Expression des algorithmes

un bon niveau d'abstraction

Groupe algorithmique de l'IREM de Grenoble
{Anne.Rasse,**Jean-Marc.Vincent**,Benjamin.Wack}@imag.fr
{Maryline.Althuser,Herve.Barbe}@ac-grenoble.fr

2017

EXPRESSION DES ALGORITHMES

Expression des algorithmes

un bon niveau d'abstraction

Groupe algorithmique de l'IREM de Grenoble
{Anne.Rasse,**Jean-Marc.Vincent**,Benjamin.Wack}@imag.fr
{Maryline.Althuser,Herve.Barbe}@ac-grenoble.fr

2017

LE MONDE

Facebook annonce une série de mesures pour lutter contre les fausses informations ***16.12.2016***

..... Facebook a également annoncé qu'il allait **ajuster l'algorithme de son fil d'actualité** pour refléter le fait que, selon l'entreprise, les utilisateurs ont tendance à moins partager un article si, après l'avoir lu, ils se sentent trompés par son titre. .....

Prix « Le Monde » de la recherche 2016 : aménager les villes grâce aux algorithmes ***23.11.2016***

Quand les algorithmes président aux destinées des élèves ***14.11.2016***

... Particulièrement bien adaptés à la tradition jacobine française, ces algorithmes ont eu depuis quinze ans le mérite de sortir l'orientation des élèves des décisions arbitraires, des passe-droits, des files d'attente et de la règle du *premier arrivé, premier servi* qui régnait par le passé. *L'**objectif de ces algorithmes est de mettre tous les élèves sur un pied d'égalité, puisque toutes les demandes sont traitées en même temps**, commente l'économiste Julien Grenet, spécialiste des algorithmes de répartition scolaire, mais cela ne fonctionne que si on garantit la transparence du processus et des critères de priorité, et si tout le monde les connaît.* Et c'est là, en France, que le bât blesse.
Développés de manière empirique, sans réels débats démocratiques à même de les justifier et d'éloigner les soupçons, les zones d'ombre de ces algorithmes occultent parfois les bénéfices apportés....

LE FIGARO

Un algorithme pour détecter les pensées suicidaires ***22/11/2016***

... Leurs réponses, passées à la **moulinette d'un algorithme**, permettront de repérer les patients à risque de rechute. ...

Google revoit son algorithme ***21/12/2016***

... Des ajustements récents dans les algorithmes de Google "aideront à faire apparaître des contenus plus crédibles et de meilleure qualité", a encore affirmé l'entreprise. ...

La France se penche sur la régulation des algorithmes ***16/12/2016***

... Axelle Lemaire a annoncé les premières mesures entreprises par Bercy pour faire face à la montée en puissance des algorithmes. ...
Le Conseil national du numérique (CNNum) a été saisi pour réfléchir à un outil grand public capable de renseigner les mauvaises expériences rencontrées par des utilisateurs avec des algorithmes, tandis que l'INRIA coordonnera le lancement d'une plateforme scientifique explorant l'enjeu éthique des algorithmes. ...

DAUPHINÉ LIBÉRÉ

Pôle emploi lance le site Bob-emploi pour aider les chômeurs textit15/11/2016

... Ce site, qui se veut un «service public citoyen», **fonctionne avec un algorithme** capable d'analyser les données sur le marché du travail, la situation de chaque chômeur, et, fait innovant, sur les parcours anonymisés de millions de chômeurs précédents, pour les mettre à disposition de l'utilisateur. ...

Êtes-vous beau ? Une intelligence artificielle note votre visage *09/10/2016*

... La beauté est subjective. Pourtant des ingénieurs de l'agence japonaise Party ont mis au point **un algorithme permettant de répondre à la question "Êtes-vous beau ?"**. Sur la plate-forme Deeplooks, une intelligence artificielle est ainsi capable de noter entre 1 et 5 la beauté de votre visage à partir d'une simple photo. ...

VOUS AVEZ DIT *algorithme* ?

PRÉPARATION 10 min
CUISSON 15 min
POUR **12 madeleines**

100 g de farine
3 g de levure chimique
100 g de beurre
1/4 de citron non traité
2 œufs
120 g de sucre en poudre

Madeleines

1 Tamisez ensemble la farine et la levure au-dessus d'un bol.
2 Faites fondre le beurre dans une petite casserole et laissez-le refroidir.
3 Hachez finement le zeste du 1/4 de citron.
4 Cassez les œufs dans une terrine, versez le sucre par-dessus. Fouettez pendant 5 minutes pour bien les faire mousser ; ajoutez le mélange farine-levure en pluie, puis le beurre et le zeste haché, sans cesser de tourner.
5 Préchauffez le four à 220 °C.
6 Beurrez légèrement la plaque à madeleines et remplissez-la de pâte seulement aux deux tiers. Mettez au four pendant 5 minutes à 220 °C puis baissez la température à 200 °c et laissez cuire encore 10 minutes.
7 Démoulez les madeleines tièdes et laissez-les refroidir.

Extrait du Larousse des Desserts par Pierre Hermé

VOUS AVEZ DIT *algorithme* ?

PRÉPARATION 10 min

CUISSON 15 min

POUR **12 madeleines**

100 g de farine

3 g de levure chimique

100 g de beurre

1/4 de citron non traité

2 œufs

120 g de sucre en poudre

Madeleines

1 Tamisez ensemble la farine et la levure au-dessus d'un bol.

2 Faites fondre le beurre dans une petite casserole et laissez-le refroidir.

3 Hachez finement le zeste du 1/4 de citron.

4 Cassez les œufs dans une terrine, versez le sucre par-dessus. Fouettez pendant 5 minutes pour bien les faire mousser ; ajoutez le mélange farine-levure en pluie, puis le beurre et le zeste haché, sans cesser de tourner.

5 Préchauffez le four à 220 °C.

6 Beurrez légèrement la plaque à madeleines et remplissez-la de pâte seulement aux deux tiers. Mettez au four pendant 5 minutes à 220 °C puis baissez la température à 200 °c et laissez cuire encore 10 minutes.

7 Démoulez les madeleines tièdes et laissez-les refroidir.

Extrait du Larousse des Desserts par Pierre Hermé

VOUS AVEZ DIT *algorithme* ?

PRÉPARATION 10 min
CUISSON 15 min
POUR **12 madeleines**

100 g de farine
3 g de levure chimique
100 g de beurre
1/4 de citron non traité
2 œufs
120 g de sucre en poudre

Madeleines

1 Tamisez ensemble la farine et la levure au-dessus d'un bol.
2 Faites fondre le beurre dans une petite casserole et laissez-le refroidir.
3 Hachez finement le zeste du 1/4 de citron.
4 Cassez les œufs dans une terrine, versez le sucre par-dessus. Fouettez pendant 5 minutes pour bien les faire mousser ; ajoutez le mélange farine-levure en pluie, puis le beurre et le zeste haché, sans cesser de tourner.
5 Préchauffez le four à 220 °C.
6 Beurrez légèrement la plaque à madeleines et remplissez-la de pâte seulement aux deux tiers. Mettez au four pendant 5 minutes à 220 °C puis baissez la température à 200 °c et laissez cuire encore 10 minutes.
7 Démoulez les madeleines tièdes et laissez-les refroidir.

Extrait du Larousse des Desserts par Pierre Hermé

Informellement :
un **algorithme** est un processus systématique pour réaliser un objectif.

UN PEU D'HISTOIRE : LES PREMIERS ALGORITHMES

Euclide d'Alexandrie
III^{ième} siècle avant JC.

Calcul sur les entiers
$PGCD(a, b)$

Muhammad ibn Musa al-Khwarizmi
IX^{ième} siècle.

Calcul algébrique
$5 = 3.x + 2$

DÉFINITION DU *Grand Robert*

algorithme [algɔʀitm] **n. m.**

ÉTYM. 1554; *algorisme,* XIII^e^; *augorisme,* v. 1230, « calcul en chiffres arabes »; au XVI^e^, « arithmétique »; anc. esp. *alguarismo,* lat. médiéval *algorithmus,* nom latinisé du grand mathématicien arabe surnommé *(')ăl-hŭwārĭzmī* (*Al-Khawarizmi,* → Algèbre) pris comme nom commun, également sous la forme *algorismus.*

◆ **Didactique.**

1 **Hist. des sc.** Système de numérotation décimale (emprunté aux Arabes).

◆ **Vx.** Règles de l'arithmétique élémentaire.

◆ Règles opératoires intervenant dans une des opérations de l'arithmétique. | *L'algorithme de la division.*

2 **Mod.** Ensemble des règles opératoires propres à un calcul. → Mathématique, cit. 0.2. | *Algorithmes du calcul intégral, des puissances, etc.* | *Algorithme d'Euclide* (ou *du plus grand commun diviseur*). | *Algorithme infinitésimal de Leibniz.*

◆ **Par ext.** Suite de règles formelles explicitée par une représentation de type mathématique et correspondant à un enchaînement nécessaire; cette représentation mathématique. | *Construction mentale à base d'algorithmes.* | *Les algorithmes de la logique formelle.* — **Inform.** Calcul, enchaînement des actions nécessaires à l'accomplissement d'une tâche. → **Automate.** | *Langage destiné aux algorithmes.* → **Algol.**

0 Intuitivement, un algorithme est un ensemble de règles qui permet de réaliser mécaniquement toute opération particulière correspondant à un type d'opération. On peut encore dire que c'est une procédure mécanique qui, appliquée à une certaine classe de symboles (symboles d'entrée), fournit, éventuellement, un symbole de *sortie* (...)
a) Un algorithme est un ensemble d'instructions de *taille finie* (...)
b)Un opérateur (humain, mécanique, optique, etc., ou électronique) réagit aux instructions et effectue le calcul.
c)Des dispositifs (papier et crayon, roues dentées, mémoires magnétiques, etc.) permettent d'effectuer, de stocker et de retrouver les différentes étapes du calcul.
d) *Les processus sont essentiellement discrets* (...)
e) La suite des opérations élémentaires à effectuer est *parfaitement déterminée* (...)

M. GROSS et A. LENTIN, Notions sur les grammaires formelles, p. 43-44.

DÉFINITIONS

Dictionnaire *Larousse*

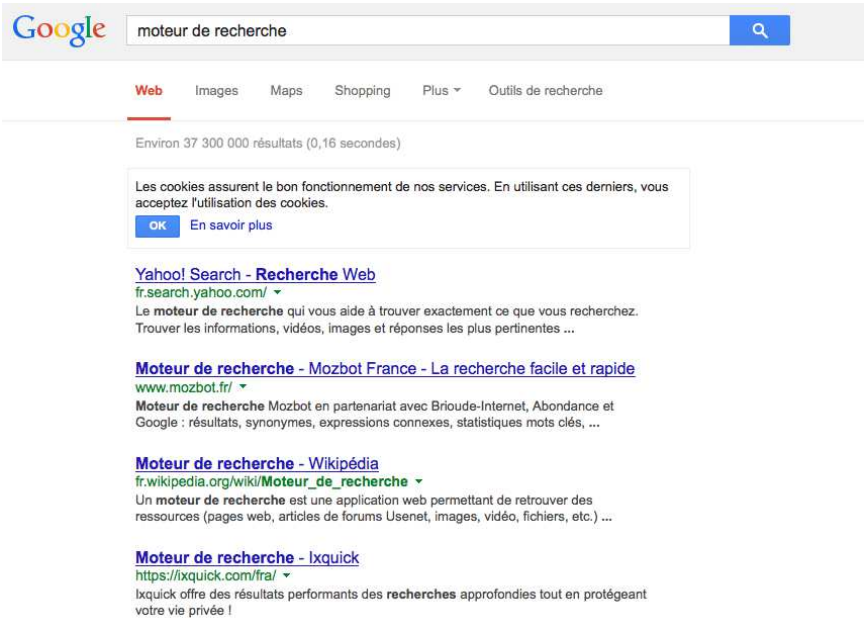
Ensemble de règles opératoires dont l'application permet de résoudre un problème énoncé au moyen d'un nombre fini d'opérations. Un algorithme peut être traduit, grâce à un langage de programmation, en un programme exécutable par un ordinateur.

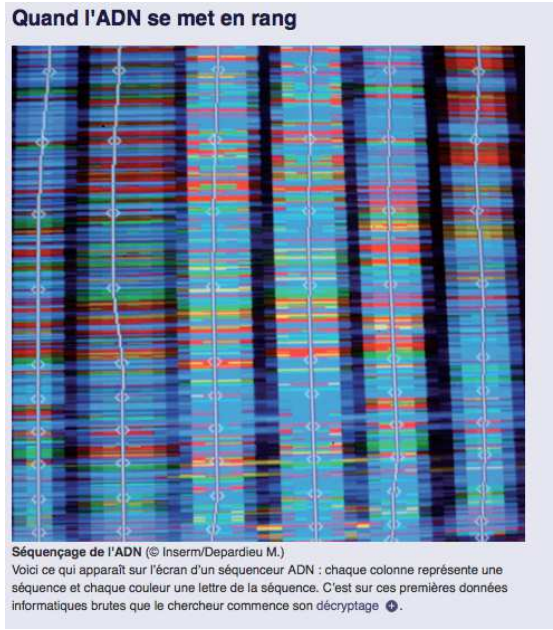
Dictionnaire *CNRTL* ALGORITHME, subst. masc.

A. MATHÉMATIQUES
1. Anciennement
a) Système de numération décimale en chiffres arabes.
b) „Ensemble des règles du calcul des nombres écrits dans le système décimal (les « quatre règles »).“ (Lal. 1968).
c) Ensemble des règles opératoires intervenant dans toute espèce de calcul. L'algorithme de la division (Lar. encyclop.), l'algorithme de la multiplication (Foulq.-St-Jean 1962) :
2. Sens mod. Ensemble de symboles et de procédés propres à un calcul algorithme du calcul intégral, algorithme du calcul des sinus, algorithme des puissances, algorithme des différences... (Lar. 19e) ; p. ext. „ensemble de formules, de signes et de conventions accessibles aux seuls initiés“ (Quillet 1965) :

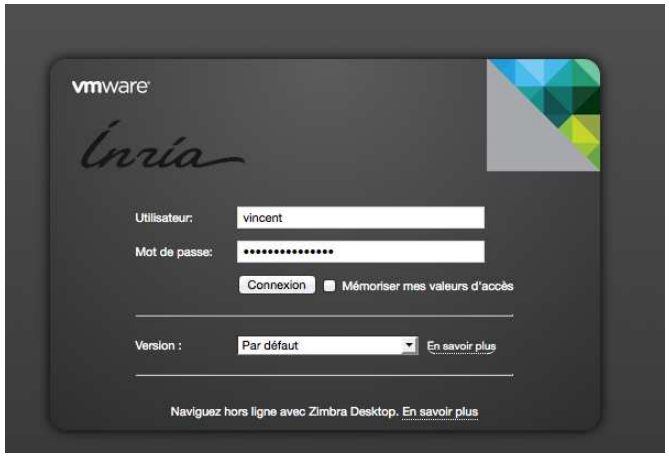
B. P. ext. [Du domaine math. au domaine du raisonnement et de la log.] Mécanisme réglant le fonctionnement de la pensée organisée et s'explicitant par des représentations analogues à celles des mathématiciens

À QUOI ÇA SERT ?





http://interstices.info/decoder-vivant



ALGORITHME

Un algorithme c'est ...

1. un moyen de communiquer à propos d'un problème/solution ;
2. une manière de résoudre un problème donné ;
3. une "formalisation" d'une méthode (qui sera prouvée et évaluée)
4. le premier pas (obligatoire) vers une implantation dans un langage de programmation

ALGORITHME

Un algorithme c'est ...

1. un moyen de communiquer à propos d'un problème/solution ;
2. une manière de résoudre un problème donné ;
3. une "formalisation" d'une méthode (qui sera prouvée et évaluée)
4. le premier pas (obligatoire) vers une implantation dans un langage de programmation

Quelques règles

1. Il y a de nombreuses manières d'écrire un algorithme
   Trouver son propre style ! ... Mais rester cohérent
2. un algorithme prends des entrées (input) et produit des sorties (output)
   Celles ci doivent être définies précisément
3. un algorithme peut utiliser d'autres algorithmes
   Approche "top-down" ... mais ces algorithmes doivent également être présentés

inspiré de Louis-Noël Pouchet

EXPRESSION D'UN ALGORITHME

Le langage algorithmique est une **convention** qui permet d'exprimer à un **lecteur**

1. l'idée de l'algorithme (principe, déroulement,...)
2. lui permettre de faire la preuve de celui-ci et de pouvoir analyser sa complexité
3. de pouvoir le traduire facilement dans un langage de programmation

Le langage algorithmique est donc plus ou moins proche d'un langage de programmation.
Exemple : tri bulle

EXPRESSION DES ALGORITHMES

TRI À BULLES : TD D'ALGORITHMIQUE

On souhaite trier un tableau d'éléments comparables. Le tableau est de taille n et les cases sont indicées de 1 à n.

Une itération ($i = 2$ à n) à chaque pas de laquelle si l'élément d'indice i est plus petit que l'élément d'indice $i - 1$ on échange l'élément d'indice i avec l'élément d'indice $i - 1$.

on répète l'itération précédente jusqu'à ce qu'il n'y ait plus d'échange

Finalement : les n éléments sont triés.

TRI À BULLES : OUVRAGE DE CORMEN ET AL.

TRI-BULLES(A)

1 **pour** $i \leftarrow 1$ **à** *longueur*$[A]$
2 **faire pour** $j \leftarrow$ *length*$[A]$ **decr jusqu'à** $i + 1$
3 **faire si** $A[j] < A[j-1]$
4 **alors** permuter $A[j] \leftrightarrow A[j-1]$

seule l'idée est donnée,

il n'y a pas de déclaration de variables,

il est implicite que A est un tableau,

on compare directement les éléments du tableau

TRI À BULLES : OUVRAGE DE WIRTH

```
procedure bubblesort;
    var i,j: index;   x: item;
begin for i := 2 to n do
        begin for j := n downto i do
                if a[j−1] .key > a[j] .key then
                begin x := a[j−1]; a[j−1] := a[j]; a[j] := x
                end
        end
end {bubblesort}
```

Program 2.4 Bubblesort

parti pris de décrire les algorithmes dans un langage de programmation ici Pascal

pas d'abstraction des opérateurs, variables définies implicitement

syntaxe et propriétés du langage de programmation.

TRI À BULLES : OUVRAGE DE BEAUQUIER ET AL.

```
procédure TRI-BULLE(t, i, j);
  pour k de i à j − 1 faire
    pour p de j − 1 à k pas −1 faire
      si c(p + 1) < c(p) alors ÉCHANGER(t, p + 1, p)
      {c(k) est la clé de l'élément t(k)}
    finpour
  finpour.
```

travaille sur les indices

procedure : paramètres d'appels

notion de clé

TRI À BULLES : OUVRAGE DE HAREL ET AL.

(1) do the following $N - 1$ times:
  (1.1) point to the first element;
  (1.2) do the following $N - 1$ times:
    (1.2.1) compare the element pointed to with the next element;
    (1.2.2) if the compared elements are in the wrong order, exchange them;
    (1.2.3) point to the next element.

met en avant la notion de pointeur

une seule variable exprimée : la taille du tableau

pas d'indice d'itération

TRI À BULLES : WIKIPEDIA FR - EN

En français

```
tri_à_bulles(Tableau T)
   pour i allant de taille de T - 1 à 1
       pour j allant de 0 à i - 1
           si T[j+1] < T[j]
               échanger(T[j+1], T[j])
```

En anglais

```
procedure bubbleSort( A : list of sortable items )
   n = length(A)
   repeat
     swapped = false
     for i = 1 to n-1 inclusive do
       /* if this pair is out of order */
       if A[i-1] > A[i] then
         /* swap them and remember something changed */
         swap( A[i-1], A[i] )
         swapped = true
       end if
     end for
   until not swapped
end procedure
```

Autres sites

https://openclassrooms.com/courses/le-tri-a-bulles
http://rosettacode.org/wiki/Sorting_algorithms/Bubble_sort
http://www.sorting-algorithms.com/bubble-sort

TRI À BULLES : OUVRAGE DE KNUTH

Algorithm B (*Bubble sort*). Records $R_1, \ldots, R_N$ are rearranged in place; after sorting is complete their keys will be in order, $K_1 \leq \cdots \leq K_N$.

B1. [Initialize BOUND.] Set BOUND $\leftarrow N$. (BOUND is the highest index for which the record is not known to be in its final position; thus we are indicating that nothing is known at this point.)

B2. [Loop on j.] Set $t \leftarrow 0$. Perform step B3 for $j = 1, 2, \ldots,$ BOUND $- 1$, and then go to step B4. (If BOUND $= 1$, this means go directly to B4.)

B3. [Compare/exchange $R_j : R_{j+1}$.] If $K_j > K_{j+1}$, interchange $R_j \leftrightarrow R_{j+1}$ and set $t \leftarrow j$.

B4. [Any exchanges?] If $t = 0$, terminate the algorithm. Otherwise set BOUND $\leftarrow t$ and return to step B2. ∎

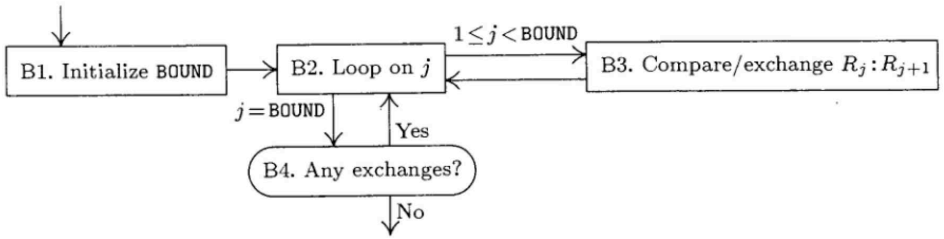


Fig. 15. Flow chart for bubble sorting.

représentation graphique

schéma itératif (steps), non modulaire

TRI BULLE : OUVRAGE DE LUC DAMAS

EXPRESSION D'UN ALGORITHME : SYNTHÈSE

Conclusion ?

EXPRESSION DES ALGORITHMES

LE CRÊPIER

Crêpier-Itératif (T)

```
for i =Taille (T) to 2 do
    k = 1 ;
    for j = 2 to i
        if T[j] > T[k]
            k = j
    Retourne-préfixe (T,k)
    Retourne-préfixe (T,n)
```

LE CRÊPIER

Crêpier-Itératif (T)

```
for i =Taille (T) to 2 do
    k = 1 ;
    for j = 2 to i
        if T[j] > T[k]
            k = j
    Retourne-préfixe (T,k)
    Retourne-préfixe (T,n)
```

Retourne-préfixe (T,i)

```
j = 1 while j < i − j + 1
    Échange (T,j,i − j + 1)
    j = j + 1
```

Échange (T,i,j)

```
x = T[i]
T[i] = T[j]
T[j] = x
```

LE CRÊPIER

Crêpier-Itératif (T)

Données: Un tableau T d'éléments comparables

Résultats: Les éléments rangés dans le même tableau par ordre croissant

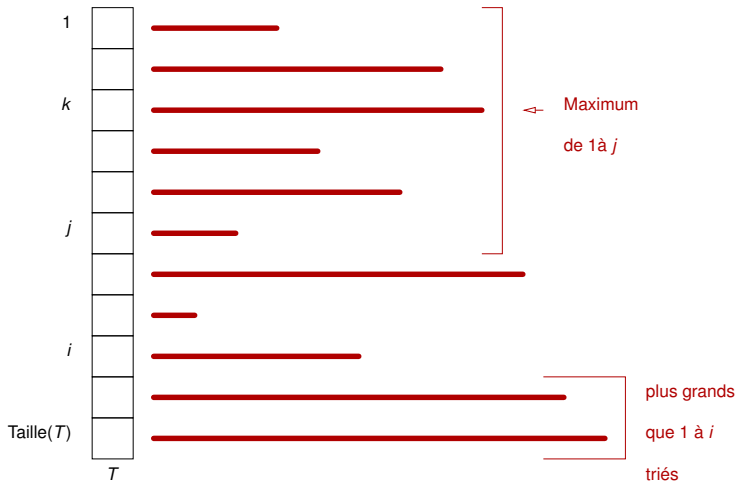
```
for i =Taille (T) to 2 do
    // Placer la bonne crêpe, la plus grande des i
       premières en position i
    k = 1 ; // indice de la plus grande crêpe
    for j = 2 to i
        // recherche de la plus grande crêpe
        if T[j] > T[k]
            k = j
    Retourne-préfixe (T,k) // positionnement de la plus
       grande crêpe au sommet de la pile
    Retourne-préfixe (T,i) // positionnement de la plus
       grande crêpe à sa position i
```

PREUVE DE L'ALGORITHME

Décrire l'état des variables de l'algorithme

Crêpier-Itératif (T)
Données: Un tableau T d'éléments comparables
Résultats: Les éléments rangés dans le même tableau par ordre croissant

```
for i = Taille(T) to 2 do
    // Placer la bonne crêpe, la plus grande
       des i premières en position i
    k = 1 ; // indice de la plus grande crêpe
    for j = 2 to i
        // recherche de la plus grande crêpe
        if T[j] > T[k]
            k = j
        ****************************************
    Retourne-préfixe (T,k)
    // positionnement de la plus grande
       crêpe au sommet de la pile
    Retourne-préfixe (T,i)
    // positionnement de la plus grande
       crêpe à sa position i
```



LE CRÊPIER (PREUVE)

Crêpier-Itératif (T)
Données: Un tableau T d'éléments comparables
Résultats: Les éléments rangés dans le même tableau par ordre croissant

```
for i =Taille (T) to 2 do
    // {Les éléments de T d'indice plus grand que i sont triés et plus grands que les
       éléments d'indice de 1 à i}
    // Placer la bonne crêpe, la plus grande des i premières en position i
    k = 1 ; // indice de la plus grande crêpe
    for j = 2 to i
        // {L'élément T[k] est plus grand que les éléments d'indice 1 à j − 1.}
        // recherche de la plus grande crêpe
        if T[j] > T[k]
            k = j
        // {L'élément T[k] est plus grand que les éléments d'indice 1 à j.}
    Retourne-préfixe (T,k) // positionnement de la plus grande crêpe au sommet de la pile
    Retourne-préfixe (T,i) // positionnement de la plus grande crêpe à sa position i
    // {Les éléments de T d'indice plus grand que i − 1 sont triés et plus grands que les
       éléments d'indice de 1 à i − 1}
```

- Les éléments **{ ..... }** sont appelées des **assertions** qui portent sur les valeurs des variables et ont une valeur logique vrai/faux.
- Dans notre exemple on montre que si l'assertion est vraie en début d'itération, elle sera vérifiée en fin d'itération (invariant d'itération).
- Il reste à montrer que l'assertion est vérifiée au début de l'itération.
- En fin d'itération, l'assertion vraie prouve la correction du programme.

LE CRÊPIER (APPROCHE RÉCURSIVE)

Crêpier-Récursif (T,n)
Données: Un tableau T d'éléments comparables de taille n
Résultats: Les éléments rangés dans le même tableau par ordre croissant

if $n \neq 1$
  $k =$**Indice-du-max** (T,n)
  `// recherche de l'indice de la valeur maximale du tableau de 1 à n`
  Retourne-préfixe (T,k)
  `// positionnement de la plus grande crêpe au sommet de la pile`
  Retourne-préfixe (T,n)
  `// positionnement de la plus grande crêpe à la position n`
  Crêpier-Récursif (T,$n-1$)
  `// appel récursif pour trier le tableau de 1 à n-1`

Expression plus élégante, plus facile à prouver par induction (par récurrence)

LE CRÊPIER : PETITE HISTOIRE

Quel est le nombre minimal $f(n)$ de retournements pour trier un tas arbitraire de n crêpes ?

Références

- Dweighter, Harry ; Garey, Michael R. ; Johnson, David S. ; Lin, Shen (1977), *Solutions of Elementary Problem E2569*, Amer. Math. Monthly 84 : 296
  Pseudonyme de Jacob E. Goodman
- Gates W.H. ; Papadimitriou, C.H. *Bounds for sorting by prefix reversal*. Discrete Math. 27 (1979), 47–57.
- Bulteau, L. ; Fertin, G. ; Rusu, I. *Pancake Flipping is Hard*. 37th International Symposium on Mathematical Foundations of Computer Science, Aug 2012, Bratislava, Slovakia. 7467, pp.247-258, 2012, Lecture Notes in Computer Science, Springer Verlag. Hal version

Valeurs de $f(n)$

- $f(n)$ est connu pour $n \leqslant 19$.
- Meilleur encadrement
$$\frac{15}{14}n \leqslant f(n) \leqslant \frac{18}{11}n + \mathcal{O}(1).$$
- le problème *MIN-SBPR* est $\mathcal{NP}$-dur
- Online Encyclopedia of Integer Sequences A058986

EXPRESSION DES ALGORITHMES

PRINCIPES POUR LA CONCEPTION D'ALGORITHMES

1. Déterminer les entrées et les sorties (spécification)
2. Trouver la structure de donnée adaptée pour ce problème
   - Ne pas hésiter à pré-traiter les entrées pour les mettre sous une forme adéquate
3. Essayer de réduire le problème à un problème connu
   - Tri, recherche, chemin dans un graphe,...
   - Vérifier si une solution existe déjà (livres, internet,...)
4. Décider de la manière d'approcher le problème : itératif/récursif/mix
   - Dépend de la manière de penser, de la facilité à trouver des invariants, de la manière de décomposer le problème en sous problèmes,...
5. **Écrire** l'algorithme
6. Faire tourner l'algorithme sur des exemples simples et des exemples "limite".
7. Donner les invariants de l'algorithme et faire la preuve
8. Évaluer la complexité de l'algorithme

PRINCIPES POUR L'EXPRESSION DES ALGORITHMES

Ma position personnelle (que certains de mes collègues ne partagent pas) :

- Un algorithme est toujours accompagné de schémas et de texte
  représentation de l'état des variables,
- utiliser les conventions
  (i,j sont des indices, x un réel, n un entier par exemple une taille de tableau,...).
- utiliser des identificateurs explicites, une variable un usage
  noms de variables, de fonctions,
- la forme est importante
  l'indentation doit permettre de comprendre la structure de l'algorithme
- mettre des commentaires dans le code et accompagner l'algorithme par une explication en français
- ne mettre que l'information importante : minimiser l'encre
- être cohérent
  utiliser le même formalisme pour toutes les présentations d'algorithmes

de manière plus générale

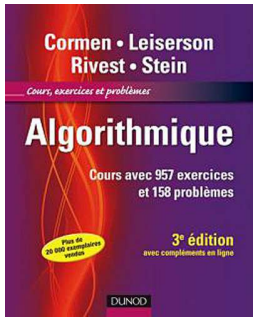
favoriser tout ce qui aide à la compréhension et éliminer ce qui peut perturber la lecture

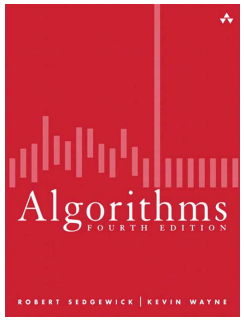
EXPRESSION DES ALGORITHMES

BIBLIOGRAPHIE : OUVRAGES DE RÉFÉRENCE

- **Algorithmique** *Thomas Cormen, Charles Leiserson, Ronald Rivest, and Clifford Stein.*. Dunod, 2010.
  Ouvrage de référence internationale en algorithmique. Très pédagogique il peut être utilisé en autoformation, lorsque les bases sont acquises. Couvre l'ensemble du domaine.
- **Algorithms** *Robert Sedgewick and Kevin Wayne*. Addison Wesley, 2011.
  Une approche thématique permettant de reprendre les différents et paradigmes de l'algorithmique. La présentation est soignée, les détails des implémentations en Java sont très utiles.

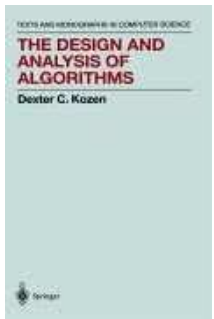
  Des versions précédentes en français : *Robert Sedgewick* Algorithmes en C ou Algorithmes en Java chez Dunod

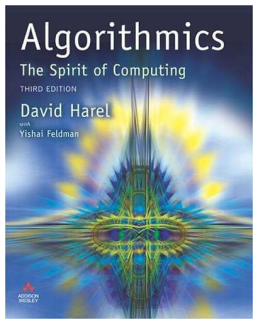


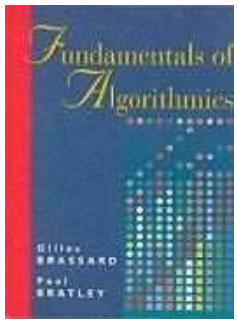


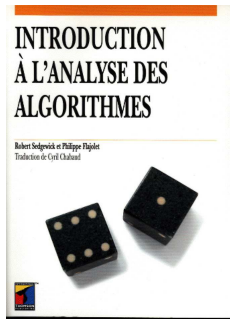
BIBLIOGRAPHIE : OUVRAGES PLUS AVANCÉS

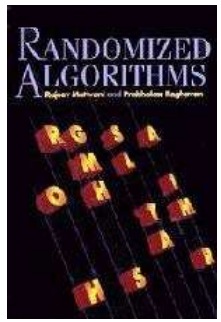
- **The Design and Analysis of Algorithms** *Dexter C. Kozen* Springer, 1991.
  Excellent ouvrage pour de l'algorithmique avancée. Présenté sous forme de séquence de lectures "indépendantes" il va directement à l'essentiel. Les principes algorithmiques sont ainsi mis en valeur.
- **Algorithmics : The Spirit of Computing** *David Harel and Yishai Feldman* Addison Wesley, 2004.
  Orienté méthodologie, cet ouvrage propose une vue transversale en abordant successivement, méthode et analyse, limitations et robustesse, extensibilité... intéressant pour le recul pris.
- **Introduction à l'analyse des algorithmes** *Robert Sedgewick and Philippe Flajolet* Addison Wesley 1995
  Ouvrage théorique sur l'analyse de la complexité des algorithmes
- **Fundamental of Algorithms** *Gilles Brassard and Paul Bratley* Prentice Hall 1996
- **Randomized Algorithms**, *R. Motwani and P. Raghavan*, Cambridge University Press, 1995.





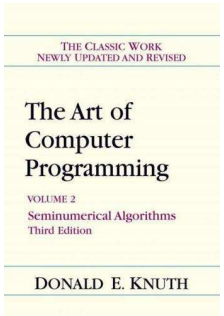


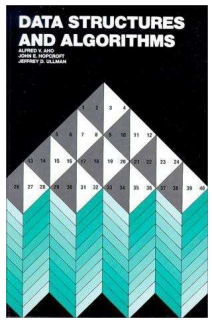


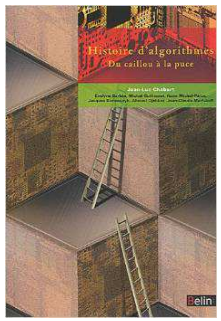


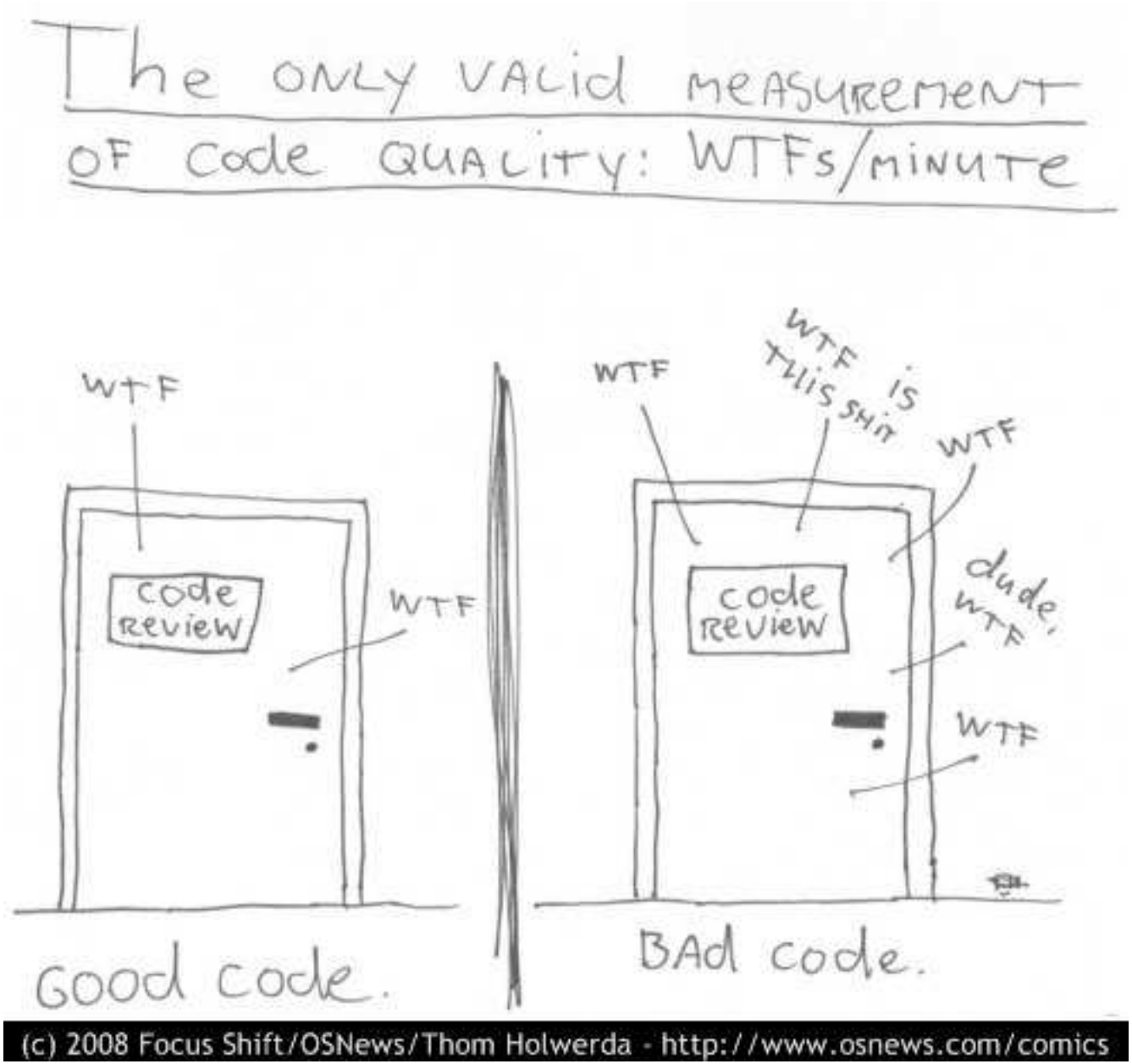
BIBLIOGRAPHIE : OUVRAGES HISTORIQUES DE RÉFÉRENCE

- **The Art of Computer Programming, Vol 1-4** *Donald E. Knuth*, Addison-Wesley, 1998.
  Ouvrage historique et encore d'actualité pour la conception et l'analyse d'algorithmes
- **Data Structures and Algorithms** *Alfred V. Aho, J.E. Hopcroft, et Jeffrey D. Ullman* Addison Wesley 1983
- *Jean-Luc Chabert et al.* **Histoires d'algorithmes** Belin 2010
  Une histoire des algorithmes avec un point de vue calcul et calcul numérique








The ONLY VALid MEASUREMENT OF Code QUALITY: WTFs/MINUTE
WTF
code review
WTF
Good code.
WTF
WTF is THIS SHIT
WTF
code review
dude, WTF
WTF
BAd code.
(c) 2008 Focus Shift/OSNews/Thom Holwerda - http://www.osnews.com/comics