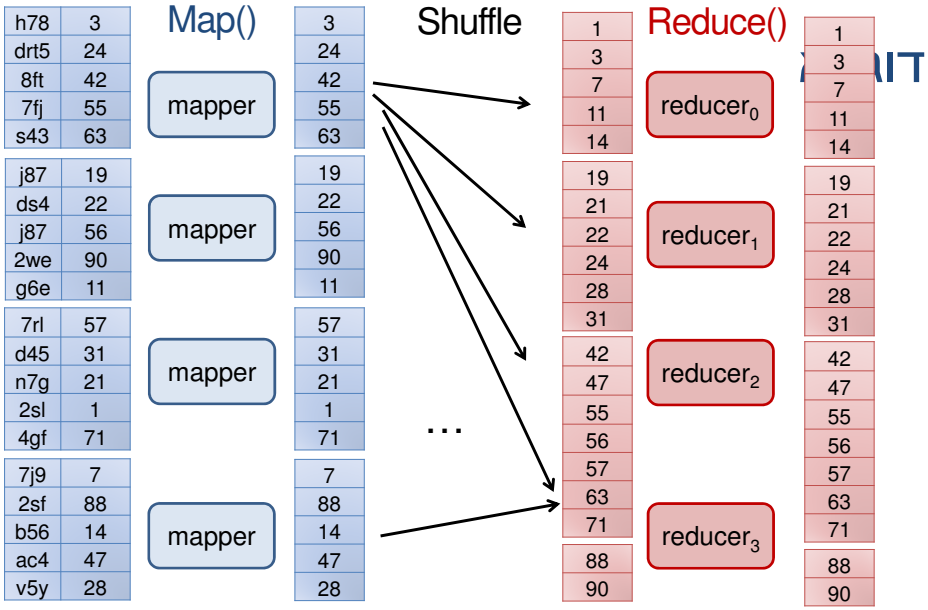
- רוצים למיין קובץ בן 20 רשומות בעזרת 4 מכונות
- R=4
- Sample: דוגמים 6 ערכים
- ממיינים את הדגימות
- בוחרים R-1=3 מפתחות לחלוקה

11 **14** 19 **31** 42 **71**

3 24 **42** 55 63 **19** 22 56 **11** 90 57 **31** 21 1 **71** 7 88 **14** 47 28

דוגמא





מיון בעזרת MapReduce: סיכום

- הנחה: ניתן לדגום מספיק ערכים כדי לקרב את התפלגות הקלט
  - במימוש המקורי: 10^5 דגימות ל- 10^9 שורות קלט
  - מה ה"קנס" אם הדגימה לא טובה מספיק?
- ה-Reduce() מתחיל רק לאחר שה-Map() הסתיים
- המיון בכל Reducer מתבצע ביעילות על דיסקים נפרדים

← R מיונים בלתי תלויים במקביל!

← זמן המיון הכולל כמשך המיון המקומי האטי ביותר (בתוספת תקורת המערכת)