

am pus B si C, cica ar fi si A, desi eu as vedea A in $O(n)$

6. Precizați care dintre următorii algoritmi pot fi implementați într-o complexitate $O(n \cdot \log_2 n)$:

- A) Algoritmul care calculează numărul minim de subșiruri strict crescătoare în care poate fi partiționat un sir.
- B) Algoritmii de sortare QuickSort si MergeSort.
- C) Algoritmul de verificare a apartenentei a n elemente la un sir cu n elemente ordonate crescător.
- D) Algoritmul de căutare binara.

A

B

C

D

7. Se considera subalgoritmul *verif* care primește ca si parametru unic de intrare un

7. Se considera subalgoritmul *verif* care primește ca si parametru unic de intrare un număr natural nenul n ($1 \leq n \leq 10^9$).

Subalgorithm *verif*(n):

```
While  $n \neq 0$  execute
  If  $n \bmod 7 > 1$  then
    return 0
  EndIf
   $n \leftarrow n \div 7$ 
```

EndWhile

return 1

EndSubalgorithm

am pus C si D, era A si D la chestiunile astea chiar am probleme... tot timpul gresesc....

Precizați care dintre următoarele afirmații sunt false referitoare la secvența de cod de mai sus:

- A) Algoritmul efectuează descompunerea în baza 7 a numărului n .
- B) Algoritmul returnează 1 dacă si numai dacă n conține doar cifrele 0 si 1 în scrierea sa în baza 7.
- C) Algoritmul verifica dacă numărul n poate fi scris ca suma de puteri distincte ale lui 7.
- D) Algoritmul returnează 0 pentru orice număr care nu este multiplu de 7.

A

B

C

D

8. Subalgoritmul *aranjare* primește ca si parametri de intrare un vector *a* care conține *n* elemente numere naturale nenule si parametru *n* număr natural strict mai mare ca 1 ($1 \leq a[i] \leq 10^9$, $1 \leq i \leq n$, $2 \leq n \leq 10^5$).

```
Subalgorithm aranjare(a, n):  
  p ← 1  
  For i ← 1, n execute  
    If a[i] MOD 2 = 0 then  
      a[i] ← a[i] + a[p]  
      a[p] ← a[i] - a[p]  
      a[i] ← a[i] - a[p]  
      p ← p + 1  
    EndIf  
  EndFor  
EndSubalgorithm
```

Precizați care dintre următoarele afirmații sunt adevărate.

- A) Sortează in ordine crescătoare elementele vectorului *a*.
- B) Rearanjează elementele vectorului in așa fel încât cele impare sa fie înaintea celor pare.
- C) Rearanjează elementele vectorului in așa fel încât cele pare sa fie înaintea celor impare.
- D) Niciuna dintre afirmațiile A, B si C nu este corecta.

A

B

C

D

Am pus C cica e D
nu mi se pare ca are vreuns sens...

9. Se considera cele 2 subprograme *ce_face1* si *ce_face2* care primesc ca si parametrii de intrare cate un vector **a** cu **n** elemente numere întregi si valoarea **n**, număr natural nenul mai mare ca 1 ($-10^9 \leq a[i] \leq 10^9$, $1 \leq i \leq n$, $2 \leq n \leq 10^3$).

```
Subalgorithm ce_face1(a, n):  
    max  $\leftarrow -10^9$   
    s  $\leftarrow 0$   
    For i  $\leftarrow 1$ , n execute  
        s  $\leftarrow s + a[i]$   
        If s > max then  
            max  $\leftarrow s$   
        EndIf  
        If s < 0 then  
            s  $\leftarrow 0$   
        EndIf  
    EndFor  
    return s  
EndSubalgorithm
```

```
Subalgorithm ce_face2(a, n):  
    For i  $\leftarrow 1$ , n execute  
        s[i]  $\leftarrow s[i-1] + a[i]$   
    EndFor  
    max  $\leftarrow -10^9$   
    For i  $\leftarrow 1$ , n execute  
        For j  $\leftarrow i$ , n execute  
            If s[j] - s[i-1] > max then  
                max  $\leftarrow s[j] - s[i-1]$   
            EndIf  
        EndFor  
    EndFor  
    return max  
EndSubalgorithm
```

Am pus B si D, dar cica e A si D
dar totusi, nu este o contrazicere???

Știind ca șirul **s** este un sir cu $10^3 + 1$ elemente nule înainte de apelul subprogramului *ce_face2*(a, n), precizați care dintre următoarele informații sunt adevărate:

- A) Pentru aceleași date de intrare cele 2 subprograme nu returnează același lucru.
- B) Ambele secvențe de cod returnează suma maxima a unei secvențe de elemente din vector.
- C) Cele 2 secvențe de cod au complexități similare.
- D) Cele 2 secvențe de cod calculează același lucru, utilizând algoritmi diferiți, de complexități diferite, primul fiind mai eficient decât al doilea.

10. Subprogramul f este definit mai jos și are ca parametru unic de intrare un număr natural nenul n ($1 \leq n \leq 10^5$).

Subalgorithm $f(n)$:

```
j ← n
While j > 1 execute
    i ← 1
    While i < j2 execute
        i ← i * 4
    EndWhile
    j ← j DIV 3
EndWhile
```

EndSubalgorithm

HELP

Precizați în ce clasă de complexități se încadrează subprogramul $f(n)$:

- A) $O(\log_2^2 n^3)$
- B) $O(\log_{12} n^2)$
- C) $O(\log_3 n * \lg n^2)$
- D) $O(\log_3 n * \log_4 n)$

-
11. Se considera subprogramul fct care primește ca și parametri de intrare 2 valori n și k numere naturale nenule ($1 \leq n \leq 10^2$, $1 \leq k \leq n$).

Subalgorithm $fct(n, k)$:

```
If k = 0 OR n = k then
    return 1
Else
    write „Zece la Info”
    return  $fct(n-1, k-1) + fct(n-1, k)$ 
EndIf
```

EndSubalgorithm

de aici ne interesează doar
algoritmul pt grila următoare

Precizați câte cuvinte se vor afișa pe ecran în urma apelului $fct(n, k)$:

- A) $C_n^k * 6 - 6$
- B) $C_n^k * 2 - 2$
- C) $C_n^k * 3 - 3$
- D) $C_n^k - 1$

12. În legătura cu subprogramul de la punctul 11. . Care informații sunt adevărate?

- A) Numărul total de apeluri pe care le va efectua calculatorul pentru apelul $fct(n, k)$ este $C_n^k \cdot 2 - 1$.
- B) Pentru $fct(10, 5)$ numărul total de apeluri (incluzând apelul inițial) este 503.
- C) Pentru $fct(15, 3)$ numărul total de apeluri (incluzând apelul inițial) este 907.
- D) Numărul total de apeluri pe care le va efectua calculatorul pentru apelul $fct(n, k)$ este $C_n^k - 1$.

dacă la 11 este corect C-ul, atunci cum aici este corect A și B?

15. Se considera subprogramele *cb1* și *cb2* care primesc ca și parametrii un șir *a* cu *n* elemente numere naturale nenule ordonate crescător, numărul natural nenul *n* și un număr întreg *val* ($1 \leq a[i] \leq 10^9$, $1 \leq i \leq n$, $-10^9 \leq val \leq 10^9$).

```
Subalgorithm cb1(a, n, val):  
  st ← 1; dr ← n; poz ← -1  
  While st < dr execute  
    mj ← (st + dr) DIV 2  
    If a[mj] ≤ val then  
      poz ← mj  
      st ← mj + 1  
    Else  
      dr ← mj - 1  
    EndIf  
  EndWhile  
  return poz  
EndSubalgorithm
```

```
Subalgorithm cb2(a, n, val):  
  st ← 1; dr ← n; poz ← -1  
  While st < dr execute  
    mj ← (st + dr) DIV 2  
    If a[mj] ≥ val then  
      poz ← mj  
      dr ← mj - 1  
    Else  
      st ← mj + 1  
    EndIf  
  EndWhile  
  return poz  
EndSubalgorithm
```

din nou avem nevoie de algoritmi pt grila urmatoare

Precizați care dintre următoarele afirmații sunt adevărate:

- A) Cele 2 subprograme verifică dacă valoarea *val* apare în șirul *a*.
- B) Cele 2 subprograme returnează aceeași valoare în cazul în care valoarea *val* apare în șir.
- C) Cele 2 subprograme returnează aceeași valoare în cazul în care valoarea *val* apare în șir în mod unicat.
- D) În cazul în care valoarea *val* nu apare în șir, subprogramele returnează -1.

16. Tot referitor la cele 2 subprograme de la punctul anterior, precizați în care dintre următoarele scenarii este necesară utilizarea ambelor funcții pentru determinarea rezultatului dorit:
- A) Pentru a verifica dacă o valoare apare într-un sir ordonat crescător.
 - B) Pentru a număra de câte ori apare o valoare într-un sir ordonat crescător.
 - C) Pentru a vedea câte valori dintr-un sir ordonat crescător aparțin unui interval dat.
 - D) Pentru a verifica cea mai apropiată valoare care apare în sir ordonat crescător de o anumită valoare care nu apare în sir.

am pus B și D, este B și C...

18. Subprogramul $f1$ este definit alăturat și primește ca și parametrii de intrare 4 numere naturale n , $prod$, d , p ($1 \leq n \leq 10^6$). La apelul inițial, variabila $prod$ are valoarea 1, variabila d are valoarea 2 și variabila p are valoarea 0. Subalgoritmul $f2$ primește ca și parametrii 3 valori numere naturale n , cnt și d . ($1 \leq n \leq 10^6$). La apelul inițial cnt are valoarea 0 și d are valoarea 1.

```

Subalgorithm  $f1(n, prod, d, p)$ :
  If  $n = 0$  then
    write prod
  Else
    If  $n \bmod d \neq 0$  then
      If  $d > \sqrt{n}$  then
         $f1(n, prod, n, 0)$ 
      Else
         $f1(n, prod, d+1, 0)$ 
      EndIf
    Else
       $p \leftarrow p + 1$ 
       $n \leftarrow n \text{ DIV } d$ 
      If  $n \bmod d \neq 0$  then
         $prod \leftarrow prod * (p + 1)$ 
      EndIf
       $f1(n, prod, d, p)$ 
    EndIf
  EndIf
EndSubalgorithm

Subalgorithm  $f2(n, cnt, d)$ :
  If  $d \geq \sqrt{n}$  then
    If  $d = \sqrt{n}$  then
       $cnt \leftarrow cnt + 1$ 
    EndIf
    write cnt
  Else
    If  $n \bmod d = 0$  then
       $cnt \leftarrow cnt + 2$ 
    EndIf
     $f2(n, cnt, d + 1)$ 
  EndIf
EndSubalgorithm

```

Deși inițial credeam că returnează aceeași chestie ca primul ar fi algoritmul de calcul al nr de divizori am luat exemplu, ca 6 și 12 și nu mi-a dat ce trebuia... așa că am pus C și D și de fapt era A B D

Precizați care dintre următoarele afirmații sunt corecte:

- A) Pentru aceleași valori ale parametrului n la apelul inițial, cele 2 subprograme afișează același valoare.
- B) Cele 2 secvențe de cod calculează și afișează același lucru.
- C) Subalgoritmul $f1$ are o complexitate mai bună decât subalgoritmul $f2$.
- D) Primul subalgoritm calculează produsul puterilor+1 a factorilor primi care apar în descompunerea în factori primi a lui n , iar cel de-al doilea subalgoritm calculează numărul de divizori ai lui n .

22. Se considera subalgoritmul *stg* care primește ca si parametrii de intrare un sir **a** cu **n** elemente numere naturale nenule, numărul natural **n** si o valoare **poz** ($1 \leq a[i] \leq 10^9$, $1 \leq n \leq 10^3$, $1 \leq \text{poz} \leq n$).

```
Subalgoritm stg(a, n, poz):  
  For i ← 1, n execute  
    If i = poz then  
      For j ← i + 1, n execute  
        a[j-1] ← a[j]  
      EndFor  
    EndIf  
  EndFor  
EndSubalgorithm
```

am pus B si D, e doar D ???
sa fie faptul ca nu scade n?

Precizați care dintre următoarele afirmații sunt adevărate:

- A) Subalgoritmul șterge toate elementele pare din șirul **a**.
- B) Subalgoritmul șterge elementul de pe poziția **poz** din șirul **a**.
- C) Subalgoritmul are o complexitate pătratică.
- D) Subalgoritmul are o complexitate liniară.

23. Precizați cate numere de 7 cifre exista cu primele 4 cifre ordonate strict crescător si ultimele 4 cifre ordonate strict descrescător. Ex. 1234321, 1269431 etc.

- A) 126
- B) 2439
- C) 4878
- D) 7608

HELP