עבודה 4 – תיאורטי

אחיעד שייקר וניצן גואטה

1. נתוני BTree הם :

בעל n צמתים, כמות הבלוקים בכל צומת היא D*(2t-1) כאשר D הוא גודל הבלוק, ובנוסף מספר הילדים בכל צומת הוא 2t, כלומר 2t מצבעים.

נתונים ה MBT הם :

בעל n צמתים, כמות המידע הנשמרת בכל צומת היא 20 byte, ובנוסף מספר המצביעים בכל צומת הוא 2t.

נחשב את היחס :

$$\frac{MBT}{BT} = \frac{n(20byte+2t*byte)}{n((2t-1)Dbyte+2t*byte)} = \frac{byte(20+2t)}{byte((2t-1)D+2t)}$$

2. לא, עדכון ה-BTree ישפיע בMBT רק על שינוי הצומת שאליה נכנס הבלוק ב-BTree וכל האבות הקדמונים שלו.

לצורך המימוש של אלגוריתמי ההכנסה והמחיקה, נוסיף את השדות הבאים :

1. מצביע לאב קדמון בBTree וב-MBT.
2. מצביע לצומת המקבילה מ MBT לBTree ולהפך.

נחשב את זמן ריצת SHA1 לצומת בודד (כאשר D- גודל כל בלוק, t – דרגת מינימום):

ניתן לראות כי לפי האלגוריתם SHA1 הוא לינארי בכמות הבייטים שמשפיעים עליו.

כמות הבייטים היא כפונקצייה של t*D מכיוון שישנם לכל היותר 2t-1 בלוקים שכל אחד בגודל D בייטים + 2t ילדים שגודל חתימת כל ילד היא 20 בייטים.

לכן, בסה״כ זמן ריצת SHA1 היא O(t*D).

נכסח מחדש את אלגוריתם ההכנסה:

א. נרד בBTree לכיוון העלה אליו אנחנו צריכים להכניס את הבלוק, אם במהלך הירידה פיצלנו צומת כלשהו בBTree ע״י splitChild , נעדכן בהתאם את האב הקדמון, ניצור צומת חדש ב-MBT, כזה שמקביל לאח שהתפצל, נעדכן את האב הקדמון לצומת החדש ואת המצביע מהאח שהתפצל לצומת המקביל לו ב-MBT (זה שיצרנו) נחשב במקביל את החתימות של הבן ואחיו שהתפצלו ובעזרת המצביע של ה-MBT נעדכן את החתימות בצמתים המקבילים. פונקציית splitChild פועלת בO(1), יצירת צומת חדש בMBT היא O(1) , עדכון מצביעים הוא O(1), חישוב חתימה בצומת בודד O(t*D) ועדכונהO(1) , בסך הכל לכל היותר $\log_t n$ ירידות בעץ עד שנגיע לעלה,

כלומר, לכל היותר $\log_t n$ פעמים שנבצע פיצול, לכן לכל היותר $O(\log_t n)$ עדכוני חתימה,

לכן בסה״כ $O(\log_t n * t * D)$.

ב. כאשר נמצא את הצומת בBTree אליו צריך להכניס הבלוק (העלה) נכניס אותו למקום המתאים לו (הכנסה לעלה ב-BTree, נעדכן מצביעים לילדים בהתאם), נחשב מחדש את החתימה של הצומת אליו הוכנס הבלוק (בעזרת המצביע לMBT נעבור לצומת המקביל ונעדכן את החתימה). הכנסה רגילה בBTree לעלה היא O(1), עדכון מצביעים לילדיםO(1) , חישוב חתימהO(t*D) ועדכונה O(1), לכן בסה״כ O(t*D).

ג. כעת, נעלה בכל פעם בעזרת המצביע לאב הקדמון של הצומת, נחשב בכל שלב את חתימת צומת האב מחדש (בכל עליה נעלה גם בMBT ובעזרת המצביעים נעדכן את החתימות) נעצור כשנגיע לצומת שאין לו אב קדמון (שורש העץ).
כלומר, בשלב זה, עדכנו את החתימות של כל הצמתים במסלול מהעלה ששונה עד לשורש. גובה העץ הוא $\log_t n$ לכן נעלה $O(\log_t n)$ צמתים בעץ , לכן לכל היותר נבצע $O(\log_t n * t * D)$ עדכוני חתימה, לכן בסה״כ $O(\log_t n * t * D)$.

חישוב זמן ריצה כולל:

$$\mathrm{O}(\log_t n * t * D) + O(t * D) + O(\log_t n * t * D) = O(\log_t n * t * D) = O(logn * t * D)$$

<u>ננסח מחדש את אלגוריתם המחיקה:</u>

נחלק למקרים לפי פונקציית המחיקה הרגילה של BTree :

1. **אם המפתח נמצא בצומת הנוכחי:**

א. במקרה הקל, נמחק ערך מעלה שאין בו מספר מינימלי של בלוקים, נחשב את החתימה של הצומת הזה מחדש, נעבור בעזרת המצביע לצומת המקביל בMBT ונעדכן את החתימה, נבצע זאת גם לכל צומת בBTree שנמצא במסלול מהעלה ממנו נמחק הבלוק ועד לשורש. מחיקת ערך מעלה זה O(1), חישוב חתימה מחדשO(t*D) ועדכונה בצומת O(1) , גובה העץ הוא $\log_t n$ לכן נעלה $O(\log_t n)$ צמתים בעץ , לכן לכל היותר נבצע $O(\log_t n * t * D)$ עדכוני חתימה, לכן בסה״כ $O(\log_t n * t * D)$.

ב. אחרת אם לילד השמאלי שלי אין מינימום בלוקים, נעתיק את הקודם (predecessor) במקום הערך המבוקש (בצומת הנוכחי) ונפעיל באופן רקורסיבי את המחיקה על הקודם בתת העץ בו הוא נמצא (נפעיל את הפונקציה על הילד השמאלי) .
(הרקורסיה תמשיך עד שנמחק ערך מהעלה ובסעיף א׳ – סעיף המחיקה מהעלה – אנחנו מעדכנים את כל החתימות של האבות הקדמונים ושל העלה עצמו). העתקת הקודם O(1), הפעלת הפונקציה מחדש O(1), לכן בסה״כ O(1).

ג. אחרת אם לילד השמאלי שלי יש מינימום בלוקים, אך לילד הימני יש יותר ממינימום, נעתיק את העוקב (successor) במקום הערך המבוקש (בצומת הנוכחי) ונפעיל באופן רקורסיבי את המחיקה על העוקב בתת העץ בו הוא נמצא (נפעיל את הפונקציה על הילד הימני) .
(הרקורסיה תמשיך עד שנמחק ערך מהעלה ובסעיף א׳ – סעיף המחיקה מהעלה – אנחנו מעדכנים את כל החתימות של האבות הקדמונים ושל העלה עצמו).
העתקת העוקב O(1), הפעלת הפונקציה מחדש O(1). לכן בסה״כ O(1).

ד. אחרת אם שני הילדים שלו בעלי מינימום בלוקים נבצע מיזוג, בעץ המקביל אנו מוחקים את הצומת שמוזגה (זאת שנמחקה באלגוריתם המיזוג), נעדכן את המצביעים בהתאם ונפעיל את הפונקציה על הצומת הממוזג באופן רקורסיבי.
(הרקורסיה תמשיך עד שנמחק ערך מהעלה ובסעיף א׳ – סעיף המחיקה מהעלה – אנחנו מעדכנים את כל החתימות של האבות הקדמונים ושל העלה עצמו). ביצוע מיזוג הוא O(1), עדכון מצביעים ומחיקת צומת מהMBT (זאת שהתאחדה בעץ BTree) זה O(1). הפעלת הפונקציה מחדש O(1). לכן בסה״כ O(1).

2. **אם המפתח לא נמצא בצומת הנוכחי:**

א. נבדוק באיזה מן הבנים של הצומת הערך צפוי להיות, אם הבן המבוקש בעל יותר ממינימום בלוקים, נפעיל את הפונקציה עליו בצורה רקורסיבית.
(הרקורסיה תמשיך עד שנמחק ערך מהעלה ובסעיף א׳ – סעיף המחיקה מהעלה – אנחנו מעדכנים את כל החתימות של האבות הקדמונים ושל העלה עצמו). בדיקת כמות הבלוקים בבן המבוקש O(1), הפעלת הפונקציה מחדש O(1), לכן בסה״כ O(1).

ב. אחרת (אם לבן המבוקש יש מינימום בלוקים) נבדוק באיזה מן הבנים של הצומת הערך צפוי להיות, אם הבן המבוקש בעל מינימום איברים, נבצע shiftOrMerge ונפעיל את הפונקציה על הצומת שבו צפוי להיות הערך. אם ביצענו shift נחשב את החתימות של הצומת ואחיו ששונה, נעבור בעזרת מצביע הMBT לצומת המקביל ונעדכן את החתימות ששונו בהתאם, ונמשיך ברקורסיה (אחרת נמשיך ברקורסיה מבלי לחשב שום חתימה). (הרקורסיה תמשיך עד שנמחק ערך מהעלה ובסעיף א׳ – סעיף המחיקה מהעלה – אנחנו מעדכנים את כל החתימות של האבות הקדמונים ושל העלה עצמו).

בדיקת כמות הבלוקים בבן המבוקש O(1),

ביצוע של merge זה O(1), ביצוע של shift זה O(1) כמו באלגוריתם שנלמד בכיתה,

עדכון חתימות בביצוע shift זה O(2*t*D) = O(t*D) , הפעלה מחדש של הפונקציה O(1).

לכן בסה״כ O(t*D).

חישוב זמן ריצה כולל: $O(\log_t n * t * D) + 4 * O(1) + O(t * D) = O(logn * t * D)$

סיבוכיות המקום: קיימים 2 עצים שהינם כצורת BTree,

לכן כעת קיימים O(n+n) = O(2n) צמתים. בנוסף, לכל צומת נוספו 2 מצביעים, אחד לאב קדמון, והשני לצומת מקביל בעץ מקביל, בסה״כ מצביעים שנוספו זה O(n+n) = O(2n).

בסך הכל סיבוכיות המקום תהיה O(4n) = O(n) .

3. הסיבה שמתכנני SHA1 בחרו לאתחל את המשתנים לערכים קבועים ולא בערכים אקראיים המוגרלים מחדש בכל הפעלה של הפונקציה היא מהסיבה שהפונקציה משתמשת בערכים של הקוד כדי לציור ערכים חדשים ע״י שימוש בערכים אלו, אם נשתמש בערכים אקראיים לא נוכל להבטיח שכאשר נריץ את הקוד לחתימה לצורך בדיקת אימות נקבל את אותם הערכים. הפונקציה לא הפיכה ויוצרת חתימה חד-חד ערכית לכל קובץ עליו תופעל, לכן, לא קיים חשש לשימוש חוזר במשתנים הנתונים כי פלט הפונקציה יהיה זהה רק עבור קבצים זהים.

הסיבה שמתכנני SHA1 בחרו לאתחל את המשתנים בריש גלי היא בכדי להראות שאין למתכנני האלגוריתם backdoor להצפנה, כלומר, חמשת המספרים שפורסמו הם מקבוצת מספרים הנקראת Nothing up my sleeve number , שזו למעשה קבוצת מספרים שאין להם תכונות מיוחדות עבור מפתחי האלגוריתם, לכן הם לא מוסתרים בפני המשתמשים. למעשה, אין למפתחים דרך לשחזר את פעולת ההצפנה כדי למצוא מה היה הפלט לפני ההצפנה ולכן פרסמו את המשתנים בריש גלי. במידה והיו בוחרים במספרים בעלי תכונות מיוחדות, קיימת סכנה של פרצת אבטחה המאפשרת גישה למידע חסוי ללא צורך באימות זהות.