דור עזריה

אינדוקציה מול רקורסיה

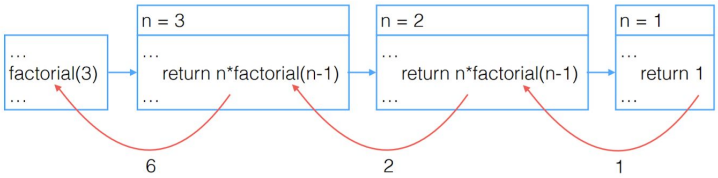
בנושא זה נציג באיזה שיטה עדיף להשתמש כדי לממש את האלגוריתם נכון יותר - באמצעות אינדוקציה או רקורסיה.
גם אם הסיבוכיות של שתי השיטות שווה, ישנם שיקולים נוספים שחשוב לשים עליהם דגש.

אז מתי עדיף להשתמש באינדוקציה ומתי רקורסיה?

מבחינת ניהול זיכרון של רקורסיה, בכל קריאה לפונקציה נוצר **עותק חדש שלה.**
ובעותק יש:

- הקצאה חדשה של כל המשתנים הפנימיים, כולל הפרמטרים המופיעים בחתימה שלה.
- השמה התחלתית של ערכים לפרמטרים לפי מה שהועבר בקריאה אליה.
- אתחול של הפקודה הבאה לבצע בקוד - בעותק החדש זו הפקודה הראשונה בפונקציה.

התהליך נראה כך:



נראה דוגמה להשוואה של רקורסיה מול אינדוקציה עבור חישוב עצרת ונראה את ההשלכות ביניהם:

חישוב עצרת

קלט - מספר אי-שלילי שלם, פלט - !n.

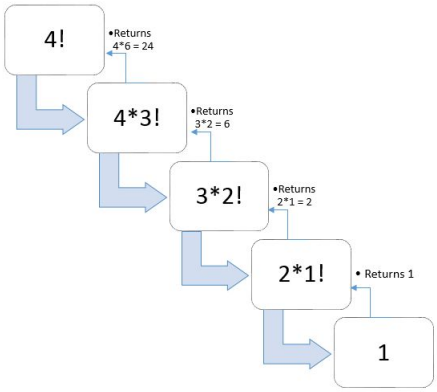
רקורסיבי	אינדוקטיבי

`public static int factorial(int n) {`
`  if(n == 0)`
`  return 1;`
 	
`  return n*factorial(n-1);`
`}`
	`public static int factorial(int n) {`
	`  int f = 1;`
	
	`  for(int i = 1 ; i <= n ; i++){`
	`  f *= i;`
	`  }`
	`  return f;`
	`}`

סיבוכיות נעבור 2n פעמים - נכנסים n פעמים כדי לחשב את הצעדים ואת גודל המחסנית ויוצאים n פעמים - סה"כ $O(n) + O(n) = O(2n) = O(n)$.	**סיבוכיות** נעבור פעם אחת על הלולאה - סה"כ $O(n)$.

קוד הרקורסיבי תואם להגדרה באינדוקציה ויפה יותר אבל דורש **יותר** זיכרון במהלך הביצוע - בתחתית הרקורסיה מוחזקים בו-זמנית **n עותקים של factorial.**

למרות שהסיבוכיות שלהם שווה, עדיף לנו להשתמש באינדוקציה, למה? - כי אפשר לראות לפי האלגוריתם בשיטת האינדוקציה כי מתבצע **n פחות** פעולות מאשר שיטת הרקורסיה המבצע 2n פעולות.



הדמייה רקורסיבית

חישוב סדרת פיבונאצ'י

הסדרה מוגדרת באינדוקציה:

$$F_1 = F_2 = 1$$
$$F_n = F_{n-2} + F_{n-1}$$

האיברים הראשונים בסדרה הם:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ……..

קלט - מספר אי-שלילי

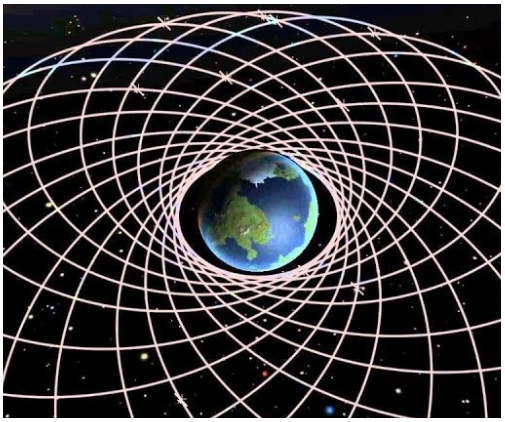
פלט - ערך באינדקס לפי סדר הסדרה.

רקורסיבי	אינדוקטיבי
אלגוריתם	אלגוריתם

`public static int fiboRec(int n) {`
`  if( n == 0 || n == 1) {`
`  return n;`
`  }`
`  return fiboRec(n-1) + fiboRec(n-2);`
`  // O(2^N)`
`}`
	`public static int fiboInductive(int n) {`
	`  int[] arr = new int[n+1];`
	`  arr[0] = 0;`
	`  arr[1] = 1;`
	
	`  for(int i = 2 ; i <= n ; i++)`
	`  arr[i] = arr[i-1] + arr[i-2];`
	
	`  return arr[n];`
	`}`

סיבוכיות כל קריאה לפונקציה גוררת שתי קריאות לפונקציה - לכן,הסיבוכיות היא $O(2^n)$	**סיבוכיות** עבור ההשמה של איבר 0 ו-1 זה $O(2)$. עבור הלולאה אנו רצים $n-1$ פעמים פחות 2 ההשמה שבוצעו כבר ולכן התחלנו לרוץ החל מאינדקס 2 כלומר הלולאה מבצעת $O(n-3) = O(n)$. סה"כ - $O(n) + O(2) = O(n)$.

קל לראות במקרה זה שהסיבוכיות בין שתי השיטות בהחלט שונה,
מן הסתם עדיף להשתמש בשיטת האינדוקטיבית כי $O(n) < O(2^n)$ לכל $n \geq 0$.



הערכת ביצועים עבור רקורסיה

- נספור כמה קריאות מתבצעות ל-fibonacci.
- נסמן ב- $T(n)$ את מספר הקריאות הנדרשות כדי לחשב את $fibonacci(n)$.

$$T(1) = T(2) = 1$$

$$T(n) = 1 + T(n-2) + T(n-1)$$

- הפתרון מקיים $T(n) = 2F_n - 1$.

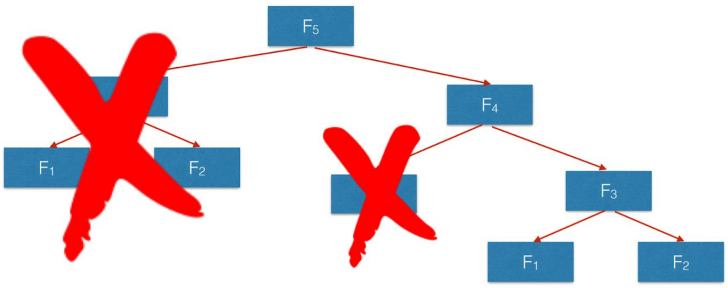
מבחינת צריכת זכרון

- בשלב מסויים בריצה פתוחים בבת אחת **n עותקים** של הפונקציה.

מה הבעיה?

אנחנו מחשבים מחדש אותם ערכים פעם אחר פעם.

- למשל, עבור n == 10, הקריאה $fibo(n-2)$ מחשבת את F_8 וגם הקריאה $fibo(n-1)$ מחשבת את F_8 באחת הקריאות הרקורסיביות שם.
- הערכים F_1 ו- F_2 מוחזרים בסה"כ F_n פעמים.
- אם נזכור את שני הערכים האחרונים, לא נצטרך לחשב מחדש ברקורסיה את כל הערכים הקודמים.



במשך הקורס נלמד על *יחס הזהב* ונראה שימוש לפתרון פיבונאצ'י.