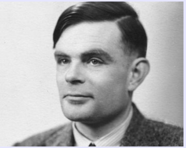
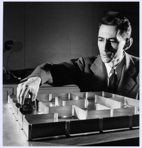
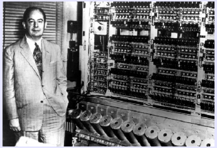

Information, Calcul et Communication

Compléments de cours

J.-C. Chappelier

Examen 1 2018 Q11

Quelle est la sortie de l'algorithme suivant sur l'entrée $L = (-3, 5, 12, -4, 3, 8, -1, -6, 4)$:

algo11

entrée : *L liste de valeurs*

sortie : *???*

$a \longleftarrow$ **taille**(L)

$R \longleftarrow \emptyset$ // Liste vide

Si $a \geq 3$

  Pour i de 1 à $a-2$

    Pour j de $i+1$ à $a-1$

      Pour k de $j+1$ à a

        Si $L[i] + L[j] + L[k] = 0$

          $R \longleftarrow (i, j, k)$

Sortir : R

A] $\emptyset$ (liste vide)

*B] $(2, 4, 7)$

C] $(5, -4, -1)$

D] $(1, 7, 9)$

Note : on aurait aussi pu ajouter :

E] (1,7,9,2,4,7) ☞ attention à la différence avec $R \longleftarrow R \oplus (i, j, k)$

F] (7,4,2) ☞ attention à l'ordre dans lequel on parcourt (et dans lequel on écrit)

Examen 1 2018 Q12

Si l'on note n la taille de la liste L, quelle est la complexité de l'algorithme de la question précédente ?

*A] $\Theta(n^3)$

B] $\Theta(2^n)$

C] $\Theta(n^2)$

D] $\Theta(n^4)$

Examen 1 2018 Q13

On s'intéresse ici à raccourcir les répétitions de trois ou plus valeurs identiques successives ; par exemple à produire la liste $(6, 6, 4, 4, 12, 4, 6)$ à partir de la liste $(6, 6, 4, 4, 4, 12, 4, 6)$ en supprimant le 4 en cinquième position car il est présent trois fois consécutives.
À noter que :

- les seules valeurs supprimées sont celles qui sont répétées successivement trois fois ou plus (l'une à la suite de l'autre) ; on ne garde alors que deux de ces valeurs (cf la valeur 4 ci-dessus) ;
- toute valeur présente une ou deux fois successivement est préservée, et l'on conserve l'ordre de la liste ;
- en sortie on ne peut donc pas avoir plus de deux valeurs identiques consécutives.

Ecrivez un algorithme itératif (c.-à-d. non récursif, mais avec des boucles) résolvant ce problème.

Examen 1 2018 Q13

Voici une solution possible :

simplifie1

entrée : *L, liste de nombres*
sortie : L', *version expurgée de L*

$n \longleftarrow$ **taille**(L)
Si $n \leq 2$
  Sortir : L

$L' \longleftarrow (L[1], L[2])$
Pour i allant de 3 à n
  Si $L[i] \neq L[i-1]$ **ou** $L[i] \neq L[i-2]$
    $L' \longleftarrow L' \oplus L[i]$

Sortir : L'

Plus de solutions et des commentaires dans le corrigé officiel.

Examen 1 2018 Q14

Déterminez la complexité de votre algorithme.
Justifiez votre réponse.

La complexité de l'algorithme précédent est en $\Theta(n)$, où n est la taille de la liste (précisez vos notations !). On ne parcourt en effet qu'une seule fois la liste.

Leçon I.2 (conception d'algorithmes) – Points clés

- approche descendante : **DÉCOMPOSEZ** le problème
- algorithmes récursifs :
  - ramener le problème à la résolution du *même* problème sur moins de données
  - penser à la condition d'arrêt
- programmation dynamique :
  stocker/mémoriser au lieu de recalculer
- problèmes de plus courts chemins
  complexité polynomimale : $\Theta(n^3)$, $\Theta(n^2)$ ou $\Theta(n)$ en fonction de la nature du problème (nombre de villes de départ/d'arrivée fixées)

Leçon I.2 (conception d'algorithmes) – Dichotomie

Écrire complètement l'algorithme de recherche par dichotomie dans une liste :

- spécifier le problème
- « couper la liste en deux » : comment faire ?
  ☞ je vous impose de passer 2 paramètres supplémentaires en entrée : indice de début de recherche et indice de fin de recherche (inclus)

On a donc :
Entrée : liste L *ordonnée*, valeur v, indice i (début), indice j (fin)
Sortie : vrai ou faux

Leçon I.2 (conception d'algorithmes) – Dichotomie

recherche

entrée : *liste L ordonnée, valeur v, indice i (début), indice j (fin)*
sortie : *vrai ou faux*

Si $j < i$
  Sortir : *faux*

$m \longleftarrow \lfloor \frac{i+j}{2} \rfloor$
Si $v = L[m]$
  Sortir : *vrai*

Sinon, **si** $v < L[m]$
  Sortir : recherche(L, v, i, m-1)

Sortir : recherche(L, v, m+1, j)

Leçon I.1b (algorithmes, complexité) – Élements maximaux

Écrire un algorithme, *récursif* cette fois, pour :

- trouver tous les éléments maximaux dans une liste

NOTE : il y a plein de façons de créer le sous-problème à rappeler dans un algorithme récursif sur des listes, parmi lesquelles (commencez par essayer celles-ci) :

- couper la liste en deux
- supprimer un élément de la liste

Leçon I.1b (algorithmes, complexité) – Élements maximaux

Version « supprimer un élément » :

elmax1

entrée : *liste L non vide*

sortie : *liste des positions des valeurs maximales*

$n \longleftarrow$ **taille**(L)

Si $n = 1$

  Sortir : (1)

$L' \longleftarrow$ **ajoute**$(1,$**elmax1**$(L[2], \cdots, L[n]))$

Si $L[1] = L[L'[1]]$

  Sortir : $(1) \oplus L'$

Sinon, si $L[1] > L[L'[1]]$

  Sortir : (1)

Sortir : L'

avec :

ajoute

entrée : *valeur v, liste L*

sortie : *liste des valeurs de L augmentées de v*

$n \longleftarrow$ **taille**(L)

$L' \longleftarrow ()$

Pour i de 1 à n

  $L' \longleftarrow L' \oplus (L[i] + v)$

Sortir : L'

Leçon I.1b (algorithmes, complexité) – Élements maximaux

Version « couper en deux » :

elmax2

entrée : *liste L non vide*

sortie : *liste des positions des valeurs maximales*

$n \longleftarrow$ **taille**(L)

Si $n = 1$

  Sortir : (1)

$m \longleftarrow \lfloor \frac{n}{2} \rfloor$

$L' \longleftarrow$ **elmax2**$(L[1], \cdots, L[m])$

$L'' \longleftarrow$ **ajoute**$(m,$**elmax2**$(L[m+1], \cdots, L[n]))$

Si $L[L'[1]] = L[L''[1]]$

  Sortir : $L' \oplus L''$

Sinon, si $L[L'[1]] > L[L''[1]]$

  Sortir : L'

Sortir : L''

Leçon I.1b (algorithmes, complexité) – Élements maximaux

Question subsidiaire : quelle est la complexité de ces algorithmes ?

NOTE : pour calculer la complexité d'algorithmes récursifs, vous avez trois moyens « pratiques » :

1. *Compter* les instructions
   l'inconvénient est que cela conduit à une équation sur la complexité (fonction), qu'il est parfois (souvent ?) difficile de résoudre
2. *dessiner* le graphe des appels depuis la taille n jusqu'à toutes les terminaisons et compter alors le nombre d'arcs
   (revoir l'exemple du cours des appels du calcul récursif des coefficients du binôme)
3. utiliser la méthode « incrémenter et compter » : de combien augmente la complexité si j'augmente la taille de l'entrée de 1 ?
   ☞ cela donne une estimation de la dérivée de la complexité (fonction)