

Algorithmique et programmation 2025

- Questionnaire 1

(01/10, 3ème séance)

Nombre de participants : 57



1. La complexité d'un algorithme est liée:

33 bonnes réponses
sur 47 répondants

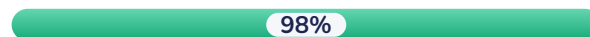
à la machine qui l'exécutera



12 votes



aux nombres d'opérations élémentaires



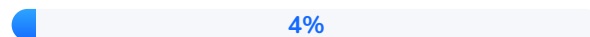
46 votes

au temps de développement



3 votes

à la facilité de lecture



2 votes



**Combien d'instructions
2. élémentaires seront exécutées par
le programme suivant ?**

20 bonnes réponses
sur 45 répondants

```
1 int x, y;  
2 x = x + y ;  
3 y = x - y ;  
4 x = x - y ;
```


3	<div><div></div></div> 24%	11 votes
4	<div><div></div></div> 7%	3 votes
5	<div><div></div></div> 0%	0 votes
6	<div><div></div></div> 22%	10 votes
7	<div><div></div></div> 2%	1 vote
<div><div></div></div> 8	<div><div></div></div> 44%	20 votes
9	<div><div></div></div> 0%	0 votes
10	<div><div></div></div> 0%	0 votes



3. En pire cas, combien d'instructions élémentaires seront exécutées par le programme suivant ?

14 bonnes réponses
sur 47 répondants

```
1 if (rand () % 2 == 0) {  
2     int temp;  
3     temp = a;  
4     a = b;  
5     b = tmp;  
6 } else {  
7     a = a + b;  
8     b = a - b;  
9     a = a - b;  
10 }
```

3	<div><div></div></div> 0%	0 votes
4	<div><div></div></div> 2%	1 vote
5	<div><div></div></div> 2%	1 vote
6	<div><div></div></div> 19%	9 votes
7	<div><div></div></div> 11%	5 votes
8	<div><div></div></div> 15%	7 votes
 9	<div><div></div></div> 32%	15 votes
10	<div><div></div></div> 23%	11 votes

Donner l'ordre de grandeur en O et

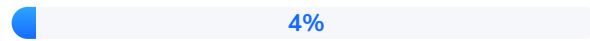


4. Ω du nombre d'instructions exécutées en pire cas par le programme suivant

35 bonnes réponses
sur 45 répondants

```
1 void afficher(int tab[], int n) {  
2     for (int i = 0; i < n; i++) {  
3         printf("%d, ", tab[i]);  
4     }  
5 }
```

$O(\log_2(n))$ et
 $\Omega(\log_2(n))$



4%

2 votes



$O(n)$ et $\Omega(n)$



78%

35 votes

$O(n^2)$ et $\Omega(n^2)$



9%

4 votes

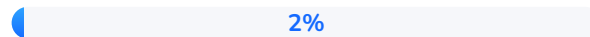
$O(2^n)$ et $\Omega(2^n)$



9%

4 votes

$O(n!)$ et $\Omega(n!)$



2%

1 vote



5. **Est-il vrai qu'il existe un algorithme avec une complexité en $O(\log_2(n))$ pour rechercher un élément dans un tableau trié de taille n ?**

34 bonnes réponses
sur 46 répondants



Oui



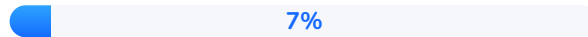
34 votes

Non



9 votes

On ne sait pas,
c'est un problème
ouvert



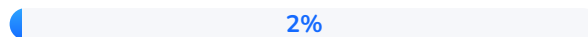
3 votes



6. **Soient g et h deux fonctions. On dit que g est dominée par h quand:**

45 bonnes réponses
sur 48 répondants

$g(n)=h(n)$



1 vote

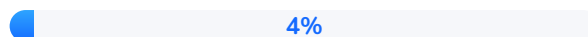


s'il existe n_0 et c
tels que $\forall n > n_0$,
on a $g(n) \leq c \cdot$
 $h(n)$



45 votes

s'il existe n_0 et c
tels que $\forall n > n_0$,
on a $c \cdot g(n) >$
 $h(n)$



2 votes



7. Quelles propositions sont vraies ?

36 bonnes réponses
sur 47 répondants



$$n = O(n)$$



35 votes

$$n = O(\log_2(n))$$



11 votes



$$4n = O(n)$$



32 votes



$$n = O(n^2)$$



28 votes



8. Quelles propositions sont vraies ?

42 bonnes réponses
sur 48 répondants



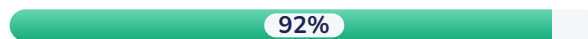
$$n = \Omega(n)$$



42 votes



$$n = \Omega(\log_2(n))$$



44 votes



$$4n = \Omega(n)$$



35 votes

$$n = \Omega(n^2)$$



6 votes

Donner la complexité



9. algorithmique en O de n du nombre d'instructions exécutées en pire cas par le programme suivant:

44 bonnes réponses
sur 46 répondants

```
1 int vais_je_avoir_une_bonne_note(int tableau[], int n) {  
2     for (int i = 0; i < n; i++) {  
3         for (int j = 0; j < n; j++) {  
4             if (tableau[i] > tableau[j]) {  
5                 return 0;  
6             }  
7         }  
8     }  
9     return 1;  
10 }
```

<input checked="" type="checkbox"/>	$O(n!)$	<div><div></div></div> 9%	4 votes
<input checked="" type="checkbox"/>	$O(n)$	<div><div></div></div> 7%	3 votes
<input checked="" type="checkbox"/>	$O(n^2)$	<div><div></div> 85%</div>	39 votes
	$O(1)$	<div><div></div></div> 2%	1 vote
	$O(\log_2(n))$	<div><div></div></div> 2%	1 vote
<input checked="" type="checkbox"/>	$O(2^n)$	<div><div></div></div> 9%	4 votes
<input checked="" type="checkbox"/>	$O(n^4)$	<div><div></div></div> 20%	9 votes

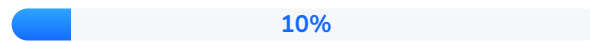


**Donner la complexité
algorithmique en Ω de n du
10. nombre d'instructions exécutées
en pire cas par le programme
suivant:**

12 bonnes réponses
sur 48 répondants

```
1 int vais_je_avoir_une_bonne_note(int tableau[], int n) {  
2     for (int i = 0; i < n; i++) {  
3         for (int j = 0; j < n; j++) {  
4             if (tableau[i] > tableau[j]) {  
5                 return 0;  
6             }  
7         }  
8     }  
9     return 1;  
10 }
```

$O(n!)$

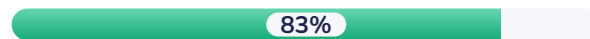


10%

5 votes



$O(n)$



83%

40 votes

$O(n^2)$



71%

34 votes



$O(1)$



71%

34 votes



$O(\log_2(n))$



92%

44 votes

$O(2^n)$



15%

7 votes

$O(n^4)$



8%

4 votes



Donner la complexité algorithmique en Θ de n du nombre d'instructions exécutées en pire cas par le programme suivant:

0 bonne réponse
sur 43 répondants

```
1 int vais_je_avoir_une_bonne_note(int tableau[], int n) {  
2     for (int i = 0; i < n; i++) {  
3         for (int j = 0; j < n; j++) {  
4             if (tableau[i] > tableau[j]) {  
5                 return 0;  
6             }  
7         }  
8     }  
9     return 1;  
10 }
```

$\Theta(n!)$



6 votes

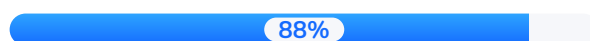


$\Theta(n)$



9 votes

$\Theta(n^2)$



38 votes

$\Theta(1)$



6 votes

$\Theta(\log_2(n))$



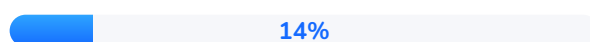
5 votes

$\Theta(2^n)$



5 votes

$\Theta(n^4)$



6 votes



12.

Donner la complexité algorithmique en Θ de n du nombre d'instructions exécutées en pire cas par le programme suivant, où n est la taille de *zozo*. On supposera que $\alpha = 0$ et $\text{zeta} = n$.

24 bonnes réponses
sur 44 répondants

```
1 int suis_je_un_zozo(char zozo[], int alpha, int zeta, char moi) {  
2     if (zeta-alpha < 1) return -1;  
3     int mu = (alpha + zeta) / 2;  
4     if (zozo[mu] == moi) {  
5         return mu;  
6     } else if (zozo[mu] < moi) {  
7         return suis_je_un_zozo(zozo, mu+1, zeta, moi);  
8     } else {  
9         return suis_je_un_zozo(zozo, alpha, mu-1, moi);  
10    }  
11 }
```

 $\Theta(n^2)$ 

2%

1 vote

 $\Theta(n^4)$ 

2%

1 vote

 $\Theta(1)$ 

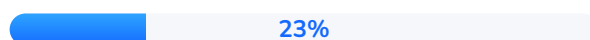
16%

7 votes

 $\Theta(\log_2(n))$ 

55%

24 votes

 $\Theta(n)$ 

23%

10 votes

 $\Theta(2^n)$ 

5%

2 votes

 $\Theta(n!)$ 

0%

0 votes

Donner la complexité algorithmique Θ de n du nombre d'instructions exécutées en pire cas par le programme suivant, où n est la taille de *zozo*. On supposera que $\alpha = 0$ et que $n > 50000$.



13.

6 bonnes réponses
sur 47 répondants

```
1 char afficher_zozos(char zozo[], int alpha) {  
2     if (alpha >= 10000) return zozo[alpha];  
3     printf("%c ", zozo[alpha]);  
4     afficher_zozos(zozo, alpha+1);  
5 }
```

$\Theta(n^2)$



2%

1 vote

$\Theta(n^4)$



0%

0 votes



$\Theta(1)$



15%

7 votes

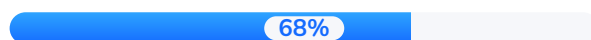
$\Theta(\log_2(n))$



6%

3 votes

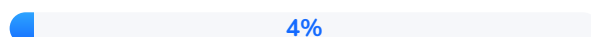
$\Theta(n)$



68%

32 votes

$\Theta(2^n)$



4%

2 votes



13%

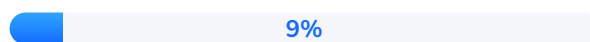
$\Theta(n!)$

6 votes



- 14.** Soit t un tableau d'entiers et n sa taille maximale, où $n > 0$. Donner la complexité algorithmique, en pire cas, en Θ de n de l'opération suivante: Ajouter un élément à la position i , où $0 \leq i \leq n - 1$. Si une valeur est présente, elle sera supprimée. On suppose que si $i - 1 \geq 0$, alors $t[i - 1]$ contient une valeur.

14 bonnes réponses
sur 43 répondants

 $\Theta(n^2)$ 

9%

4 votes

 $\Theta(n^4)$ 

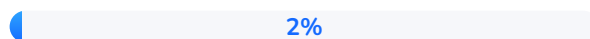
0%

0 votes

 $\Theta(1)$ 

33%

14 votes

 $\Theta(\log_2(n))$ 

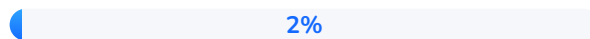
2%

1 vote

 $\Theta(n)$ 

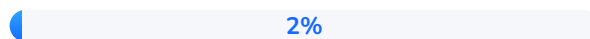
53%

23 votes

 $\Theta(2^n)$ 

2%

1 vote

 $\Theta(n!)$ 

2%

1 vote

Soit t un tableau d'entiers et n sa taille maximale, où $n > 0$. Donner la complexité algorithmique, en pire cas, en Θ de n de l'opération suivante: Ajouter l'élément à la position i , où $0 \leq i \leq n - 1$. Contrainte: l'ordre initial du tableau devra être préservé. On suppose que $t[n - 1]$ ne contient pas de valeur et que si $i - 1 \geq 0$, alors $t[i - 1]$ contient une valeur.



15. position i , où $0 \leq i \leq n - 1$.

35 bonnes réponses
sur 45 répondants

$\Theta(n^2)$



9%

4 votes

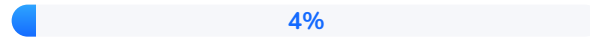
$\Theta(n^4)$



0%

0 votes

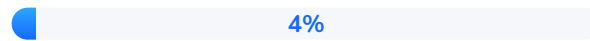
$\Theta(1)$



4%

2 votes

$\Theta(\log_2(n))$



4%

2 votes



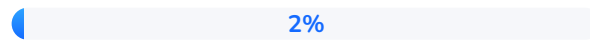
$\Theta(n)$



80%

36 votes

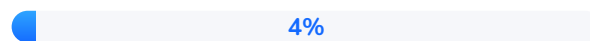
$\Theta(2^n)$



2%

1 vote

$\Theta(n!)$



4%

2 votes

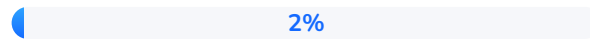
Soit t un tableau d'entiers et n sa taille maximale, où $n > 0$. Donner la complexité algorithmique, en meilleur cas, en Θ de n de l'opération suivante: Ajouter



16. l'élément à la position i , où $0 \leq i \leq n - 1$. Contrainte: l'ordre initial du tableau devra être préservé. On suppose que $t[n - 1]$ ne contient pas de valeur, et que si $i - 1 \geq 0$, alors $t[i - 1]$ contient une valeur.

38 bonnes réponses
sur 45 répondants

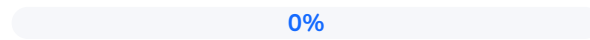
$\Theta(n^2)$



2%

1 vote

$\Theta(n^4)$



0%

0 votes



$\Theta(1)$



87%

39 votes

$\Theta(\log_2(n))$



0%

0 votes

$\Theta(n)$



13%

6 votes

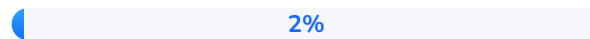
$\Theta(2^n)$



0%

0 votes

$\Theta(n!)$



2%

1 vote



17.

Soit l une liste chaînée de n éléments. Donner la complexité algorithmique, en pire cas, en Θ de n de l'opération suivante: Lire l'élément situé en $i^{\text{ème}}$ position dans la liste, où $i \in \mathbb{N}$

0 bonne réponse
sur 0 répondant

$\Theta(n^2)$

0%

0 votes

$\Theta(n^4)$

0%

0 votes

$\Theta(1)$

0%

0 votes

$\Theta(\log_2(n))$

0%

0 votes



$\Theta(n)$

0%

0 votes

$\Theta(2^n)$

0%

0 votes

$\Theta(n!)$

0%

0 votes



18.

Soit l une liste chaînée de n éléments. Donner la complexité algorithmique, en meilleur cas, en Θ de n de l'opération suivante:
Lire l'élément situé en $i^{\text{ème}}$ position dans la liste, où $i \in \mathbb{N}$

0 bonne réponse
sur 0 répondant

$\Theta(n^2)$

0%

0 votes

$\Theta(n^4)$

0%

0 votes



$\Theta(1)$

0%

0 votes

$\Theta(\log_2(n))$

0%

0 votes

$\Theta(n)$

0%

0 votes

$\Theta(2^n)$

0%

0 votes

$\Theta(n!)$

0%

0 votes

Soit l une liste chaînée de n éléments. Donner la complexité algorithmique, en pire cas, en Θ



19. de n de l'opération suivante:
Supprimer l'élément situé en première position dans la liste.
Contrainte: l'ordre initial de la liste devra être préservée.

0 bonne réponse
sur 0 répondant

$\Theta(n^2)$

0%

0 votes

$\Theta(n^4)$

0%

0 votes



$\Theta(1)$

0%

0 votes

$\Theta(\log_2(n))$

0%

0 votes

$\Theta(n)$

0%

0 votes

$\Theta(2^n)$

0%

0 votes

$\Theta(n!)$

0%

0 votes

Soit t un tableau d'entiers et n sa taille maximale, où $n > 0$. Donner la complexité algorithmique, en



20. meilleur cas, en Θ de n de l'opération suivante: Supprimer l'élément à la position 1. Contrainte: l'ordre initial du tableau devra être préservé.

0 bonne réponse
sur 0 répondant

$\Theta(n^2)$

0%

0 votes

$\Theta(n^4)$

0%

0 votes

$\Theta(1)$

0%

0 votes

$\Theta(\log_2(n))$

0%

0 votes



$\Theta(n)$

0%

0 votes

$\Theta(2^n)$

0%

0 votes

$\Theta(n!)$

0%

0 votes

Soit l une liste chaînée de n éléments. Donner la complexité algorithmique, en pire cas, en Θ



21. de n de l'opération suivante:

Rechercher un élément e .

Hypothèse: la liste est triée par ordre croissant des valeurs.

0 bonne réponse

sur 0 répondant

$\Theta(n^2)$

0%

0 votes

$\Theta(n^4)$

0%

0 votes

$\Theta(1)$

0%

0 votes

$\Theta(\log_2(n))$

0%

0 votes



$\Theta(n)$

0%

0 votes

$\Theta(2^n)$

0%

0 votes

$\Theta(n!)$

0%

0 votes

Soit l une liste chaînée de n éléments. Donner la complexité algorithmique, en pire cas, en Θ



22. de n de l'opération suivante:
Échanger l'élément en première position avec l'avant dernier élément. On suppose que les deux éléments existent.

0 bonne réponse
sur 0 répondant

$\Theta(n^2)$

0%

0 votes

$\Theta(n^4)$

0%

0 votes

$\Theta(1)$

0%

0 votes

$\Theta(\log_2(n))$

0%

0 votes



$\Theta(n)$

0%

0 votes

$\Theta(2^n)$

0%

0 votes

$\Theta(n!)$

0%

0 votes

Soit M une matrice de n lignes et m colonnes représentée par un tableau à deux dimensions. Donner la complexité algorithmique, en pire cas, en Θ



23. de n de l'opération suivante:
Rechercher la position i, j , si elle existe, de l'entier v où $0 \leq i \leq n - 1$ et $0 \leq j \leq m - 1$. On suppose que $n = m$ et que v est dans le tableau.

0 bonne réponse
sur 0 répondant



$\Theta(n^2)$

0%

0 votes

$\Theta(n^4)$

0%

0 votes

$\Theta(1)$

0%

0 votes

$\Theta(\log_2(n))$

0%

0 votes

$\Theta(n)$

0%

0 votes

$\Theta(2^n)$

0%

0 votes

$\Theta(n!)$

0%

0 votes