

## Seminar 11 - Exerciții complexități

---

1. \*Căror clase de complexități aparține funcția următoare? \*

$$T(n) = 4n^2 + \frac{6}{7}n + 240$$

*Check all that apply.*

- ☐  $\Omega(1)$
- ☐  $\theta(1)$
- ☐  $\Omega(n)$
- ☐  $O(1)$
- ☐  $\Omega(n^3)$
- ☐  $O(n)$
- ☐  $\theta(n)$
- ☐  $\theta(2^n)$
- ☐  $O(n^2)$
- ☐  $O(n^3)$
- ☐  $\Omega(2^n)$
- ☐  $O(2^n)$
- ☐  $\theta(n^3)$
- ☐  $\theta(n^2)$
- ☐  $\Omega(n^2)$

2. \*<sup>1</sup>Se dă un algoritm care primește ca date de intrare trei tipuri de input de mărime n.

Pentru tipul 1 de date de intrare, complexitatea ca timp de execuție este  $\theta(n^4)$ , iar probabilitatea de a avea acest input este de  $\frac{1}{n^2}$

Pentru tipul 2 de date de intrare, complexitatea ca timp de execuție este  $\theta(n^3)$ , iar probabilitatea de a avea acest input este  $\frac{1}{n}$ .

Pentru tipul 3 de date de intrare, complexitatea ca timp de execuție este  $\theta(n)$ , iar probabilitatea de a avea acest input este  $1 - \frac{1}{n} - \frac{1}{n^3}$ . Calculați:

- a) Complexitatea în cazul defavorabil
- b) Complexitatea în cazul favorabil
- c) Complexitatea în cazul mediu
- d) Overall complexity

3. Care este complexitatea funcției f1? \*

```
def f1(n):  
    p = 1  
    for i in range(1, n+1):  
        p = p*i  
    return p
```

Mark only one oval.

- ☐ 0(1)
- ☐ 0(n)
- ☐  $\theta(n)$
- ☐  $\theta(1)$

---

<sup>1</sup> Cerințe preluate și adaptate din quiz-uri de la cursul Data Structures, MIE 2020/2021, prof. Zsuzsanna Oneț-Marian

4. Care este complexitatea funcției f3? \*

```
import random

def f3(n, m):
    a = 0
    b = 0
    for i in range(n):
        a += random.randint(1, 100)
    for j in range(m):
        b += random.randint(1, 50)
```

Mark only one oval.

- ☐  $\theta(n \cdot m)$  time,  $\theta(1)$  space
- ☐  $\theta(n + m)$  time,  $\theta(n + m)$  space
- ☐  $\theta(n + m)$  time,  $\theta(1)$  space
- ☐  $\theta(n \cdot m)$  time,  $\theta(n + m)$  space

5. Care este complexitatea ca timp de execuție a funcției f2? Calculați complexitatea în caz favorabil, defavorabil și caz mediu. \*

```
def f2(lst):
    """
    Verifica daca exista un numar par in lista
    :param lst: lista de numere naturale
    :type lst: list
    :return: True daca in lista exista un numar par, False altfel
    :rtype: bool
    """
    poz = 0
    n = len(lst)
    while poz < n and lst[poz] % 2 != 0:
        poz += 1

    return poz < n
```

---

---

---

---

---

6. Care este complexitatea (ca timp de execuție) pentru funcția f4? Scrieți calculul complexității cu ajutorul sumelor. \*

```
def f4(n):  
    a = 0  
    for i in range(n):  
        for j in range(i, n):  
            a = a + i + j
```

---

---

---

---

---

7. Care este complexitatea (ca timp de execuție) pentru funcția f5? \*

```
def f5(n):  
    k = 0  
    for i in range(n / 2, n + 1):  
        j = 2  
        while j <= n:  
            k = k + n / 2  
            j = j * 2  
    return k
```

Mark only one oval.

- ☐  $\theta(n)$
- ☐  $\theta(n \cdot \log n)$
- ☐  $\theta(n^2)$
- ☐  $\theta(n^2 \cdot \log n)$

8. Care este complexitatea pentru funcția f6? Calculați complexitatea în caz favorabil, defavorabil și caz mediu.

```
def f6(n):  
    for i in range(1, 2 * n - 5):  
        j = i + 1  
        ok = True  
        a = 0  
        while j <= 2 * n and ok:  
            a = random.randint(1, 10)  
            if a % 5 == 0:  
                ok = False  
            j = j + 1
```

---

---

---

---

---

9. Care este complexitatea (ca timp de execuție, overall complexity) pentru funcția f7? \*

```
def f7(n):  
    a = 0  
    i = n  
    while i > 0:  
        a += i  
        i /= 2  
    return a
```

Mark only one oval.

☐  $O(n)$

☐  $O(\sqrt{n})$

☐  $O(\frac{n}{2})$

☐  $O(\log n)$

☐  $\theta(n)$

☐  $\theta(\log n)$

☐  $\theta(\sqrt{n})$

☐  $\theta(\frac{n}{2})$

10. Care este complexitatea pentru funcția f8?

```
def f8(n):  
    a = 0  
    i = n  
    while i > 0:  
        j = 1  
        while j < n:  
            for k in range(0, n, 2):  
                a += i + j + k  
            j = j * 2  
        i = i / 2  
    return a
```

---

---

---

---

---

11. Care este complexitatea pentru funcția f9?

```
def f9(n):  
    a = 0  
    for i in range(n):  
        for j in range(i):  
            a += j  
            ok = random.randint(1, 10)  
            if ok % 2 == 0:  
                break  
    return a
```

---

---

---

---

---

12. Care este complexitatea pentru funcția recursive\_f1? Scrieți relația de recurență.

```
def recursive_f1(n):  
    if n <= 0:  
        return 1  
    else:  
        return 1 + recursive_f1(n - 1)
```

---

---

---

---

---

13. Care este complexitatea pentru funcția recursive\_f2? Scrieți relația de recurență.

```
def recursive_f2(n):  
    if n <= 1:  
        return 1  
    else:  
        return 1 + recursive_f2(n - 5)
```

---

---

---

---

---

14. Care este complexitatea pentru funcția recursive\_f3? Scrieți relația de recurență.

```
def recursive_f3(n):  
    if n <= 0:  
        return 1  
    else:  
        return 1 + recursive_f3(n / 2)
```

15. Care este complexitatea pentru funcția recursive\_f4? Scrieți relația de recurență.

```
def recursive_f4(n, m, o):  
    if n <= 0:  
        print('m is', m, '& n is', n)  
    else:  
        recursive_f4(n - 1, m + 1, o)  
        recursive_f4(n - 1, m, o + 1)
```

16. Care este complexitatea pentru funcția recursive\_f5? Scrieți relația de recurență.

```
def recursive_f5(n):  
    print(n)  
    for i in range(n):  
        print('*' * n)  
    if n < 1:  
        return 1  
    else:  
        return 1 + recursive_f5(n - 1)
```