ניתוח זמן ריצה

(על קצה המזלג)

קרן כליף

ביחידה זו נלמד:

- מהו ניתוח זמן ריצה
  - מוטיבציה
  - הגדרה
- ניתוחי זמן ריצה בסיסיים
  - קבוע
  - לינארי
  - ריבועי
  - לוגריתמי

משפחות של פונקציות

- לינאריות:

כאשר a ו- b קבועים $f(n) = a \cdot n + b$

- ריבועיות:

כאשר a,b ו- c קבועים $f(n) = a \cdot n^2 + b \cdot n + c$

- מעריכיות (אקספוננציליות):

$f(n) = 2^n$

ניתוח זמן ריצה - מוטיבציה

- כאשר אנחנו מריצים את התוכניות שלנו, נראה לנו שהן **רצות מאוד מהר**, ואנחנו לא עוצרים לחשוב כמה עבודה באמת מבוצעת ע"י המחשב
- ב**"עולם האמיתי"** לעיתים יש חשיבות לכל מיקרו שנייה שהתוכנית רצה
- נרצה **לדעת להעריך** האם התוכנית שכתבנו יעילה, כלומר כמה עבודה המחשב באמת מבצע
- נרצה **לבחון** האם יש דרך לכתוב את התוכנית בצורה יותר יעילה

זמן ריצה - הגדרה

- זמן ריצה של תוכנית הוא סדר גודל של **מספר הפעולות שהתוכנית מבצעת ביחס לגודל הקלט**
- סדר גודל אינו מספר מדויק אלא הערכה של מספר הפעולות המבוצעות בתוכנית

ניתוח זמן ריצה – זמן ריצה קבוע לעומת זמן ריצה תלוי משתנה

- כמה זמן לוקח למחשב לחשב את:

```
x = x + y;
```

  - התשובה תלויה במהירות המחשב, אך הנחה סבירה היא שפעולה זו על אותו מחשב בזמנים שונים תיקח זמן זהה

- כמה זמן לוקח למחשב לחשב את:

```
for (i=1 ; i <= n ; i++)
   x += i;
```

  - במקרה זה זמן הריצה תלוי בגודלו של n שאינו ידוע מראש, ולכן צריך לקחתו בחשבון בזמן הניתוח התיאורטי, שכן יהיה הבדל משמעותי אם n=10 או n=100,000,000...

ניתוח זמן ריצה – זמן ריצה קבוע O(1)

- כאשר מספר הפעולות של התוכנית קבוע בכל הרצה, נאמר כי זמן הריצה הוא קבוע, ונסמנו ב- O(1)

```
void main()
{
    int  num;
    printf("Please  enter  a  number: ");
    scanf("%d", &num);
    printf("The  number  is  %d \n",  num);
}
```

- בתוכנית זו מספר הפעולות תמיד יהיה זהה, לא משנה מה ערכו של num

ניתוח זמן ריצה – זמן ריצה קבוע O(1)

מציאת הערך הגדול והערך הקטן:

```
void main()
{
     int num1, num2, small, big;

     printf("Please enter 2 numbers: ");
    scanf("%d%d", &num1, &num2);

     small = num1;
     big = num2;

     if (small > big)
     {
          small = num2;
         big = num1;
     }
}
```

בעת חישוב ניתוח זמן ריצה נסתכל תמיד על המקרה
הגרוע ביותר, כלומר המקרה בו יהיו הכי הרבה פעולות ולכן בדוגמא זו נניח כי ה- if מבוצע

- עדיין במקרה זה מספר הפעולות קבוע בכל הרצה, אפילו אם ערכם של num1 ו- num2 מאוד גדול, ולכן זמן הריצה של תוכנית זו הוא O(1)

ניתוח זמן ריצה – תלות בגודל הקלט (לינארי)

חישוב עצרת:

```
void main()
{
    int i, num, fact=1;

    printf("Enter a number: “);
    scanf(“%d”, &num);
    for (i=1 ; i <= num ; i++)
      fact *= i;
    printf(“%d != %d \n”, num, fact);
}
```

מספר הפעולות בתוכנית זו **תלוי בערכו של num**:
התוכנית תבצע תמיד את הפעולות הקבועות של קבלת הקלט והדפסת התוצאה, אבל מספר הפעמים שהלולאה תרוץ תלוי בקלט.

- מספר הפעולות בתוכנית זו הוא O(1) עבור הפעולות הקבועות ועוד O(1) פעולות בכל איטרציה של הלולאה, ומאחר ויש num איטרציות:

$O(main) = O(1) + num*O(1) = O(1) + O(num*1) = O(1) + O(num)$

כאשר num מאוד גדול זמן הריצה הקבוע זניח ולכן נאמר כי:

$O(main) = O(1) + O(num) = O(num)$

ניתוח זמן ריצה – תלות בגודל הקלט (לינארי) (2)

הדפסת המספרים הזוגיים עד מספר מסוים:

```
void main()
{
     int num, i;

    printf("Please enter a number: “);
    scanf(“%d”, &num);

    printf("All even numbers from 1 to %d:“, num);
    for (i=0 ; i < num  ; i+=2 )
        printf(“%d “, i);

    printf(“\n”);
 }
```

- מספר הפעולות בתוכנית זו הוא O(1) עבור הפעולות הקבועות ועוד O(1) פעולות בכל איטרציה של הלולאה, ומאחר ויש num/2 איטרציות:

$$O(main) = O(1) + (num/2)*O(1) = O(1) + O((num/2)*1) = O(1) + O(num/2) = O(num/2)$$

- כאשר num מאוד גדול זמן הריצה הקבוע זניח ולכן נאמר כי:

$$O(main) = O(num/2) = O(num)$$

ניתוח זמן ריצה – תלות בגודל הקלט (ריבועי)

הדפסת מלבן:

```
void main()
{
    int width, height, i, j;

    printf("Enter a rectangle's width and height: “);
    scanf(“%d%d”, &width, &height);
    for (i=1 ; i <= height ; i++)
    {
        for (j=1 ; j <= width ; j++)
            printf(“*”);
        printf(“\n”);
     }
}
```

O(1)

width*O(1) = O(width)

heigth*O(width) = O(height*width)

סה"כ זמן הריצה של התוכנית: O(height *width)

אם width == height == n
סדר גודל זמן הריצה היה $O(n^2)$

ניתוח זמן ריצה – תלות בגודל הקלט (ריבועי)

הדפסת משולש:

```
void main()
{
   int num, i, j;

   printf("Please enter the base of the triangle: ");
   scanf("%d", &num);

   for (i=1 ; i <= num ; i++)
   {
       for (j=1 ; j <= i ; j++)
     printf("*");
       printf("\n");
   }
}
```

בדוגמא זו מספר הפעולות בלולאה הפנימית שונה בכל איטרציה, ולכן נבדוק ביתר קפידה כמה פעולות באמת מתבצעות: באיטרציה הראשונה של הלולאה הראשית לולאה הפנימית תרוץ פעם אחת, באיטרציה השנייה פעמיים וכו'..

- סדר גודל זמן הריצה של התוכנית הוא מספר הפעולות בסה"כ של הלולאה הראשית: 2+1+...+num, וזוהי סדרה חשבונית שסכומה:

$$O(main) = O(num*(num+1)/2) = O((num^2+num)/2) = O(num^2 + num) = O(num^2)$$

ניתוח זמן ריצה – תלות בגודל הקלט (לוגריתמי)

הדפסת חזקה:

```
void main()
{
      int num, i;

      printf("Please enter a number: “);
      scanf(“%d”, &num);

      printf("All 2 powers till %d:\n“, num);
      for (int i=1 ; i <= num ; i*=2)
            printf(“%d “, i);

      printf(“\n”);
{
```

- מספר הפעולות בתוכנית זו הוא O(1) עבור הפעולות הקבועות ועוד O(1) פעולות בכל איטרציה של הלולאה, אך מהי כמות האיטרציות של הלולאה?

ניתוח זמן ריצה – תלות בגודל הקלט (לוגריתמי)

- למשל עבור n=128:
  - כמות האיטרציות היא כמות הפעמים שנכפיל ב- 2 עד שנגיע ל- n:

    1□2□4□8□16□32□64□128
  - אפשר לשאול כמה פעמים הכפלנו את 2 בעצמו עד קבלת 128:

    $128 = 2^{?}$
  - וזהו בדיוק פעולת ה- log:

    $\log_2 128$
- לכן, כמות הפעולות שתרוץ עבור לולאה שקצב ההתקדמות שלה הוא כפל/חילוק הוא log

ניתוח זמן ריצה לוגריתמי

כלומר, ככל שבסיס הלוג יותר גדול,
כך כמות הפעולות קטנה (יותר יעיל)

```
for (int i=1 ; i <= 240 ; i*=2)
    …..
```

- כמות האיטרציות:

1 □2□4□8□16□32□64□128□256

$\log_2 256 = 8$

```
for (int i=1 ; i <= 240 ; i*=3)
    …..
```

- כמות האיטרציות: 1□3□9□81□243

$\log_3 243 = 5$

השוואת סדרי גודל

- כאשר אנו כותבים תוכנית נשאף שזמן הריצה יהיה נמוך ככל שניתן

- נזכור כי:

$$O(1) < O(\log_3 n) < O(\log_2 n) < O(n) < O(n^2) < O(n^3)$$

חיבור סדרי גודל

```
void foo(int n, int m)
{
    int i, j;

    for (int i=1 ; i <= n; i++)
      printf("%d ", i);

    printf("\n");

    for (int i=1 ; i <= m; i++)
      printf("%d ", i);
}
```

מאחר ולא ידוע לנו היחס בין n ל-m,
נאמר שסדר הגודל הוא O(n+m)

ביחידה זו למדנו:

- מהו ניתוח זמן ריצה
  - מוטיבציה
  - הגדרה
- ניתוחי זמן ריצה בסיסיים
  - קבוע
  - לינארי
  - ריבועי
  - לוגריתמי

תרגיל 1: מהו זמן הריצה של התוכניות הבאות?

```
void main()
{
   int n, i;

   printf("Enter a number: ");
   scanf(“%d”, &n);

   printf("All even from 0 till %d:\n“, n);

   ...
}
```

```
for (i=0 ; i <= n ; i+=2)
    printf(“%d  “, i);
```

O(n)

```
for (i=0 ; i <= n ; i++)
{
    if (i%2 == 0)
         printf(“%d  “, i);
}
```

O(n)

ובכל זאת, למרות שהיעילות זהה,
נעדיף את הפתרון העליון

תרגיל 2: מהו זמן הריצה של התוכניות הבאות?

```
void main()
{
   int n, i;

   printf("Enter a number: ");
   scanf(“%d”, &n);

   printf("All 2 power from 1 till %d:\n“, n);

   ...
}
```

```
for (i=1 ; i <= n ; i*=2)
     printf(“%d  “, i);
```

$O(\log_2(n))$

```
for (i=1 ; i <= n ; i++)
{
    // check if i is a 2 power
    int j;
    for (j=1 ; j < i ; j*=2);

    if (j == i)
         printf(“%d  “, i);
}
```

$O(n*\log_2(n))$