Rappels sur le pseudo code

Le pseudo code d'un algorithme débute toujours par les caractéristiques suivantes :

- Nom de l'algorithme
- Données (en entrée)
- Résultats (en sortie)
- Initialisation des variables locales

Rappels sur les structures utilisées dans les algorithmes

- Bloc « tant que » :

Structures répétitives (tant que)

Tant que CONDITION faire instruction ;
Instruction ;
Fin tant que

- bloc « si » :

Structures alternatives (si) => exécution exclusive.

Si CONDITION alors instruction si vrai
Sinon instruction si faux
Fin si

Rappels sur les structures utilisées dans les algorithmes (suite)

- Bloc « pour » :

  pour I allant de d à f faire instruction ; //séparateur d'instruction

  Instruction ;

  Fin pour

- bloc début

début
instruction ;
instruction ;
Fin

- Tableaux

A[i] élément du tableau avec A[0] 1er élément du tableau

- Chaînes de caractère : « »

- Affectation : A ← 14

Algorithme Recherche Linéaire
représentation graphique

4	17	5	179	16	8	16	50

Exemple : trouver la valeur 179 dans le tableau ci-dessus

Algorithme Recherche Linéaire
pseudo code

Algo : recherche-lineaire
Entrée : 1 tableau T, 1 variable v
Sortie : 1ère position de v dans T ou (-1) si v n'est pas dans T
Variable locale : pos

```
Pos <-- 0
        Tant que pos < taille T
        et T(pos) != v faire
        Pos <-- pos + 1
fin tant que
si pos >= taille T alors retourner -1
        sinon retourner pos
        fin si
```

Algorithme Recherche Linéaire
1ère implémentation scheme

```
(define (recherche-lineaire T v)
(let ((pos 0))
     ( let (boucle ()
          (if (<= pos (vector-length T))
          -1
          (if (equal? V (vector-ref T pos))
          pos
          (begin
          (set! Pos (+ 1 pos))
     (boucle)))))))
```

Algorithme Recherche Linéaire
2éme implémentation scheme

```
(define (recherche-lineaire T v)
      (let (boucle ((pos 0))
            (if (<= pos (vector-length T))
            -1
            (if (equal? V (vector-ref T pos))
            pos
            (boucle (+ 1 pos))))))
```

Algorithme Recherche Linéaire
comptage des opérations

3 opérations significatives ont été notées en rouge dans le pseudo code.

L'exercice consiste à compter la réalisation ces 3 opérations dans le cas d'1 recherche dans 1 tableau à n cases.

Dans 1 tableau à n cases, chaque cas a 1 probabilité égale à 1/n de se produire.

Algorithme Recherche Linéaire
comptage des opérations

Quel est le nombre moyen d'opérations réalisées dans l'algorithme ?

0*1/n + 1*1/n + 2*1/n + …. + (n - 1)*1/n
Cela équivaut à faire
(1 + 2 + …. + (n - 1))/n

Réalisation de la somme de Gauss :
L'astuce est de poser la somme DEUX fois
On écrit la somme de droite à gauche (ordre décroissant)
Et aussi de gauche à droite (ordre croissant)

S = 1 + 2 + 3 + … + (m - 1) + m m termes
S = m + (m - 1) + …. + 2 + 1 m termes

Algorithme Recherche Linéaire
comptage des opérations

2S = (m +1) + (m + 1) + (m + 1) + ….. + (m + 1)
2S = (m + 1)* m
Soit
S = m(m + 1)/2 ; formule générale

Dans notre cas particulier, nous avons 1 somme des (n - 1)/n premiers termes.
En appliquant la formule, cela donne :

S = n(n - 1)/2n soit
S = (n - 1)/ 2

Algorithme Recherche Linéaire
Croissance des algorithmes

Algorithme en O(n)
F est élément de O(g) s'il existe x0 et a tel que pour tout x > x0, f(x) < ag(x)

Dans notre cas de la recherche linéaire :
S = (n - 1) / 2
S < n
L'algorithme est en O(n)

Algorithme en Ω(n)
F est élément de Ω(g) s'il existe x0 et a tel que pour tout x > x0, f(x) > ag(x)

Algorithme en Θ(n)
F est élément de Θ(g) s'il existe x0, a1 et a2 tel que pour tout x > x0, a1g(x) < f (x) <a2g(x)

Algorithme Recherche Linéaire
bilan de la croissance et recherche d'amélioration

L'algorithme de recherche linéaire a une croissance en O(an) avec 0 < a < n.

Ce n'est pas l'algorithme de recherche dans un vecteur le plus rapide.

Peut-on améliorer cela ?

1/ trier les valeurs du tableau T en ordre croissant

3	7	8	12	24	31	40	50

Algorithme Recherche Linéaire
bilan de la croissance et recherche d'amélioration

1/ trier les valeurs du tableau T en ordre croissant (suite)
S'il on veut rechercher la variable v de valeur 9.

on part de la position 0 (pos) et on progresse (pos + 1) de case en case.
Lorsque l'on arrive à la valeur 12 du tableau , on sait que v = 9 n'est pas dans le tableau.
le programme retourne - 1.

Peut-on encore améliorer l'algo ?

Oui (il reste du temps avant la fin du cours)

2/ Algo de recherche par dichotomie

Algorithme Recherche Linéaire
bilan de la croissance et recherche d'amélioration

2/ Algo de recherche par dichotomie (suite)

Cette recherche suppose que le tableau soit trié dans l'ordre croissant :

a/ On choisit le milieu du tableau. Si le tableau possède un nombre de case pair, ce sera le rang n/2.
b/ On compare la valeur de v avec l'élément le plus petit ou plus grand que l'élément pivot.

Dans notre exemple, 3 comparaisons sont réalisées contre 4 dans le recherche non dichotomique.
En effet, à chaque itération, la taille du tableau T est divisée par 2.

n n/2 n/4
La progression est en O(log2(n))

Algorithme de recherche de la valeur maximum d'1 ensemble de valeurs

Nom de l'Algo : MAX
Données d'entrées : tableau T
Résultat : le rang de la plus grande valeur de T
Initialisation : vmax <-- 0
Pour i allant de 1 à longueur de T -1
Faire
    si T(i) < T(max) alors max <-- i
    fin si
Fait
Retourner max

Algorithme de recherche de la valeur maximum d'1 ensemble de valeurs
Implémentation scheme

```
(define (max T)
        (let ((imax 0)
              (l (vector-length T)))
        (do ((i 1 (+ i 1)))
            ((> i l)
               imax)
        (if (> (vector-ref T imax))
        (set! lmax i)))))
```

//procédure compteur
(set! n (+ 1 n))
//initialisation de n
(define n 0) au début et (set! n 0) à chaque exécution de programme.