

תרגול נוסף של שאילתות מסוג GROUP BY , הצפנה - המשך

פתרון מפורט לשיעורי הבית, כפי שעברנו עליו בכתה:

**לכתוב שאילתה שתיתן את רשימת מספרי זיהוי הספרים (isbn) של הספרים שנישאלו על ידי שלושה סטודנטים לפחות, מהספריות שנימצאות בניו יורק.**

פיתרון

מצד אחד יש לנו את המשימה לבדוק את טבלת ההשאלות - borrow כיוון שרוצים לבדוק ספרים שהושאלו מספר פעמים (מעל 3), אבל מצד שני רוצים רק מספריות שנימצאות בניו יורק, ובטבלה זו למרבה הצער, אין לנו את הערים של הספריות, אלא רק את שמותיהן ( LNAME , ISBN, ID :השדות הם: )

ולכן על מנת קודם כל, בשלב ראשון, למצוא רק את הספרים שהושאלו מספריות ניו יורקיות, עלינו לחבר עם: join בין הטבלה borrow והטבלה: library ( שבה יש שדות: LNAME ו- CITY ), בצורה הבאה (inner join על isbn וגם שהעיר היא ניו יורק):

```
select isbn
from borrow b, library l
where l.lname=b.lname
and l.city='New York'
```

שאילתה זו מבודדת לנו את כל הספרים בטבלה: borrow (כלומר אלה שנישאלו) מספריות בניו יורק. זהו השלב הראשון. כעת המשימה בשלב השני היא "לקבץ" את הספרים שנישאלו מספר פעמים שווה או יותר מ-3, ולכן נעזר בהוראת ה- GROUP BY שלמדנו, ונפעיל אותה על הרשימה שהתקבלה מהשלב הראשון (כלומר על השאילתה שכתבנו כבר) בצורה הבאה:

```
select isbn, count(*)
from
  ( select isbn
    from borrow b, library l
    where l.lname=b.lname
    and l.city='New York'
  )
group by isbn
having count(*)>=3
```

שימו לב מה קרה כאן. הפכנו את השאילתה מהשלב הראשון, לתת-שאילתה ומיקמנו אותה בתוך שאילתת ה- group by , כלומר התייחסנו אל השאילתה מהשלב הראשון כמו לטבלה עצמאית.

מה קיבלנו בשלב השני? רשימה של מספרי ספרים ועל יד כל מספר ספר את מספר הפעמים שהוא מופיע ב- borrow (כלומר הושאל) ואלה כמובן רק מספריות ניו יורקיות.

כעת מה שנותר לעשות זה משהו פשוט. הרי בתוצאה הסופית לא מעניין אותנו כמה סטודנטים בדיוק שאלו ספרים אלה, אם זה 3 או 7 או 100, רק את מספרי הספרים, ולכן מכל הקונסטרוקציה שבנינו, אנחנו ניצור תת-שאילתה, וכעת נכתוב:

```
select isbn
from
    ( כל הקונסטרוקציה )
```

ולכן התשובה הסופית תהיה:

```
select isbn
from (
    select isbn, count(*)
    from (
        select isbn
        from borrow b, library l
        where l.lname=b.lname
    )
    group by isbn
    having count(*)>=3
)
```

שאלתה נוספת שעליה עבדנו:

לכתוב שאילה שתיתן את רשימת כל הסטודנטים ששאלו את כל הספרים ששאל הסטודנט בעל ID של '384'

חלקכם הסתבכתם עם זה. כדי לצאת מההסתבכות, אם שאלתה טיפה מורכבת, הייתי ממליץ לנתח ולפרק אותה לגורמים. לעיתי קרובות יש להתחיל מהחלק הפנימי והחוצה. במקרה זה קודם נכתוב מהם כל הספרים ששאל הסטודנט בעל ID של '384'.

```
select isbn
from borrow
where ID='384'
```

כעת נותר לנו לבחור מאותה טבלה (borrow) את רשימת הסטודנטים (ID) אבל כאשר לכל סטודנט הושאלו (לפחות) כל הספרים שהצגנו בשאלתה הפנימית. לשם כך היה עלינו לזכור את השימוש בקודה ALL משולבת עם סימן השיוויון.

```
select ID
from borrow
where isbn = ALL
(
    select isbn
    from borrow
    where ID='384'
)
```

כעת זה נראה פשוט, לא?

### הצפנה

מטרת חלק זה של ההרצאה זה לדבר על אלגוריתם פופולארי של אבטחת מידע בשם: RSA (על שם: Rivest, Shamir Adleman). לשם כך יש קודם כל להבין מספר אלמנטים מקדימים.

האלגוריתם מבוסס על שני מערכי חזקות  $e$  ו- $d$  כאשר  $e$  הוא המפתח הציבורי להצפנה (Encrypt) ו- $d$  הוא המפתח הפרטי של הצופן לפיענוח (Decrypt). אם  $P$  תהיה ההודעה (plain text) ו- $C$  יהיה הטקסט המוצפן, המשוואה שתתקיים היא:  $C = P^e \text{ mod } n$  (Ciphertext). המקבל של הטקסט המוצפן יפעיל את הפעולה הנגדית:  $P = C^d \text{ mod } n$  ובכך יוכל להבין את  $P$ .

המודולו  $n$  הוא מספר גדול שנייצר בעת בניית הצופן. תהליך ההצפנה והפיענוח נעשה בעזרת חישוב מודולרי של חזקות (כפי שעוד ניראה) שיכול להתבצע בזמן סביר (ניקרא גם זמן פולינומיאלי), אולם פירוק המודולו לגורמים דורש זמן אקספוננציאלי (לשבירת הצופן). את ההבדל אפשר לתאר כהבדל בין: העלאה בחזקה להוצאת שורש. אין עדיין אלגוריתם מהיר למציאת שורש כמו להעלאה בחזקה. את ההבדל בין זמן פולינומיאלי (Polynomial) וזמן אקספוננציאלי (Exponential) אוכל להדגים כמו ההבדל בין  $X^2$  ו- $2^x$  - תציבו למשל  $x=30$  או  $x=100$  ותיווכחו בהבדל.

פיצוח האלגוריתם דורש הוצאת שורש  $e$  של  $C$  מודולו  $n$ . (זו אריתמטיקה מודולרית שדורשת המון זמן)

במילים אחרות, כיוון שבוב יודע את  $d$  (ניקרא Trap door) הוא יכול לפענח בזמן פולינומיאלי את ההודעה. אם יום אחד מישהו ימציא אלגוריתם פולינומיאלי עבור שורש  $e$  של מספר גדול מאד - כל ההצפנות בעולם יהיו בצרה צרורה.

יש עוד אפשרות לשבור את הצופן אם ננסה לפרק לגורמים מכפלה של שני ראשוניים גדולים, אבל גם זה מצריך זמן אקספוננציאלי (ניראה זאת בהמשך).

אריתמטיקה מודולרית הופכת למאד שימושי כשמדובר בהצפנה והעיקרון פשוט. אחת הסיבות לאריתמטיקה מודולרית זה שעובדים עם מספרים גדולים מאד שעלולים לא להיכנס למחשב. ולכן העבודה עם המודולוס מקילה על החישובים.

$$a \text{ mod } b = \text{rem}$$

$\text{rem}$  זוהי השארית של החלוקה של  $a$  ב- $b$ . כמו הפעולה של  $\%$  בשפת  $C\#$ . באריתמטיקה מודולרית אנו מעוניינים אך ורק בשארית (לא במנה). **המרחב של כל השאריות בחלוקה במספר  $n$ , מכונה:  $Z$  של  $n$ .** למשל:  $Z(10) = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$  - זהו סט המספרים בין 0 ל- $n-1$ .

ה'למה' הראשונה שנראה זה ש:

$$(a*b) \text{ mod } n = [(a \text{ mod } n) * (b \text{ mod } n)] \text{ mod } n$$

לדוגמה:

$$(7*9) \text{ mod } 4 = [(7 \text{ mod } 4) * (9 \text{ mod } 4)] \text{ mod } 4 = 3$$

זו למה כמעט טריביאלית, אבל אוכיח אותה ע"מ להראות מה זו הוכחה מתמטית פשוטה (אולי זה ישכנע את חלקכם ללמוד מתמטיקה באוניברסיטה ואת האחרים לא)

תחילה ניכתוב את  $a$  ו- $b$  בעזרת כפולות של המודולוס  $n$ :  
כאשר:

$$k_1 \geq 0; 0 \leq k_2 < n; k_3 \geq 0; 0 \leq k_4 < n$$

$$a=7; b=8; n=3$$

$$7=2*3+1; 8=2*3+2 \quad (k_1=2; k_2=1; k_3=2; k_4=2)$$

$$a=6; b=3; n=3$$

$$6=2*3+0; 3=1*3+0 \quad (k_1=2; k_2=0; k_3=1; k_4=0)$$

תחילה נעבוד על הצד השמאלי של הטענה, דהיינו:  $(axb) \text{ mod } n$

$$axb = (k_1n+k_2)(k_3n+k_4) = k_1k_3n^2 + k_2k_3n + k_1k_4n + k_2k_4 = k_1k_3n^2 + (k_2k_3+k_1k_4)n + k_2k_4$$

כעת נייצג את:  $k_2k_4$  כמכפלה של המודולוס:  $k_2k_4 = k_5n + k_6$  (כיוון ש- $k_2k_4$  הוא מספר שלם זה אפשרי, למשל:  $k_2=4; k_4=5$  אז:  $k_2k_4 = 4*5 = 20 = 6*3 + 2$  - כלומר כאן  $k_5=6$  ו- $k_6=2$ )

וכעת נכתוב מחדש את :  $axb = k$  :

$$axb = k_1k_3n^2 + (k_2k_3 + k_1k_4)n + k_5n + k_6$$

כאשר מבצעים את:  $n \bmod n$  נקבל:  $axb \bmod n = k_6$  (כל היתר היו כפולות של  $n$ )

כעת נעבור לצד הימני של הטענה, דהיינו:  $[(a \bmod n)(b \bmod n)] \bmod n$   
 כיוון שהצבנו את :  $b = k_3n + k_4$  ;  $a = k_1n + k_2$ , לאחר ביצוע  $n \bmod n$  על  $a$ , נקבל:  $k_2$ , ועל  $b$  נקבל  $k_4$   
 כלומר:  $(a \bmod n)(b \bmod n) = k_2k_4$

נחזור להצבה של:  $k_2k_4 = k_5n + k_6$  ולכן אם נבצע  $n \bmod n$ , נקבל:

$$(k_2k_4) \bmod n = (k_5n + k_6) \bmod n = k_6$$

כלומר שגם בצד שמאל וגם בצד ימין קיבלנו בסוף את  $k_6$  - כלומר הם שווים.

עוד טענה שלא נוכיח:  $a^n \bmod b = (a \bmod b)^n$  לדוגמה:  $10^{30} \bmod 3 = (10 \bmod 3)^{30} = 1$   
 כללים אלה נועדו להקל על חישובים בחשבון מודולארי.

**מהם הפכים מכפליים כללית?** מספרים שאם נכפיל אותם אחד בשני, נקבל 1. למשל 4 ו-1/4.

כעת נדבר על הנושא של **הפכים מכפליים באריתמטיקת מודולו** (Modular Multiplicative inverse)  
 מהם הפכים מכפליים באריתמטיקת מודולו? אלה מספרים שאם נכפיל אותם אחד בשני וניקח את המודולו  $n$  של המכפלה, נקבל: 1 כלומר בהינתן מספר שלם  $a$  ומספר שלם  $n$  נחפש את  $b$  שהוא ההפך המכפלי של  $a$  במודולו  $n$  (כאשר גם  $a$  וגם  $b$  גדולים שווים לאחד וקטנים מ- $n$ )

לדוגמה ההפך המכפלי של 3 במודולו 11, הוא: 4, כיוון ש-  $3 \times 4 \bmod 11 = 1$  (ההפך המכפלי שנירצה הוא קטן מהמודולו, כלומר במקרה זה קטן מ-11) כלומר הוא ב  $Z$  של  $n$ .

ההפך המכפלי  $b$  של מספר  $a$  במודולו  $n$ , קיים **אם ורק אם**  $a$  ו- $n$  הם ראשוניים ביחס אחד לשני. כלומר אין להם מחלק משותף גדול מ-1. במקרה של הדוגמה היו אלה 3 ו-11. כלומר  $a$  ו- $n$  הם ראשוניים ביחס אחד לשני (ניראה זאת בהמשך)

מהם מספרים שווים (או חופפים) מודולו? (Congruent modulo). ניראה זאת בדוגמה:

$$20 \bmod 3 = 2$$

$$11 \bmod 3 = 2$$

במצב כזה נאמר ש- 20 ו-11 הם שווים מודולו 3.

זוכרים את **האלגוריתם של אוקלידס** למציאת מחלק משותף גדול ביותר של 2 מספרים? בעיקרון חלוקה של מודולו עד שמגיעים ל-0 וה- GCD הוא מה שקיבלנו לפני האפס. השתמשנו רק בשאריות.

למשל: מצא את ה- GCD של 175 ו-126:

$$175 \bmod 126 = 49; 126 \bmod 49 = 28; 49 \bmod 28 = 21; 28 \bmod 21 = 7; 21 \bmod 7 = 0$$

לכן 7 הוא המחלק המשותף.

היום נילמד את האלגוריתם המורחב של אוקלידס ונישמור גם על המנות באריתמטיקת מודולו והמטרה היא לחשב את ההפך המכפלי מודולו  $n$  של  $a$ . כלומר נפעיל את האלגוריתם הבא שמוצא את ה- GCD, ובאותה הזדמנות גם את ההפך המכפלי, אם הוא קיים (במידה וה- gcd יצא 1)

ניראה דוגמה של ההפך המכפלי של 28 במודולו 75.

למעשה מה שאנחנו מוצאים זה צמד מספרים:  $s$  ו- $t$ , כך שיתקיים:  $s*a + t*b = \gcd(a,b)$  ובגלל ש- $s$  הוא המכפיל של המודולו, וה- $\gcd$  אמור להיות 1, מכאן נובע ש- $t$  יהיה ההפך המכפיל של  $b$ .

האלגוריתם:

// Use the letters r,t,s for intermediate variables , q for quotient, a, b as input (all ints)  
 // The GCD will be at the end in r1 and the multiplicative inverse in t

a->r1; b->r2; 1->s1; 0->s2; 0->t1; 1->t2 //a=75, b=28

while(r2 > 0)

{

q= r1 / r2;

r1 - q\*r2 -> r; r2 -> r1; r -> r2;

s1 - q\*s2 -> s; s2 -> s1; s -> s2;

t1 - q\*t2 -> t; t2 -> t1; t -> t2;

}

r1 -> gcd; t1 -> t; s1->s;

if (gcd==1) {

if (t < 0) t += a; //The multiplicative is bet. 1 and n-1

return t;

}

שיעורי בית

לתכנת את האלגוריתם המורחב של אוקלידס בשפת תכנות לבחירתכם.