<u>מבני נתונים עבודה 4 – החלק התיאורטי:</u>

1. נתון B-Tree
עם דרגה מינימלית t .
המכיל n צמתים – כל הצמתים, כולל השורש, מלאים.
בצמתי העץ מאוחסנים בלוקים של קובץ – גודל כל בלוק הוא D.
נשים לב כי
$n(2t-1)D$ – זהו הגודל של כל הצמתים מכיוון שיש לנו n צמתים ו $(2t-1)$ בלוקים כאשר גודל כל בלוק הוא D.
$2tn$ - זהו הגודל של המצביעים של צמתי העץ מכיוון שיש לנו n צמתים ולכל אחד מהם יש $2t$ בנים.
$20n$- זהו הגודל של כל צמתי עץ ה Merkle-B-Tree מכיוון שיש לנו n צמתים וכל אחד מהם הוא 20 בייטים.
לכן היחס בין שני העצים יהיה:

$$\frac{n(2t-1)D+2tn}{20n+2tn}$$

2. נתונים B-Tree ולצידו MBT שנגזר ממנו כאשר מטרתנו היא לשמור חתימה עדכנית עבור ה B-Tree בכל רגע נתון. **ברור שעדכון ה B-Tree על ידי הכנסת בלוק חדש או מחיקת בלוק קיים אינו מחייב חישוב מחדש של כל צמתי ה MBT** ולכן נראה זאת על ידי עדכון האלגוריתם שכתבנו.

<u>Insert:</u> נוסיף שדה נוסף של MerkleBNode לכל אחד מהnodes של העץ, ההכנסה לעץ תהיה זהה למה שכתבנו מלבד לדבר אחד, לכל אחד מהnodes שהכנסנו ניצור גם MerkleBNode חדש ובמהלך החזרה בעץ הרקורסיה נעדכן גם את ערכי הhash בכל אחד מהMBTs שנמצא בBNode.

אין צורך לעדכן את הערכים שנמצאים בתחתית העץ מכיוון שערכי hash שלהם לא צריכים להשתנות.
זמן הריצה של האלגוריתם יהיה $O(t\log_t n)$.
סיבוכיות המקום של האלגוריתם תהיה $O(\log_t n)$.

<u>Delete:</u> מכיוון שהוספנו שדה נוסף של MerkleBNode לכל אחד מהnodes, בפונקציית ה Delete המעודכנת נבצע מחיקה של ה MerkleBNode במהלך העלייה בחזרה בעץ הרקורסיה בכל אחד מארבעת הקייסים ובנוסף נבצע עדכון לילדים כאשר הפונקציה shiftOrMergeChildIfNeeded נקראת והתנאי getChildAt(childIndx).isMinSize() מתקיים .
זמן הריצה של האלגוריתם יהיה $O(t\log_t n)$.
סיבוכיות המקום של האלגוריתם תהיה $O(\log_t n)$.

3. נשים לב כי בכל פונקציית גיבוב קריפטוגרפית המשתמשת בשיטת "מרקל דמגרד", בכל שלב, הפלט של הגיבוב לבלוק הקודם משמש כקלט לגיבוב הבלוק הנוכחי(בצורת וקטור אתחול חדש).
השיטה משתמשת בגיבוב הבלוקים עד כה על מנת ליצור גיבוב המתחשב גם בבלוקים הקודמים, דבר המתבטא בעצם בתלות בווקטור האתחול ולכן אם נבחר להשתמש בווקטור אתחול שונה נקבל גיבוב שונה עבור קלט זהה.
במקרה שלנו, המשתנים הללו משמשים לקביעת הפלט של SHA1 כווקטור אתחול.
הסיבה שבגללה לא הפריע למפתחי האלגוריתם לפרסם את הערכים ההתחלתיים היא שערכים אלו לא מספקים backdoor לאלגוריתם מכיוון שבחירת ערכים קבועים אלו היא שרירותית.
בנוסף, הפונקציה SHA1 משמשת ליצירת חתימה של B-Tree. אם נבחר משתנים לפונקציה באופן אקראי, לא ניתן יהיה להשתמש בה למטרת אימות נתונים (כפי שקורה שמורידים קבצים מהאינטרנט-מקבלים hash code שמאמת את נכונות הקובץ) , דבר שעלול להוביל לחוסר משמעות כיוון שלא תהיה עקביות.

מסיבה זאת מפתחי השיטה פרסמו את הפונקציות ברש גליי וללא ערכים אקראיים.