ALG1 : qu'est-ce qu'un algorithme ?

loig.jezequel@univ-nantes.fr

Définitions informelles

Algorithme — source : Cormen et al.

Procédure de calcul bien définie qui prend en entrée un ensemble de valeurs et qui donne en sortie un ensemble de valeurs.

Algorithme — source : Cormen et al.

Séquence d'étapes de calcul qui transforment une entrée en sortie.

Algorithme — source : Académie française

Méthode de calcul qui indique la démarche à suivre pour résoudre une série de problèmes équivalents en appliquant dans un ordre précis une suite finie de règles.

Un exemple d'algorithme : l'addition

Problème

entrée

Étant donnés $\boxed{x \text{ et } y}$ deux nombres entiers positifs exprimés en base 10, calculer la valeur de la somme $\boxed{x+y}$.

sortie

Principe de l'algorithme

Additionner les chiffres deux à deux, de droite à gauche, en propageant les éventuelles retenues.

Une instance du problème : $x = 234$ et $y = 56$

$$
\begin{array}{cccc}
 & 2 & 3 & 4 \\
+ & & 5 & 6 \\
\hline
\end{array}
$$

Un exemple d'algorithme : l'addition

entrée

Problème

Étant donnés $\boxed{x \text{ et } y}$ deux nombres entiers positifs exprimés en base 10, calculer la valeur de la somme $\boxed{x + y}$.

sortie

Principe de l'algorithme

Additionner les chiffres deux à deux, de droite à gauche, en propageant les éventuelles retenues.

Une instance du problème : $x = 234$ et $y = 56$

$$
\begin{array}{cccc}
 & 2 & {}^{1}3 & 4 \\
+ & & 5 & 6 \\
\hline
 & & & 0
\end{array}
$$

Un exemple d'algorithme : l'addition

Problème

entrée

Étant donnés $\boxed{x \text{ et } y}$ deux nombres entiers positifs exprimés en base 10, calculer la valeur de la somme $\boxed{x + y}$.

sortie

Principe de l'algorithme

Additionner les chiffres deux à deux, de droite à gauche, en propageant les éventuelles retenues.

Une instance du problème : $x = 234$ et $y = 56$

$$
\begin{array}{cccc}
 & 2 & {}^{1}3 & 4 \\
+ & & 5 & 6 \\
\hline
 & & 9 & 0
\end{array}
$$

Un exemple d'algorithme : l'addition

Problème

entrée

Étant donnés $\boxed{x \text{ et } y}$ deux nombres entiers positifs exprimés en base 10, calculer la valeur de la somme $\boxed{x + y}$.

sortie

Principe de l'algorithme

Additionner les chiffres deux à deux, de droite à gauche, en propageant les éventuelles retenues.

Une instance du problème : $x = 234$ et $y = 56$

$$
\begin{array}{cccc}
 & 2 & {}^{1}3 & 4 \\
+ & & 5 & 6 \\
\hline
 & 2 & 9 & 0
\end{array}
$$

Un exemple d'algorithme : l'addition, suite

Notations

On note $x = x_n x_{n-1} \dots x_1 x_0$ et $y = y_m y_{m-1} \dots y_1 y_0$ et on suppose (sans perte de généralité) que $n \geq m$.

variables

Algorithme

boucles

Soit $\boxed{i = 0}$ et soit $\boxed{r_0 = 0}$.

Tant que $i \leq m$ poser $z_i = (x_i + y_i + r_i)_0$ et $r_{i+1} = (x_i + y_i + r_i)_1$ puis augmenter i de 1.

Tant que $i \leq n$ poser $z_i = (x_i + r_i)_0$ et $r_{i+1} = (x_i + r_i)_1$ puis augmenter i de 1.

Si $r_{n+1} = 0$, retourner le nombre $z_n z_{n-1} \dots z_1 z_0$ et sinon (si $r_{n+1} > 0$), retourner le nombre $r_{n+1} z_n z_{n-1} \dots z_1 z_0$.

conditionnelle

Un exemple d'algorithme : le PGCD

Problème

Étant donnés deux entiers positifs a et b, calculer le plus grand diviseur commun de a et b.

Algorithme

Tant que $b \neq 0$, poser $b = a \mod b$ et $a = b$. Retourner a.

Une instance du problème : $a = 594$ et $b = 459$

Un exemple d'algorithme : le PGCD

Problème

Étant donnés deux entiers positifs a et b, calculer le plus grand diviseur commun de a et b.

Algorithme

Tant que $b \neq 0$, poser $b = a \mod b$ et $a = b$. Retourner a.

Une instance du problème : $a = 594$ et $b = 459$

1. $b = 594 \mod 459 = 135$, $a = 459$

Un exemple d'algorithme : le PGCD

Problème

Étant donnés deux entiers positifs a et b, calculer le plus grand diviseur commun de a et b.

Algorithme

Tant que $b \neq 0$, poser $b = a \mod b$ et $a = b$. Retourner a.

Une instance du problème : $a = 594$ et $b = 459$

1. $b = 594 \mod 459 = 135$, $a = 459$
2. $b = 459 \mod 135 = 54$, $a = 135$

Un exemple d'algorithme : le PGCD

Problème

Étant donnés deux entiers positifs a et b, calculer le plus grand diviseur commun de a et b.

Algorithme

Tant que $b \neq 0$, poser $b = a \mod b$ et $a = b$. Retourner a.

Une instance du problème : $a = 594$ et $b = 459$

1. $b = 594 \mod 459 = 135$, $a = 459$
2. $b = 459 \mod 135 = 54$, $a = 135$
3. $b = 135 \mod 54 = 27$, $a = 54$

Un exemple d'algorithme : le PGCD

Problème

Étant donnés deux entiers positifs a et b, calculer le plus grand diviseur commun de a et b.

Algorithme

Tant que $b \neq 0$, poser $b = a \mod b$ et $a = b$. Retourner a.

Une instance du problème : $a = 594$ et $b = 459$

1. $b = 594 \mod 459 = 135$, $a = 459$
2. $b = 459 \mod 135 = 54$, $a = 135$
3. $b = 135 \mod 54 = 27$, $a = 54$
4. $b = 54 \mod 27 = 0$, $a = 27$

Un exemple d'algorithme : le PGCD

Problème

Étant donnés deux entiers positifs a et b, calculer le plus grand diviseur commun de a et b.

Algorithme

Tant que $b \neq 0$, poser $b = a \mod b$ et $a = b$. Retourner a.

Une instance du problème : $a = 594$ et $b = 459$

1. $b = 594 \mod 459 = 135$, $a = 459$
2. $b = 459 \mod 135 = 54$, $a = 135$
3. $b = 135 \mod 54 = 27$, $a = 54$
4. $b = 54 \mod 27 = 0$, $a = 27$
5. $b = 0$, on retourne 27

Quelques exemples de problèmes d'algorithmique

Biologie

Beaucoup de problèmes autour de l'ADN et notamment le problème d'alignement de séquences ADN.

Internet

Problèmes de routage, de recherche et classement de données, de compression.

Industrie

Problèmes de logistique et planification avec optimisation des ressources.

Remarque

Les algorithmes pour résoudre ces problèmes sont trop complexes pour les traiter dans ce cours, mais les techniques qu'on abordera sont à la base de ces algorithmes.

Pourquoi étudier l'algorithmique ?

Évolution des performances des ordinateurs

Tous les deux ans environ le nombre de transistors dans les micro-processeurs – et donc la puissance de calcul – double (loi de Moore), la mémoire augmente aussi rapidement.

Mais. . .

- exécuter un algorithme a quand même toujours un coût en énergie et en temps,
- et certains problèmes sont intrinsèquement trop complexes pour être résolus par force brute.

Donc savoir concevoir des algorithmes performants est important.

Algorithmes performants : un exemple[1]

Cas A

Rangement de n éléments par tri par insertion bien programmé : $2n^2$ opérations, sur un ordinateur puissant : 1 milliard d'opérations par seconde.

Cas B

Rangement de n éléments par tri fusion mal programmé : $50n\log_2(n)$ opérations, sur un ordinateur peu puissant : 10 millions d'opérations par seconde.

Si $n = 10^6$

Dans le cas A l'exécution nécessite $2.(10^6)^2$ opérations et dure donc $2.(10^6)^2/10^9 = 2000$ secondes alors que dans le cas B l'exécution nécessite $50.10^6.\log_2(10^6)$ opérations et dure donc $50.10^6.\log_2(10^6)/10^7 = 100$ secondes.

[1] Cormen et. al, les tris seront abordés lors d'un prochain cours

Pourquoi étudier l'algorithmique ? suite et fin

Même avec une puissance de calcul et une mémoire infinies

Terminaison Un algorithme mal conçu peut boucler sans arrêt, et donc ne jamais donner de réponses.

Validité Un algorithme mal conçu peut donner des réponses incorrectes.

Implantation Un algorithme mal compris peut être mal codé, le code exécuté par l'ordinateur ne correspondra alors pas à ce qui est attendu (terminaison et validité ne sont plus assurées).