SDA – Seminar 2 – Complexități

- Folosim complexități pentru a compara diferiți algoritmi care efectuează aceeași operație.
- În general, vorbim de complexitate de timp, care depinde de numărul de operații efectuate de un algoritm. Numărul de operații de obicei depinde și de datele de intrare pentru algoritmul respectiv, de aceea în general este notat în funcție de mărimea datelor de intrare *n*, cu T(n).
    - De exemplu $T(n) = 4n^2 + n + \log_2 n + 7$
- În general nu ne interesează T(n), adică numărul exact de operații ci doar clasa din care face parte.
    - O – mare – este o limită superioară pentru T(n).
        - $T(n) \in O(n^2)$, $T(n) \in O(n^3)$, $T(n) \in O(2^n)$, etc.
    - Ω – mare – este o limită inferioară pentru T(n).
        - $T(n) \in \Omega(1)$, $T(n) \in \Omega(n)$, $T(n) \in \Omega(n^2)$
    - Θ – mare – o singură valoare
        - $T(n) \in \theta(n^2)$
- Complexitatea poate să depindă și de valoarea exactă a parametrilor, nu doar de mărimea lor. În acest caz avem 3 valori pentru complexitate:
    - Caz favorabil
    - Caz defavorabil
    - Caz mediu
- În general folosim notația θ (fiind valoarea exactă). Notația O este folosită în 2 cazuri:
    - Avem caz favorabil, defavorabil, mediu.
    - Nu putem calcula notația θ exact. În acest caz folosim cea mai mică valoare O posibilă.

1. Adevărat sau fals?
    a. $n^2 \in O(n^3)$ - Da
    b. $n^3 \in O(n^2)$ – Nu
    c. $2^{n+1} \in \theta(2^n)$ – Da
    d. $n^2 \in \theta(n^3)$ – Nu
    e. $2^n \in O(n!)$ - Da
    f. $\log_{10} n \in \theta(\log_2 n)$ - Da
    g. $O(n) + \theta(n^2) = \theta(n^2)$ - Da
    h. $\Theta(n) + O(n^2) = O(n^2)$ – Da
    i. $O(n) + O(n^2) = O(n^2)$ – Da
    j. $O(f) + O(g) = O(\max\{f,g\})$ – Da
    k. $O(n) + \theta(n) = O(n)$ – Nu – este $\theta(n)$

Algoritm	Complexitate timp				Complexitate spațiu
	CF	CD	CM	Total	
Căutare secvențială	Θ(1)	Θ(n)	Θ(n)	O(n)	Θ(1)
Căutare Binară	Θ(1)	Θ($\log_2 n$)	Θ($\log_2 n$)	O($\log_2 n$)	Θ(1)
Sortare prin selecție	Θ(n^2)	Θ(n^2)	Θ(n^2)	Θ(n^2)	Θ(1) – in place
Sortare prin inserție	Θ(n)	Θ(n^2)	Θ(n^2)	O(n^2)	Θ(1) – in place
Sortare prin metoda bulelor	Θ(n)	Θ(n^2)	Θ(n^2)	Θ (n^2)	Θ(1) – in place
Sortare rapidă	Θ(n $\log_2 n$)	Θ(n^2)	Θ(n $\log_2 n$)	Θ(n^2)	Θ(1) – in place
Sortare prin interclasare	Θ(n $\log_2 n$)	Θ(n $\log_2 n$)	Θ(n $\log_2 n$)	Θ(n $\log_2 n$)	Θ(n)- out of place

2. Construiți un algoritm cu timpul θ(n $\log_2 n$)

```
Pentru i = 1, n execută
      j ← i
      Câttimp j ≠ 0 execută
            j = ⌊j/2⌋
      sf_câttimp
sf_pentru
```

3. Calculați complexitatea pentru cei 2 algoritmi:

```
gasit ← fals
Pentru i = 1, n executa
      Dacă x_i = a atunci
            gasit ← adevarat
      sf_daca
sf_pentru
```

$\left.\begin{matrix} CF: \theta(n) \\ CD: \theta(n) \end{matrix}\right\} => \theta(n)$

```
gasit ← false
câttimp gasit = fals și i < n execută
      Dacă x_i = a atunci
            gasit ← adevarat
      sf_daca
      i ← i + 1
sf_cattimp
```

CF: θ(1)
CD: θ(n)
CM: sunt n + 1 cazuri posibile (orice pozitie sau niciunul)

$$T(n) = \sum_{I \in D} P(I) * E(I) = \frac{1}{n+1} + \frac{2}{n+1} + \ldots + \frac{n}{n+1} = \frac{n*(n+1)}{2*(n+1)} = \frac{n}{2} \in \theta(n)$$

CT: O(n)

4. X este un tablou de întregi nenegativi, fiecare element e <= n

```
k← 0
Pentru i = 1, n execută
      Pentru j = 1, x_i execută
            k ← k + x_j
   sf_pentru
sf_pentru
```

$$T(n) = \sum_{i=1}^{n} \sum_{j=1}^{xi} 1 = \sum_{i=1}^{n} x_i = s\ (suma\ elementelor)$$

Fie $x_i = \begin{cases} 1, dacă\ i\ este\ pătrat\ perfect \\ 0, altfel \end{cases}$ s = √n

T(n) ∈ θ(max {n, s})

5. Se consideră problema de a determina dacă un șir arbitrar de numere $x_1...x_n$ conține cel puțin 2 termeni egali. Arătați că acest algoritm poate fi realizat în $\theta(n \log_2 n)$.
   Soluție: MergeSort pe șir.

6. Calculați complexitatea:

```
Pentru i = 1,n execută
      @op elementară
sf_pentru
i ← 1
k ← adevarat
Câttimp i <= n – 1 și k execută
      j ← i
      k_1 ← adevărat
      Câttimp j <= n și k_1 execută
            @ op elementară (k_1 poate fi modificat)
            j ← j + 1
      sf_câttimp
      i ← i + 1
      @op elementara (k poate fi modificat)
sf_câttimp
```

CF: k, k_1 poate deveni fals după primul pas => θ (n)

CD: k, k_1 nu devin false niciodată

$$T(n) = \sum_{i=1}^{n-1} \sum_{j=i}^{n} 1 = \sum_{i=1}^{n-1} n - i + 1 = \sum_{i=1}^{n-1} n - \sum_{i=1}^{n-1} i + \sum_{i=1}^{n-1} 1 =$$

$$n * (n-1) - \frac{n*(n-1)}{2} + n - 1 \, \epsilon \, \theta \, (n^2)$$

CM: pentru un i fixat, k_1 poate deveni fals după 1,2, ..., n-i+1 execuții.

Prob: $\frac{1}{n-i+1}$

$$\frac{1}{n-i+1} + \frac{2}{n-i+1} + \ldots + \frac{n-i+1}{n-i+1} = \frac{(n-i+1)*(n-i+2)}{2\,(n-i+1)} = \frac{(n-i+2)}{2}$$

Pentru ciclul câttimp exterior, k poate deveni fals după 1, 2, ..., n-1 iterații.

$$T(n) = \frac{1}{n-1} * \frac{n-1+2}{2} + \frac{2}{n-1} * \frac{n-2+2}{2} + \ldots + \frac{n-1}{n-1} * \frac{n-(n-1)+2}{2} =$$

$$\frac{1}{2*(n-1)} * \sum_{i=1}^{n-1} i*(n-i+2) = \cdots$$

$$= \frac{1}{2*(n-1)} * \left(\frac{n*(n-1)*n}{2} - \frac{(n-1)*n*(2n-1)}{6} + 2 * \frac{(n-1)*n}{2} \right)$$

$$= \frac{1}{2} * \left(\frac{n^2}{2} - \frac{2*n^2 - n}{6} + n \right) = \frac{1}{2} * \left(\frac{3n^2 - 2n^2 + 5n}{6} \right) \in \theta(n^2)$$

Complexitatea totală: $O(n^2)$

7. Calculați complexitatea:

```
Subalg p(x,s,d) este:
      Daca s < d atunci
            m ← [(s+d)/2]
            Pentru i = s, d-1, execută
                  @op. Elementare
            sf_pentru
            Pentru i = 1,2 execută
                  p(x, s, m)
            sf_pentru
      sf_daca
sf_subalg

apel: p(x, 1, n)
```

$$T(n) = \begin{cases} 2 * T\left(\frac{n}{2}\right) + n \, , daca \, n > 1 \\ 0, \quad altfel \end{cases}$$

$T(n) = 2T\left(\frac{n}{2}\right) + n$ $\qquad n = 2^k$

$$T(2^k) = 2 * T(2^{k-1}) + 2^k$$
$$2 * T(2^{k-1}) = 2^2 * T(2^{k-2}) + 2^k$$
$$2^2 * T(2^{k-2}) = 2^3 * T(2^{k-3}) + 2^k$$
$$\dots$$
$$2^{k-1} * T(2) = 2^k * T(1) + 2^k$$

$$T(2^k) = 2^k * T(1) + k * 2^k = k * 2^k = n * log_2 n \rightarrow T(n) \in \theta(n \log_2 n)$$

8. Calculați complexitatea:

```
s ← 0
Pentru i = 1, n^2 execută
      j ← i
      Câttimp j ≠ 0 execută
            s ← s + j
            j ← j – 1
      sf_câttimp
sf_pentru
```

$$T(n) = \sum_{i=1}^{n^2} \sum_{j=1}^{i} 1 = \sum_{i=1}^{n^2} i = \frac{n^2 * (n^2 + 1)}{2} \in \theta(n^4)$$

9. Calculați complexitatea:

```
s ← 0
pentru i = 1, n^2 execută
      j ← i
      Câttimp j ≠ 0 execută
            s ← s + j – 10 * [j/10]
            j ← [j/10]
      sf_câttimp
sf_pentru
```

- Ciclul câttimp se execută de $\log_{10}$ i ori.
- $T(n) \in \theta(n^2 \log_{10} n) \Rightarrow T(n) \in \theta(n^2 \log_2 n)$

10. Calculați complexitatea:

```
Subalg operație(n, i) este
      Daca n > 1 atunci
            i ← 2 * i
            m ← [n/2]
            operatie (m, i-2)
            operatie (m, i-1)
            operatie (m, i+2)
            operatie (m, i+1)
      altfel
            scrie i
      sf_daca
sf_subalg
```

$$T(n) = \begin{cases} 4 * T\left(\frac{n}{2}\right) + 2, n > 1 \\ 1, \quad altfel \end{cases}$$

$$T(n) = 4 * T\left(\frac{n}{2}\right) + 2$$
$$T(2^k) = 4 * T(2^{k-1}) + 2$$
$$4 * T(2^{k-1}) = 4^2 * T(2^{k-2}) + 4 * 2$$
$$4^2 * T(2^{k-2}) = 4^3 * T(2^{k-3}) + 4^2 * 2$$
$$\dots$$
$$4^{k-1} * T(2) = 4^k * T(1) + 4^{k-1} * 2$$
$$T(n) = 4^k + 2 * (4 + 4^2 + \dots + 4^{k-1}) = 4^k + 2 * \frac{4^k - 2}{3} = n^2 + 2 * \frac{n^2 - 2}{3} \in \theta(n^2)$$