תרגילים ביעילות: ניתוח יעילות

תרגיל 5: ניתוח ☆

א. עבור כל אחד מקטעי התוכניות הבאים, נתח את יעילותם לפי אורך קלט, צעד בסיסי, פונקצית זמן הריצה וסיבוכיות זמן הריצה.

ב. השווה בין יעילותם. איזה קטע יעיל יותר? נמק.

א

```
for (int i = 1 ; i<=n ; i++)
    if (i%5 == 0)
        Console.WriteLine (i);
```

ב

```
for (int i = 5 ; i <= n ; i = i+5)
    Console.WriteLine (i);
```

תרגיל 6: ניתוח ☆☆

```
static int Points(int[] a)
{
  int i = 0, count = 0;
  while (i<a.Length-1)
  {
      while (i<a.Length-1 && a[i] == a[i+1] )
            i++;
      i++;
      count++;
  }
  if (a[a.Length-1] ==a[a.Length-2])
      return count-1;
  return count;
}
```

א. מה מטרת הפעולה?

ב. מה סיבוכיות זמן הריצה של הפעולה? נמק תשובתך.

תרגיל 7: ניתוח ☆☆

א. עבור כל אחת מהפעולות הבאות: נתח את יעילותן לפי אורך קלט, צעד בסיסי, פונקצית זמן הריצה וסיבוכיות זמן הריצה.

ב. השווה בין יעילותן. איזו פעולה יעילה יותר? נמק.

פעולה א

```
static void P1(int num)
{
  double top = Math.Sqrt(num);
  for (int i = 1 ; i<=top ; i++)
   if (num % i == 0)
   {
      Console.WriteLine (i+ " "+ num/i);
      if (i!=num/i)
         Console.WriteLine (num/i+ " "+ i);
   }
}
```

פעולה ב

```
static void P2 (int num)
{
 for (int i = 1 ; i<= num ; i++)
    for (int j=1 ; j<= num ; j++)
       if (i*j == num)
          Console.WriteLine(i+ " "+j);
}
```

תרגיל 8: מנייה ☆☆

נתון מערך המכיל את 40 ציוני תלמידי הכיתה במבחן במדעי המחשב. המורה מעוניין לדעת כמה תלמידים בכתה קיבלו את כל אחד מהציונים 0-100. הצע שני פתרונות: האחד בעזרת מערך מונים, והשני ללא מערך עזר, והשווה בין יעילותם.

נסרק עם CamScanner

תרגיל 9: איברים משותפים ★★

כתוב פעולה המקבלת שני מערכים ממוינים של מספרים שלמים ומדפיסה את האיברים המשותפים לשני המערכים. על הפעולה להיות בסיבוכיות O(n).

דגשים לסיכום הפרק

✓ **פונקצית זמן ריצה**

- פונקצית זמן הריצה היא פונקציה המתארת את מספר יחידות הזמן שידרשו לביצוע אלגוריתם. מחשבים את פונקצית זמן הריצה על-ידי מניית ההוראות שהאלגוריתם מבצע. החישוב מתייחס להחלטה שכל הוראה אורכת יחידת זמן אחת.
- פונקצית זמן הריצה מחושבת תמיד עבור **המקרה הגרוע ביותר (Worst Case)**. המשמעות של המקרה הגרוע ביותר היא התייחסות למקרה בו האלגוריתם יבצע את המספר הגדול ביותר של הוראות, גם אם בדרך כלל הוא מבצע פחות.

✓ **סיבוכיות** זמן הריצה של אלגוריתם - **סדר הגודל** של האלגוריתם - נקבעת לפי הגורם המשמעותי ביותר בפונקצית זמן הריצה, תוך התעלמות מהקבוע הכלול באיבר זה מאחר ובערכים גדולים של אורך הקלט הוא הופך להיות זניח. סיבוכיות זמן הריצה מסומנת ב-O. כאשר משך זמן הריצה של האלגוריתם קבוע (אינו תלוי באורך קלט), הסיבוכיות היא O(1).

- כאשר אלגוריתם א' מזמן אלגוריתם ב' ומעוניינים לבחון את יעילותו של אלגוריתם א', יש להתחשב בהוראות המתבצעות באלגוריתם ב'.
- בדרך כלל, כאשר רוצים לחשב סיבוכיות זמן ריצה, לא מחשבים קודם את פונקצית זמן הריצה, אלא מעריכים לפי מבנה הלולאות. יש לזכור להתייחס גם ללוגיקה ולא רק למבנה הלולאות. לדוגמה לולאה מקוננת אינה בהכרח בעלת סיבוכיות $O(n^2)$ (ראה תרגיל 6).

✓ **צעד בסיסי** הוא קבוצת הוראות החוזרות על עצמן הכי הרבה פעמים במהלך ביצוע אלגוריתם. משך זמן הביצוע של הצעד הבסיסי הוא קבוע ואינו תלוי באורך הקלט.

✓ **אורך הקלט**

- אורך הקלט הוא הגורם המשפיע על "כמות העבודה" שהאלגוריתם מבצע.
- המונחים צעד בסיסי ואורך קלט מתייחסים להערכת משך ביצוע של לולאות.
- שים ♥: למרות שהמילה קלט מופיעה במונח "אורך הקלט" היא אינה קשורה בקלט של מספרים.
- הגדרת אורך הקלט צריכה תמיד להתייחס לפרמטר כלשהו **הנהוג לסימון כ- n**, ויש להגדיר את מה n מייצג. הגדרת המשמעות של n היא הכרחית. השימוש ב- n הוא כללי ולא תלוי במשתנים בהם משתמש האלגוריתם. n יכול להיות גדול ככל שיידרש.

✓ קיימים שני סוגים של **שיפור בסיבוכיות של אלגוריתמים**:

שיפור בסדר גודל – כאשר השיפור הוא בפונקציה המתארת את הסיבוכיות.

שיפור בקבוע – כאשר הפונקציה המתארת את הסיבוכיות זהה, אך הקבוע הכופל אותה יותר קטן.

כאשר אלגוריתם אינו תלוי באורך קלט אלא מבצע תמיד מספר מקסימלי קבוע של פעולות – גם אם הוא גדול מאד – הסיבוכיות של האלגוריתם היא **סיבוכיות קבועה ומסומנת ב- O(1).**

נסרק עם CamScanner