Aix-Marseille-Université

Test 3 UE Algorithmie et programmation 2026

Contrôle L1

11-02-2026

Les réponses sont à donner directement sur le sujet. Un espace est réservé pour chaque réponse.

Équivalences

Difficulté : ★ ☆ ☆ Question :

Quelles sont les égalités correctes ?

① une fonction en $\mathcal{O}(n)$ est aussi en $\mathcal{O}(n^2)$

② une fonction en $\mathcal{O}(1)$ est aussi en $\Theta(1)$

③ une fonction en $\mathcal{O}(n)$ est aussi en $\Theta(n)$

④ une fonction en $\mathcal{O}(n^2)$ est aussi en $\mathcal{O}(n)$

Difficulté : ★ ☆ ☆ Question :

Quelle est la complexité $C(n)$ qui correspond à l'équation :

$$C(n) = \begin{cases} \mathcal{O}(1) + C(n-3) & \text{si } n > 3 \\ \mathcal{O}(1) & \text{sinon} \end{cases}$$

① $C(n) = \Theta(n^2)$

② $C(n) = \Omega(n^2)$

③ $C(n) = \mathcal{O}(n) + \mathcal{O}(1)$

④ $C(n) = \Omega(1)$

⑤ $C(n) = \mathcal{O}(n^2)$

⑥ $C(n) = \Theta(n)$

Difficulté : ★ ★ ☆ Question :

Quelle est la complexité de l'algorithme suivant :

```
algorithme remplace(T: [entier], i: entier, j: entier):
  c := T[i]
  pour chaque (k := entier) de [0..T.longueur[:
    si T[k] == c :
        pour chaque (l := entier) de [0..T.longueur[:
          T[l] ← T[j]
```

① $C(n) = \mathcal{O}(1)$ avec $n = T.\text{longueur}$

② $C(n) = \mathcal{O}(n)$ avec $n = T.\text{longueur}$

③ $C(n) = \mathcal{O}(n^2)$ avec $n = T.\text{longueur}$

Zéro de fonction par dichotomie (GEI 2023)

Difficulté : ★ ★ ☆ Question :

On considère deux nombres réels a et b et une fonction réelle f continue sur l'intervalle $[a, b]$ avec comme contrainte que les deux valeurs $f(a)$ et $f(b)$ soient de signes contraires. On cherche à l'aide de l'algorithme de dichotomie comment trouver le zéro de la fonction f sur l'intervalle $[a, b]$, c'est-à-dire trouver x tel que $f(x) = 0$. On rappelle brièvement cet algorithme qui consiste à subdiviser en deux parties un intervalle et choisir celui dans lequel existe un zéro de la fonction. On répète le processus jusqu'à ce que l'erreur absolue soit inférieure à une valeur ϵ définie initialement. L'erreur absolue de la méthode de dichotomie, après n étapes, est au plus égale à :

① $(b-a)/n$

② $(b-a)/(n+1)$

③ $(b-a)/2^n$

④ $(b-a)/2^{n+1}$

⑤ $\log_2((b-a)/\epsilon)$

Difficulté : ★ ★ ☆ Question :

Écrivez un algorithme qui implémente la recherche dichotomique. Sa signature devra être `dichotomie(f: (réel) -> réel, a: réel, b: réel, epsilon: réel) -> réel`.

On se placde toujours dans les conditions de validité de l'algorithme. On suppose que l'utilisateur les connait et ne se trompe pas. On supposera donc que $f(a)$ et $f(b)$ sont de signe contraire.

algorithme dichotomie(f: (réel) → réel, a: réel, b: réel, epsilon: réel) → réel:
  (c := réel) ← $(b-a)/2$
  tant que ($|b-a| >$ epsilon **:**
    si $((f(a) \geq 0)$ **et** $(f(c) \leq 0))$ **ou** $(f(a) \leq 0)$ **et** $(f(c) \geq 0)$ **:**
      $b \leftarrow c$
    sinon :
      $a \leftarrow c$
    $c \leftarrow (b-a)/2$
  rendre c

Difficulté : ★ ★ ★ Question :

Donnez la complexité de l'algorithme précédent. Explicitez bien les paramètres que vous utilisez pour la mesurer.

À chaque étape $b-a$ va être divisé par 2 et l'algorithme va s'arrêter lorsque cet longueur sera plus petite que epsilon. Au bout de n étapes cette longueur vaut $(b-a)/2^n$, le nombre total d'étapes sera le plus petit n tel que $(b-a)/2^n \leq$ epsilon c'est a dire : $n = \log_2((b-a)/\text{epsilon})$. On a donc une complexité de $C(a, b, \text{epsilon}) = \Theta(\log_2((b-a)/\text{epsilon}))$ puisque toutes les autres opérations sont en temps constant.