Révisions

Donnez un algorithme pour les problèmes suivants :

Algorithme 1 : recherche(**d** T : tableau d'entier, **d** x : entier, **r** present : booléen, **r** PP : entier)

Données : T 1 tableau d'entiers ; x un entier
Résultat : Si T contient un élément de valeur x, *present* est `true`, sinon *present* est `false` et PP est un des éléments de T les plus proches de x, c'est à dire $PP \in T$ et $\forall i \in [0..taille(T)[, |x - PP| \leq |x - T[i]|$

Algorithme 2 : plusPetitEcart(**d** T : tableau d'entier) : entier

Données : T : tableau d'entier
Résultat : renvoie le plus petit écart entre 2 éléments distincts de T, autrement dit $min\{|T[i] - T[j]|$ avec $i, j \in [0, taille(T)[$ et $i \neq j\}$

Algorithme 3 : Somme ?(**d** A : tableau ,**d** B : tableau , **d** Som :entier, **r** existe : Booléen, **r** i :entier, **r** j : entier)

Données : A et B 2 tableaux d'entiers de même taille n et Som un entier
Résultat : si il existe un élément de A et un élément de B dont la somme vaut Som, *existe* est `true` et $A[i] + B[j] = Som$, sinon *existe* est `false`

Algorithme 4 : LignesEgales ?(**d** T : Tableau) :Booléen

Données : $T[0...n-1, 0...n-1]$ un tableau de n lignes et n colonnes et composé de 0 et de 1
Résultat : renvoie Vrai si et seulement si T possède 2 lignes identiques, c'est à dire ssi $\exists i, j \in [0...n[$ tels que $i \neq j$ et $\forall k \in [0...n[, \quad T[i, k] = T[j, k]$

Algorithme 5 : InsérerTabTrié(**dr** T : tableau d'entier, **d** m : entier, **d** e : entier)

Données : $m < taille(T) - 1$, le sous-tableau $T[0..m]$ est trié $\nearrow$; le sous-tableau $T[m+1..taille(T)[$ est *"vide"*
Résultat : modifie T en y insérant e parmi les m premiers éléments : le sous-tableau $T[0..m+1]$ est trié

Preuve d'arrêt

Algorithme 6 : BB(**d** N : entier, **r** f : entier)

Données : $N \in \mathbb{N}^*$
Résultat : $f = ?$
Variable : $n \in \mathbb{N}$
début
  $n \leftarrow N$; $f \leftarrow 7!$;
  tant que $n \neq 7$ **faire**
    si $n > 7$ **alors** $f \leftarrow f \times n$; $n \leftarrow n - 1$
    sinon
      $n \leftarrow n + 1$; $f \leftarrow f/n$
    fin si
  fin tq
fin algorithme

Algorithme 7 : AA(**d** n : entier) : entier

Données : $n \in \mathbb{N}$
Résultat : Renvoie ? ? ?
Variables : $i, r, x \in \mathbb{N}$
début
  i ← 0 ; r ← 1 ; x ← 0 ;
  tant que $i \leqslant n$ **faire**
    i ← i + 1 ; r ← r + x - i ; x ← x + 3;
  fin tq
  renvoyer `r`
fin algorithme

Algorithme 8 : PGCD(**d** N, M : entier) : entier

Données : $N, M \in \mathbb{N}^*$
Résultat : renvoie $pgcd(N, M)$
Variables : $n, m \in N$
début
  $n \leftarrow N$; $m \leftarrow M$;
  tant que $n \neq 0$ `et` $m \neq 0$ **faire**
    si $m > n$ **alors** $m \leftarrow m - n$
    sinon
      $n \leftarrow n - m$
    fin si
  fin tq
  renvoyer $m + n$
fin algorithme

Algorithme 9 : gcd(**d** n : entier, **d** p : entier) :entier

Données : $n, p \in \mathbb{N}, n \neq 0$ *ou* $p \neq 0$
Résultat : renvoie le pgcd de n et p
début
  si `p=0` **alors**
    renvoyer `n`
  sinon
    renvoyer `gcd(p,n mod p)`
  fin si
fin algorithme

1. Établir la preuve d'arrêt de l'algorithme 6. Que calcule cet agorithme ?
2. L'algorithme 7 s'arrête-t-il ? Pourquoi ?
3. On considère l'algorithme 8. Dites pour chacune des expressions suivantes si leur valeur est dans $\mathbb{N}$ et, dans l'affirmative, si elle décroît strictement à chaque itération. m ; n ; $m - n$; $n - m$; $|m - n|$; $m + n$; $m * n$
   Cet algorithme s'arrête-t-il ?
4. L'algorithme 9 s'arrête-t-il ? Pourquoi ?

Propriété invariante

1. Montrez que l'algorithme 13 s'arrête, démontrez les invariants $A = 3(X - C) + 1$; $B = 3(X - C)^2$. On admet sans le montrer l'invariant $Z = (X - C)^3$. Que calcule alors cet algorithme ?
2. Parmi les égalités suivantes, quelles sont celles vérifiées par l'itération de l'algorithme 11 : $(r_1 = i)$, $(r_1 = i-1)$, $(r_1 = fib(i))$, $(r_2 = r_1 + 1$ si $i \neq 1)$, $(r_2 = fib(i + 1))$? En déduire la valeur à renvoyer pour que l'algorithme soit correct. Rappel fib(0)=0, fib(1)=1 et fib(n)=fib(n-1)+fib(n-2) pour $n > 1$.
3. Faites une trace de l'algorithme 8 pour les données $M = 48, N = 30$. Pour chaque itération calculez l'ensemble des entiers divisant à la fois m et n. Que constatez-vous ? Quelle propriété, faisant intervenir l'ensemble des diviseurs communs à N et M, semble invariante pour cette itération ?
4. Faites la trace d'exécution de l'algorithme 7 pour $n = 5$. 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. En déduire la valeur de la variable r à la fin du tant que. Que calcule l'algorithme 7 ?
5. Montrez que l'algorithme 12 s'arrête. Faites une trace de l'algorithme pour $a = 18$. Donnez un invariant liant les valeurs de p et de i, un autre liant q et i. Montrez ces invariants. Montrez que l'algorithme est correct.
6. (*) Montrez en donnant un invariant que l'algorithme 6 est correct.

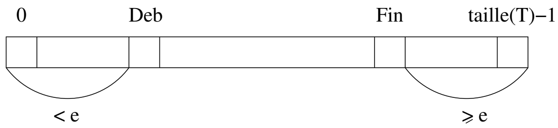
Écriture d'algorithme et de preuve

1. Donnez un algorithme calculant le produit de 2 entiers par simples additions successives. Vous utiliserez un `Tant Que` pour votre itération. Faites-en la preuve.
2. On cherche un algorithme pour le problème suivant :

   Algorithme 10 : rechDichoVersion2(**d** T : tableau d'entier,**d** e : entier, **r** present : booléen, **r** ind : entier)

   Données : T un tableau d'entiers triés ;$e \in \mathbb{N}$
   Résultat : `present` =Vrai si et seulement si $e \in T$; Si `present` alors `ind`= $\min\{i \in [0 \ldots taille(T)[: T[i] = e\}$

   Modifiez l'algorithme de recherche dichotomique vu en cours pour avoir l'invariant suivant :

3. Un enseignant a stocké les notes (entiers compris entre 0 et 20) de ses étudiants dans un tableau en les rangeant par valeurs croissantes. Comment peut-il vérifier si le nombre d'étudiants ayant obtenu la moyenne est strictement supérieur au nombre d'étudiants n'ayant pas eu la moyenne ? Il veut à présent savoir combien d'étudiants n'ont pas eu la moyenne. Donnez un algorithme.
4. Un paquet de jeu de cartes (représenté par un tableau d'entiers (différents 2 à 2)) a été trié, puis coupé une fois (coupe non vide). On obtient par exemple le tableau

3	4	5	6	7	9	1	2

   Écrivez un algorithme qui trouve par dichotomie le lieu de coupe. Sur l'exemple le lieu de coupe est situé entre les indices 5 et 6, on convient de renvoyer l'indice 6.

Algorithme 11 : f(**d** n : entier) :entier

Données : n un entier strictement positif
Résultat : Renvoie $fib(n)$
Variables : $i \in \mathbb{N}, r_1 \in \mathbb{N}, r_2 \in \mathbb{N}, r_3 \in \mathbb{N}$;
début
  $i \leftarrow 0$; $r_1 \leftarrow 0$; $r_2 \leftarrow 1$;
  tant que $i < n$ **faire**
    $r_3 \leftarrow r_1$; $r_1 \leftarrow r_2$; $r_2 \leftarrow r_3 + r_2$; $i \leftarrow i + 1$;
  fin tq
  renvoyer ????;
fin algorithme

Algorithme 12 : Logentier(**d** a : entier ; **r** i : entier)

Données : a un entier strictement positif
Résultat : $i = \lfloor Log_2 a \rfloor$, i.e. i vérifie que $2^i \leqslant a < 2^{i+1}$
Variables p, q, i : Entiers ;
début
  $i \leftarrow 0$; $p \leftarrow 1$; $q \leftarrow 2$;
  tant que $q \leq a$ **faire**
    $p \leftarrow q$; $q \leftarrow p + p$; $i \leftarrow i + 1$;
  fin tq
fin algorithme

Algorithme 13 : CC(**d** X :entier, **r** Z :entier)

Données : $X \in \mathbb{N}$
Résultat : $Z \in \mathbb{N}$
Variables : $A, B, C \in \mathbb{N}$
début
  $A \leftarrow 1$; $B \leftarrow 0$; $C \leftarrow X$; $Z \leftarrow 0$;
  tant que $C > 0$ **faire**
    $Z \leftarrow Z + A + B$; $B \leftarrow B + 2 \times A + 1$;
    $A \leftarrow A + 3$; $C \leftarrow C - 1$;
  fin tq
fin algorithme