ניתוח החלק השני בפרוייקט שאילתות

מגישים:

אלכס לוינזון 308636141

הלל דודיאן 318593720

מבנה האינדקס, בחלק השני של הפרוייקט, נותר זהה לחלוטין למבנה האינדקס בחלק הראשון, ולכן גודלו נשאר זהה, וכן מספר הקבצים שהוא יוצר.

השינוי שבוצע בחלק זה הוא בתהליך יצירת האינדקס, שהפך להרבה יותר יעיל מבחינת זמן ריצה.

<u>תהליך יצירת האינדקס</u>

בשלב הראשון, עוברים על כל הסקירות, עבור כל סקירה שומרים את הmeta data בקובץ **parsed_reviews.** כאשר מגיעים לטקסט של הסקירה, מוסיפים את כל מילות הסקירה לרשימה הכללית שמחזיקה את כל המילים של כל הסקירות. בכל פעם שמגיעים ל200 אלף סקירות נמחק כפילויות מרשימה זו.
כאשר סיימנו לעבור על כל הסקירות נמחק כפילויות בפעם האחרונה ונמיין.
לאחר מכן, נכתוב את כל הרשימה של כל המילים למחרוזת אחת גדולה, ובמקביל נשמור את מיקומי כל תחילת מילה ברשימה נפרדת.
את רשימת כל המילים (זו ששמרנו לפני המחרוזת הארוכה) נמחק כי היא כבר לא רלוונטית.
בשלב השני, נעבור שוב על קובץ הסקירות, ועבור כל מילה של כל סקירה נייצר זוג בפורמט הבא:
(מיקום המילה במילון, מספר המסמך שבו המילה נמצאת).

כאשר גודל הרשימה עובר את גודל הבאפר הקבוע במערכת, נכתוב את הרשימה שנוצרה בצורה בינארית לקובץ, נמשיך בתהליך זה עד שנעבור על כל הסקירות.

במידה וגודל הרשימה הכללי שנכתב לקובץ אינו חזקה של 2, נשלים אפסים עד המספר הקרוב שהוא חזקה של 2.

בשלב השלישי, נרצה למיין את הזוגות ששמרנו בקובץ. נעזר באלגוריתם merge sort כאשר יישמנו אותו לפי אותם עקרונות בדיוק שלמדנו בכיתה.

בשלב הרביעי והאחרון, עברנו על הרשימה הממוינת ויצרנו רשימות תפוצה עבור כל מילה.
כלומר קראנו את הקובץ הממוין של הזוגות:
(מיקום המילה במילון, מספר המסמך שבו המילה נמצאת)
התעלמנו מכל האפסים, ומהקובץ הזה יצרנו קובץ שמחזיק את רשימות התפוצה של כל המילים במילון, כלומר קובץ של הזוגות, בפורמט הבא:
(מספר המסמך שבו המילה נמצאת, השכיחות שלה במסמך זה)
ואותו קידדנו לפי שיטת הקידוד שפרטנו עליה בניתוח לתרגיל הראשון, בהסבר על הקובץ **doc_freq.bin**.

שאר התהליך יצירת האינדקס נותר זהה לחלק הראשון של הפרויקט כולל כל הפונקציות של ה-**IndexReader** שנותרו ללא שינוי.

ניתוח זמני ריצה וגודל האינדקס

זמני ריצה

המחשב שבו השתמשנו לצורך ביצוע הבדיקות הינו מחשב עם מפרט קצה:

CPU: i7 8700K
RAM: 32GB DDR4
Storage: 275GB SSD Ultra-Fast
OS: Windows 10 Professional

הערה חשובה: על מנת שהתוכנה תעבוד בצורה אופטימלית מבחינת הגדרות הבאפר, חובה להריץ את התוכנה על מחשב שבו יש לפחות 16 ג'יגה בייט של זיכרון עבודה (RAM). במידה ומריצים את התוכנה על מחשב שבו יש פחות זיכרון עבודה, חובה לשנות את גודל ה buffer ל 4194304. הסבר על איפה בדיוק לשנות ניתן למצוא בreadme אך נציין זאת גם כאן:

1. **בקובץ IndexMergeSort בשורה 4, להגדיר BUFFER_SIZE = 4194304**
2. **בקובץ IndexWriter בשורה 11, להגדיר BUFFER_SIZE = 4194304**
3. **בקובץ ReviewsParsing בשורה 8, להגדיר BUFFER_SIZE = 4194304**

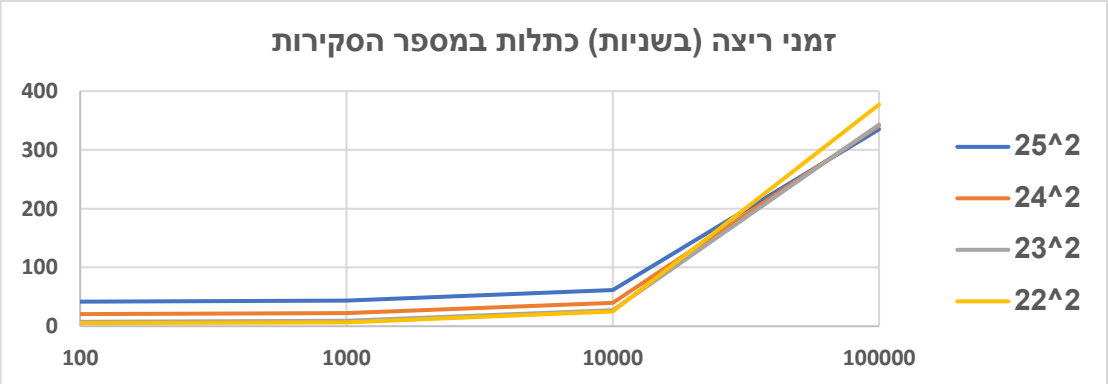
כמו כן נציין שלא הוגדר על ידי המרצה על איזו מערכת לפתח או להריץ את התוכנה ולכן, מתוך שיקולים הנדסיים ומהניסיון שצברנו, העדפנו להשאיר את גודל ה buffer על 8388608 ולחייב את המשתמש להריץ את התוכנה על מחשב עם 16 ג'יגה ראם.

באפר מעל 2^25 לא אפשר יצירת אינדקס על מיליון סקירות כיוון שהוא מאפשר אחסון של מידע רב מידי ב RAM.

באפר קטן מ- 2^22 גרם לזמני ריצה איטיים עבור יותר מ10 אלף סקירות
לכן זה הטווח האפשרי לבחירת באפר. עבור המחשב שעליו ביצענו את הבדיקות, קבענו את הבאפר לגודל **2^23 לאחר ביצוע מספר של בדיקות שבהן הגענו למסקנה שזמני הריצה עם הגדרה זו הינם אופטימליים. גודל הבאפר = מספר האלמנטים שנשמרים ברשימה (list).** סוג האלמנט שנשמר ברשימה לא משנה. עם באפר זה בוצעו כלל פעולות הקריאה והכתיבה מ/לקבצים (I/O).

להלן טבלת זמן ההרצה של בניית האינדקס על גודל באפר משתנה ומספר סקירות משתנה. הנתונים רשומים בשניות. גם במחשב העל שבו השתמשנו לצורך ביצוע הבדיקות, זמן הריצה של 10 מיליון סקירות לא היה סביר ולכן לא חיכינו עד שיסיים. לכן לא כללנו עבורו את כלל התוצאות.

#Reviews / Buffer size	2^22	2^23	2^24	2^25
100	5.27	7.5	20.6	41.8
1,000	6.81	9	22.2	43.6
10,000	24.91	27	39.9	61.5
100,000	377	343	341	335
1,000,000	--	--	--	17700
10,000,000	--	--	--	--

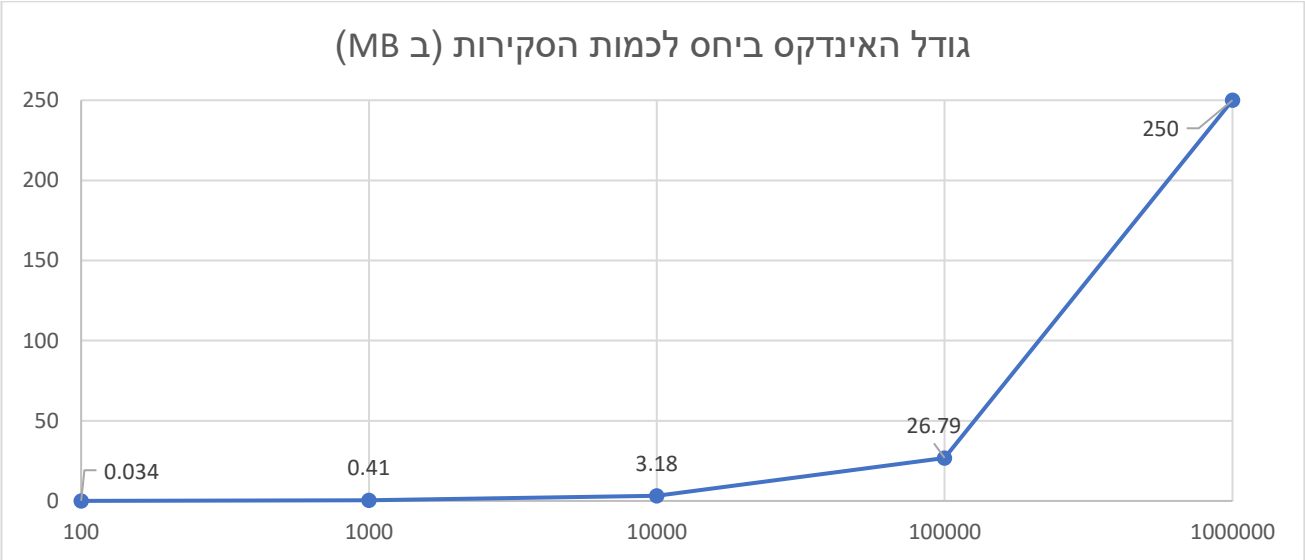


עבור מיליון סקירות לקח 12 דקות המעבר הראשון על ה Raw Data – כלומר יצירת המילון.
ההספק עבור המעבר השני על ה raw data הוא כ200 אלף סקירות כל חמש דקות
לכן כל המעבר על הסקירות בפעם הראשונה וגם השנייה – לוקח 25 דקות.

גודל האינדקס

Num of reviews	Index Size
100	35.5KB
1,000	420KB
10,000	3.18MB
100,000	26.8MB
1,000,000	250MB

לפי הטבלה ניתן לראות שאכן היחס נשמר בצורה דיי לינארית ביחס לכמות הסקירות. חשוב רק לזכור שמספר המילים השונות אינו בהכרח משתנה בצורה לינארית ביחס לכמות הסקירות. התוצאות שקבלנו מתאימות לציפיותינו.



ניתוח זמני הרצה של שאילתות על 100 מילים אקראיות:

line registers fats straps recoveries alcoholism poisons replacements selection dynamics motions intensity speed specialist sectors interests acid appropriations cab warship vine art rattles phases propeller diagram thyristor stator pin weld explanation transmissions terminology copy drives discipline manual fighting languages airship scratches decibels books whirl chapters steps cages lots nineties thins extensions base stems streaks sort category weights pieces diamond wells destination credits stations fluid miss stick compilers resolution breasts diameter tries tender centimeters calculations navigator try october seams profits religion churns ignition fasteners transmitters bend processors beans presumption address slopes insulation piece decay percent chips tractors development amplitude oscillators camp

פונקציית getTokenFrequency

Num of reviews	Time in seconds
100	0.001
1,000	0.001
10,000	0.001
100,000	0.001
1,000,000	0.001

פונקציית getReviewsWithToken

Num of reviews	Time in seconds
100	0.001
1,000	0.006
10,000	0.027
100,000	0.229
1,000,000	1.508