Aix-Marseille-Université

Test 1 UE Algorithmie et programmation 2026

Contrôle 1A

10-06-2025

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

Nom/Prenom	

Barème. Pour chaque question :

- ne pas répondre donne 0 point,
- répondre de façon exacte donne : $+\frac{2}{3}$ points
- répondre de façon inexacte[1] :
  - pour une réponse de type VRAI/FAUX : $-\frac{2}{3}$ points,
  - pour une réponse libre : $-\frac{2}{3}$ points,
  - pour une réponse de type 1 parmi 3 : $-\frac{1}{3}$ points.

[1]le nombre de points négatifs varie pour que si l'on répond de façon aléatoire l'espérance soit nulle

Base théorique

Difficulté : ★ ☆ ☆ Question :

Quelle est la différence entre un programme et un algorithme ?

Un algorithme s'arrête quelques soient ses entrées.

Difficulté : ★ ★ ☆ Question :

Que démontre le théorème de Rice ?

① Que l'on peut pas démontrer qu'un algorithme donné résolve un problème donné

② Que l'on peut pas trouver un ensemble fini de caractéristiques qui prouvent qu'un algorithme quelconque résolve un problème donné

③ Que l'on peut pas trouver un ensemble fini de caractéristiques qui prouvent qu'un algorithme donné résolve un problème donné

Difficulté : ★ ★ ☆ Question :

Que signifie le problème de l'arrêt d'un programme ?

① Que l'on peut pas démontrer qu'un programme donné s'arrête pour une entrée donnée

② Que l'on peut pas trouver un ensemble fini de caractéristiques qui prouvent qu'un algorithme donné s'arrête pour toutes les entrées possibles

③ Que l'on peut pas trouver un ensemble fini de caractéristiques qui prouvent qu'un programme quelconque s'arrête pour une entrée donnée

Difficulté : ★ ★ ☆ Question :

Quelles sont les conséquences des deux questions précédentes pour nos propres algorithmes ?

Pour chaquen de nos programmes, Il faudra prouver que c'est un algorithme et démontrer qu'il résoud bien le problème annoncé. Cela ne peut pas être fait automatiquement via une preuve générique ou un algorithme.

Difficulté : ★ ★ ★ Question :

Combien existe-t-il d'algorithmes ?

① Il existe $\epsilon > 0$ tel qu'il existe autant d'algorithme que de réels dans $[0, \epsilon]$

② Autant que de nombres entiers

③ Autant que le nombre de fonctions de $\mathbb{N}$ dans $\mathbb{N}$

④ Autant que de multiples de π

⑤ Autant que de nombres réels

Il y a autant de multiples de π (π, 2π, 3π, ...) que de nombres entiers. Pour le reste c'est du cours : il y a autant de fonctions de $\mathbb{N}$ dans $\mathbb{N}$ que de réels dans $[0, 1]$ et il y a autant de réels dans $[0, \epsilon]$ qu'il n'y en a dans $[0, 1]$.

Remplacement de valeurs

Difficulté : ★ ★ ★ Question :

Écrire (et démontrer) un algorithme de signature `remplace(T: [entier], i: entier, j: entier)` qui remplace tous les éléments de T valant initialement `T[i]` par des éléments valant `T[j]`.

algorithme remplace(T: [entier], i: entier, j: entier):
  $c := T[i]$
  pour chaque ($k :=$ entier) de $[0..T.$longueur$[$**:**
    si $T[k] == c$ **:**
      $T[k] \leftarrow T[j]$

L'invariant de boucle est : pour tout $l \geq k$:

$$\begin{cases} T[l] = T[j] & \text{si } T[l] \text{ valait initialement } c \\ T[l] & \text{inchangé sinon} \end{cases}$$

L'invariant est trivialement vérifié et donc en fin de boucle on a bien que seules les valeurs des $T[k]$ valant initialement c ont été modifiées. Comme c vaut la valeur initiale de $T[i]$ l'algorithme résout bien le problème demandé.

Attention cependant, l'utilisation de la variable c est indispensable. En effet l'algorithme suivant (c'est la première idée qui vient) appelé avec `remplace_faux([1, 2, 1], 0, 1)` transforme le tableau en `[2, 2, 1]` à la place de `[2, 2, 2]` car au bout de la première itération $T[0]$ vaut 2.

algorithme remplace_faux(T: [entier], i: entier, j: entier):
  pour chaque ($k :=$ entier) de $[0..T.$longueur$[$**:**
    si $T[k] == T[i]$ **:**
      $T[k] \leftarrow T[j]$

Difficulté : ★ ☆ ☆ Question :

Pourquoi l'algorithme de la question précédente ne rend-il rien ?

Parce que le tableau est modifié par l'algorithme.