ב"ה סיוון ה'תשפ"ב

מגישות:

עדן איוב edeneyov 211698881

ראשית דובדבני reshitd 212075667

חלק ב' – ניסויים

3. 1. $|Q_1| = |\{i^2 \ mod \ q | \ 0 \le i < q\}| = 3286$

$|Q_2| = |\{(-1)^i \cdot i^2 \ mod \ q | 0 \le i < q\}| = 6571$

2. עבור $QPHashTable$, כל פעולות ההכנסה הושלמו בהצלחה, ללא זריקת חריגים.

עבור $AQPHashTable$, נזרקו בכל יצירה של טבלה כארבעה חריגים מסוג TableIsFullException.

השוני בין התוצאות נובע מהעובדה שכאשר הסימנים של i^2 אינם מתחלפים, כשהטבלה קרובה להתמלאות הצעדים חוזרים על עצמם (כפי שניתן לראות ממספר השאריות האפשרי בQ_1 – רק חצי ממספר המקומות בטבלה. לעומת זאת, מכיוון ש6571 הוא ראשוני ושארית החלוקה שלו ב4 היא 3, גודל הצעדים אינו חוזר על עצמו, ובסך הכל מתקבלת תמורה של כל המקומות האפשריים. כלומר, מגיעים למקום תפוס רק כאשר כל הטבלה מלאה.

3. על פי תורת המספרים, אם p ראשוני, יש לו בדיוק $\frac{p-1}{2}$ שאריות ריבועיות שונות מ0.

אכן, גם עבור $p = 6571$ נקבל $\frac{6571}{2} = 3285$. נוסיף 1 למספרים עם שארית 0 ונקבל $|Q_1| = 3285 + 1 = 3286$.

לכל p ראשוני, יש בדיוק $\frac{p-1}{2}$ שאריות שאינן ריבועיות שונות מ0. בQ_2 אנו מוסיפים על ידי ה$(-)$ עוד $\frac{p-1}{2}$ שאריות שאינן ריבועיות ומקבלים $|Q_2| = 3286 + 3285 = 6571$.

4. נשים לב שזמני הריצה מושפעים מהצטברות ראשונית - איברים שמופו לאותו תא התחלתי (התנגשויות), מהצטברות משנית – היווצרות של רצפים זהים של סדרות בדיקה, ומזמן חישוב של הפונקציה. מדרישות התרגיל, ההצטברות הראשונית זהה בכל סוגי הטבלאות.

1.

Class	***Running Time***
LPHashTable	1291279700 *nano sec*
QPHashTable	1337802100 *nano sec*
AQPHashTable	1208855700 *nano sec*
DoubleHashTable	1326754100 *nano sec*

אין הבדלים גדולים בין זמני הריצה. ניתן לראות שזמן הריצה הקצר ביותר הוא של *AQPHashTable*, שאומנם דורשת חישוב מסובך מעט יותר מ*QPHashTable* והרבה יותר מ*LPHashTable* אך היא מונעת הצטברות משנית – לא נוצרים רצפים של צעדים, אין סדרת בדיקות דומה לחברתה, ולכן ההכנסה היא מיידית.

לאחר מכן מגיע *LPHashTable*. בשיטה זו, כפי שראינו בהרצאה, נוצרים רצפים ארוכים של תאים תפוסים – נוצרת הצטברות משנית. כלומר, כשהטבלה מתחילה להתמלא, כל איבר לא מוכנס מיידית, אלא לאחר הפעלה של פונקציית הצעד שוב ושוב. עם זאת, החישוב של הפונקציה הליניארית הוא הפשוט ביותר, והדבר מפצה על ההצטברות המשנית.

החישוב של *DoubleHashTable* הוא המסובך ביותר, והדבר בא לידי ביטוי בזמן הריצה, אך בשיטה זו לא נוצרים רצפים, וכך רוב ההכנסות קורות מיידית.

זמן הריצה הארוך ביותר הוא של *QPHashTable*. החישוב של הפונקצייה הריבועית מסובך יותר מזו של הליניארית, וגם בשיטה זו נוצרים רצפים (אומנם משניים, ולא מיידיים כמו בליניארית) של תאים תפוסים, שדורשים את הפעלת פונקציית הצעד שוב ושוב.

2.

Class	***Running Time***
LPHashTable	5160794200 *nano sec*
AQPHashTable	4510178000 *nano sec*
DoubleHashTable	4849774000 *nano sec*

ההבדל בביצועים כאשר אנו ממלאים כמעט את כל הטבלה, לעומת הסעיף הקודם בו מילאנו חצי טבלה, שונה בהתאם לסוג הטבלה. ההבדל ב*LPHashTable* הוא פי 3.99 – ההבדל המשמעותי ביותר. ברור שהדבר נובע מהצטברות רצפים ארוכים של תאים תפוסים, שמתרחשת מתכונותיה של הפונקציה הליניארית, עד שבהכנסות האחרונות מתבצע חישוב של פונקציית הצעד מספר רב של פעמים – כאורך כל הטבלה.

ההבדל ב*AQPHashTable* הוא פי 3.73. סיבוכיות הפונקציה גדלה ככל שמתקדמים בסדרת הבדיקות, ומובן שכשמוכנסים יותר איברים זמן הריצה גדל.

ההבדל ב*DoubleHashTable* הוא פי 3.65 – ההבדל הקטן ביותר. פיזור האיברים יעיל, וסיבוכיות הפונקציה לכל הכנסה נשארת זהה בכל נסיון בסדרת ההכנסות, ומובן שכשמוכנסים יותר איברים זמן הריצה גדל.

5.

Iterations	***Running Time***
First 3 iterations	12728465800 *nano sec*
Last 3 iterations	49410711300 *nano sec*

קיים הבדל בין זמני הביצוע: שלוש האיטרציות האחרונות לוקחות יותר מפי שלושה זמן מהביצוע של שלוש האיטרציות הראשונות. בכל אחת מהאיטרציות, מספר המקומות שמסומנים כמחוקים בטבלת הhash גדל. לכן, מאיטרציה לאיטרציה הפונקציה צריכה לעבור על הרבה יותר מקומות בסדרת הבדיקה בהכנסה לטבלה, מה שמשתקף בהתארכות זמן הביצוע.