1. Care dintre următoarele afirmaţii referitoare la tehnica Greedy este adevărată?
   1. **Tehnica Greedy este eficientă şi este folosită pentru determinarea soluţiei optime a unei probleme.**
   2. Tehnica Greedy oferă soluţia optimă a unei probleme după generarea tuturor soluţiilor posibile.
   3. Tehnica Greedy construieşte soluţia unei probleme componentă cu componentă cu posibilitatea modificării componentelor deja alese la paşii anteriori.
   4. Tehnica Greedy construieşte soluţia optimă a unei probleme prin recombinarea soluţiilor optime a tuturor subproblemelor acesteia.
2. Care dintre următoarele afirmaţii referitoare la tehnica Divide et Impera este adevărată?
   1. Tehnica Divide et Impera este folosită pentru determinarea tuturor soluţiilor unei probleme prin împărţirea recurentă a problemei date în subprobleme şi combinarea optimă a soluţiilor acestora.
   2. Tehnica Divide et Impera construieşte soluţia unei probleme componentă cu componentă.
   3. **Tehnica Divide et Impera este folosită pentru determinarea soluţiei unei probleme prin împărţirea recurentă a problemei date în subprobleme şi combinarea soluţiilor acestora.**
   4. Tehnica Divide et Impera este folosită pentru determinarea soluţiei optime a unei probleme prin împărţirea recurentă a problemei date în subprobleme şi combinarea soluţiilor optime ale acestora.
3. Se dă următorul subprogram scris în pseudocod:

**Functie\_test(i, g)**

**dacă g>0**

**dacă i < g**

Functie\_test(i+1, g)

Scrie i

Functie\_test(i+1, g)

**sfârşit dacă**

Functie\_test(i+1, g-1)

**sfârşit dacă**

**sfârşit Functie\_test**

Ce se afişează la apelul Functie\_test(3, 7)?

1. 65646465365656465445
2. **65646565365646565545**
3. 65546563365646565545
4. 65646564365646564545
5. Se consideră o structură max heap cu n componente. Algoritmul pentru determinarea valorii maxime are complexitatea:
6. O(log n)
7. O(n log n)
8. O(n)
9. **O(1)**
10. Se consideră o structură max heap cu n componente. Algoritmul pentru determinarea valorii minime are complexitatea:
11. O(log n)
12. O(n log n)
13. **O(n)**
14. O(1)
15. Se consideră o structură max heap cu n elemente. Algoritmul de căutare a unei valori în heap are complexitatea:
16. O(log n)
17. **O(n)**
18. O(n log n)
19. O(n2)
20. Se consideră o structură max heap cu n elemente. Algoritmul de ștergere a valorii maxime în heap are complexitatea:
21. **O(log n)**
22. O(n)
23. O(n log n)
24. O(1)
25. Se consideră o structură max heap cu n elemente. Algoritmul de restabilire a proprietății de max heap în urma modificării uneia dintre valori are complexitatea:
26. O(1)
27. O(n)
28. O(n log n)
29. **O(log n)**
30. Se consideră o structură max heap cu n componente. Algoritmul pentru determinarea valorii celei mai apropiate de valoarea medie a elementelor din stuctură are complexitatea:
31. O(log n)
32. O(n log n)
33. **O(n)**
34. O(1)
35. Se consideră un vector cu n componente. Algoritmul optim de construcţie a unei structuri heap pornind de la elementele iniţiale are complexitatea:
36. O(log n)
37. O(n log n)
38. O(n2)
39. **O(n)**

**Target**

**# Secvenţă pseudocod**

* **de calculat complexitatea**
* **determinare de output**

**# Identificarea oportunităţii folosirii unei anumite tehnici pentru rezolvarea unei probleme specifice.**

**# Compararea metodelor de rezolvare a unei probleme prin intermediul complexităţii.**

**# Recunoaşterea tehnicilor menţionate(greedy, divide et impera) precum şi a caracteristicilor acestora.**

**# Cunoaşterea recursivităţii şi a mecanismului recursiv.**

**# Pentru o problemă dată, recunoaşterea algoritmului de rezolvare cu complexitatea cea mai mică / mare. (Calculul şi compararea de complexităţi)**