Quiz Complexități

Întrebările din acest quiz au fost luate și adaptate de pe pagina www.commonlounge.com

Obs. Nu este necesar să înțelegeți ce face codul din întrebare pentru a răspunde la întrebări.

Întrebări

Întrebare 1:

```
funcție fibonacci(n: întreg) este:
      i, j, k, t: întreg
      j = 1
      i = 0
      pentru k = 1, n+1, 1 execută
            t = i + j
            i = j
            j = t
      sf_pentru
      returnează j
sf_funcție
```

Cât este complexitatea de timp a algoritmului?

a. Θ(1)
b. Θ(log n)
c. Θ(n)
d. Θ(n*logn)
e. Θ(n^2)
f. Θ(n^3)

Întrebare 2:

```
funcție fibonacci(n: întreg) este:
      i, j, k, t: întreg
      j = 1
      i = 0
      pentru k = 1, n+1, 1 execută
            t = i + j
            i = j
            j = t
      sf_pentru
      returnează j
sf_funcție
```

Un calculator poate să efectueze aproximativ 10^8 (100,000,000) operații pe secundă, unde o operație este o adăugare, atribuie, incrementare, decrementare, scădere, multiplicare, etc. Cât timp va dura execuția codului pentru **n = 100**?

a. Mai puțin de o secundă
b. Între o secundă și un minut
c. Între un minut și o oră
d. Mai mult de o oră

Întrebare 3:

```
funcție fibonacci(n: întreg) este:
      i, j, k, t: întreg
      j = 1
      i = 0
      pentru k = 1, n+1, 1 execută
            t = i + j
            i = j
            j = t
      sf_pentru
      returnează j
sf_funcție
```

Un calculator poate să efectueze aproximativ 10^8 operații pe secundă, unde o operație este o adăugare, atribuie, incrementare, decrementare, scădere, multiplicare, etc. Cât timp va dura execuția codului pentru **n = 10,000**?

a. Mai puțin de o secundă
b. Între o secundă și un minut
c. Între un minut și o oră
d. Mai mult de o oră

Întrebare 4:

```
funcție fibonacci(n: întreg) este:
      i, j, k, t: întreg
      j = 1
      i = 0
      pentru k = 1, n+1, 1 execută
            t = i + j
            i = j
            j = t
      sf_pentru
      returnează j
sf_funcție
```

Un calculator poate să efectueze aproximativ 10^8 operații pe secundă, unde o operație este o adăugare, atribuie, incrementare, decrementare, scădere, multiplicare, etc. Cât timp va dura execuția codului pentru **n = 1,000,000**?

a. Mai puțin de o secundă
b. Între o secundă și un minut
c. Între un minut și o oră
d. Mai mult de o oră

Întrebare 5:

```
funcție fibonacci(n: întreg) este:
      i, j, k, t: întreg
      j = 1
      i = 0
      pentru k = 1, n+1, 1 execută
            t = i + j
            i = j
            j = t
      sf_pentru
      returnează j
sf_funcție
```

Un calculator poate să efectueze aproximativ 10^8 operații pe secundă, unde o operație este o adăugare, atribuie, incrementare, decrementare, scădere, multiplicare, etc. Cât timp va dura execuția codului pentru **n = 100,000,000**?

a. Mai puțin de o secundă
b. Între o secundă și un minut
c. Între un minut și o oră
d. Mai mult de o oră

Întrebare 6:

```
funcție fibonacci(n: întreg) este:
      i, j, k, t: întreg
      j = 1
      i = 0
      pentru k = 1, n+1, 1 execută
            t = i + j
            i = j
            j = t
      sf_pentru
      returnează j
sf_funcție
```

Un calculator poate să efectueze aproximativ 10^8 operații pe secundă, unde o operație este o adăugare, atribuie, incrementare, decrementare, scădere, multiplicare, etc. Cât timp va dura execuția codului pentru **n = 10,000,000,000**?

a. Mai puțin de o secundă
b. Între o secundă și un minut
c. Între un minut și o oră
d. Mai mult de o oră

Întrebare 7:

```
funcție fibonacci(n: întreg) este:
      i, j, k, t: întreg
      j = 1
      i = 0
      pentru k = 1, n+1, 1 execută
            t = i + j
            i = j
            j = t
      sf_pentru
      returnează j
sf_funcție
```

Un calculator poate să efectueze aproximativ 10^8 operații pe secundă, unde o operație este o adăugare, atribuie, incrementare, decrementare, scădere, multiplicare, etc. Cât timp va dura execuția codului pentru **n = 1,000,000,000,000**?

a. Mai puțin de o secundă
b. Între o secundă și un minut
c. Între un minut și o oră
d. Mai mult de o oră

Întrebare 8:

```
funcție fibonacci(n: întreg) este:
      i, j, k, t: întreg
      j = 1
      i = 0
      pentru k = 1, n+1, 1 execută
            t = i + j
            i = j
            j = t
      sf_pentru
      returnează j
sf_funcție
```

Cât este complexitatea de spațiu a algoritmului?

a. Θ(1)
b. Θ (n)
c. Θ (n^2)
d. Θ (n^3)

Întrebare 9:

```
funcție fibonacci(n: întreg) este:
      a, ta, b, tb, c, rc, tc, d, rd: întreg
      a = 1
      b = 1
      c = 1
      rc = 0
      d = 0
      rd = 1

      cât timp n > 0 execută
            dacă n mod 2 == 1 atunci
                  tc = rc
                  rc = rc*a + rd*c
                  rd = tc*b + rd*d
            sf_dacă

            ta = a
            tb = b
            tc = c
            a = a*a + b*c
            b = ta*b + b*d
            c = c*ta + d*c
            d = tc*tb + d*d

            n = n / 2

      sf_cât timp
      returnează rc
sf_funcție
```

Cât este complexitatea de timp a algoritmului?

a. Θ(1)
b. Θ(log n)
c. Θ(n)
d. Θ(n*logn)
e. Θ(n^2)
f. Θ(n^3)

Întrebare 10:

```
funcție fibonacci(n: întreg) este:
      a, ta, b, tb, c, rc, tc, d, rd: întreg
      a = 1
      b = 1
      c = 1
      rc = 0
      d = 0
      rd = 1

      câttimp n > 0 execută
            dacă n mod 2 == 1 atunci
                  tc = rc
                  rc = rc*a + rd*c
                  rd = tc*b + rd*d
            sf_dacă

            ta = a
            tb = b
            tc = c
            a = a*a + b*c
            b = ta*b + b*d
            c = c*ta + d*c
            d = tc*tb + d*d

            n = n / 2

      sf_câttimp
      returnează rc
sf_funcție
```

Cât este complexitatea de spațiu a algoritmului?

a. Θ(1)
b. Θ(n)
c. Θ(n^2)
d. Θ(n^3)

Întrebare 11:

```
funcție fibonacci(n: întreg) este:
      fib: întreg[n]
      k: întreg
      fib[0] = 0
      fib[1] = 1
      pentru k = 2, n+1, 1 execută
            fib[k] = fib[k-1]+fib[k-2]
      sf_pentru

      returnează fib[n]
sf_funcție
```

Cât este complexitatea de timp a algoritmului?

a. Θ(1)
b. Θ(log n)
c. Θ(n)
d. Θ(n*logn)
e. Θ(n^2)
f. Θ(n^3)

Întrebare 12:

```
funcție fibonacci(n: întreg) este:
      fib: întreg[n]
      k: întreg
      fib[0] = 0
      fib[1] = 1
      pentru k = 2, n+1, 1 execută
            fib[k] = fib[k-1]+fib[k-2]
      sf_pentru

      returnează fib[n]
sf_funcție
```

Cât este complexitatea de spațiu a algoritmului?

a. Θ(1)
b. Θ(n)
c. Θ(n^2)
d. Θ(n^3)

Întrebare 13:

```
funcție fibonacci(n: întreg) este:
      fib: întreg[n]
      k: întreg
      fib[0] = 0
      fib[1] = 1
      pentru k = 2, n+1, 1 execută
            fib[k] = fib[k-1]+fib[k-2]
      sf_pentru

      returnează fib[n]
sf_funcție
```

Un număr întreg ocupă 4 bytes de memorie. Câtă memorie folosește algoritmul pentru **n = 10,000,000?** (1 MB = 1024 bytes)

a. Aproximativ 1 MB
b. Aproximativ 4 MB
c. Aproximativ 10 MB
d. Aproximativ 40 MB
e. Aproximativ 100 MB
f. Aproximativ 400 MB

Întrebare 14:

```
funcție matrix_multiply(n: întreg, first: întreg[][], second: întreg[][]) este:

       result: întreg[n][n]
       m, n, p, q, c, d, k, sum, row, col: întreg
       sum = 0
       pentru row = 0, n, 1 execută
              pentru col = 0, n, 1 execută
                     result[row][col] = 0
                     pentru k = 0, n, 1 execută
                            result[row][col] = result[row][col] +
                                               first[row][k] * second[k][col]
                     sf_pentru
              sf_pentru
       sf_pentru
       returnează result
sf_funcție
```

Cât este complexitatea de timp a algoritmului?

a. Θ(1)
b. Θ(log n)
c. Θ(n)
d. Θ(n*logn)
e. Θ(n^2)
f. Θ(n^3)

Întrebare 15:

```
funcție matrix_multiply(n: întreg, first: întreg[][], second: întreg[][]) este:

       result: întreg[n][n]
       m, n, p, q, c, d, k, sum, row, col: întreg
       sum = 0
       pentru row = 0, n, 1 execută
              pentru col = 0, n, 1 execută
                     result[row][col] = 0
                     pentru k = 0, n, 1 execută
                            result[row][col] = result[row][col] +
                                               first[row][k] * second[k][col]
                     sf_pentru
              sf_pentru
       sf_pentru
       returnează result
sf_funcție
```

Cât este complexitatea de spațiu a algoritmului (ignorând datele de intrare)?

a. Θ(1)
b. Θ(n)
c. Θ(n^2)
d. Θ(n^3)

Întrebare 16:

```
funcție matrix_multiply(n: întreg, first: întreg[][], second: întreg[][]) este:

       result: întreg[n][n]
       m, n, p, q, c, d, k, sum, row, col: întreg
       sum = 0
       pentru row = 0, n, 1 execută
              pentru col = 0, n, 1 execută
                     result[row][col] = 0
                     pentru k = 0, n, 1 execută
                            result[row][col] = result[row][col] +
                                               first[row][k] * second[k][col]
                     sf_pentru
              sf_pentru
       sf_pentru
       returnează result
sf_funcție
```

Un calculator poate să efectueze aproximativ 10^8 operații pe secundă, unde o operație este o adăugare, atribuie, incrementare, decrementare, scădere, multiplicare, etc. Cât timp va dura execuția codului pentru **n = 100?**

a. Mai puțin de o secundă
b. Între o secundă și un minut
c. Între un minut și o oră
d. Mai mult de o oră

Întrebare 17:

```
funcție matrix_multiply(n: întreg, first: întreg[][], second: întreg[][]) este:

       result: întreg[n][n]
       m, n, p, q, c, d, k, sum, row, col: întreg
       sum = 0
       pentru row = 0, n, 1 execută
              pentru col = 0, n, 1 execută
                     result[row][col] = 0
                     pentru k = 0, n, 1 execută
                            result[row][col] = result[row][col] +
                                               first[row][k] * second[k][col]
                     sf_pentru
              sf_pentru
       sf_pentru
       returnează result
sf_funcție
```

Un calculator poate să efectueze aproximativ 10^8 operații pe secundă, unde o operație este o adăugare, atribuie, incrementare, decrementare, scădere, multiplicare, etc. Cât timp va dura execuția codului pentru **n = 1,000?**

a. Mai puțin de o secundă
b. Între o secundă și un minut
c. Între un minut și o oră
d. Mai mult de o oră

Întrebare 18:

```
funcție matrix_multiply(n: întreg, first: întreg[][], second: întreg[][]) este:

       result: întreg[n][n]
       m, n, p, q, c, d, k, sum, row, col: întreg
       sum = 0
       pentru row = 0, n, 1 execută
              pentru col = 0, n, 1 execută
                     result[row][col] = 0
                     pentru k = 0, n, 1 execută
                            result[row][col] = result[row][col] +
                                               first[row][k] * second[k][col]
                     sf_pentru
              sf_pentru
       sf_pentru
       returnează result
sf_funcție
```

Un calculator poate să efectueze aproximativ 10^8 operații pe secundă, unde o operație este o adăugare, atribuie, incrementare, decrementare, scădere, multiplicare, etc. Cât timp va dura execuția codului pentru **n = 10,000?**

a. Mai puțin de o secundă
b. Între o secundă și un minut
c. Între un minut și o oră
d. Mai mult de o oră

Întrebare 19:

```
funcție matrix_multiply(n: întreg, first: întreg[][], second: întreg[][]) este:

        result: întreg[n][n]
        m, n, p, q, c, d, k, sum, row, col: întreg
        sum = 0
        pentru row = 0, n, 1 execută
                pentru col = 0, n, 1 execută
                        result[row][col] = 0
                        pentru k = 0, n, 1 execută
                                result[row][col] = result[row][col] +
                                                   first[row][k] * second[k][col]
                        sf_pentru
                sf_pentru
        sf_pentru
        returnează result
sf_funcție
```

Un număr întreg ocupă 4 bytes de memorie. Câtă memorie ocupă vectorul *result* pentru **n = 1500?** (1 MB = 1024 bytes)

a. Aproximativ 1 MB
b. Aproximativ 4 MB
c. Aproximativ 10 MB
d. Aproximativ 40 MB
e. Aproximativ 100 MB
f. Aproximativ 400 MB

Avem 2 algoritmi care respectă următoarele condiții:

Algoritmul L (linear) are complexitate de timp Θ(n), mai exact face aproximativ 100*n operații.

Algoritmul Q (pătratic) are complexitate de timp Θ(n^2), mai exact face aproximativ 10*n^2 + 10*n operații.

La următoarele 4 întrebări, trebuie să vă decideți care algoritmi va rula mai repede pentru diferite valori ale lui n, presupunând că toate operațiile durează la fel de mult.

Întrebare 20:

Care algoritm este mai rapid pentru **n=5?**

a. Algoritmul L
b. Algoritmul Q

Întrebare 21:

Care algoritm este mai rapid pentru **n=20?**

a. Algoritmul L
b. Algoritmul Q

Întrebare 22:

Care algoritm este mai rapid pentru **n=100?**

a. Algoritmul L
b. Algoritmul Q

Întrebare 23:

Care algoritm este mai rapid pentru **n=2000?**

a. Algoritmul L
b. Algoritmul Q

Întrebare 24:

Alegeți cea mai bună clasă de complexitate asimptotică pentru funcția următoare: **3/2 * n**

a. Θ(1)
b. Θ(n)
c. Θ(n^2)
d. Θ(n^3)
e. Θ(2^n)
f. Niciuna dintre aceste funcții

Întrebare 25:

Alegeți toate clasele de complexitate asimptotică de care aparține funcția următoare: **3/2 * n** (mai multe răspunsuri corecte sunt posibile)

a. O(1)
b. O(n)
c. O(n^2)
d. O(n^3)
e. O(2^n)
f. Ω(1)
g. Ω (n)
h. Ω (n^2)
i. Ω (n^3)
j. Ω (2^n)
k. Θ(1)
l. Θ(n)
m. Θ(n^2)
n. Θ(n^3)
o. Θ(2^n)

Întrebare 26:

Alegeți cea mai bună clasă de complexitate asimptotică pentru funcția următoare: **2n^2 + 5n + 1000**

a. Θ(1)
b. Θ(n)
c. Θ(n^2)
d. Θ(n^3)
e. Θ(2^n)
f. Niciuna dintre aceste funcții

Întrebare 27:

Alegeți toate clasele de complexitate asimptotică de care aparține funcția următoare: **2n^2 + 5n + 1000** (mai multe răspunsuri corecte sunt posibile)

a. O(1)
b. O(n)
c. O(n^2)
d. O(n^3)
e. O(2^n)
f. Ω(1)
g. Ω (n)
h. Ω (n^2)
i. Ω (n^3)
j. Ω (2^n)
k. Θ(1)
l. Θ(n)
m. Θ(n^2)
n. Θ(n^3)
o. Θ(2^n)

Întrebare 28:

Alegeți cea mai bună clasă de complexitate asimptotică pentru funcția următoare: **100n^3 + 18*nlogn**

a. Θ(1)
b. Θ(n)
c. Θ(n^2)
d. Θ(n^3)
e. Θ(2^n)
f. Niciuna dintre aceste funcții

Întrebare 29:

Alegeți toate clasele de complexitate asimptotică de care aparține funcția următoare: **100n^3 + 18*nlogn** (mai multe răspunsuri corecte sunt posibile)

a. O(1)
b. O(n)
c. O(n^2)
d. O(n^3)
e. O(2^n)
f. Ω(1)
g. Ω (n)
h. Ω (n^2)
i. Ω (n^3)
j. Ω (2^n)
k. Θ(1)
l. Θ(n)
m. Θ(n^2)
n. Θ(n^3)
o. Θ(2^n)

ntrebare 30:

Alegeți cea mai bună clasă de complexitate asimptotică pentru funcția următoare: **50**

a. Θ(1)
b. Θ(n)
c. Θ(n^2)
d. Θ(n^3)
e. Θ(2^n)
f. Niciuna dintre aceste funcții

Întrebare 31:

Alegeți toate clasele de complexitate asimptotică de care aparține funcția următoare: **50** (mai multe răspunsuri corecte sunt posibile)

a. O(1)
b. O(n)
c. O(n^2)
d. O(n^3)
e. O(2^n)
f. Ω(1)
g. Ω (n)
h. Ω (n^2)
i. Ω (n^3)
j. Ω (2^n)
k. Θ(1)
l. Θ(n)
m. Θ(n^2)
n. Θ(n^3)
o. Θ(2^n)

Întrebare 32:

Alegeți cea mai bună clasă de complexitate asimptotică pentru funcția următoare: **5n*(n^2 + n)**

a. Θ(1)
b. Θ(n)
c. Θ(n^2)
d. Θ(n^3)
e. Θ(2^n)
f. Niciuna dintre aceste funcții

Întrebare 33:

Alegeți toate clasele de complexitate asimptotică de care aparține funcția următoare: **5n*(n^2 + n)** (mai multe răspunsuri corecte sunt posibile)

a. O(1)
b. O(n)
c. O(n^2)
d. O(n^3)
e. O(2^n)
f. Ω(1)
g. Ω (n)
h. Ω (n^2)
i. Ω (n^3)
j. Ω (2^n)
k. Θ(1)
l. Θ(n)
m. Θ(n^2)
n. Θ(n^3)
o. Θ(2^n)

Întrebare 34:

Alegeți cea mai bună clasă de complexitate asimptotică pentru funcția următoare: **n^2*2^n**

a. Θ(1)
b. Θ(n)
c. Θ(n^2)
d. Θ(n^3)
e. Θ(2^n)
f. Niciuna dintre aceste funcții

Întrebare 35:

Alegeți toate clasele de complexitate asimptotică de care aparține funcția următoare: **n^2*2^n** (mai multe răspunsuri corecte sunt posibile)

a. O(1)
b. O(n)
c. O(n^2)
d. O(n^3)
e. O(2^n)
f. Ω(1)
g. Ω (n)
h. Ω (n^2)
i. Ω (n^3)
j. Ω (2^n)
k. Θ(1)
l. Θ(n)
m. Θ(n^2)
n. Θ(n^3)
o. Θ(2^n)

Întrebare 36:

Alegeți cea mai bună clasă de complexitate asimptotică pentru funcția următoare: **3^n**

a. Θ(1)
b. Θ(n)
c. Θ(n^2)
d. Θ(n^3)
e. Θ(2^n)
f. Niciuna dintre aceste funcții

Întrebare 37:

Alegeți toate clasele de complexitate asimptotică de care aparține funcția următoare: **3^n** (mai multe răspunsuri corecte sunt posibile)

a. O(1)
b. O(n)
c. O(n^2)
d. O(n^3)
e. O(2^n)
f. Ω(1)
g. Ω (n)
h. Ω (n^2)
i. Ω (n^3)
j. Ω (2^n)
k. Θ(1)
l. Θ(n)
m. Θ(n^2)
n. Θ(n^3)
o. Θ(2^n)

Întrebare 38:

Adevărat sau fals?

a. $n^2 \in O(n^3)$
b. $n^3 \in O(n^2)$
c. $2^{n+1} \in \Theta(2^n)$
d. $2^{2n} \in \Theta(2^n)$
e. $n^2 \in \Theta(n^3)$
f. $2^n \in O(n!)$
g. $\log_{10} n \in \Theta(\log_2 n)$
h. $(n + m)^2 \in O(n^2 + m^2)$
i. $3^n \in O(2^n)$
j. $\log_2 3^n \in O(\log_2 2^n)$

Răspunsuri:

1. C
2. A
3. A
4. A
5. B
6. C
7. D
8. A
9. B
10. A
11. C
12. B
13. D
14. F
15. C
16. A
17. B
18. D
19. C
20. B
21. A
22. A
23. A
24. B
25. B, C, D, E, F, G, L
26. C
27. C, D, E, F, G, H, M
28. D
29. D, E, F, G, H, I, N
30. A
31. A, B, C, D, E, F, K
32. D
33. D, E, F, G, H, I, N
34. E
35. E, F, G, H, I, J, O
36. F
37. F, G, H, I, J
38.
    a. Adevărat
    b. Fals
    c. Adevărat
    d. Fals
    e. Fals
    f. Adevărat
    g. Adevărat
    h. Adevărat
    i. Fals
    j. Adevărat