23.02.2015

סיכום שיעור (המשך העיסוק בראשוניים, במסגרת הצפנה)

הכל עד כאן טוב ויפה אם היו לנו **מספרים ראשוניים גדולים מאד** (בעלי כ-154 ספרות כ״א). איך נמצא כאלה?
ישנן 2 אפשרויות לחיפוש ראשוניים, או לבדיקה אם מספר הוא ראשוני או לא. אפשרות אחת היא **אלגוריתם דטרמיניסטי**, כלומר, אלגוריתם שעבור כל מספר יתן תשובה מדוייקת ב-100% אם המספר הוא ראשוני או לא. האפשרות השניה זהו אלגוריתם הסתברותי (probabalistic)

לדוגמה, האלגוריתם הפשוט ביותר הוא אלגוריתם החלוקה - לקחת מספר n, וליצור לולאה שתתחיל ממחלק של 2 ותגדיל את המחלק ב-1, בכל סיבוב וכל פעם תנסה לחלק את: n במספר התורן. אם תהיה חלוקה - המספר אינו ראשוני, וכך נמשיך עד לשורש של n. אם כל החלוקות כשלו, n הוא ראשוני.

זהו אלגוריתם איטי. הבה ונחשב את הסיבוכיות שלו. אם נניח שכל פעולת חלוקה של סיפרה דורשת ביט אחד, ובמספר n ישנם n(b) ביטים, מספר פעולות החילוק יהיה: שורש של 2 בחזקת: n(b) או:

(2^n(b))^0.5 = 2^n(b)

כאשר n שואף לאינסוף, n/2 שווה ל- n למעשה , כלומר סיבוכיות אקספוננציאלית של : O(2^n)

לדוגמה: נניח של- n יש 200 ספרות בינאריות, אז צריכים 2^100 פעולות של ביט. בהנחה שהמחשב יכול לעשות מיליארד פעולות בשנייה 2^30, זה יקח 2^70 שניות שהן כמו נצח (מעל 30 טריליון שנה).

ישנם מספר אלגוריתמים דטרמיניסטיים לקביעה אם מספר הוא ראשוני או לא, שעובדים רק על סוג מספרים מסוים, למשל **מספרי מרסן** (Mersenne) על שם הנזיר הצרפתי מארין מרסן, אלה מספרים שיכולים להיות מיוצגים כ- 2^n - 1 עד לאוקטובר 2014 המספר הראשוני מסוג מרסן הגדול ביותר הידוע הוא: 2^57,885,161 - 1 זה מספר עם 17,425,170 ספרות. (ישנה קבוצה של אנשים שעובדת על זה מ-1997, פרוייקט שניקרא: GIMP קיצור של: Great Internet Mersenne Prime Search.

בשנת 2002 פותח אלגוריתם דטרמיניסטי שעובד בזמן פולינומיאלי, לקביעת ראשוניות על כל מספר. זה ניקרא AKS על שם שלושה מדענים הודים מ- IIT (Agrawal, Kayal, Saxena) הסיבוכיות שלו היא:
O((log2 n(b))^12) - כאשר n(b) זה מספר הביטים במספר: n. והוא מבוסס על **המשפט הקטן של פרמה** שאומר שעבור כל מספר ראשוני p, ועבור כל מספר שלם a השונה מאפס: **a^p mod p =a**
לדוגמה: p=7 ו- a=2 אז: 2^7 mod 7=128 mod 7 = 2
וכן על העובדה שמספר p>2 הוא ראשוני אם ורק אם:

(x-a)^p (mod p) = (x^p-a) (mod p)

עבור כל מספר a שאין לו מחלק משותף עם p. אלגוריתם זה קצת מורכב (דורש ידע על תאורית מקדמים בינומיים שהיא הכללה של המשפט הקטן של פרמה הנ״ל).

אנחנו נעבוד עם **אלגורים הסתברותי שקל ליישום**, והבדיקה אם מספר ראשוני או לאו, תתן לנו תוצאה בהסתברות גבוהה ככל שנירצה, שהמספר הוא ראשוני. זהו האלגוריתם של מילר-רבין.

בעיקרון למספרים ראשוניים ישנן כל מיני תכונות, אבל תכונות אלה לעיתים נכונות גם עבור לא ראשוניים. אם נבדוק הרבה תכונות כאלה והמספר n יעבור את כל הבדיקות, נוכל לקבוע במידה גדולה של ודאות שהמספר n הוא ראשוני.

כעת, אני טוען שאם p הוא מספר ראשוני, אזי עבור כל מספר a, כך ש- 1<= a <= p-1 מתקיים:
a^(p-1) mod p = 1

זה נובע מהמשפט הקטן של פרמה הנ״ל: a^p mod p = a

23.02.2015

(*) a=a^p mod p= [axa^(p-1)] mod p = (a mod p)(a^(p-1) mod p)
ומכיוון ש: a<p הרי: a mod p =a
ולכן נחליף ב (*) את- (a mod p) ב- a, ונקבל:
a= ax(a^(p-1) mod p) => 1=a^(p-1) mod p) /a צימצום ב-

דוגמה: a=4 , p=7
a^(p-1) mod p= 4^(7-1) mod 7= 4^6 mod 7 = 4096 mod 7 = 1

על פי הטענה, אם עבור a כלשהו בין 1 ל- n-1 מתקיים: a^n mod n != 1 זה אומר ש- n אינו ראשוני.
מצד שני יכול גם מספר שאינו ראשוני להיות בעל תכונה זו: למשל 15
4^14 mod 15 = 268,435,456 mod 15 = 1

כלומר אפשר לפסול מספר n כלא ראשוני אם גילינו ש: a^n mod n != 1 , אבל לא להכריז על n כראשוני גם אם: a^n mod n = 1

משפט נוסף אומר שאם n>2 הוא ראשוני אז למשוואה:
x^2 mod n =1

יש בדיוק 2 פתרונות עבור x בתחום של [0,1,……n-1] שהם: 1 ו- n-1

למשל אם n=7 אז הפתרונות הם 1 ו- 6.
1^2 mod 7 = 1 פתרון
2^2 mod 7 = 4
3^2 mod 7 = 2
4^2 mod 7 = 2
5^2 mod 7 = 4
6^2 mod 7 = 1 פתרון

גם במקרה הזה, לרוע המזל, ישנם מספרים שאינם ראשוניים שעבורם המצב הוא דומה, למשל 22
1^mod 22=1; 2^2mod 22= 4; 3^2mod 22= 9; 4^2mod 22= 16 ; 5^2mod 22=3;
6^2mod 22=14; 7^2mod 22=5; 8^2mod 22=20; 9^2mod 22=15…….. **21^2mod 22=1**

אבל למספרים שאינם ראשוניים יכולים להיות יותר פתרונות, למשל n=8
1^2mod 8=1
3^2mod 8=1
5^2mod 8=1
7^2mod 8=1
זה למעשה אומר שאם ביצענו את הבדיקה הזו וראינו שיש פתרון שאינו 1 או n-1 אז בטוח ש- n אינו ראשוני, למרות שלא ניתן לאמר שאם יש רק את 1 ו- n-1 כפתרונות אז n ראשוני.

באלגוריתם של מילר-רבין נשתמש בבדיקות הנ״ל של ראשוניות בצורה איטרטיבית ומספר שיעבור את כל הבדיקות ולא יפסל כ- ״לא ראשוני״ יהיה בהסתברות גבוהה ככל שנירצה: ראשוני.
ננצל את האלגוריתם שמוצא חזקות במהירות (Fast Exponentiation) כדי לבדוק את שני התנאים הנ״ל אם המספר הוא ׳אולי׳ ראשוני אם עבר את התנאים בהצלחה. את a בוחרים באקראיות בין 1 ל n-1

מה מספר הפעמים שנריץ את: PrimalityTesting ונקבל Perhaps Prime כדי להיות די בטוחים ש- n הוא ראשוני?

ישנה למה שלא נוכיח כאן, שאומרת שאם n הוא אי זוגי ואם ניתן ל- a ערכים שונים m פעמים כאשר n>a>1 ועבור כל אחד מאלה, נקבל חזרה: Perhaps Prime אז הסיכוי ש- n הוא ראשוני הוא לפחות:

23.02.2015

1- 1/2^m כלומר אפשר להריץ את האלגוריתם 100 פעמים כדי לקבל בסבירות גבוהה למדי, מספר ראשוני (סבירות של: 0.99999999999999999999999999999 30 פעם 9)

כעת נותרה הבעיה של מציאת ראשוני גדול (האלגוריתם הנ״ל רק בודק אם מספר הוא ראשוני או לא).
הנה הצעה פרקטית: נייצר מחרוזת בינארית של 2048 ספרות כאשר הראשונה 1 והאחרונה 1, ויתר ה-2046 ניבחרות באורח אקראי. נריץ את מילר-רבין 100 פעם ואם עבר, יהיה לנו ראשוני. אם כשל, נייצר עוד מספר כנ״ל. כמה פעמים נידרש עד למציאת ראשוני?

ישנו עוד משפט מתורת המספרים שאומר שהסיכוי שכזה מספר רנדומלי הוא ראשוני, הינו בערך:
1/(ln(2^(n))

כאשר n זה מספר הביטים (במקרה שלנו 2048) , כך שהסיכוי הוא: 0.0007 =(1/ln(2^2048))

אם כך נסכם כמה פעמים צריך לבחור מספר רנדומאלי בן 2048 ספרות בינאריות (מספר עשרוני עם 617 ספרות) : 100 הרצות של מילר-רבין כפול הסיכוי של מציאת ראשוני בעל 2048 ספרות בינאריות:
100x(1/0.0007) =~ 143,000
הרצת רבין מילר 143,000 3פעם היא בהחלט בכוחו של מחשב אישי מינימלי.

שיעורי בית

לכתוב את האלגוריתם (PrimalityTesting) שבודק את התנאים הנ״ל, כלומר הקלט יהיה מספר, והאלגוריתם יחזיר אם המספר ״יכול להיות ראשוני״ , או: ״בטוח שאינו ראשוני״