
Aucun document autorisé. Les téléphones doivent être rangés.
Ce contrôle n'est pas anonyme. Incrivez sur la copie vos nom, prénom et mention de Licence, sans replier le coin supérieur droit de la copie.
Lisez attentivement le sujet. Pour répondre à une question vous pouvez utiliser le résultat d'une question précédente même si vous n'avez pas traitée cette dernière.
Lorsqu'une question vous demande la complexité d'un algorithme, vous devez donner son ordre de grandeur (notation θ).

---

1. L'algorithme suivant a pour donnée un entier naturel et renvoie comme résultat un entier naturel. Vous devez deviner ce que calcule cet algorithme et en faire la preuve :

Algorithme 1 : truc(**d** n : entier) : entier

```
Données : n ∈ ℕ
Résultat : Renvoie ? ? ?
Variables : i, r, x ∈ ℕ
début
    i ← 0 ; r ← 1 ; x ← 0 ;
    tant que  i ⩽ n faire
        i ← i + 1;
        r ← r + x - i;
        x ← x + 3;
    fin tq
1   renvoyer  r
fin algorithme
```

1.a **(0,5 point)** Faites la preuve d'arrêt de cet algorithme.

1.b **(0,5 point)** Faites la trace de l'exécution de l'algorithme pour $n = 5$.

1.c **(4 points)** Donnez un invariant liant les valeurs des variables x et i et un invariant liant les valeurs des variables r et i. Prouvez ces 2 invariants.

1.d **(1 point)** Déduire des invariants la valeur de la variable r à la fin du tant que (ligne 1). Que calcule l'algorithme `truc` ?

2. **(5 points)** Écrivez un algorithme `plusProche` qui étant donné T un tableau d'entiers trié et e un entier renvoie un des éléments de T les plus proches de e.

Ex si T est le tableau | 3 | 5 | 9 | 14 | 15 | 19 |

`plusProche(T,6)` renvoie 5 car 5 est l'élément de T le plus proche de 6.

`plusProche(T,14)` renvoie 14 car 14 est l'élément de T le plus proche de 14.

`plusProche(T,7)` renvoie soit 5 soit 9 car 5 et 9 sont les deux éléments de T les plus proches de 7 et ils sont tous les deux aussi proches de 7.

Votre algorithme pour `plusProche` doit avoir une complexité dans le pire des cas en $\theta(log_2(n))$ où n est la taille du tableau. Vous donnerez un invariant significatif vérifié par votre algorithme.

3. Soit T un tableau d'entiers. On cherche dans T la longueur du plus grand sous-tableau trié. Formellement on cherche à calculer `lgSousTabTrié(T)` défini par :

$$max(\{j-i+1 : i \in [0 \dots taille(T)-1], j \in [i \dots taille(T)-1] \; et \; T[i] \leqslant T[i+1] \leqslant T[i+2] \leqslant \dots \leqslant T[j-1] \leqslant T[j]\})$$

ex

si T est le tableau | 13 | 5 | 19 | 14 | 16 | 19 | 8 | `lgSousTabTrié(T)=3` car | 14 | 16 | 19 | est le plus long sous tableau trié de T.

si T est le tableau | 3 | 9 | 7 | 4 | 4 | 3 | 1 | `lgSousTabTrié(T)=2` car les plus longs sous-tableaux triés de T sont | 3 | 9 | et | 4 | 4 |.

3.a **(2 points)** Un premier algorithme pour ce problème est :

Algorithme 2 : lgSousTabTrié1(**d** T : tableau d'entiers) : entier

Données : T un tableau d'entiers
Résultat : Renvoie la longueur d'un plus long sous-tableau trié de T
Variables : $i, j, lgMax \in \mathbb{N}$;
début
  lgMax ← 0 ;
  pour i `de 0 à taille(`T`)-1` **faire**
    j ← i+1 ;
    tant que `j`$\leqslant$`taille(T)-1 et T[j-1]` $\leqslant$ `T[j]` **faire**
      si `lgMax < j-i+1` **alors**
        lgMax ← j-i+1 ;
      fin si
      j ← j+1;
    fin tq
  finpour
  renvoyer `lgMax`;
fin algorithme

Quel est le meilleur des cas pour cet algorithme ? Quelle est la complexité dans le meilleur des cas ?

Quel est le pire des cas pour cet algorithme ? Quelle est la complexité dans le pire des cas ?

3.b **(7 points)** Écrivez un second algorithme pour `lgSousTabTrié` qui améliore la complexité dans le pire des cas de `lgSousTabTrié1`. Quelle est cette complexité dans le pire des cas ? Vous donnerez des invariants significatifs et précis (rôles des variables) vérifiés par votre algorithme.