

פרק 4 רקורסיה

1. הזמן הנדרש

10 שעות.

2. מבנה הפרק

מבוא – חפיסת השוקולד

- בעיה זו מדגימה את הפתרון הרקורסיבי של חלוקת שוקולד. יש לשים לב לדיוק מסוים ואף לדון בו בכיתות חזקות: אם רק ילד אחד היה צריך לחלק את השוקולד, הוא לא יכול היה להשלים את המשימה בהצלחה מבלי שהשוקולד היה נמס. כלומר בפתרון זה, בנוסף לרקורסיה, קיים שימוש בהרבה תלמידים (הרבה מעבדים), ואכן פתרון רקורסיבי יכול להיות יעיל מאוד כאשר ישנה אפשרות להשתמש במספר מעבדים בו זמנית.
- מורים המשתמשים במודל האנשים הקטנים להסברת נושא הרקורסיה עלולים להרגיש שיש סתירה בין הדוגמה ובין המודל. בבעיית השוקולד הילדים נקראים שוב ושוב לעזור במשימת החלוקה, בעוד במודל האנשים הקטנים, לכל אדם יש את המשימה הייחודית שלו והוא אינו שותף במשימות נוספות. מורים אלה יכולים להימנע מדיון בבעיית השוקולד, אך יש לשים לב שעיקר הדוגמה הוא להמחשת התהליך החוזר על עצמו ולא הרבה מעבר לכך.

א – אז מהי (בדיוק) רקורסיה?

יש להתמקד במושגים החדשים המוזכרים בסעיף ולנסות לתת לתלמידים תחושה ראשונית על מהותו של התהליך הרקורסיבי. ברור שהבנה מעמיקה תיווצר עם ריבוי הדוגמאות ופתרון.

ב – פתרון רקורסיבי של בעיה פשוטה

- בפרק מוצגות כמה דרכים למעקב אחר התהליך הרקורסיבי. דרכי המעקב משתנות לפי סוגי הרקורסיה ומורכבות המעקב. ברור שיש עוד דרכי מעקב, וכן שבכתיבת ספר סטטי אין דרך להדגים את כל הדרכים הדינמיות המתאימות למעקב אחר רקורסיה. לשם כך יעילים יותר צג המחשב או הלוח והגירים. אין הכרח לעקוב אחר התהליכים בדרכים המופיעות בספר. השתמשו בכל אחת מהדרכים המוכרות והנחות לכם.
- ניתן לייעל את הפעולה לחישוב עצרת על ידי כך שחוסכים בזימון רקורסיבי אחד:
`if (n==0 || n==1) return 1`
אם התלמידים מעלים אפשרות זו כדאי להתייחס אליה.

- כאשר פעולה רקורסיבית ניתנת לניסוח בעזרת שתי אפשרויות (או יותר) באופן הזה:

```
if {...}
    return {...}
else {...}
    // קריאה רקורסיבית
```

אין צורך לכתוב את ה-else, כיוון שאם התנאי מתקיים הפעולה נעצרת על ידי ה-return, אבל צורת הכתיב המלאה של התנאי ברורה יותר, ולכן לפעמים התהליך ייכתב במלואו.

תשובה לשאלת המחשבה המופיעה בסעיף זה:

? נסו לכתוב את הפעולה reverse(...) / Reverse(...) ללא תנאי העצירה וראו מה יקרה.

תשובה:

התהליך לא ייפסק כלל!

מערכת השפה אינה מסוגלת לאבחן שהתכנית לא תעצור אם תורשה להמשיך לרוץ, אבל כאשר מחסנית זמן הריצה מתמלאת, המערכת אינה יכולה לבצע זימונים נוספים. התוכנית תיעצר בגלל חוסר מקום במחסנית זמן ריצה, ותיווצר שגיאה בזמן ריצה. הסבר זה מתאים רק לתלמידים המכירים את המושג מחסנית זמן ריצה (לאחר שלמדו את פרק 8 – מחסנית).

ג – אלגוריתם רקורסיבי לחישוב מספרי פיבונצ'י

- ניתן לכתוב פתרון יותר "אלגנטי" לחישוב מספרי פיבונצ'י, שיהיה יעיל יותר:
בג'אוה:

```
public static int fibonacci(int k)
{
    if (k==1 || k==2)
        return k-1;

    return (fibonacci(k-1) + (fibonacci(k-2)));
}
```

בסישרפ:

```
public static int Fibonacci(int k)
{
    if (k==1 || k==2)
        return k-1;

    return (Fibonacci(k-1) + (Fibonacci(k-2)));
}
```

- יש להפנות את תשומת הלב של התלמידים כי המציין של מספרי פיבונצ'י מתחיל ב-1 ($k \geq 1$) ולא ב-0, כפי שהם רגילים מהמערך.
- יעיל יותר לפתור את המשימה של חישוב מספרי פיבונצ'י על ידי לולאה ולא על ידי זימונים רקורסיביים, באופן הזה (בגיאווה הפעולה `fibon(...)` תיכתב כמובן באות קטנה):

```
public static int Fibon(int num)
{
    int beforeLast = 0;
    int last = 1;
    int temp;

    if (num == 1 )
        return 0;
    if (num == 2)
        return 1;
    for (int i=3; i<= num; i++)
    {
        temp = last;
        last = beforeLast + last;
        beforeLast = temp;
    }
    return last;
}
```

היתרון של הפעולה המבוצעת על ידי לולאה הוא בכך שלא נעשים חישובים כפולים, כפי שקורה בפעולה הממומשת באמצעות רקורסיה. לכן במקרה הזה הלולאה יעילה יותר מהדרך הרקורסיבית. ניתוח יעילות מראה כי לגירסה הרקורסיבית יעילות מעריכית בעוד שלגירסה האיטרטיבית יעילות לינארית.

תשובה לשאלת המחשבה המופיעה בסעיף זה:

? כאשר מזמנים את `Fibonacci(...)` / `Fibonacci(...)` עבור הערך 5,

כמה פעמים מחושב `fibonacci (1)` וכמה פעמים מחושב `fibonacci (2)`?

תשובה:

`fibonacci (1)` יחושב פעמיים, ו-`fibonacci (2)` יחושב שלוש פעמים.

ד – סוגי רקורסיה

המאפיין המרכזי של רקורסיית "הלוך-חזור" הוא פעולה נוספת (אחת לפחות) המתבצעת אחרי הקריאה הרקורסיבית. אם הרקורסיה מחזירה ערך, ויש באלגוריתם הרקורסיבי שימוש בערך זה אזי הרקורסיה היא מסוג "הלוך-חזור". כך הדבר בשתי הדוגמאות שראינו בפרק עד כה, שבהן מוחזר ערך מהקריאה הרקורסיבית, וזה המצב בדרך כלל. אך ייתכן כי הרקורסיה אינה מחזירה ערך, ועדיין היא מהסוג "הלוך-חזור", כיוון שמתבצעת פעולה כלשהי אחרי הקריאה הרקורסיבית.

תשובה לשאלת המחשבה המופיעה בסעיף זה :

? מה מחזירה הפעולה $mysterySum(\dots)$ עבור $n = 3$?

תשובה:

הפעולה מחזירה 7.

ה – רקורסיה על טיפוס נתונים אחרים

הערה כללית: יחידת העיבוד המרכזית (ה-CPU) הנמצאת בלב כל מחשב, ואחראית על הרצת כל התוכניות, היא בעצם מחשבון משוכלל שפעולותיו הבסיסיות כוללות את הפעולות האריתמטיות. לכן פעולה אריתמטית (כגון: חיבור, חיסור, כפל, חילוק וכד') היא פעולה "זולה" ומתבצעת מהר. בניגוד לכך, זימון פעולה הוא פעולה "יקרה" יחסית. הקריאה לפעולה היא תהליך המצריך העתקת הרבה משתנים ממקומות שונים בזיכרון (כגון: המשתנים המועברים לפעולה, מצב הזיכרון לפני הכניסה לפעולה, פינוי מקום לערך ההחזרה של הפעולה – אם קיים, ועוד). לכן מציאת מקסימום במערך באופן רקורסיבי, כפי שנעשית בפרק, יעילה פחות ממעבר איטרטיבי על המערך בעזרת לולאה. אף שדבר זה אינו משנה את יעילות הפעולה במונחים שילמדו בפרק 5 – יעילות, חשוב שהתלמיד יבין כי קריאה לפעולה יקרה יותר מביצוע פעולה אריתמטית בסיסית.

ו – דוגמה: האלגוריתם של אוקלידס

האלגוריתם של אוקלידס מתבסס על העובדה הזו:

"אם מספר מחלק שני מספרים, אזי הוא מחלק גם את השארית של חלוקת הגדול שביניהם במשנהו".

בכיתות חזקות ניתן לבקש הוכחה לעובדה זו.

הנה הוכחה אפשרית בעזרת דוגמה: נניח כי המספר הוא 6. כלומר, אם 2 מספרים מתחלקים ב-6 אז גם השארית של החלוקה ביניהם מתחלקת ב-6. (אפשר כמובן לקחת עוד נעלם במקום 6 אך זה יותר קשה להבנה ומיותר).

את המספר הראשון שמתחלק ב-6 נסמן ב-6a

את המספר השני שמתחלק ב-6 נסמן ב-6b

נניח כי המספר השני גדול מהמספר הראשון.

כדי לחשב את שארית החלוקה יש להחסיר את המספר הקטן מהמספר הגדול כמה פעמים שאפשר:

שארית החלוקה היא:

$$6b - k6a = 6(b - ka)$$

ברור לעין כי גם השארית מתחלקת ב-6.

תשובה לשאלת המחשבה המופיעה בסעיף זה :

? מהו תנאי העצירה של האלגוריתם של אוקלידס?

תשובה:

תנאי העצירה הוא : ששארית החלוקה של שני המספרים תהיה אפס.

ז – פעולת עזר רקורסיבית

לרוב אנו רגילים לדבר על "הסתרת המימוש" בהקשר של מחלקות. עיקרון זה מקבל ביטוי מעניין בצורת הכתיבה המוצגת בסעיף. המתכנת לא צריך להיות מודע לצורת המימוש של הפעולה. כך גם ניתן לשנות את מימוש הפעולה מרקורסיבית לאיטרטיבית, ולא יהיה כל שינוי בקוד שהשתמש כותב.

אנו נראה בהמשך היחידה מקרה נפוץ שבו שימוש בפעולת עזר רקורסיבית מועיל ואף נחוץ : לעתים נוח וקל לכתוב פעולה רקורסיבית על תכונה פרטית של מחלקה, אך אי אפשר לכלול אותה בממשק. במקום זאת מציעים בממשק פעולה שמשמשת בה כפעולת עזר. למשל, למחלקה רשימה יש תכונה פרטית שהיא שרשרת חוליות. קל לכתוב פעולה רקורסיבית לחישוב אורך של שרשרת. פעולה זו מקבלת הפניה לחוליה הראשונה בשרשרת, מפעילה את עצמה רקורסיבית עם ההפניה לחוליה הבאה בשרשרת, ומוסיפה 1 לערך המתקבל. אבל, פעולה לחישוב אורך בממשק הרשימה, היא פעולה ללא פרמטרים. הפתרון הוא שימוש בפעולה הרקורסיבית כפעולת עזר פרטית.

מושגים

לשלושה מהמושגים אין במכוון תרגום לאנגלית. המושג העברי אינו מושג אוניברסלי משמעותי ולכן לא היה טעם בתרגומו המאוץ לאנגלית.

3. תרגילים

פתרונות לכל התרגילים מופיעים באתר המרכז להוראת המדעים, האוניברסיטה העברית בירושלים ונגישים רק לציבור המורים :

http://sites.huji.ac.il/science/unit4_2007/

שימו לב כי התרגילים הבאים עוסקים באותו הנושא, אך צורת השאלה שונה :

10 ו-14 : הדפסת המחרוזת בסדר הפוך.

9 ו-17 : חישוב מספר הספרות.

שאלות 1-7 : מעקב אחר פעולות רקורסיביות

שאלה 1

רמת קושי : קלה (רקורסיית זנב).

שאלות 2–4

רמת קושי : קלה-בינונית.

שאלות 5–7

רמת קושי : בינונית ומעלה.

שאלות 8–12 : השלמה קוד של פעולות רקורסיביות

שאלות 8–10

רמת קושי : קלה.

בשאלה 10 כדאי לכוון את התלמידים לחפש בממשק של המחלקה String / string פעולה המוציאה תת-מחרוזת מתוך מחרוזת נתונה, כך שימצאו את : substring(...) / Substring(...).

שאלות 11–12

רמת קושי : קלה ומעלה.

שאלות 13–14 : בחירת פעולה רקורסיבית אחת העונה על הדרישה מתוך 3 אפשרויות נתונות.

שאלה 13

רמת קושי : קלה.

שאלה 14

רמת קושי : קלה ומעלה.

שאלות 15–16 : הרצת פעולות רקורסיביות הפועלות על צבים (עצמים ממחלקת Turtle).

מומלץ לבקש מהתלמידים לעקוב אחרי הפעולות ולהגיד מה לדעתם הן מבצעות, עוד לפני הרצת הפעולות במחשב.

שאלה 15

רמת קושי : קלה.

שאלה 16

רמת קושי : קלה ומעלה.

ב. כתיבת רקורסיות

תרגילים 17–28 : תרגילים על ספרות ומספרים, תרגילים על מערך ותרגילי הדפסה.

רמת קושי : קלה-בינונית.

ג. תרגילי אתגר

תרגיל 29: פרמוטציות

ניתן לתאר את האלגוריתם באופן הזה:

1. עבור על כל איברי המערך באמצעות האינדקס i , ועבור כל איבר בצע:
2. חשב את הפרמוטציות על כל האיברים פרט לאיבר i
3. עבור כל פרמוטציה הוסף את האיבר ה- i לתחילתה.

כלומר הקטנת הבעיה נעשית באמצעות הורדת האיבר במקום ה- i ושליחת יתר האיברים לפעולה בצורה רקורסיבית.

כמובן שהפתרון ייעשה באמצעות אינדקסים.

תרגיל 30: מגדלי האנוי

בשאלה זו נשתמש ברעיון הזה:

1. העבר $n-1$ דסקיות ממוט a למוט b .
2. העבר דיסקית ממוט a למוט c .
3. העבר $n-1$ דסקיות ממוט b למוט c .

תרגיל 31: בעיית שמונה המלכות

שאלה זו היא קלה להבנה, אך הפתרון שלה קשה. באמצעותה ניתן להדגים את הכוח של הרקורסיה.

ניתן לתאר את האלגוריתם באופן הזה:

1. הצב את המלכה הראשונה במקום $(1, 1)$.
2. עבור לשורה הבאה והצב את המלכה במקום הראשון שבו אינה מאוימת על ידי המלכה הראשונה.
3. המשך באופן המתואר בשורה 2. כלומר, עבור כל שורה חפש את המקום הראשון שאינו מאויים על ידי המלכות הקודמות.
4. אם הגעת לשורה שבה אין מקום להציב בו את המלכה באופן שלא תהיה מאוימת על ידי המלכות הקודמות, חזור לשורה הקודמת והזז את המלכה למקום הבא שאינו מאויים על ידי המלכות האחרות.
5. המשך בתהליך עד שתציב את כל המלכות.

הערה חשובה: שאלה זו מתבססת באופן מהותי על תהליך ה"חזור" של התהליך הרקורסיבי. אם לא ניתן להציב את המלכה בשורה כלשהי, התוכנית תחזור שורה אחת אחורה ומשם תמשיך את התהליך.

תרגיל 32 : בעיית מסעי פרש

בפתרון הבעיה נשתמש ברעיון הזה :

1. עבור המיקום הנתון של הפרש : בצע כל מהלך אפשרי ממיקום זה. מהלך הוא אפשרי אם הוא חוקי לפי חוקי השחמט ואם עדיין לא ביקרו במיקום זה.
2. סמן את המיקום החדש (כדי לזכור שהיינו בו).
3. עבור כל מהלך שביצעת זמן את הפעולה באופן רקורסיבי.
4. אם הגעת למקום שממנו אינך יכול להתקדם, חזור.

תרגיל 33 : בעיית המבוך

בבעיה זו נשתמש ברעיון הזה :

בצע כל צעד אפשרי ממיקום נתון. עבור כל צעד שביצעת הפעל את הפעולה באופן רקורסיבי על המיקום החדש. אם הגעת ליעד, החזר "אמת", אחרת – החזר "שקר".

חשוב לסמן את המקום שהיינו בו, ואם כבר היינו במקום מסוים חשוב לזכור שאין לחזור אליו.

באמצעות תרגיל זה, ניתן לדון עם התלמידים בפרמטרים שהפעולה צריכה לקבל (מיקום המשתנה ברקורסיה, יעד קבוע).

גם שאלה זו מתבססת על תהליך ה"חזור" הרקורסיבי. שהרי במקרה שנתקענו במיקום מסוים ואיננו יכולים להמשיך, הפעולה חוזרת שלב אחד אחור ומנסה את האפשרות הבאה עבור מיקום קודם.