פרק 8 – יעילות של אלגוריתמים

הצגת הפרק

מטרת הפרק

הצגה אינטואיטיבית של המושגים יעילות של אלגוריתם וזמן ביצוע של אלגוריתם. זמן הביצוע מוסבר תחילה דרך מספר הפעולות המתבצעות במהלך הביצוע, ומיד לאחר מכן דרך המרכיב העיקרי באלגוריתם המשפיע על מספר הפעולות המתבצעות – מספר הפעמים שלולאת האלגוריתם מתבצעת.

הצגת המושגים יעילות וזמן ביצוע מחולקת למספר מרכיבים: ראשית, הצגת יעילות כקנה מידה חדש לבחינת אלגוריתם, נוסף לקנה המידה נכונות שהוצג עד כה. שנית, הבחנה שהמרכיב העיקרי המשפיע על יעילות אלגוריתם, מבחינת זמן ביצוע, הוא מספר הפעמים שהלולאה מתבצעת. שלישית, הדגשה שפיתוח אלגוריתם יעיל, אשר בו מתבצעת הלולאה מספר קטן ככל האפשר של פעמים, נעשה תוך ניתוח וניצול המאפיינים של הקשר קלט-פלט.

מושגים

- יעילות של אלגוריתם
- פעולות יסוד
- זמן ביצוע של אלגוריתם

דגשים ודידקטיקה

- במהלך הצגת חיפוש ברשימה ממוינת אנו מעוניינים בהבחנה אינטואיטיבית בלבד בין זמני הביצוע של דרכים שונות לפתרון בעיה. לכן, אין להעמיק בניתוח שתי הדרכים המתוארות ואין לדון באופן יישומן על ידי תוכנית מחשב. אך כדאי להדגים כל אחת מן הדרכים. כדאי להציג על הלוח רשימה של 30-20 שמות, לבצע חיפוש בשתי הדרכים, ולספור עבור כל דרך את מספר ההשוואות המתבצעות במהלך החיפוש. בחיפוש סדרתי עשוי להיות מספר ההשוואות כאורך הרשימה, ואילו בחיפוש בינארי יהיו לכל היותר 5 השוואות.
- לפני הצגת אלגוריתמים 2 ו-3 בפתרון בעיה 1 כדאי לחזור לדוגמאות הקלט-פלט שבשאלה 8.1, ולעודד את התלמידים לנסות ולזהות מאפיינים של הקלט-פלט אשר על פיהם ניתן ליעל את לולאת האלגוריתם.

פתרונות

שאלה 8.2

ציינו עבור כל אחד מהקלטים הבאים את מספר הפעמים שתתבצע הלולאה בכל אחד משלושת האלגוריתמים שפיתחנו:

א. 1000

ב. 2000

תשובה 8.2

א. 1000 פעמים באלגוריתם 1, 500 פעמים באלגוריתם 2, 31 פעמים באלגוריתם 3.

ב. 2000 פעמים באלגוריתם 1, 1000 פעמים באלגוריתם 2, 44 פעמים באלגוריתם 3.

שאלה 8.3

בקטע התוכנית הבא מחושבת המכפלה של שני נתוני קלט חיוביים שלמים בשימוש בפעולת חיבור בלבד:

```
x = in.nextInt();
y = in.nextInt();
sum = 0;
for(i = 1; i <= x ; i++)
    sum = sum + y;
System.out.println( "The product is " + sum);
```

ידוע שאחד מנתוני הקלט גדול באופן משמעותי מהשני, אך לא ידוע אם זה הנתון הראשון או השני. השתמשו במאפיין זה של הקלט כדי לשנות את קטע התוכנית הנתון כך שיהיה יעיל ככל שניתן.

הדרכה: שימו לב שיש חשיבות לבחירת המשתנה אשר על פיו נקבע מספר הפעמים שתתבצע הלולאה.

מהו מספר הפעמים שתתבצע הלולאה של קטע התוכנית הנתון, ומהו מספר הפעמים שתתבצע הלולאה של קטע התוכנית החדש?

תשובה 8.3

הפתרון היעיל ביותר הוא לרוץ בלולאה עד לערך הקטן יותר מבין x ו-y, ולהשתמש בערך הגדול יותר מביניהם כיחידת הסיכום.

```
x = in.nextInt();
y = in.nextInt();
if(x < y)
{
     limit = x;
     unit = y;
}
else
{
     limit = y;
     unit = x;
}
sum = 0;
for(i = 1; i <= limit ; i++)
    sum = sum + unit;
System.out.println( "The product is " + sum);
```

מספר הפעמים שתתבצע לולאת קטע התוכנית הנתון בשאלה יהיה כערכו של נתון הקלט הראשון, ומספר הפעמים שתתבצע לולאת קטע התוכנית החדש יהיה כגודלו של הערך ה**קטן** מבין שני נתוני הקלט. כאשר נתון הקלט הראשון גדול באופן משמעותי מנתון הקלט השני, מספר הפעמים שתתבצע לולאת קטע התוכנית החדש יהיה קטן יותר באופן משמעותי מן המספר עבור קטע התוכנית הנתון בשאלה.

שאלה 8.4

פתחו אלגוריתם מבלי ליישמו שיהיה יעיל ככל האפשר, והקלט שלו הוא שני מספרים שלמים גדולים מ-1 שאינם מחלקים זה את זה, והפלט שלו הוא כל המספרים השלמים החיוביים המחלקים את שני מספרי הקלט. תארו לפי ערכו של הקלט את מספר הפעמים שתתבצע לולאת האלגוריתם.

תשובה 8.4

האלגוריתם הבא מבטא את הרעיון לפתרון, שהוא שילוב רעיון הפתרון של שאלה 8.3 עם רעיון הפתרון של בעיה 1.

1. *קלוט שני מספרים שלמים חיוביים ב-*num1*, וב-*num2
2. *אם* num2 > num1
   2.1. *השם את השרש של* $\sqrt{num1}$ *ב-*limit
3. *אחרת*
   3.1 *השם את השרש של* $\sqrt{num2}$ *ב-*limit
3. *עבור כל מספר שלם חיובי* i *הקטן מ-*limit ***בצע***:
   3.1. *אם* i *מחלק את* num1 *ובמ את* num2 *ללא שארית*
      3.1.1. *הדפס את ערכו של* i
      3.1.2. *הדפס את ערכם של* num1/i *ושל* num2/i

מספר הפעמים שתתבצע לולאת האלגוריתם יהיה שורשו של הקטן מבין שני נתוני הקלט.

שאלה 8.5 (מתקדמת)

נניח שבבעיה 1 הפלט הדרוש הוא כל המספרים השלמים החיובים הקטנים מ-N ש**אינם** מחלקים את נתון הקלט N. מה יהיה מספר הפעמים שתתבצע הלולאה באלגוריתם לפתרון הבעיה החדשה?

תשובה 8.5

מספר הפעמים שתתבצע לולאת האלגוריתם לפתרון הבעיה החדשה יהיה num-2, כיוון שתחום המספרים שיש לסרוק הוא בין 2 ל-num-1. הפלט יכלול מספרים בין 2 לבין num/2 (כמעט עבור כל קלט) ואת **כל** המספרים בין (num/2)+1 ל-num-1.

שאלה 8.6

יש לפתח אלגוריתם אשר הקלט שלו הוא מספר שלם חיובי N, והפלט שלו הוא כל המספרים השלמים החיוביים אשר קטנים מ-N והשורש שלהם הוא מספר שלם.

למשל עבור הקלט 50 הפלט יהיה 49 36 25 16 9 4 1.

האלגוריתם הבא, הכולל משתנים מטיפוס שלם הוא פתרון אפשרי:

1. *קלוט מספר שלם חיובי* num-ב
2. *עבור כל מספר שלם חיובי* i *שקטן מ-*num *בצע*:
   2.1. *אם השורש של* i *הוא מספר שלם*
      2.1.1. *הדפס את ערכו של* i

א. מהו מספר הפעמים שתתבצע הלולאה של האלגוריתם הנתון?

ב. ישמו בביטוי בוליאני בשפת Java את התנאי "השורש של i הוא מספר שלם"

ג. אפשר לכתוב אלגוריתם אשר זמן-הביצוע שלו יהיה קצר בהרבה מהאלגוריתם הנתון, וזאת ב"ייצור" מספרי הפלט, באמצעות העלאה בריבוע של כל המספרים השלמים אשר ריבועם קטן מן הקלט.

האלגוריתם החלקי הבא מבוסס על הרעיון המתואר:

1. *קלוט מספר שלם חיובי* num-ב
   1.1. *עבור כל מספר שלם חיובי* i *הקטן מ-*_____ *בצע*:
      1.1.1. *הדפס את ערכו של* i^2

השלימו את האלגוריתם ותארו את מספר הפעמים שתתבצע הלולאה של אלגוריתם זה.

תשובה 8.6

א. מספר הפעמים שתתבצע הלולאה של האלגוריתם הנתון הוא num.

ב. `(int)Math.sqrt(num)== Math.sqrt(num)`

ג. כותרת הלולאה באלגוריתם היעיל תהיה:

עבור כל מספר שלם חיובי i *הקטן מ-* $\sqrt{\text{num}}$ *בצע*:

מספר הפעמים שתתבצע הלולאה באלגוריתם זה הוא הערך השלם של $\sqrt{\text{num}}$

שאלה 8.7

יש לפתח וליישם אלגוריתם אשר הקלט שלו הוא שני מספרים שלמים חיוביים, כך שהמספר השני גדול מהראשון. הפלט הדרוש הוא הכפולות של המספר הראשון אשר קטנות מהמספר השני או שוות לו. למשל עבור הקלט 1000 200 הפלט יהיה 1000 800 600 400 200.

משפטי התוכנית הבאים הם יישום של אלגוריתם לפתרון הבעיה:

```
x = in.nextInt();
y = in.nextInt();
for (i = x; i <= y; i++)
    if (i % x == 0)
        System.out.println(i);
```

א. כמה פעמים תתבצע הלולאה עבור הקלט 1000 200?

ב. תארו בצורה כללית את מספר הפעמים שתתבצע הלולאה על פי ערכי נתוני הקלט x ו-y.

ג. בפתרון המוצג i גדל בקפיצות של 1. ניתן לשפר את יעילות הפתרון הנתון בשינוי הקפיצות של i לקפיצות גדולות מ-1, קפיצות אשר מתאימות למרחק בין זוג מספרי פלט עוקבים (שימו לב שהמרחק בין כל זוג מספרי פלט עוקבים הוא אחיד). כתבו פתרון יעיל יותר המבוסס על הרעיון המתואר, ותארו את מספר הפעמים שתתבצע לולאת הפתרון החדש.

תשובה 8.7

א. 801

ב. מספר הפעמים שתתבצע הלולאה הוא: y-x+1.

ג. התכונה המאפיינת את ערכי הפלט היא שהמרחק בין כל זוג מספרי פלט עוקבים הוא כגודל נתון הקלט הראשון x. לכן ניתן להגדיל את i בקפיצות בגודל x:

```
x = in.nextInt();
y = in.nextInt();
for (i = x; i <= y; i += x)
        System.out.println(i);
```

מספר הפעמים שתתבצע לולאת הפתרון החדש הוא הערך של x/y.

שימו ♥: שיפור זה הינו שיפור משמעותי. למשל, בפתרון החדש תתבצע הלולאה 5 פעמים בלבד עבור הקלט 1000 200 , לעומת 801 פעמים בפתרון המוצג בתחילת השאלה.

שאלה 8.8

נתונה לולאת ה-**for** הבאה:

```
for (i = 1 ; i <= 30000 ; i++)
    if (i % 500 == 0)
        System.out.println(i);
```

א. מהו מספר הפעמים שתתבצע הלולאה?

ב. מהי מטרת הלולאה?

ג. כתבו לולאה יעילה הרבה יותר להשגת אותה המטרה. מהו מספר הפעמים שתתבצע הלולאה היעילה שכתבתם? פי כמה מספר זה קטן מתשובתכם בסעיף א?

תשובה 8.8

א. 30000 פעמים

ב. מטרת הלולאה היא הצגה כפלט של כל הכפולות של 500 הקטנות או שוות ל-30000.

ג. לולאה יעילה יותר להשגת אותה מטרה:

```
for (i = 500; i <= 30000; i += 500)
      System.out.println(i);
```

לולאה זו תתבצע 60 פעמים בלבד.