

1. מחלקת רקורסיה



התבניות שבמחלקה

1.1 - רקורסיה תחילה

1.2 - רקורסיה בסוף

1.3 - רקורסיה כפולה

1.4 - הפרד ומשול

1.5 - רקורסיה דו-מימדית (העשרה)



רקורסיה הינה דרך להתבוננות ופתרון בעיות על-ידי "הצרת" בעיה לבעיה זזה, אשר בה גודל הנתונים (ולפעמים אף כמותם) קטן יותר, או קרוב יותר למקרה בסיסי, אשר פתרונו מיידי. חוקרים במתמטיקה ומדעי המחשב הוכיחו שכל בעיה אלגוריתמית, אשר ניתנת לפתרון בדרך לא רקורסיבית, ניתנת לפתרון גם בדרך רקורסיבית. לכן, הכרת גישה רקורסיבית לפתרון בעיות הינה נדבך בסיסי במדעי המחשב. במקרים רבים מתאים יותר לגשת לפתרון בעיה בצורה רקורסיבית. בחומר הלימוד של "יסודות מדעי המחשב" הוצג מספר מצומצם של מקרים כאלה (בסוף חומר הלימוד של "יסודות-2"). בחומר הלימוד של "עיצוב תכנה" מופיעים מקרים כאלה שוב ושוב, בפרקים שונים, ובפרט בפרק של "עצים".

נציג כאן **דוגמה קצרה** (הלקוחה ממחקרו של דוד גינת) לאפקטיביות של התבוננות רקורסיבית. נתונים שני מספרים שלמים חיוביים X ו- Y , ונתון ש- $Y > X$. יש לחשב את מספר הפעולות המינימלי מסוג $+1$ ו- $\times 2$ הדרושות כדי להגיע מ- X ל- Y (למשל, עבור 10 ו-21 יהיה מספר זה 2 – תחילה $\times 2$ ואחר-כך $+1$, ואילו עבור 10 ו-17 יהיה מספר זה 7 – עקב 7 פעולות החיבור $+1$).

פותרים רבים של שאלה זו מנסים דרכים שונות של "הליכה קדימה". בפרט, רבים מציעים להפעיל תחילה כמה שיותר פעולות כפל, וכשלא ניתן יותר לכפול – להפעיל את פעולות ה- $+1$. דרך זו אינה נכונה! למשל, עבור 10 ו-100 תהיה ההתקדמות משמאל לימין: 100, 99, ..., 83, 82, 81, 80, 40, 20, 10 – בסך-הכל 23 פעולות. ניתן להפעיל הרבה פחות פעולות, וזאת על-ידי התבוננות רקורסיבית.

ניתן לשאול "מהי הפעולה האחרונה בסדרת הפעולות המינימלית?" התשובה לשאלה זו די ברורה: אם Y זוגי אזי פעולה זו צריכה להיות $\times 2$, ואם Y הוא א-זוגי – אזי פעולה זו היא $+1$. "נלך רקורסיבית אחורה" עם פעולה זו, ושוב נשאל את אותה השאלה, וכך הלאה, עד שנגיע ל- X . בדרך זו יתברר לנו למשל שעבור הנתונים 10 ו-100 מספיקות 6 פעולות בלבד!!! הליכה רקורסיבית תוביל לסדרת המספרים הבאה: 50, 25, 24, 12, 11, 10.

חישובים רקורסיביים נעשים בצורות שונות, אשר במקרים רבים נגזרות מתבניות שחוזרות שוב ושוב. בפרק זה אנו מציגים חמש תבניות כאלה. (פתרון הבעיה לדוגמה לעיל ניתן ליישום הן באמצעות התבנית 1.1 והן באמצעות התבנית 1.2). תבניות הן סכמות כלליות, אך הצגת כל אחת מהן נעשית באמצעות דוגמה. אין להיתפש לדוגמה במונח של החישוב המסויים שהיא מציגה, אלא כביטוי לסכמה כללית. השאלות המוצגות אחרי כל תבנית ממוקדות נקודה זו.

אנו מאמינים שהתנסות חוזרת ונשנית של שימוש בתבניות בשאלות השונות, ובהקשרים שונים, מהווה נדבך מרכזי בלימוד והטמעה של הנושא רקורסיה. אנו מציעים לכל לומד לבחון את הפתרונות הרקורסיביים שבתבניות ובפתרון השאלות הן בהיבט של פיתוח "חוקים רקורסיביים" (כגון הדוגמה לעיל) והן בהיבט של מעקב מפורט אחרי מהלך ביצוע חוקים אלו.

1.1 - רקורסיה תחילה

נקודת מוצא: מספר שלם חיובי n .

מטרה: הדפסת כל המספרים מ-1 עד n .

אלגוריתם: הדפס-בסדר-עולה (n)

$n = 1$ ρ ρ ρ

n ρ ρ ρ

אחרת

הדפס-בסדר-עולה ($n-1$)

n ρ ρ ρ

הערות

- משמעות המושג "תחילה" היא שהפעולה הראשונה במהלך הביצוע של כל "רמה רקורסיבית", הינה הזימון הרקורסיבי של "הרמה" הבאה. החישוב (בדוגמה שבתבנית – ההדפסה) מתבצע רק עם החזרה ("העלייה") מן הקריאה הרקורסיבית.
- כדי להבין כדרוש את ההערה הקודמת, כדאי לפרוש מהלך של ביצוע לדוגמה של האלגוריתם שבתבנית.
- תנאי העצירה בדוגמה הוא $n=1$. בדוגמה זו תנאי זה מתאים כיון שנקודת המוצא היא מספר שלם חיובי, ובכל זימון רקורסיבי ערכו מוקטן ב-1. ישנם פתרונות רקורסיביים בהם נקודת המוצא או ההקטנה בזמן הזימון עשויות להיות שונות, ובהם כדאי להשתמש בסימן כגון " $<$ " במקום הסימן " $=$ ".
- לפעמים, במקרה העצירה (הבסיס) אין לבצע חישוב כלשהו. בפעמים כאלה ייכתב תנאי שאין בו חלק של "אחרת". למשל:

$n > 0$ ρ ρ ρ

$n-1$ ρ ρ ρ

n ρ ρ ρ

(פעולה המקבלת מספר שלם חיובי n, ומדפיסה את המספרים מ-1 עד n *)*

Pascal

```
procedure print_up (n: integer);
begin
    if n = 1 then
        write (n)
    else
        begin
            print_up (n-1);
            write (n);
        end;
end;
```

/ פעולה המקבלת מספר שלם חיובי n, ומדפיסה את המספרים מ-1 עד n */*

C

```
void print_up (int n)
{
    if (n == 1)
        printf ("%d ",n);
    else
    {
        print_up (n-1);
        printf ("%d ",n);
    }
}
```

שאלות

שאלה 1.1.1

נתונה סדרת הנסיגה הכתובה בלשון מתמטית:
 $a_1 = 3$
 $a_n = a_{n-1} + 2$
פתח אלגוריתם רקורסיבי שיקבל מקום n בסדרה ויחזיר את האיבר ה- n -י בסדרה זו.

■

שאלה 1.1.2

פתח אלגוריתם רקורסיבי שיקבל מספר זוגי וחיובי n וידפיס בסדר עולה את כל הספרות הזוגיות הקטנות מ- N .

■

שאלה 1.1.3

פתח אלגוריתם רקורסיבי שיקבל מספר שלם וחיובי n ויחזיר את סכום ספרותיו.

■

שאלה 1.1.4

פתח אלגוריתם רקורסיבי שיקבל שני מספרים טבעיים a ו- b ויחזיר את מכפלתם על ידי פעולות חיבור וחסור בלבד. הנחייה: $a * b = a + a + \dots + a$ (b פעמים)

■

שאלה 1.1.5

פתח אלגוריתם רקורסיבי שיקבל שני מספרים שלמים וחיוביים ויחזיר את מנת החלוקה של המספר הראשון בשני תוך שימוש בפעולות חיבור וחסור בלבד.

■

שאלה 1.1.6

פתח אלגוריתם רקורסיבי שיקבל כפרמטר מערך בגודל N של מספרים שלמים ויחזיר את סכום איבריו.

■

שאלה 1.1.7

פתח אלגוריתם רקורסיבי שיקבל מספר שלם חיובי n בבסיס 10 ויחזיר את ערכו בבסיס 2.

■

שאלה 1.1.8

פתח אלגוריתם רקורסיבי שיקבל מספר שלם וחיובי n ויחזיר מספר חדש המורכב מספרות המספר בסדר הפוך.
דוגמא: עבור המספר 416982 יוחזר המספר 289614

■

שאלה 1.1.9

נתונה פונקציה רקורסיבית.

- א. בצע מעקב עבור הקריאה `what (123)`
ב. רשום באופן כללי מה מבצע האלגוריתם.

Pascal הפונקציה בשפת

```
function what (n: integer): integer;
var
  k: integer;
begin
  if (n < 10) then
    what := n
  else
    begin
      k := what ((n div 100)*10 + n mod 10);
      what := k*10 + (n mod 100) div 10;
    end;
end;
```

C הפונקציה בשפת

```
int what (int n)
{
  int k;
  if (n < 10)
    return n;
  else
  {
    k = what ((n/100)*10 + n%10);
    return (k*10 + (n%100) / 10);
  }
}
```

■

1.2 - רקורסיה בסוף

נקודת מוצא: מספר שלם וחיובי n .

מטרה: הדפסת כל המספרים מ- n עד 1.

אלגוריתם: הדפס-בסדר-יורד (n)

אם $n = 1$ אז

הדפס n

אחרית

הדפס n

הדפס-בסדר-יורד ($n-1$)

הערות

- משמעות המושג "בסוף" היא שהפעולה האחרונה במהלך הביצוע של כל "רמה רקורסיבית", הינה הזימון הרקורסיבי של "הרמה" הבאה. החישוב (בדוגמה שבתבנית – ההדפסה) מתבצע לפני זימון זה.
- כדי להבין כדרוש את ההערה הקודמת, כדאי לפרוש מהלך של ביצוע לדוגמה של האלגוריתם שבתבנית.
- תבנית זו דומה מאד לתבנית של ביצוע איטרטיבי בלולאה, שבו לאחר סיום החישוב הנוכחי של גוף הלולאה מתחיל חישוב נוסף.

| | |
|---|---------------|
| <pre>(* פעולה המקבלת מספר שלם חיובי n, ומדפיסה את המספרים מ- n עד 1 *) procedure print_down (n: integer); begin if n = 1 then write (n) else begin write (n); print_down (n-1); end; end;</pre> | <i>Pascal</i> |
|---|---------------|

| | |
|--|---|
| <pre>/* פעולה המקבלת מספר שלם חיובי n, ומדפיסה את המספרים מ- n עד 1 */ void print_down (int n) { if (n == 1) printf ("%d ", n); else { printf ("%d ", n); print_down (n-1); } }</pre> | C |
|--|---|

שאלות

1.2.1 שאלה

נגדיר "מספר מתחלף" כמספר בו כל זוג ספרות שכנות (או צמודות) הינו בעל זוגיות שונה. פתח אלגוריתם רקורסיבי שיקבל כפרמטר מספר שלם חיובי n ויחזיר "אמת" אם הוא "מספר מתחלף" ו"שקר" אחרת. דוגמא: עבור המספר 254781 יוחזר "אמת" (ספרות המספר הן זוגי/אי-זוגי לסירוגין). עבור המספר 247589 יוחזר "שקר".

■

1.2.2 שאלה

פתח אלגוריתם רקורסיבי שיקבל כפרמטר מספר שלם וחיובי n ויחזיר "אמת" אם קיים במספר לפחות זוג אחד של ספרות צמודות בעל אותה זוגיות, ו"שקר" אחרת.

■

1.2.3 שאלה

פתח אלגוריתם רקורסיבי שיקבל כפרמטרים שני מספרים שלמים וחיוביים a ו- b ויחזיר את שארית החלוקה של המספר הראשון בשני תוך שימוש בפעולות חיבור וחסור בלבד.

■

1.2.4 שאלה

פתח אלגוריתם רקורסיבי שיקבל כפרמטר מחרוזת st ויחזיר "אמת" אם המחרוזת מהווה פלינדרום, ו"שקר" אחרת.

■

1.2.5 שאלה

פתח אלגוריתם רקורסיבי שיקבל כפרמטר מספר שלם וחיובי n ויחזיר את ספרות המספר מהסוף להתחלה. דוגמא: עבור המספר 87263 יודפס 36278

■

1.3 - רקורסיה כפולה

(דוגמא: סדרת פיבונאצ'י)

נקודת מוצא: מספר שלם וחיובי n .

מטרה: חישוב ערכו של האיבר ה- n בסדרת פיבונאצ'י.

אלגוריתם: פיבונאצ'י (n)

אם $n < 3$ אזי

החזר $n-1$

אחרת

החזר פיבונאצ'י ($n-1$) + פיבונאצ'י ($n-2$)

הערות

- בתבנית זו ישנם שני זימונים רקורסיביים. בכל אחד מן הזימונים מועבר ערך שונה. הזימונים השונים יכולים להיות באותו ביטוי (כפי שמוצג בדוגמת פיבונאצ'י לעיל), או בביטויים שונים בהוראות נפרדות בפסאודו-קוד (או קוד). ייתכן גם זימון משולש (ואף יותר) באופן דומה לזימון הכפול.
- כדי להבין כדרוש את ההערה הקודמת, כדאי לפרוש מהלך של ביצוע לדוגמה של האלגוריתם שבתבנית. בפרט, כדאי לשים לב בפרישה לאופני העצירה השונים של "מסלולים" שונים של זימונים רקורסיביים: בדוגמת פיבונאצ'י שבתבנית לעיל תהיה עצירה במקרים בהם סוף "מסלול" הוא $n=1$ ובמקרים אחרים בהם סוף "מסלול" הוא $n=2$.
- כדאי לשים לב לשימוש בסימן " $<$ " בתנאי העצירה, ולהבינו בהקשר של ההערה הקודמת וההערה אודות השימוש בסימן " $=$ " המופיעה בתבנית 1.1.
- בדוגמה של התבנית פיבונאצ'י לעיל זוג הזימונים הרקורסיביים הינו מסוג "רקורסיה תחילה" (בדומה לתבנית 1.1) כיון שפעולת החישוב (חיבור) מתבצעת לאחר החזרה משני הזימונים הרקורסיביים.
- זימון כפול מאפשר לפעמים התבוננות נוחה, אך לא בהכרח יעילה. בפרט, הוא עלול להוביל לזימונים חוזרים ונשנים של חישובים זהים. למשל בדוגמת פיבונאצ'י לעיל, כפי שניתן להיווכח על-ידי פרישת מהלך ביצוע פשוט. במקרים כאלה כדאי לנסח את התבנית הרקורסיבית תחילה, כדי לבטא התבוננות רקורסיבית מאירה, ואחר-כך לנסות להמירה למבנה לא רקורסיבי (לולאה).

Pascal

האלגוריתם מקבל מספר שלם וחיובי n ומחזיר את ערך האיבר במקום (
) ה-n בסדרת פיבונאצ'י.

```
function fibonacci (n: integer): integer;  
begin  
  if n < 3 then  
    fibonacci := n-1      {הסדרה מתחילה ב- 0, 1}  
  else  
    fibonacci := fibonacci (n -1) + fibonacci (n -2) ;  
end;
```

C

האלגוריתם מקבל מספר שלם וחיובי n ומחזיר את ערך האיבר במקום /
/ ה-n בסדרת פיבונאצ'י.

```
int fibonacci (int n)  
{  
  if (n < 3)  
    return n-1;          // הסדרה מתחילה ב- 0, 1  
  else  
    return fibonacci (n -1) + fibonacci (n -2);  
}
```

שאלות

שאלה 1.3.1

נתונה הסדרה :

$$\Leftrightarrow 1, 2, 3, 6, 4, 13, 7, 24, 11, 42, \dots$$

- שלושת האיברים הראשונים בסדרה הם : 1, 2, 3
 - כל איבר שנמצא במקום זוגי בסדרה, החל מהמקום ה-4, הוא סכום שלושת האיברים הקודמים לו.
 - כל איבר הנמצא במקום אי-זוגי בסדרה, החל מהמקום ה-5, הוא ההפרש המוחלט שבין שני האיברים במקומות הזוגיים שלפניו.
- פתח אלגוריתם רקורסיבי לחישוב האיבר ה- n -י בסדרה זו (הקלט לפעולה יהיה המקום ה- n -י והערך שיוחזר יהיה ערכו של האיבר במקום זה).



שאלה 1.3.2

רונן נמצא בראש סולם בעל n שלבים. ברצונו לרדת מהסולם. בכל פעם הוא יכול לרדת שלב בודד או שני שלבים. פתח אלגוריתם רקורסיבי שיקבל כפרמטר את מספר השלבים בסולם ויחזיר את מספר הדרכים השונות בהן ניתן להגיע לתחתיתו.

שתי דרכים תקראנה "שונות" אם קיים לפחות קטע אחד השונה ביניהן.

דוגמא : אם מספר 1 מייצג ירידה של שלב אחד, והמספר 2 מייצג ירידה של שני שלבים, אזי 1, 1, 1, 1, 2 והדרך 1, 1, 1, 2, 1 הן שתי דרכים שונות.



שאלה 1.3.3

- פתח אלגוריתם רקורסיבי שיקבל כפרמטר מספר שלם וחיובי n וידפיס את כל המספרים הבינאריים בעלי n ספרות.
- פתח אלגוריתם רקורסיבי שיקבל כפרמטר מספר שלם וחיובי n , יחשב ויחזיר את מספר המחרוזות הבינאריות באורך n שלא מופיע בהן הרצף 11.



1.4 - הפרד ומשול

(דוגמא: מקסימום בסדרה)

נקודת מוצא: סדרת מספרים.

מטרה: מציאת מקסימום בעזרת פירוק הסדרה לשתי תת-סדרות ומציאת המקסימום בכל תת-סדרה בנפרד.

אלגוריתם: מצא-מקסימום (סדרת איברים)

את הסדרה מכילה איבר גודל n

הוא המקסימלי

אחת

מצא-מקסימום (חלק מסדרת המספרים) n - $max1$

מצא-מקסימום (שאר האיברים) n - $max2$

החזר את הערך של $max1$ ו- $max2$

הערות

- בתבנית זו ישנם שני זימונים רקורסיביים (וייתכנו גם יותר), אשר מאופיינים בכך שכל אחד מהם מופנה לחישוב בחלק אחר של תחום הנתונים, או סדרת הנתונים. עם החזרה מן הזימונים הרקורסיביים ישנו עיבוד של התוצאות שהוחזרו, ומוחזרת תוצאה מתאימה ל"רמה" הרקורסיבית הקודמת.
- ניתן להסתכל על חישובים מן הסוג שבתבנית, של "הפרד ומשול", כמקרים מיוחדים של רקורסיה כפולה (תבנית 1.3), שבה אין חוסר יעילות מן הסוג המתואר בהערה האחרונה של רקורסיה כפולה. חישובים אלו הם בדך-כלל יעילים מאד.
- הרעיון של "הפרד ומשול" הינו רעיון שימושי ביותר בפתרון בעיות אלגוריתמיות שונות. רעיון זה הינו למשל הבסיס לאלגוריתם של "מיון-מיזוג" (Merge-Sort). רעיון זה שימושי בחישובים שונים המתבצעים על מבנה הנתונים "עץ בינארי".
- במקרים רבים מחולק תחום הנתונים (או סדרת הנתונים) לשני חלקים זרים שגודלם שווה. זהו המקרה ב"מיון-מיזוג" וזהו המקרה המומלץ עבור הדוגמה שבתבנית לעיל (כלומר, "חלק מסדרת המספרים" יהיה חצי מן המספרים ש"הועברו" לטיפול "ברמת הרקורסיה" הנוכחית). במקרים כאלו מספר הרמות הרקורסיביות נמוך ביותר ($\log N$) עבור סדרת נתונים בגודל N). ומכאן היעילות הרבה של שימוש בתבנית זו. כדאי להיווכח בכך באמצעות פרישת ביצוע לדוגמה.

Pascal

(* *right*-ו-*left* המערך ואת קצות המערך במערך ואת קצות המערך ואת קצות המערך ומחזיר את ערך האיבר הגדול ביותר.
*) הנחות: המערך וערכי הקצה מאותחלים ותקינים.

```
function max_in_arr (var a: arr_type; left, right: integer): integer;
var
  middle, max1, max2: integer;
begin
  if left = right then
    max_in_arr := a[left]
  else
    begin
      middle := (left + right) div 2;
      max1 := max_in_arr (a, left, middle);
      max2 := max_in_arr (a, middle+1, right);
      max_in_arr := max (max1, max2);
    end;
end;
```

C

/* *right*-ו-*left* המערך ואת קצות המערך במערך ואת קצות המערך ומחזיר את ערך האיבר הגדול ביותר.
*/ הנחות: המערך וערכי הקצה מאותחלים ותקינים.

```
int max_in_arr (arr_type a, int left, int right)
{
  int middle, max1, max2;
  if (left == right)
    return a[left];
  else
  {
    middle = (left + right) / 2;
    max1 = max_in_arr (a, left, middle);
    max2 = max_in_arr (a, middle+1, right);
    return max (max1, max2);
  }
}
```

הפעולה *max* מקבלת שני ערכים שלמים ומחזירה את הערך הגדול מביניהם.

שאלות

שאלה 1.4.1

נתון מערך בגודל N המכיל מספרים. פתח אלגוריתם שיחזיר את סכום איברי המערך. חישוב הסכום יעשה בשיטת הפרד ומשול.

■

שאלה 1.4.2

מערך יקרא "ממויין עולה" אם בכל תת-מערך חלקי לו, סכום האיברים של מחצית שמאלית קטן מסכום האיברים של מחצית ימנית. (הנחייה: אם אורך תת-המערך אי-זוגי מספר איברי תת המערך השמאלי יהיה קטן ב-1 ממספר איברי תת-המערך הימני)

דוגמא: המערך $2, 4, 7, 8, 9, 12, 17, 20$ נקרא מערך "ממויין עולה"
הסכום $58 > 21 =$ הסכום

פתח אלגוריתם רקורסיבי, בשיטת הפרד ומשול, שיקבל מערך בגודל N ($N \geq 2$) ויחזיר 'אמת' אם המערך "ממויין עולה" ו-'שקר' אחרת.

■

שאלה 1.4.3

פתח אלגוריתם רקורסיבי, בשיטת הפרד ומשול, שיקבל מערך בגודל N ($N \geq 2$), ויחזיר את ההפרש הגדול ביותר בין שני איברים כלשהם במערך.

■

שאלה 1.4.4

נתון מערך בגודל N המכיל מספרים. פתח אלגוריתם רקורסיבי שיבצע חיפוש-בינארי. הפעולה תבוצע בשיטת הפרד ומשול.

■

שאלה 1.4.5

נתון מערך בגודל N המכיל מספרים. פתח אלגוריתם רקורסיבי שיבצע מיון-מיזוג. הפעולה תבוצע בשיטת הפרד ומשול.

■

שאלה 1.4.6

נתון מערך בגודל N המכיל מספרים. פתח אלגוריתם רקורסיבי שיבצע מיון-מהיר. הפעולה תבוצע בשיטת הפרד ומשול.

(על מיון-מהיר Quick Sort ניתן לקרוא באתר:

<http://www.cs.hope.edu/~algnim/ccaa/selection.html>)

■

1.5 - רקורסיה דו מימדית

(דוגמא: דרכים בסריג)

נקודת מוצא: נקודה ברביע החיובי.

מטרה: חישוב מספר הדרכים האפשריות להגעה לראשית הצירים כאשר מותר לצעוד צעד אחד שמאלה או צעד אחד למטה.

אלגוריתם: דרכים (x,y)

$x = 0$ / $y = 0$ / $x = 0$ / $y = 0$ / $x = 0$ / $y = 0$

החזרה 1

אחרית

החזרה דרכים $(x-1,y)$ + דרכים $(x,y-1)$

הערות

- בתבנית זו ישנם שני זימונים רקורסיביים, שבכל אחד מהם שני פרמטרים מספריים, ובכל אחד מהם מוקטן פרמטר אחר. ההתייחסות לכל אחד מן הפרמטרים היא כאל "מימד" נפרד, ולכן כל אחד מן הזימונים הרקורסיביים מבטא הקטנה של ערך ב"מימד" אחר.
- העצירה של החישוב הרקורסיבי מתקיימת כאשר הערכים בשני ה"מימדים" הם הערכים הבסיסיים (הערכים 0 בדוגמה שבתבנית).
- הדוגמה שבתבנית ממחישה את המתואר לעיל באמצעות חישובים בסריג, שהינו למעשה מטריצה דו-מימדית. כדאי לפרוש דוגמה של מהלך ביצועה.
- פרישת החישוב במקרה של חישוב מספר הדרכים בסריג תציג מסלולים רבים של "ירידה ברקורסיה", שכל אחד מהם מחושב בנפרד. למעשה, כל מסלול מלא הינו דרך נפרדת על הסריג. צורת הכתיבה הרקורסיבית נוחה ואינטואיטיבית. יחד עם זאת, ניתן לייעל את אופן החישוב על-ידי כתיבה לא רקורסיבית, תוך שימוש בטבלה. כתיבה זו הינה מורכבת, ומשלבת את הנושא של "תכנות דינמי", שהוא מעבר לחומר הלימוד בבית-ספר תיכון.
- מספר ה"מימדים" (פרמטרים) יכול להיות גדול מ-2, אך עיקרון ההקטנה של פרמטר נפרד בכל זימון רקורסיבי יישמר. במקרה שהמספר גדול מ-2 נתייחס לתבנית כתבנית של רקורסיה "רב-מימדית".

Pascal

) האלגוריתם מקבל קואורדינטות של נקודה ברביע הראשון ומחזיר (את מספר הדרכים השונות ממנה לנקודה 0,0. הנחה: x,y מאותחלים.

```
function ways (x, y: integer) : integer;
begin
    if (x = 0) or (y = 0) then
        ways := 1
    else
        ways := ways (x-1, y) + ways (x, y-1) ;
end;
```

C

/ האלגוריתם מקבל קואורדינטות של נקודה ברביע הראשון ומחזיר / את מספר הדרכים השונות ממנה לנקודה 0,0. הנחה: x,y מאותחלים.

```
int ways (int x, y)
{
    if (x == 0 || y == 0)
        return 1;
    else
        return (ways (x-1, y) + ways (x, y-1));
}
```

שאלות

שאלה 1.5.1

במשחק כדורסל יש שלושה סוגי קליעות:
א. מעבר לקשת - מקנה 3 נקודות.
ב. בטווח הקשת - מקנה 2 נקודות.
ג. קליעת עונשין - מקנה 1 נקודות.

פתח אלגוריתם אשר יקבל כפרמטר את התוצאה הסופית של המשחק $x:y$ ויחזיר את מספר האפשרויות השונות בהן ניתן להגיע לתוצאה זו.

■

שאלה 1.5.2

נתונה הפונקציה הרקורסיבית הבאה:

הפונקציה בשפת Pascal

```
(* _____ : טענת כניסה :  
_____ *)
```

```
function trouble (a, b: real) : real;  
begin  
  if a >= b then  
    trouble := (a + b) / 2  
  else  
    trouble := trouble (trouble (a+2, b-1) , trouble (a+1, b-2));  
end;
```

C הפונקציה בשפת

```
/* _____ : טענת כניסה :  
_____ */
```

```
float trouble (float a, float b)  
{  
  if (a >= b)  
    return (a + b) / 2.0;  
  else  
    return trouble (trouble (a+2, b-1) , trouble (a+1, b-2));  
}
```

א. מה יוחזר כתוצאה מהקריאה הרקורסיבית : trouble (3, 6)
תאר את תוצאת החישוב בעזרת עץ מעקב.

ב. מה יוחזר כתוצאה מהקריאה הרקורסיבית : trouble (0, 6)
תאר את תוצאת החישוב בעזרת עץ מעקב.

ג. השלם את טענות הכניסה והיציאה.

■

שאלה 1.5.3

צעד של פרש בלוח שחמט מורכב מתנועה שתי משבצות לכיוון צפון. דרום. מזרח או מערב ותנועה של משבצת אחת לימין או לשמאל. פתח אלגוריתם רקורסיבי שיחשב: בכמה דרכים שונות ניתן להגיע ממשבצת (8,8) למשבצת (1,1) במגבלות הבאות:

א. ב-10 צעדים בדיוק.

ב. ב-4 צעדים בדיוק.

ג. ב-11 צעדים בדיוק.



שאלות סיכום

שאלה 1

פתח אלגוריתם רקורסיבי המקבל כפרמטר מספר שלם ומחזיר את סכום הספרות הזוגיות פחות סכום הספרות האי זוגיות.

■

שאלה 2

פתח אלגוריתם רקורסיבי בשיטת הפרד ומשול המקבל כפרמטר מערך בגודל N ומחזיר את הערך המקסימלי ואת הערך המינימלי מבין איברי המערך.

■

שאלה 3

מערך maze בגודל $(N \times M)$ מייצג מבוך שבו על הנוסע למצוא מסלול מפינה שמאלית עליונה אל הפינה הימנית תחתונה. המבוך מתואר באמצעות 0 המייצג מעבר ו-1 המייצג קיר. הנוסע רשאי לנוע ממשבצת פנויה למשבצת פנויה הצמודה לה באותה שורה או באותה עמודה, אך אסור לו לדלג מעל משבצת או לנוע באלכסון.

תחילת המסלול בפינה השמאלית עליונה של המבוך וסיומו בפינה הימנית תחתונה. פינות אלו פנויות למעבר (כלומר מכילות את הערך 0).

פתח אלגוריתם המקבל כפרמטר את המערך maze ומדפיס הודעה אם קיים מסלול במבוך. אם קיים - יש להדפיס את מיקום המשבצות מסוף המסלול ועד תחילתו.

■

דרך אחרת לספירת מספר הקירות במערך היא בדרך של הפרד ומשול: מספר הקירות במבצר הוא הסכום שנוצר מהאיבר של האלכסון הראשי + מספר הקירות בשורה של איבר זה + מספר הקירות בעמודה של איבר זה + מספר הקירות בשאר המערך (הדו-מימדי).

מספר-הקירות-במבצר (arr, i)
 } פעולה המקבלת את מטריצה arr המתארת מבצר, ומספר i הנמצא על האלכסון הראשי ומחזירה את מספר התאים במטריצה שערכם הוא 1.
 { הנחות: איברי המערך הם 0 או 1, $i \leq \text{MAX_ROW}$, $j \leq \text{MAX_COL}$

אם $(i \leq \text{MAX_ROW})$ ו- $(i \leq \text{MAX_COL})$ אזי
 החזר את $\text{arr}[i, i]$
 + מספר-הקירות-בשורה (arr, i, i+1)
 + מספר-הקירות-בעמודה (arr, i+1, i)
 מספר-הקירות-במבצר (arr, i+1)

מספר-הקירות-בשורה (arr, i, j)
 } פעולה המחזירה את מספר התאים בשורה i במערך דו-מימדי arr שערכם 1.
 { הנחות: איברי המערך הם 0 או 1, $i \leq \text{MAX_ROW}$, $j \leq \text{MAX_COL}$

אם $j \leq \text{MAX_COL}$ ואם $i \leq \text{MAX_ROW}$ אזי
 החזר 0
 אחרת -
 החזר את $\text{arr}[i, j]$ + מספר-הקירות-בשורה (arr, i, j+1)

מספר-הקירות-בעמודה (arr, i, j)
 } פעולה המחזירה את מספר התאים בעמודה j במערך דו-מימדי arr שערכם 1.
 { הנחות: איברי המערך הם 0 או 1, $i \leq \text{MAX_ROW}$, $j \leq \text{MAX_COL}$

אם $i \leq \text{MAX_ROW}$ ואם $j \leq \text{MAX_COL}$ אזי
 החזר 0
 אחרת -
 החזר את $\text{arr}[i, j]$ + מספר-הקירות-בעמודה (arr, i+1, j)

אלגוריתם: סמן-וספור-חדרים-במבצר (A)

} פעולה המקבלת מטריצה A המתארת מבצר, ומסמנת וסופרת את מספר גושי התאים במטריצה שערכם 0.

תא יחשב שייך לגוש-תאים אם לפחות אחת הצלעות שלו משותפת לגוש-התאים.

{ הנחות: המטריצה מאותחלת במבנה המבצר: 1 לקיר, ו-0 לחלל
counter ← 0

Row r / שורה שנייה r / שורה לפני אחרונה, r :

Col c / מעמודה שנייה c / עמודה לפני אחרונה, c :

אם $arr[r, c] = 0$, אזי

counter ← 1

סמן-חדר (arr, row, col)

return counter

סמן-חדר (arr, row, col)

} פעולה המקבלת "כתובת" של חדר במבצר arr (שורה = row, עמודה = col)

ומסמנת אותו ואת כל התאים הסמוכים לו שהם חלק מאותו חדר.

{ סימון הביקור בתא יהיה בערך השונה מ-0.

אם $arr[r, c]$ אינו קרי ולא סומן שביקרו בו אזי

arr[r, c] ← 1

סמן-חדר (arr, row, col + 1)

סמן-חדר (arr, row + 1, col)

סמן-חדר (arr, row, col - 1)

סמן-חדר (arr, row - 1, col)



