

ניתוח סיבוכיות

```
void Stage1(int n) {
    for(int i=0; i<n; i++) { System.out.println("Operation"); }
}
```

$$T(n) = \sum_{i=1}^n 1 = n \in O(n)$$

```
void Stage2(int complexity) {
    for(int i=0; i<complexity; i++) { System.out.println("Operation"); }
}
```

$$T(\text{complexity}) = \sum_{i=1}^{\text{complexity}} 1 = \text{complexity} \in O(\text{complexity})$$

```
void Stage3(int n) {
    for(int i=n; i>0; i--) { System.out.println("Operation"); }
}
```

זוהי ל-Stage1 ולכן:

$$T(n) = \sum_{i=1}^n 1 = n \in O(n)$$

```
void Stage4(int n) {
    for(int i=0; i<n; i++) {
        for(int j=0; j<n; j++) {
            // Basic Operation 1
            // Basic Operation 1
        }
    }
}
```

$$T(n) = \sum_{i=1}^n \sum_{j=1}^n 2 = \sum_{i=0}^n 2 \cdot n = 2 \cdot n \cdot n = 2n^2 \stackrel{c=2, n_0=1}{\sim} O(n^2)$$

```
void Stage5(int n) {
    for(int i=n; i>0; i--) {
        for(int j=0; j<n; j++) {
            // Basic Operation 1
        }
    }
}
```

$$T(n) = \sum_{i=1}^n \sum_{j=1}^n 2 = \sum_{i=0}^n 2 \cdot n = 2 \cdot n \cdot n = 2n^2 \stackrel{c=2, n_0=1}{\sim} O(n^2)$$

```
void Stage4dot1(int n) {
    for(int i=0; i<n; i++) {
        for(int j=0; j<i; j++) {
            // Basic Operation 1
        }
    }
}
```

$$T(n) = \sum_{i=1}^n \sum_{j=1}^i 1 = \sum_{i=1}^n i = 1 + 2 + \dots + n = 1 \cdot S_{[1,n]} = n \cdot \frac{1+n}{2} = \frac{n^2}{2} + n \in O(n^2)$$

או לחלופין ניתן לראות ישיר (הגיון) כי: (הסבר בתרגול)

$$T(n) = 1 \cdot (0 + 1 + 2 + \dots + n) = 1 \cdot S_{[1,n]} = n \cdot \frac{1+n}{2} = \frac{n^2}{2} + n \in O(n^2)$$

```
int f(int i) { return i + 1; }
void Stage6(int n) {
    for(int i=0; i<n; i=f(i)) { System.out.println("Operation"); }
}
```

$$T(n) = \sum_{i=0}^n 1 = n \in O(n)$$

```
int g(int i) { return i * 2; }
void Stage7(int n) {
    for(int i=0; i<n; i=g(i)) { System.out.println("Operation"); }
}
```

∞

```
void Stage8(int n) {
    for(int i=1; i<n; i=g(i)) { System.out.println("Operation"); }
}
```

נשים לב כי בכל איטרציה מצבעים פעולה אחת, נשים לב כי אם נעקוב אחרי i , אז הערכים שהוא מקבל הינם $i = 1, i = 2, i = 4 \dots$ עד לתנאי העצירה ש- $i = n$
 בגלל שאנחנו מחשבים חסם עליון, נניח כי $n = 2^k$ עבור $k \in \mathbb{Z}^+$ ולכן נקבל כי
 $k = \log_2(n)$, מכיוון שבכל איטרציה מבצעים פעולה אחת, נקבל כי הסיבוכיות היא:
 $T(n) = 1 \cdot \log_2 n \in O(\log n)$

```
int g2(int i) { return i * 2020; }
void Stage8dot1(int n) {
    for(int i=1; i<n; i=g2(i)) {
        for(int zvi = 0; zvi < 2021; zvi++) System.out.println("Operation");
    }
}
```

נשים לב כי בכל איטרציה מבצעים 2021 פעולות, נשים לב כי אם נעקוב אחרי i , אז הערכים שהוא מקבל הינם $i = 1, i = 2020, \dots$ עד לתנאי העצירה ש- $i = n$
 בגלל שאנחנו מחשבים חסם עליון, נניח כי $n = 2020^k$ עבור $k \in \mathbb{Z}^+$ ולכן נקבל כי
 $k = \log_{2020}(n)$, מכיוון שבכל איטרציה מבצעים 2021 פעולות, נקבל כי הסיבוכיות היא:
 $T(n) = 2021 \cdot \log_{2020} n \in O(\log n)$

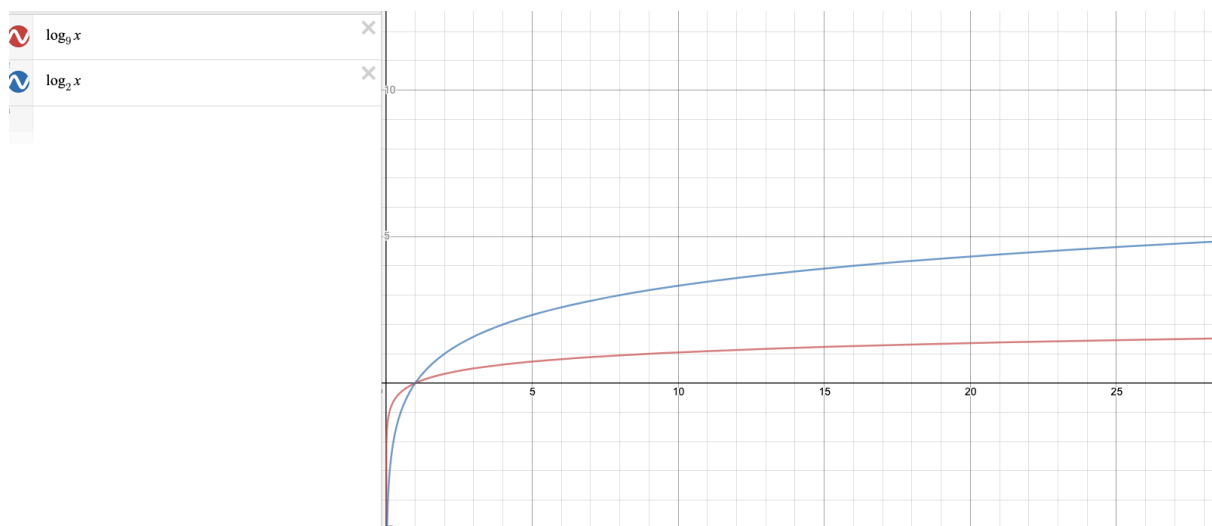
```
int k(int i) { return i * (int)(
    // Note: [0,1)*8 + 2 -> [0,8) + 2 -> [2,10)
    Math.random()*8 + 2); }
void Stage9(int n) {
    for(int i=1; i<n; i=k(i)) { System.out.println("Operation"); }
}
```

נשים לב כי בכל איטרציה מצבעים פעולה אחת, נשים לב כי אם נעקוב אחרי i , אז הערכים שהוא מקבל הינם כפולה של האיטרציה הקודמת במספר בין 2 ל-9 כולל, עד לתנאי העצירה ש- $i = n$

בגלל שאנחנו מחשבים חסם עליון, נניח כי $n = 2^k$ עבור $k \in \mathbb{Z}^+$ ולכן נקבל כי
 $k = \log_2(n)$, מכיוון שבכל איטרציה מבצעים פעולה אחת, נקבל כי הסיבוכיות היא:
 $T(n) = 1 \cdot \log_2 n \in O(\log n)$

עבור המקרה הטוב ביותר, נשים לב כי i יכול לקפוץ בקפיצות של 9, ולכן כמות האיטרציות לכל הפחות הינה $\log_9 n$ ולכן:

$$\log_2 n \geq T(n) \geq \log_9 n$$



```
int p(int i) { return (int) Math.pow(i,2); }
void Stage10(int n) {
    for(int i=5; i<n; i=p(i)) { System.out.println("Operation"); }
}
```

$$5^{2^{2^{\dots}}} = 5^{2^k} \leq n \Rightarrow k \leq \log_2 \log_5(n)$$

```
void Stage11(int n) {
    // for(int i=0; i<n; i++) {
    for(int i=1; i<=n; i*=2) {
        for(int j=0; j<n; j++) {
            // Basic Operation
        }
    }
}
```

$$T(n) = n \cdot \log_2 n$$

```
void Stage12(int n) {
    // for(int i=0; i<n; i++) {
    for(int i=1; i<=n; i*=2) {
        for(int j=0; j<i; j++) {
            // Basic Operation
        }
    }
}
```

נשים לב כי i מקבל ערכים $n, \dots, 4, 2, 1$, לפיכך, נגדיר משתנה k שרוץ מ-0 עד $\log n$, נגדיר את i להיות 2^k ונקבל:

$$T(n) = \sum_{k=0}^{\log n} \sum_{j=1}^i 1 = \sum_{k=0}^{\log n} \sum_{j=1}^{2^k} 1 = \sum_{k=0}^{\log n} 2^k = 2^{\log n + 1} - 1 = 2 \cdot 2^{\log n} - 1 = 2n - 1 = \theta(n)$$

עד עכשיו:

```
// [V]: for(int x=1; x < n; x*=3)
// [V]: for(int x=n; x > 1; i/=3)
// [V]: for(int x=2; x < n; x = Math.pow(x,3))
// [V]: for(int x=n; x > 2; x = Math.pow(x,1/3))
```

```
void Stage13(int n) {
    int x = 1;
    while (x < n) {
        x = x * 3;
    }
}
```

$$T(n) = \log_3 n \in O(\log n)$$

```
void Stage14 (int n){
    int x = n; // Only change this
    while (x > 1) {
        x = x / 3; // Same
    }
}
```

$$T(n) = \log_3 n \in O(\log n)$$

```
void Stage15(int n){
    int x = 2; // Inf if n > 1
    while (x < n) {
        x = x * x * x;
    }
}
```

∞

```
void Stage16(int n){
    int x = 2; // This changed
    while (x < n) {
        x = x * x * x;
    }
}
```

$$2^{3^k} \leq n \Rightarrow k \leq \log_3 \log_2(n)$$

```
void Stage17(int n){
    double x = n; // Only change this
    while (x > 2) {
        x = Math.pow(x, 1/3);
    }
}
```

$$2^{3^k} \leq n \Rightarrow k \leq \log_3 \log_2(n)$$

המרות:

```
forInit;
while(Expression) {
    Statement
    forUpdate;
}
```

```
for(forInit; Expression; forUpdate) { Statement }
```

ולכן:

```
// Stage13: for(int x=1; x < n; x*=3)
// Stage14: for(int x=n; x > 2; i/=3)
// Stage15: for(int x=1; x < n; x = Math.pow(x,3))
// Stage16: for(int x=2; x < n; x = Math.pow(x,3))
// Stage17: for(int x=n; x > 2; x = Math.pow(x,1/3))
```

```
void Stage18(int n) {
    for(int i=2; Math.pow(2,i) < n; i++) {}
}
```

$$2^i \leq n \Rightarrow i \leq \log_2(n)$$