Algorithmique III. L2 Informatique I41.

TD 1. Généralités [1]

EXERCICE 1. Calculer la somme des valeurs obtenues en lançant 10 fois de suite un dé à 6 faces constitue-t-il un algorithme ? Justifiez votre réponse.

Solution. Dans le monde physique, les conditions nécessaires à ce que l'expérience soit reproductible — c'est-à-dire telles qu'en relançant les dés 10 fois de suite, on obtienne exactement la même répartition — sont impossibles à satisfaire, il ne s'agit donc pas d'un algorithme.

NB. Les tirages dits "aléatoires" dans les langages de programmation sont en fait des générateurs *pseudo-aléatoires*, la suite des valeurs fournies par ces fonctions "`random`" est parfaitement déterminée.

EXERCICE 2. Un *organigramme* est une description schématique d'un algorithme. Les instructions sont rangées dans des boites reliées par des flèches indiquant leur chronologie.

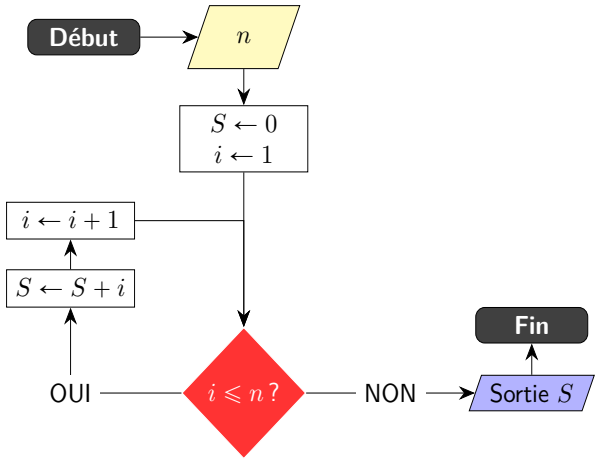

FIGURE 1. Organigramme du programme calculant la somme des n premiers entiers.

1. Version du 16 février 2023, 07 : 09

Ces boites ont des formes différentes selon la nature des instructions qu'elles contiennent, ovales pour le début et la fin, trapézoïdales pour les entrées/sorties (sur fond jaune pour les entrées et bleu pour les sorties ici), losanges pour les tests, et rectangulaires pour les autres instructions.

L'algorithme calculant la somme des n premiers entiers naturels non-nuls est présenté sous forme d'organigramme en figure 1. Écrivez sous forme d'*organigramme* l'algorithme de résolution d'une équation polynomiale $aX^2 + bX + c = 0$ dans le corps des réels $\mathbb{R}$ (les coefficients du polynôme ne sont pas nécessairement non-nuls).

Solution. Si les coefficients du polynôme peuvent être nuls, le degré de l'équation peut être $\leqslant 2$, ce qui complique un peu l'analyse et demande une étude de cas. L'organigramme est présenté en figure 2.

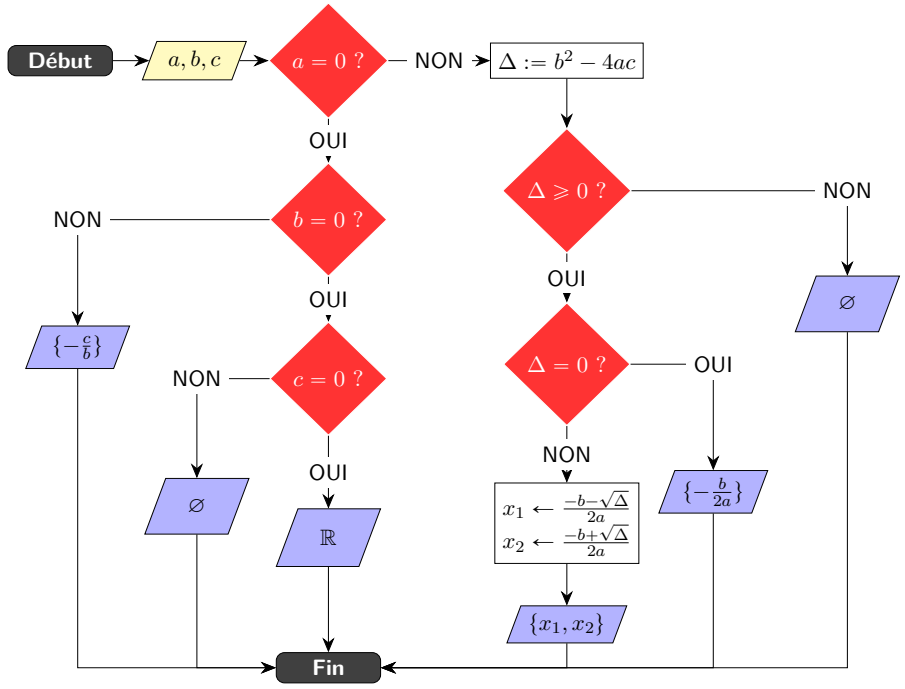

FIGURE 2. Organigramme de l'algorithme de calcul des solutions d'une équation polynomiale de degré $\leqslant 2$.

EXERCICE 3. Proposez un schéma d'encodage pour décrire les positions des pièces sur un échiquier (voir la figure 3). On rappelle qu'il y a 6 types de pièces : pions, tours, fous, cavaliers, rois et reines.

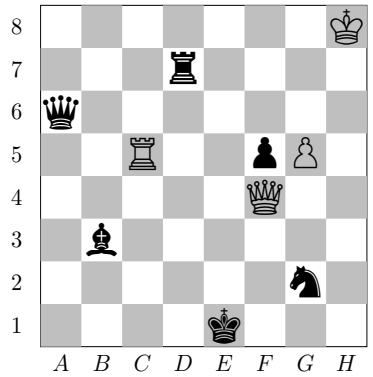

Figure 3. Une position sur un échiquier.

Solution. Il y a 6 pièces de natures différentes, pion, tour, cavalier, fou, roi, reine pour chacune des 2 couleurs, noir ou blanc, soit 13 combinaisons différentes sur une cas de l'échiquier en intégrant la case vide. On peut, par exemple, encoder l'échiquier par une matrice 8×8 dont les éléments sont des entiers codés sur 4 bits :

- 0000 : case vide
- 0001 : pion noir.
- 0010 : tour noire.
- 0011 : cavalier noir.
- 0100 : fou noir.
- 0101 : reine noire.
- 0110 : roi noir.
- 0111 : inutilisé.
- 1000 : inutilisé
- 1001 : pion blanc.
- 1010 : tour blanche.
- 1011 : cavalier blanc.
- 1100 : fou blanc.
- 1101 : reine blanche.
- 1110 : roi blanc.
- 1111 : inutilisé.

EXERCICE 4. Proposez un schéma d'encodage pour une carte d'identité. Ne tenez compte que des informations visuelles qui y sont inscrites.

Solution. Une carte d'identité contient les informations suivantes :

(1) Les noms, prénoms, taille, genre, nationalité, date et lieu de naissance du titulaire.

(2) Une photographie d'identité noir et blanc en 254dpi au format 28×38mm avec 32 nuances de gris.

(3) Le numéro du document, la date de délivrance et d'expiration.

(4) L'adresse du titulaire.

On peut supposer que :

(1) Les champs textuels, noms, prénoms, lieu de naissance et adresse, sont codés sur la base d'un alphabet Σ de 64 caractères, leurs longueurs sont de 32 caractères sauf pour l'adresse sur 128 caractères.

(2) La taille en cm peut être codée sur 8 bits (la différence de taille entre la personne la plus grande au monde et la plus petite n'excède pas 200cm), le genre sur 1 bit, la nationalité sur 8 bits (il n'y a que 197 états reconnus par l'ONU).

(3) Les différentes dates peuvent être codées sur 16 bits : 8 bits pour la position du jour dans l'année (de 1 à 366, sachant que l'on peut recalculer le mois et le jour du mois à partir de cette information) et 8 bits pour l'année (à partir de 1900 par exemple).

(4) Le numéro du document est constitué de 9 chiffres ou lettres, sur la base du même alphabet Σ que les noms.

(5) Une densité de 254dpi (*dot per inch*, i.e. point par pouce) correspond à 10 pixels par millimètre puisqu'un pouce mesure 2.54mm, soit 100 pixels au mm^2. La photographie ayant une surface de $1\,064\text{mm}^2$ elle contient donc 106 400 pixels. Il faut 5 bits pour coder les 32 nuances d'un pixel, on a donc besoin de 532 000 bits, soit 66 500 octets.

On note $\mathscr{B} := \{0, 1\}$, $O := \mathscr{B}^8$ et $R := \mathscr{B}^{16}$. Les informations d'une carte d'identité sont donc codées par un élément du produit cartésien (dans

l'ordre où elles ont été présentées) :

$$\underbrace{\Sigma^{32}}_{\text{nom}} \times \underbrace{\Sigma^{32}}_{\text{prénom}} \times \underbrace{O}_{\text{taille}} \times \underbrace{\mathscr{B}}_{\text{genre}} \times \underbrace{O}_{\text{nation}} \times \underbrace{R}_{\text{naiss.}} \times$$
$$\underbrace{\Sigma^{32}}_{\text{lieu}} \times \underbrace{O^{65\,500}}_{\text{photo}} \times \underbrace{\Sigma^{9}}_{\text{numéro}} \times \underbrace{R}_{\text{déliv.}} \times \underbrace{R}_{\text{exp.}} \times \underbrace{\Sigma^{128}}_{\text{addr.}}$$

EXERCICE 5. Proposez un schéma d'encodage pour un labyrinthe constitué uniquement de segments horizontaux et verticaux.

Solution. On pourrait, par exemple, coder un labyrinthe comme une matrice de carrés dont chacun des 4 côtés peut être ouvert (0) ou fermé (1). Chaque carré sera ainsi codé par un nombre hexadécimal pour les 16 combinaisons possibles d'ouvertures, les bits suivraient l'ordre NORD, EST, SUD, OUEST des faces du carré. Par exemple :

$1101 \equiv D$ $\qquad$ $0100 \equiv 8$

Ainsi, ce labyrinthe constitué de 3×4 carrés :

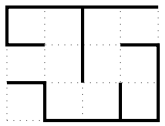

FIGURE 4. Labyrinthe.

est codé par la matrice 3×4 :

$$\begin{pmatrix} B & C & 9 & A \\ A & 4 & 1 & C \\ C & 3 & 6 & 7 \end{pmatrix}$$

Notons que l'on pourrait trouver un schéma d'encodage deux fois plus économique en espace puisqu'ici chaque segment intérieur du labyrinthe est codé deux fois. Plus généralement un labyrinthe est codé par une matrice $n \times m$ d'éléments de

$$H := \{n \in \mathbb{N} \mid 0 \leqslant n \leqslant 16\}.$$

Il faut là aussi remarquer que la notation hexadécimale est elle-même un encodage des entiers de H.

EXERCICE 6. 1. Écrivez un algorithme qui décide si un mot $u = u_1 u_2 \dots u_n$ est un palindrome ou non en le parcourant "simultanément" de la gauche vers la droite et de la droite vers la gauche à l'aide de deux indices i et j initialisés respectivement à 1 et n.

2. Montrez que votre algorithme s'arrête.

3. Justifiez votre algorithme.

4. Combien d'instructions sont exécutées dans le meilleur des cas et dans le pire des cas ?

Solution. 1. On déplace les deux indices i et j en sens opposés tant qu'ils ne se sont pas rencontrés et tant que les symboles à ces positions opposées sont identiques :

```
ALGO EstPalindrome(u):booléen
DONNEES
 · u: mot
VARIABLES
 · i,j: entiers
DEBUT
 · i ← 1
 · j ← |u|
 · TQ ((i < j) ET (u[i] = u[j])) FAIRE
 ·   · i ← i + 1
 ·   · j ← j - 1
 · FTQ
 · RENVOYER (i >= j)
FIN
```

2. D'après la condition de boucle, l'algorithme s'arrête s'il existe un couple de symboles opposés distincts et dans le cas contraire, il faut étudier l'autre condition $i < j$ de la conjonction, i.e. $j - i > 0$. Considérons la suite $u_i := j - i$ dont les valeurs sont fixées par le couple (i, j) avant chaque entrée dans la boucle `TQ`. Après l'initialisation et avant la première entrée dans la boucle, on a $u_1 = n-1$ et chaque passage dans la boucle décremente

i et incrémente j on en déduit que $u_{i+1} = u_i - 2$ et $u_i = n - 2i + 1$. On a donc

$$(u_i \leqslant 0) \;\Leftrightarrow\; i \geqslant \frac{n+1}{2}.$$

Si la longueur n du mot est impaire, disons $n = 2k+1$, alors on sort de la boucle pour $i = j = k+1$. Si $n = 2k$, on sort de la boucle pour $i = k+1$ et $j = k$.

3. Tout d'abord, les termes $u[i]$ et $u[j]$ qui sont comparés sont bien ceux qui doivent l'être puisque i et j sont initialisés aux extrémités opposées du mot à analyser et sont respectivement incrémentés et décrémentés d'une position. Si l'on sort de la boucle et que $i < j$ alors nécessairement $u[i] \neq u[j]$, le mot n'est donc pas un palindrome et l'algorithme renvoie bien *faux*. Dans le cas contraire, notons qu'avant chaque entrée dans la boucle, on a l'assertion suivante :

$$\forall r \in [\![1, i-1]\!] \quad u[r] = u[n-r+1]. \tag{1}$$

Nous avons vu qu'en sortant de la boucle on a $i = j = k+1$ si le mot est de longueur impaire, on a donc comparé toutes les paires précédentes, cette dernière étant inutile puisqu'elle correspond au symbole au milieu du mot u. Dans le cas pair, toutes les paires ont été comparées positivement, donc dans les deux cas la proposition $i \geqslant j$ est vraie si et seulement si le mot u est un palidrome.

4. Dans le meilleur des cas, la première comparaison échoue et seuls l'initialisation et le premier test auront été exécutés. Dans le pire des cas, le mot est un palindrome et il y aura eu $\lceil \frac{n}{2} \rceil + 1$ tests.

EXERCICE 7. † On rappelle la deuxième mouture de l'algorithme du cours qui décide si un mot $u = u_1 u_2 \dots u_n$ est un palindrome :

(1) On initialise une variable i à 1.
(2) Tant que $(i \leqslant \lfloor \frac{n}{2} \rfloor$ et $u_i = u_{n-i+1})$, incrémenter i.
(3) Si $i > \lfloor \frac{n}{2} \rfloor$ alors u est un palindrome, sinon ce n'est pas un palindrome.

On se propose d'évaluer le nombre *moyen* de tests effectués par cet algorithme, c'est-à-dire le nombre de fois où l'algorithme va évaluer l'expression logique (on néglige le test final après la boucle qui permet de conclure) :

$$\left(i \leqslant \left\lfloor \frac{n}{2} \right\rfloor\right) \quad \wedge \quad (u_i = u_{n-i+1}). \tag{2}$$

On veut calculer cette moyenne en fonction de la longueur n des mots. La motivation initiale pour cet algorithme était évidemment liée à la langue française, mais dans ce cas l'analyse en moyenne nécessiterait une étude exhaustive des mots de notre langue pour chaque longueur n possible.

Pour simplifier le problème et permettre une analyse plus générale, on suppose que l'on dispose d'un alphabet A de cardinal q, que l'espace des instances de l'algorithme est le langage A^n et que toutes les instances sont équiprobables. Notons que cette hypothèse d'équiprobabilité est très éloignée de la réalité, un mot de longueur 3 de la langue française qui commence par qu est certainement suivi par l'une des voyelles e, i ou o.

L'expérience aléatoire consiste donc à tirer un mot u de A^n "au hasard", l'espace d'échantillonnage Ω est alors le langage A^n. On note $p(u)$ le nombre de fois où l'expression (2) a été évaluée lors de l'exécution de l'algorithme pour l'instance u. Pour $k \in [\![1, \lfloor \frac{n}{2} \rfloor + 1]\!]$, on définit l'évènement

$$\Omega_k := \{u \in \Omega \mid p(u) = k\}.$$

(1) On supposera pour simplifier les expressions que v désigne le mot miroir de u. Justifiez que

$$\text{Prob}\left[\Omega_k\right] = \begin{cases} \text{Prob}\left[u_k \neq v_k\right] \times \displaystyle\prod_{i=1}^{k-1} \text{Prob}\left[u_i = v_i\right] & \text{si } k \in [\![1, \lfloor \frac{n}{2} \rfloor]\!], \\ \displaystyle\prod_{i=1}^{\lfloor \frac{n}{2} \rfloor} \text{Prob}\left[u_i = v_i\right] & \text{si } k = \lfloor \frac{n}{2} \rfloor + 1. \end{cases} \tag{Π}$$

(2) Montrez que

$$\forall i \in [1, \lfloor \frac{n}{2} \rfloor], \quad \text{Prob}\left[u_i = v_i\right] = \frac{1}{q} \text{ et } \text{Prob}\left[u_i \neq v_i\right] = 1 - \frac{1}{q}$$

(3) Montrez que les évènements Ω_k sont incompatibles.

(4) Montrez qu'aucun des ensembles Ω_k n'est vide.

(5) Montrez que la réunion des ensembles Ω_k est l'ensemble $\Omega := A^n$.

(6) En déduire que le nombre moyen de tests pour traiter un mot de longueur n est égal à

$$(\lfloor\frac{n}{2}\rfloor+1)\left(\frac{1}{q}\right)^{\lfloor\frac{n}{2}\rfloor}+(q-1)\underbrace{\sum_{k=1}^{\lfloor\frac{n}{2}\rfloor}k\left(\frac{1}{q}\right)^{k}}_{S}. \tag{3}$$

(7) Calculez l'expression (3). Indication : cf. chapitre de combinatoire pour calculer la somme S.

(8) Quelle est le nombre moyen de tests effectués, asymptotiquement sur la longueur des mots ?

Solution. (1) Par définition, $\mathrm{Prob}\,[\Omega_k]$ est la probabilité que la condition (2) soit évaluée k fois. Il ne faut pas perdre de vue que la dernière évaluation fait sortir de la boucle, la condition n'est donc satisfaite que $k-1$ fois. Que le mot soit de longueur paire ou impaire, c'est un palindrome si et seulement si $k=\lfloor\frac{n}{2}\rfloor+1$. Si ce n'est pas un palindrome, i.e. $1\leqslant k\leqslant\lfloor\frac{n}{2}\rfloor$, on a $u_i=u_{n-i+1}$ pour tout $i\in[\![1,k-1]\!]$ et $u_k\neq u_{n-k+1}$. Les contraintes sur ces k lettres sont indépendantes, ce qui justifie le premier produit dans l'expression (Π). Si ce n'est pas un palindrome, le raisonnement est le même mais sur l'intégralité des paires de symboles comparés.

(2) Le mot étant tiré au hasard, pour deux indices distincts i et j de l'intervalle $[1,\lfloor\frac{n}{2}\rfloor]$ la probabilité $\mathrm{Prob}[u_i=u_j]=\frac{1}{q}$. En effet, parmi les q^2 couples (a,b) de lettres possibles de A, il y a q couples (a,a). Les évènements $u_i=v_i$ et $u_i\neq v_i$ étant complémentaires, on a $\mathrm{Prob}[u_i\neq u_j]=1-\mathrm{Prob}[u_i=u_j]$.

(3) Soit k et ℓ deux entiers de l'intervalle $[\![1,\lfloor\frac{n}{2}\rfloor+1]\!]$ avec $k<\ell$. Si $x\in\Omega_k\cap\Omega_\ell$, alors $x_k\neq x_{n-k+1}$ car $x\in\Omega_k$ et $x_k=x_{n-k+1}$ car $k<\ell$ et $x\in\Omega_\ell$ ce qui est contradictoire. Les ensembles Ω_k sont donc deux-à-deux disjoints.

(4) Supposons que $q\geqslant 2$ et $A:=\{a_1,a_2,\ldots,a_q\}$. Pour tout $k\in[\![1,\lfloor\frac{n}{2}\rfloor]\!]$, il est aisé de construire un mot $u\in\Omega_k$, par exemple $a_1^{k-1}a_2a_1^{n-k}$ et pour $k=\lfloor\frac{n}{2}\rfloor+1$ le palindrome a_1^n.

(5) Si $u\in\Omega_k$, il faut noter qu'à la sortie de la boucle principale, la variable i a pour valeur k. Ainsi, avec les notations introduites, et pour un mot u donné de longueur n, l'algorithme (dont la justesse a été démontrée en cours) détermine la valeur de k telle que $u\in\Omega_k$, ce qui fournit le résultat demandé.

(6) Les 3 questions précédentes montrent que $(\Omega_k)_{k\in[\![1,\lfloor\frac{n}{2}\rfloor+1]\!]}$ est une partition de l'ensemble Ω des mots de longueur n. L'analyse d'un mot de Ω_k nécessitant k tests, le nombre moyen de tests, que l'on notera $\overline{T}$, est donc égal à

$$\overline{T}:=\sum_{k=1}^{\lfloor\frac{n}{2}\rfloor+1}k\,\mathrm{Prob}[\Omega_k]. \tag{4}$$

En reprenant les résultats des deux premières questions, on obtient tout d'abord

$$\mathrm{Prob}\,[\Omega_k]=\begin{cases}(q-1)\times\left(\frac{1}{q}\right)^k & \text{si } k\in[\![1,\lfloor\frac{n}{2}\rfloor]\!],\\ \left(\frac{1}{q}\right)^{\lfloor\frac{n}{2}\rfloor} & \text{si } k=\lfloor\frac{n}{2}\rfloor+1.\end{cases}$$

et en remplaçant dans (4) :

$$\overline{T}=(\lfloor\frac{n}{2}\rfloor+1)\left(\frac{1}{q}\right)^{\lfloor\frac{n}{2}\rfloor}+(q-1)\sum_{k=1}^{\lfloor\frac{n}{2}\rfloor}k\left(\frac{1}{q}\right)^{k}.$$

(7) On rappelle (cf. exercices de combinatoire de première année) que pour un entier n et un réel $q\neq 1$, on a

$$\sum_{k=1}^{n}kq^k=\frac{q^{n+1}\big(n(q-1)-1\big)+q}{(q-1)^2}. \tag{5}$$

On a donc en notant $m:=\lfloor\frac{n}{2}\rfloor$:

$$S=\frac{\frac{m(q-1)-1}{q^{m+1}}+q}{(\frac{1-q}{q})^2}=\frac{q^{m+1}+m(1-q)-q}{q^m(q-1)^2}.$$

Et donc

$$\begin{aligned}\overline{T} &= \frac{(m+1)}{q^m} + \frac{q^{m+1} + m(1-q) - q}{q^m(q-1)} \\ &= \frac{(m+1)(q-1) + q^{m+1} + m(1-q) - q}{q^m(q-1)} \\ &= \frac{q^{m+1} - 1}{q^m(q-1)}\end{aligned}$$

Soit finalement

$$\overline{T}(n,q) = \frac{q^{\lfloor \frac{n}{2} \rfloor + 1} - 1}{q^{\lfloor \frac{n}{2} \rfloor}(q-1)}. \tag{6}$$

(8) On repart de l'expression (6) pour obtenir :

$$\overline{T}(n,q) = \frac{q}{q-1} - \frac{1}{q^{\lfloor \frac{n}{2} \rfloor}(q-1)}$$

on en déduit que

$$\lim_{n \to +\infty} \overline{T}(n,q) = \frac{q}{q-1}.$$