מבני נתונים – משימה 4

מאור אסייג 318550746

רפאל שטרית 204654891

המחלקה להנדסת חשמל ומחשבים, התכנית להנדסת מחשבים

מבני נתונים 202.1.1011

תשובות – חלק א'

1. יחס זיכרון

נתון B-Tree עם דרגה מינימלית t, המכיל n צמתים מלאים. בכל צומת מאוחסנים בלוקים, כל בלוק בגודל D. חשבו את היחס בין המקום הדרוש לאחסון ה B-Tree והמקום הדרוש לאחסון ה Merkle-B-Tree הנגזר ממנו.

- **גודל זיכרון B-Tree :** n צמתים, לכל צומת 2t-1 בלוקים, גודל כל בלוק D. בעץ ישנם n מצביעים לצמתים (לכל צומת מצביע, ומצביע באובייקט העץ ל שורש) **ובסה"כ :**

$B - Tree\ Memory : [n * (2t - 1) * D\ +\ n]\ byts.$

- **גודל זיכרון Merkle-B-Tree :** n צמתים, לכל צומת חתימת SHA1, גודל כל חתימה 20 בתים. בעץ ישנם n מצביעים לצמתים (לכל צומת מצביע, ומצביע באובייקט העץ ל שורש) ובסה"כ :

$Merkle\ B - Tree\ Memory : [n * 20\ +\ n] = 21 * n\ bytes.$

- **היחס** הינו : $\boxed{\dfrac{B{-}Tree\ size}{MBT\ size} = \dfrac{(2t-1)*D+1}{21}\ [bytes]}$

2. עדכון MBT בפעולות B-Tree

נתונים $B-Tree$ ולצידו MBT שנגזר ממנו. המטרה היא לשמור חתימה עדכנית עבור ה B-Tree בכל רגע נתון. האם עדכון ה $B-Tree$ ע"י הכנת בלוק חדש או מחיקת בלוק קיים מחייב עדכון כל צמתי ה MBT ?

לא, ננסח מחדש את אלגוריתמי ההכנסה והמחיקה, תוך כדי פירוט השינוי בפונקציות העזר שהוצגו בהרצאה.

שדות חדשים לצומת B-Tree וה MBT

- $Boolean\ change = false$ יעזור לנו לדעת האם נדרש שינוי בחתימת הצומת המקבילית בMBT.
- לאובייקט עץ $B-Tree$ יש מצביע גם לשורש של עץ ה MBT.

הכנסה $Insert(T, k)$

1. אם השורש מלא (2t – 1 מפתחות) :
   - 1.1 **B-Tree :** צור צומת חדשה שתהיה השורש החדש של העץ.
     MBT : צור צומת חדשה שתהיה השורש החדש של העץ.
   - 1.2 עבור 2 העצים נגדיר את הצומת הישן כבן היחידי של השורש החדש.
   - 1.3 נקרא ל $SplitChild(BTree\ root, MBT\ root, 1)$ פונקציה זו עברה שינוי ונציג אותה בהמשך.
   - 1.4 קרא ל $InsertNonFullN(BTree\ root, k)$
2. אחרת :
   - 2.1 קרא ל $InsertNonFullN(BTree\ root, k)$
3. קרא ל $UpdateMBT(MBT\ root)$

ניתוח זמן ריצה : $O(t)$ $SplitChild$ במקרה הגרוע תיקרא כמספר הרמות $log_t n$ זמן הריצה של $InsertNonFullN$ $O(tlog_t n)$ הינו ולכן **בסה"כ :** $T(n) = O(tlog_t n)$.

סיבוכיות מקום : בהכנסה אנו ניגשים במקרה הגרוע בכל רמה לצומת אחת, מכיוון שהדבר נעשה באופן מקבילי ב MBT נקבל בסה"כ : $DISK\ Access :\ O(h) = O(\log_t n)$.

פיצול $SplitChild(node\ B, node\ M, index)$

1. **B-Tree :** צור צומת חדשה $BNew$ וקבע את מספר המפתחות שלה ל $t-1$.
   MBT : צור צומת חדשה $MNew$.
2. **B-Tree :** העתק ל $BNew$ את $t-1$ המפתחות של הבן הראשון מהצומת B החל מאינדקס $index$.
3. נסמן את הבן הראשון של B כ y. אם y לא עלה :
   3.1 **B-Tree :** העתק t מצביעים החל מאינדקס $index$ אל $BNew$.
   MBT : העתק t מצביעים החל מאינדקס $index$ אל $MNew$.
4. **B-Tree :** הוסף ל B מצביע ל$BNew$ באינדקס ה $index+1$, תוך כדי הזזה שמאלה של שאר המצביעים ($shift\ left$).
   MBT : הוסף ל M מצביע ל$MNew$ באינדקס ה $i+1$, תוך כדי הזזה שמאלה של שאר המצביעים ($shift\ left$).
5. **B-Tree :** בצע ב B הזזה שמאלה ($shift\ left$) של האינדקסים החל מהסוף ועד אינדקס $index$.
6. בצע :
   6.1 העבר מ y את המפתח באינדקס t אל B באינדקס $index$.
   6.2 עדכן ב 1 את מספר המפתחות ב B.

6.3 הפחת ב y את מספר המפתחות ב $t-1$.

7. עדכן את השדה $change = true$ בצמתים $B, Bnew, y$.

ניתוח זמן ריצה : במקרה הגרוע נצטרך להעביר סדר גודל של $O(t)$ איברים בין צמתים, ולכן **בסה"כ** : $T(n) = O(t)$.

סיבוכיות מקום : בפיצול אנו ניגשים ל 3 צמתים ולכן **בסה"כ** : $O(3) = O(1)$: $DISK\ Access$.

פונקציית עזר	$InsertNonFullN(node\ B, node\ k)$

1. **B-Tree :** עדכן את השדה $change = true$ בצומת B.
2. **קרא ל** $InsertNonFull(node\ B, node\ k)$ שהוצגה בהרצאה.

ניתוח זמן ריצה : במקרה הגרוע נצטרך להגיע לכל הרמות בעץ ו להעביר סדר גודל של $O(t)$ איברים בין צמתים, ולכן **בסה"כ** : $T(n) = O(tlog_t n)$.

סיבוכיות מקום : בהכנסה במקרה הגרוע נבקר בכל הצמתים בעץ וניגש ל $O(3)$ צמתים ולכן **בסה"כ** : $O(h) = O(\log_t n)$: $DISK\ Access$.

פונקציית עזר $UpdateMBT(node\ M)$

1. **MBT** : אם $M.change\ =\ true$ וגם M לא עלה :
   1.1 עבור כל אחד מהמצביעים בדוק האם הצומת המשויכת למצביע עם $change = true$.
   1.1.1 קרא ל $UpdateMbt(M.C_i)$.
   1.2 עדכן את החותמת של הצומת $M.key = hash(Node)$.
2. אחרת אם אם $M.change\ =\ true$ וגם M עלה :
   2.1 עדכן את החותמת של הצומת $M.key = hash(LeafNode)$.

ניתוח זמן ריצה : נעבור על כל הצמתים בעץ שעבורם $change\ = true$. מאלגוריתמי ההכנסה ישנם במקרה הגרוע $log_t n$ צמתים כאלו, עבור כל צומת (נניח שהיא לא עלה) נבדוק גם אם לבנים שלה $change\ =\ true$ וגם נחשב מחדש את החתימה ולכן **בסה"כ** :

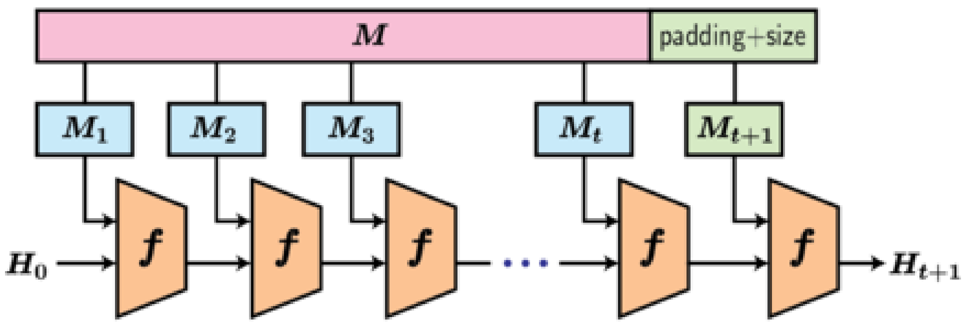
$$T(n) = (2t-1)O(log_t n) + O(log_t n) * O(HashFunction) =$$
$$= O(\max[Hash, t] * \log_t n).$$

סיבוכיות מקום : נעבור על כל הצמתים בעץ שעבורם $change\ = true$. מאלגוריתמי ההכנסה ישנם במקרה הגרוע $log_t n$ צמתים כאלו, ולכן **בסה"כ** :

$DISK\ Access:\ O(h)\ =\ O(\log_t n)$.

3. מה הסיבה שמתכנני *SHA1* בחרו לאתחל את המשתנים לערכים קבועים ובריש גלי, ולא בערכים אקראיים המוגרלים מחדש בכל הפעלה של הפונקציה ?

תשובה : *SHA-1* עובד לפי שיטת צופן-זרם (בשונה משיטת צופן-בלוקים) הערכים המאותחלים משורשרים לבלוק הראשון ומכניסים את השרשור לפונקציית התמצות, ובכל שלב פונקציית התמצות מקבלת שרשור של הבלוק עם הפלט של פונקציית התמצות הקודמת.
אורך הערכים נבחרו בצורה כזו שידמו את אורך הפלט של פונקציית התמצות.
אוסף המספרים הללו לא נבחרו באופן מיוחד (אף על פי שלכל אחד מהמספרים תכונות מתמטיות שלא משפיעות על התהליך) וערכים אלו נשארים קבועים כדי שנוכל להבטיח פלט של פונקציית תמצות התחלתית (עבור השלב הראשון) קבועה.



שיטת מרקל דמגרד, המיושמת ב *SHA-1*.

Until next time, thank you.