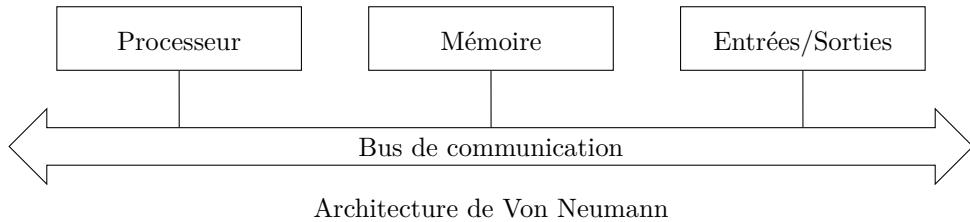


Automatisation du traitement de l'information

Qu'est-ce qu'un ordinateur ?

L'organisation générale d'un ordinateur est composée, à minima, d'un processeur et d'une mémoire. Cette structure n'a pas varié depuis les années 1940 et a été formalisée pour la première fois par Von Neumann en 1945 .



Précisons le vocabulaire :

- * Un **périphérique d'entrée** permet de fournir des informations au système.
- * Un **périphérique de sortie** permet de faire sortir des informations du système.
- * Un **périphérique d'entrée/sortie** permet l'échange d'informations dans les deux sens.
- * La **mémoire vive** ou temporaire ne persiste pas hors tension.
- * La **mémoire morte** ou persistante permet une mémorisation permanente.
- * Le **processeur** est le composant électronique qui permet d'effectuer les calculs.

Proposer des exemples :

1. Des périphériques d'entrée : *Clavier, souris, scanner, microphone...*
2. Des périphériques de sortie : *Écran, imprimante...*
3. Des périphériques d'entrée/sortie : *Écran tactile, imprimante/scanner...*
4. Un support de mémoire vive : *barrette de mémoire RAM (Random Access Memory), registres...*
5. Des supports de mémoire morte : *Disque dur, SSD (solid-state drive), clé USB, Disque optique (CD, Blu-ray)...*
6. Un constituant fondamental des processeurs : *Le transistor*

Histoire du traitement de l'information

Le mot français « informatique » apparaît dans le langage au début des années 1960. Il est le néologisme par contraction des mots « information » et « automatique ». Ce terme désigne aujourd'hui le domaine industriel en rapport avec l'ordinateur et la science du traitement de l'information par des algorithmes.

Les premières Machines - Quelques exemples

(À faire à la maison) À l'aide des documents 1 à 4 (pages suivante) :

1. Indiquer le travail que pouvait réaliser les premières machines
 - La première, la pascaline pouvait calculer des additions et des soustractions...*
 - Le métier Jacquard tissait sur commande à cartes perforées...*
 - La machine analytique était sensée pouvoir réaliser n'importe quelle calcul...*
2. Calculer le nombre de combinaisons qui sont possibles sur une colonne de carte perforée IBM.
 - (L'invention d'une machine à cartes perforées, utilisée pour le recensement de 1890 aux États-Unis, est à la base du développement de IBM)*
 - Sur une colonne de 12 lignes peuvent être inscrit $2^{12} = 4096$ codes différents*

Document 1 : La pascaline

La pascaline est une machine à calculer mécanique inventée en **1642** par **Blaise Pascal** (1623-1662). Cette machine pouvait faire les additions et les soustractions. Blaise Pascal inventa sa machine pour aider son père, surintendant et perceuteur de taxes, à faire ses calculs. La machine ne connut pas de succès commercial en raison de son prix élevé.



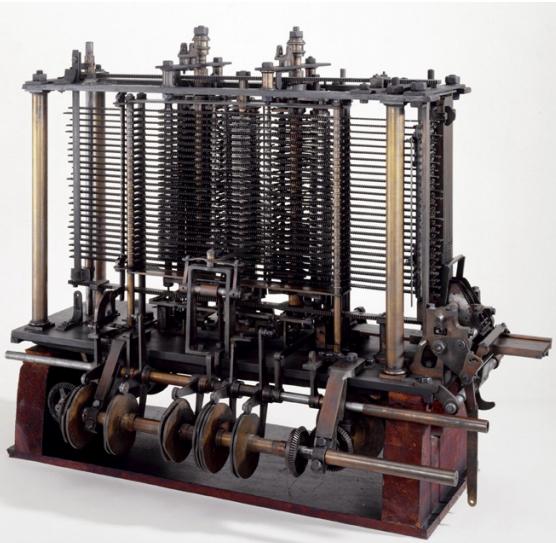
Document 2 : Le métier Jacquard

Au début du **XIX^e siècle**, Joseph-Marie **Jacquard** invente une machine à tisser mécanique capable de produire des tissus présentant des motifs complexes, en suivant des instructions contenues dans des cartes perforées défilant dans la machine.



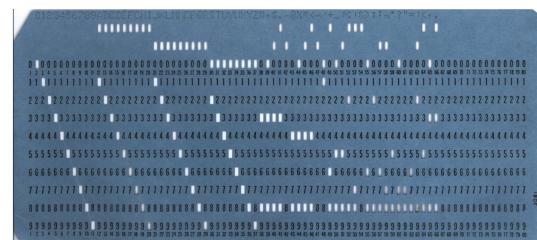
Document 3 : La machine analytique de Babbage

Afin d'aider les marins lors de leur navigation, **Babbage** envisage en 1822 de créer une machine qui produise une feuille de route en calculant la carte du ciel jour après jour pour les semaines à venir. Il réalise en **1834**, qu'il peut créer une machine plus performante, capable d'exécuter n'importe quelle tâche qui pourrait lui être décrite. Inspiré par le métier à tisser Jacquard, il envisage que cette description, qui n'est qu'une suite de calculs, soit réalisée par des cartes perforées. Cette machine ne sera jamais finalisée.



Document 4 : Les cartes perforées

Elles existeront jusqu'à la **fin des années 1970**. Elles présentent l'énorme avantage de dissocier le temps d'écriture de l'information et le temps de sa lecture. Standardisée par la société IBM, les **cartes perforées** présentent alors 80 colonnes et 12 lignes. Chaque colonne code, en binaire (0 ou 1), un caractère.



Les précurseurs de la programmation

Ada Lovelace

Ada Lovelace (1815-1852) est la fille du poète anglais Lord Byron. En 1833, elle rencontre Charles Babbage, qui est notamment l'inventeur d'une machine à calculer programmable connue sous le nom de machine analytique. Ada Lovelace traduit en anglais un article écrit en français sur la machine analytique de Babbage. Elle complète cet article de commentaires qui formalisent les idées de Babbage.

Dans sa note A, elle promet que la machine sera capable de calculs abstrait. Elle entrevoit que de telles machines ne seront pas limitées à des calculs numériques mais pourront réaliser des opérations sur toute sorte d'objets. Elle ajoute qu'un jour, elles pourrait même composer de la musique.

Ada Lovelace commence sa Note G en affirmant que, malgré ses pouvoirs impressionnantes, on ne peut pas vraiment dire que la Machine Analytique « pense ». Cette partie de Note G est ce que Alan Turing appellera plus tard « l'objection de Lady Lovelace ». Néanmoins, poursuit Lovelace, la machine peut faire des choses extraordinaires. Pour illustrer sa capacité à gérer des problèmes encore plus complexes, Lovelace fournit son programme de calcul des nombres de Bernoulli.

Les nombres de Bernoulli

Les nombres de Bernoulli (B_n) peuvent être calculés de proche en proche en résolvant les équations suivantes, écrites à partir du triangle de Pascal. (Il y a un décalage entre la notation moderne et celle utilisée dans le document, que l'on prend comme référence ici)

Triangle de Pascal

1	1						
1	2	1					
1	3	3	1				
1	4	6	4	1			
1	5	10	10	5	1		
1	6	15	20	15	6	1	
1	7	21	35	35	21	7	1

Équations donnant (B_n)

(En notation moderne, il y a un premier terme avant B_0 égal à 1)

$$1 + 2B_0 = 0$$

$$1 + 3B_0 + 3B_1 = 0$$

$$1 + 4B_0 + 6B_1 + 4B_2 = 0$$

$$1 + 5B_0 + 10B_1 + 10B_2 + 5B_3 = 0$$

$$1 + 6B_0 + 15B_1 + 20B_2 + 15B_3 + 6B_4 = 0$$

$$1 + 7B_0 + 35B_1 + 35B_3 + 21B_4 + 7B_5 = 0$$

1. Calculer :

$$B_0 = \frac{-1}{2} \dots \quad B_1 = \frac{1}{6} \dots \quad B_2 = 0 \dots \quad B_3 = \frac{-1}{30} \dots \quad B_4 = 0 \dots \quad B_5 = \frac{1}{42} \dots$$

2. On remarque, et on peut démontrer que tous les termes d'indice pair sont nuls, sauf B_0 qui vaut $\frac{-1}{2}$. Le suivant non nul est donc B_7 . Le programme proposé permet de le calculer (pour $n = 4$).

L'équation à résoudre est : $1 + 9B_0 + 36B_1 + 126B_3 + 84B_5 + 9B_7 = 0$.

Elle est utilisée sous la forme $A_0 + A_1B_1 + A_3B_3 + A_5B_5 + B_7 = 0$

Quel devrait-être la valeur de A_0 ? $A_0 = \frac{1+9B_0}{9} = (1 - \frac{9}{2}) \times \frac{1}{9} = \frac{-7}{2} \times \frac{1}{9} = \frac{-7}{18}$

Vérifier que cela correspond à la formule $-\frac{1}{2} \frac{2n-1}{2n+1} = A_0$ pour $n = 4$. $\frac{-1}{2} \frac{2 \times 4 - 1}{2 \times 4 + 1} = \frac{-1}{2} \frac{7}{9} = \frac{-7}{18}$

3. Quels noms portent les variables utilisées ? V_1, V_2, V_3 (L'exposant gauche donne le nombre d'affectations).....

4. Sont-elles initialisées ? (préciser) Les trois premières oui : à 1, 2 et n (pour calculer B_{2n-1}).....

5. Quelles opérations sont utilisées ? (Seconde colonne) Les opérations +, -, × et ÷.....

6. Lesquelles des instructions de base de la programmation sont utilisées (parmi l'affectation, les conditionnelles et les boucles) ? (Préciser)

L'affectation est utilisée à chaque étape (détailée colonnes 4 et 5).....

Les instructions conditionnelles ne sont pas explicitement utilisées.....

Une boucle est clairement utilisée : « Here follows a repetition of ...».....

Il y a même des boucles imbriquées (les accolades en colonnes 1 et 2).....

En 1977, le département de la Défense des États Unis dénomme un langage de programmation Ada (inspiré du langage Pascal) en l'honneur d'Ada Lovelace. Ce langage, très structuré, est utilisé pour des projets nécessitant un très haut niveau de fiabilité (domaines militaire, aéronautique, etc).

Diagram for the computation by the Engine of the Numbers of Bernoulli. See Note G. (page 729 et seq.)

Nature of Operation.		Data.		Working Variables.										Result Variables.					
Number of Operation.				Statement of Results.															
Variables acted upon.	Variables receiving results.	Indication of change in the value on any Variable.		V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8	V_9	V_{10}	V_{11}	V_{12}	V_{21}	V_{22}	V_{23}	V_{31}
1	\times	$V_2 \times V_3$	V_4, V_5, V_6, V_7	$\begin{cases} V_2 = V_2 \\ V_3 = V_3 \\ V_4 = V_4 \\ V_5 = V_5 \\ V_6 = V_6 \\ V_7 = V_7 \end{cases}$	$= 2n$	\dots	2	n	$2n$	$2n$	0	0	0	0	0	0	0	0	
2	-	$-V_4 - V_1$	$2V_4$	$\begin{cases} V_4 = V_4 \\ V_1 = V_1 \end{cases}$	$= 2n - 1$	\dots	1	\dots	$2n - 1$	\dots	0	0	0	0	0	0	0	0	
3	+	$V_5 + V_1$	$2V_5$	$\begin{cases} V_5 = V_5 \\ V_1 = V_1 \end{cases}$	$= 2n + 1$	\dots	1	\dots	$2n + 1$	\dots	0	0	0	0	0	0	0	0	
4	+	$V_6 + 2V_4$	V_1, V_1	$\begin{cases} V_6 = V_6 \\ V_4 = V_4 \\ V_1 = V_1 \end{cases}$	$= 2n - 1$	\dots	1	\dots	$2n - 1$	\dots	0	0	0	0	0	0	0	0	
5	+	$V_{11} + V_2$	$2V_{11}$	$\begin{cases} V_{11} = V_{11} \\ V_2 = V_2 \end{cases}$	$= 1$	\dots	2	\dots	2	\dots	0	0	0	0	0	0	0	0	
6	-	$-V_{13} - 2V_{11}$	V_{13}, V_{11}	$\begin{cases} V_{13} = V_{13} \\ V_{11} = V_{11} \end{cases}$	$= -\frac{1}{2} \cdot \frac{2n - 1}{2n + 1} = A_0$	\dots	1	\dots	$2n - 1$	\dots	0	0	0	0	0	0	0	0	
7	-	$-V_3 - V_1$	V_{10}	$\begin{cases} V_3 = V_3 \\ V_1 = V_1 \end{cases}$	$= n - 1 (= 3)$	\dots	1	\dots	n	\dots	0	0	0	0	0	0	0	0	
8	+	$V_2 + 0V_7$	$1V_7$	$\begin{cases} V_2 = V_2 \\ V_7 = V_7 \\ 0V_7 = 0V_7 \end{cases}$	$= 2 + 0 = 2$	\dots	1	\dots	2	\dots	0	0	0	0	0	0	0	0	
9	+	$V_6 + 1V_7$	$3V_{11}$	$\begin{cases} V_6 = V_6 \\ V_7 = V_7 \\ 1V_7 = 1V_7 \end{cases}$	$= \frac{2n}{2} = A_1$	\dots	1	\dots	$2n$	\dots	2	\dots	0	0	0	$\frac{2n}{2} = A_1$	\dots	\dots	
10	\times	$V_{21} \times V_{12}$	V_{12}	$\begin{cases} V_{21} = V_{21} \\ V_{12} = V_{12} \\ 3V_{12} = 3V_{12} \end{cases}$	$= B_1 \cdot \frac{2n}{2} = B_1 A_1$	\dots	1	\dots	$2n$	\dots	0	0	0	0	0	$B_1 \cdot \frac{2n}{2} = B_1 A_1$	\dots	\dots	
11	+	$V_{12} + V_{13}$	$2V_{13}$	$\begin{cases} V_{12} = V_{12} \\ V_{13} = V_{13} \end{cases}$	$= -\frac{1}{2} \cdot \frac{2n - 1}{2n + 1} + B_1 \cdot \frac{2n}{2}$	\dots	1	\dots	$2n - 1$	\dots	0	0	0	0	0	0	0	0	
12	-	$-V_{10} - V_1$	$2V_{10}$	$\begin{cases} V_{10} = V_{10} \\ V_1 = V_1 \end{cases}$	$= n - 2 (= 2)$	\dots	1	\dots	$n - 2$	\dots	0	0	0	0	0	$n - 2$	\dots	\dots	
13	-	$-V_6 - V_1$	$2V_6$	$\begin{cases} V_6 = 2V_6 \\ V_1 = V_1 \end{cases}$	$= 2n - 1$	\dots	1	\dots	$2n - 1$	\dots	0	0	0	0	0	$\frac{2n}{2} = A_1$	\dots	\dots	
14	+	$+V_1 + V_7$	$2V_7$	$\begin{cases} V_1 = V_1 \\ V_7 = V_7 \end{cases}$	$= 2 + 1 = 3$	\dots	1	\dots	$2n$	\dots	2	\dots	0	0	0	$\frac{2n}{2} = A_1$	\dots	\dots	
15	+	$+V_6 + 2V_7$	V_8	$\begin{cases} V_6 = V_6 \\ V_7 = V_7 \\ 2V_7 = 2V_7 \end{cases}$	$= 2n - 1$	\dots	1	\dots	$2n$	\dots	0	0	0	0	0	$\frac{2n}{2} = A_1$	\dots	\dots	
16	\times	$V_8 \times 3V_{11}$	$4V_{11}$	$\begin{cases} V_8 = V_8 \\ 3V_{11} = 3V_{11} \end{cases}$	$= \frac{2n}{2} \cdot \frac{2n - 1}{3} = B_1 A_1$	\dots	1	\dots	$2n - 1$	\dots	0	0	0	0	0	$B_1 A_1$	\dots	\dots	
17	-	$-V_6 - V_1$	$3V_6$	$\begin{cases} V_6 = V_6 \\ V_1 = V_1 \end{cases}$	$= 2n - 2$	\dots	1	\dots	$2n - 2$	\dots	0	0	0	0	0	$\frac{2n}{2} = A_1$	\dots	\dots	
18	+	$+IV_1 + V_7$	$3V_7$	$\begin{cases} IV_1 = IV_1 \\ V_7 = V_7 \end{cases}$	$= 3 + 1 = 4$	\dots	1	\dots	$2n - 1$	\dots	0	0	0	0	0	$\frac{2n}{2} = A_1$	\dots	\dots	
19	+	$+3V_6 + 2V_7$	IV_9	$\begin{cases} 3V_6 = 3V_6 \\ 2V_7 = 2V_7 \\ IV_9 = IV_9 \end{cases}$	$= \frac{2n - 2}{4} = A_2$	\dots	1	\dots	$2n - 1$	\dots	0	0	0	0	0	$\frac{2n}{2} \cdot \frac{2n - 1}{3} = A_2$	\dots	\dots	
20	\times	$IV_9 \times IV_{11}$	$5V_{11}$	$\begin{cases} IV_9 = IV_9 \\ 5V_{11} = 5V_{11} \end{cases}$	$= 3 + 1 = 4$	\dots	1	\dots	$2n - 1$	\dots	0	0	0	0	0	$\frac{2n}{2} \cdot \frac{2n - 1}{3} = A_2$	\dots	\dots	
21	\times	$IV_{22} \times IV_{12}$	$0V_{12}$	$\begin{cases} IV_{22} = IV_{22} \\ 0V_{12} = 0V_{12} \end{cases}$	$= B_3 \cdot \frac{2n}{2} = B_3 A_2$	\dots	1	\dots	$2n - 1$	\dots	0	0	0	0	0	$B_3 A_2$	\dots	\dots	
22	+	$+3V_{12} + 2V_{13}$	$3V_{13}$	$\begin{cases} 3V_{12} = 3V_{12} \\ 2V_{13} = 2V_{13} \end{cases}$	$= A_0 + B_1 A_1 + B_3 A_2$	\dots	1	\dots	$2n - 2$	\dots	0	0	0	0	0	$\{A_3 + B_1 A_1 + B_2 A_2\}$	\dots	\dots	
23	-	$-3V_{10} - V_1$	$3V_{10}$	$\begin{cases} 3V_{10} = 3V_{10} \\ V_1 = V_1 \end{cases}$	$= n - 3 (= 1)$	\dots	1	\dots	$n - 3$	\dots	0	0	0	0	0	$n - 3$	\dots	\dots	
24	+	$+4V_{13} + 0V_{24}$	W_{24}	$\begin{cases} 4V_{13} = 0V_{13} \\ 0V_{24} = 0V_{24} \end{cases}$	$= B_7$	\dots	1	\dots	$n + 1$	\dots	0	0	0	0	0	\dots	\dots	\dots	
25	+	$+IV_1 + V_3$	V_3	$\begin{cases} IV_1 = IV_1 \\ V_3 = V_3 \end{cases}$	$= n + 1 = 4 + 1 = 5$	\dots	1	\dots	$n + 1$	\dots	0	0	0	0	0	\dots	\dots	\dots	

Here follows a repetition of Operations thirteen to twenty-three.

B_7	\dots																
-------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------

by a Variable-card.
by a Variable-card.

Alan Turing

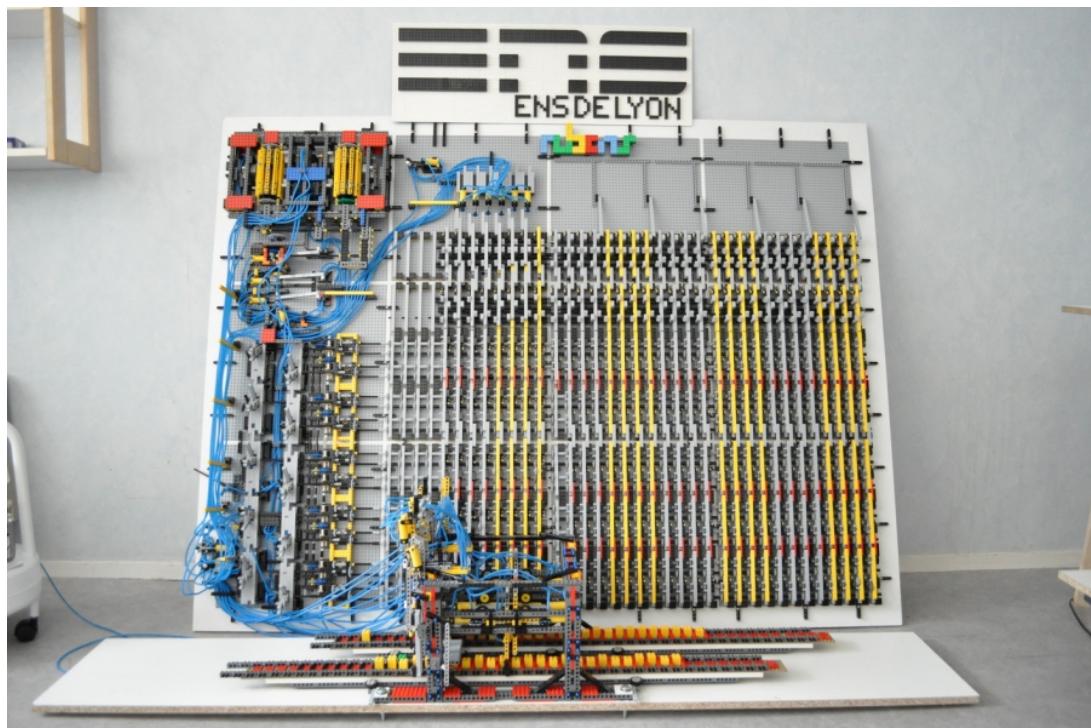
En 1936, le mathématicien et cryptologue, Alan Turing a imaginé un modèle abstrait pour définir une notion qui jusqu'alors était restée intuitive : la calculabilité. Il cherche à définir la différence entre ce qui est calculable et ce qui ne l'est pas. Ce qui est calculable doit pouvoir peut se décomposer en un nombre fini d'étapes, pouvant être réalisées par une machine.

Il imagine alors le moyen pour une machine de reproduire certaines actions humaines :

« je fais le pari que d'ici 50 ans, une personne non prévenue n'aura plus le moyen de distinguer les réponses données par un ordinateur de celles données par un humain, et ce, sur n'importe quel sujet. »

Cette machine se décrit très simplement. Elle utilise un ruban qui contient une suite de cases dans lesquelles sont inscrites des données (0 et 1). La machine est capable de lire ce qu'il y a dans les cases, de se déplacer sur le ruban d'une case à gauche ou à droite et d'écrire dans une case.

Ce modèle est aujourd'hui connu sous le nom de machine de Turing et est toujours utilisé en informatique théorique pour résoudre les problèmes de calculabilité.



Von Neumann

En s'inspirant de ses prédécesseurs, il élabore en 1945, la première description d'un ordinateur dont le programme est stocké dans sa mémoire : ici, l'ordinateur dispose d'un processeur pour exécuter les instructions et d'une structure de stockage unique (Mémoire vive RAM) pour conserver à la fois les instructions et les données demandées ou produite par le calcul. C'est le premier modèle de calculateur universel programmable concret dont le premier modèle sera commercialisé en 1951.

- Montrer (en utilisant tous les documents de ce cours) comment les avancées technologiques se sont construites en utilisant les technologies antérieurs.

La première machine à calculer, la pascaline et les cartes perforées du métier Jacquard sont les bases de la machine analytique de Babage. C'est en en faisant la traduction qu'Ada Lovelace posa les bases de la programmation et initia la notion de machine universelle. Alan Turing formalisa la notion et Von Neumann en fit une description concrète.

- Expliquer pourquoi Alan Turing peut être considéré comme le père de l'intelligence artificielle.

Alan Turing décrit ce qui est « calculable » et prévoit que son modèle de machine, basé uniquement sur le calcul binaire... sera capable à terme de soutenir une conversation avec un humain sur n'importe quel sujet (sans que celui-ci ne devine... qu'il parle avec une machine).

Les ordinateurs autour de nous

Expliquer en quoi les objets ci-dessous, de la vie courante, sont des ordinateurs et indiquer par qui ils sont programmables :

1. Thermostat d'ambiance



2. Smartphone



3. Box internet



4. Ordinateur de bord d'une voiture



Tous les appareils cités possèdent un ou plusieurs processeurs et au minimum des boutons (entrées) et un écran (sortie)....

Il possèdent également tous une mémoire (au minimum pour stocker les réglages de l'utilisateur).....

Le thermostat d'ambiance est programmable/configurable par l'utilisateur (réglage de la température selon les heures)....

Le smartphone est programmable par l'utilisateur (averti) qui peut (télécharger et) installer des programmes écrits par des professionnels.

La box internet est programmable par le fournisseur d'accès et configurable par l'utilisateur.....

L'ordinateur de bord d'une voiture est programmable par le constructeur.....

Numérisation de l'information

2

Pour qu'une machine puisse opérer de manière automatique sur des informations, une première étape consiste à les numériser. On transforme ainsi sous forme de suites de nombres, les textes, les sons, les images et les vidéos. Les programmes eux-même, ne sont pour une machine que des données à traiter. On peut se donner une bonne image de la programmation de bas niveau (langage machine et assembleur) avec l'émulateur <http://www.peterhigginson.co.uk/RISC/> (SELECT > add puis RUN par exemple)

Le système binaire

Comme les cartes perforées des premiers calculateurs qui ne contenaient que des informations binaires : un trou ou pas, les ordinateurs modernes ne manipulent toujours que des informations binaires : une tension électrique ou pas. Formellement, on note 1 ou 0 ces deux informations et on appelle **bit** : « Binary digit » ce plus petit élément d'information qui ne peut donc prendre que deux valeurs.

Numération

On apprend à l'école comment écrire un nombre en base 10, c'est à dire en utilisant les chiffres 0, 1, 2, 3, 4, 5, 6, 7, 8 et 9, en les positionnant de droite à gauche (unités, dizaines, centaines, milliers, ...). De la même façon, on peut les écrire en base 2, c'est à dire en utilisant les chiffres 0 et 1, en les positionnant de droite à gauche (unité, « deuzaire », ...):

Exemple

Pour compter le nombre d'étoiles ci-dessous :

On peut faire des paquets de 10 :

$$10^3 = 1000 \quad | \quad 10^2 = 100 \quad | \quad 10 \quad | \quad 1$$

*****	*****	*****	***
-------	-------	-------	-----

Il y a 0 paquet de 1000, 0 paquet de 100, 1 paquet de 10 et 3 étoiles : 13 en décimal.

On peut aussi faire des paquets de 2 :

$$2^3 = 8 \quad | \quad 2^2 = 4 \quad | \quad 2 \quad | \quad 1$$

**	**	**	**	**	**
----	----	----	----	----	----

Il y a 1 paquet de 8, 1 paquet de 4, 0 paquet de 2 et 1 étoile : 1101 en binaire.

(Les Shadoks font des paquets de 4 - épisode 44 de la saison 2 de « Et voilà les Shadoks » de 1969)

- Déterminer l'écriture binaire du nombre dont l'écriture décimal est : 228. (Préciser votre démarche.)

On cherche à décrire 228 avec une succession de 1 et 0 correspondant à des puissances de 2 dans 228.....

On cherche la plus grande inférieure à 228 : 1, 2, 4, 8, 16, 32, 64, 128, 256. C'est 128 = 2⁷.....

Alors 228 = 128 + 100. Et on recommence avec 100 : jusqu'à 228 = 128 + 64 + 32 + 4 = 11100100₂.....

- Retrouver l'écriture décimale du nombre binaire 10010010111

10010010111 = 2¹⁰ + 2⁷ + 2⁴ + 2² + 2 + 1.....

= 1024 + 128 + 16 + 4 + 2 + 1 = 1175.....

Octet

Un octet est une séquence de 8 bits. En anglais, octet se dit « Byte » (\neq « bit »).

- Pourquoi le nombre 228 est-il codable sur un octet ? *Il s'écrit en binaire avec 8 chiffres.....*
- Quel est le plus grand entier codable sur un octet ? *11111111₂ = 100000000₂ = 11 = 256 - 1 = 255..*
- Combien d'informations distinctes sont codables sur un octet ? *2⁸ = 256 (c'est à dire de 0 à 255).....*

Le codage ASCII

Le code ASCII (« American Standard Code for Information Interchange ») fait son apparition dans les années 1960. Son objectif était de standardiser le codage des caractères. À l'époque les informations se codaient sur 1 octet (contre 8 de nos jours sur des systèmes dit « 32 bits »). Mais un bit de cet octet était requis pour contrôler sa transmission. La table ASCII définissait donc 128 caractères (sur 7 bits).

Exemple – Codage ASCII des lettres

lettre	A	B	C	D	E	F	G	H	...	Z
code	65	66	67	68	69	70	71	72	...	90
lettre	a	b	c	d	e	f	g	h	...	z
code	97	98	99	100	101	102	103	104	...	122

En général les langages de programmations donnent des fonctions permettant de coder ou de décoder en ASCII. (CODE et CAR avec un tableau, `ord` et `chr` en Python...)

Exemple – Fonctions de codage

The screenshot shows two examples of encoding letters:

- Avec un tableau** (Using a table):
 - Top table (A1 to C1): Shows letter A with code 65.
 - Bottom table (B1 to C1): Shows letter A with code 65.
- Avec le langage Python**:


```
>>> ord('A')
65
>>> chr(65)
'A'
```

(À faire à la maison)

- Quels sont les caractères codés entre le 'z' et le 'a' ? (Détailler votre méthode.)

```
>>> for k in range(91,97):
    .....
    .....
    print(k,chr(k),end=' ')
.....
91 [ 92 \ 93 ] 94 ^ 95 - 96 ^
```

- Vérifier que l'écriture binaire des codes des lettres majuscules et minuscules ne diffère que d'un bit. Lequel ? (Détailler votre méthode.)

De A à Z les codes vont de 65 = 1000001₂ à 90 = 1011010₂

De a à z les codes vont de 97 = 1100001₂ à 122 = 1111010₂

On passe donc d'une majuscule à une minuscule en changeant le bit de poids 5 uniquement

Autre approche possible : 97 - 65 = 98 - 66 = ... = 122 - 90 = 32 = 2⁵

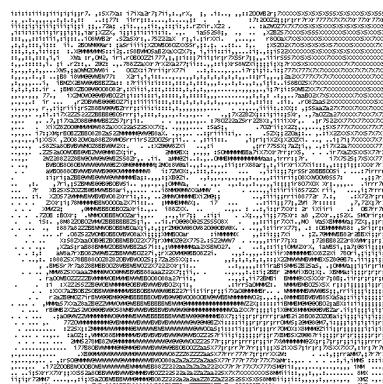
Art ASCII

Le codage ASCII a inspiré les programmeurs puis les artistes depuis les années 1960. Ils en ont fait un art qui consiste à produire des images à l'aide des glyphes des caractères.

La forme la plus simple d'art ASCII est la combinaison de deux ou trois caractères pour exprimer une émotion en texte :

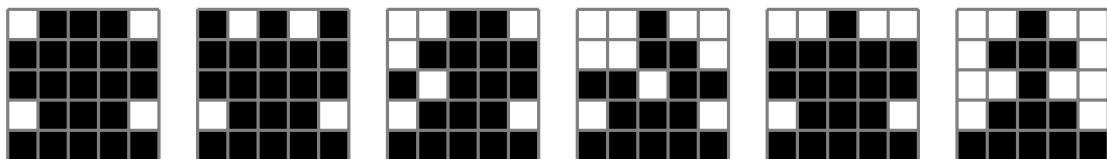
:	:)	:('
sourire	triste	pleure
;)	:D	:p
clin d'œil	rire	tire la langue

Un exemple plus complexe ci-contre.



Le codage d'une image

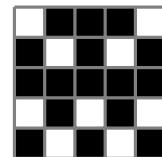
Il suffit de quelques carrés noirs ou blancs pour représenter quelque chose. Avec une grille de 5 par 5, on peut par exemple suggérer les pièces d'un jeu d'échec :



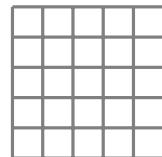
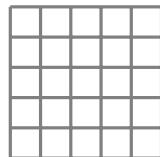
On appelle **pixel** (pour « picture element ») chacune de ces cases et les premières technologies codaient 0 une case blanche et 1 une case noire.

- Proposer un codage de l'image ci-contre inspirée de Space invaders (Shoot'em up de 1978)

01110 . 10101 . 11111 . 01010 . 10101



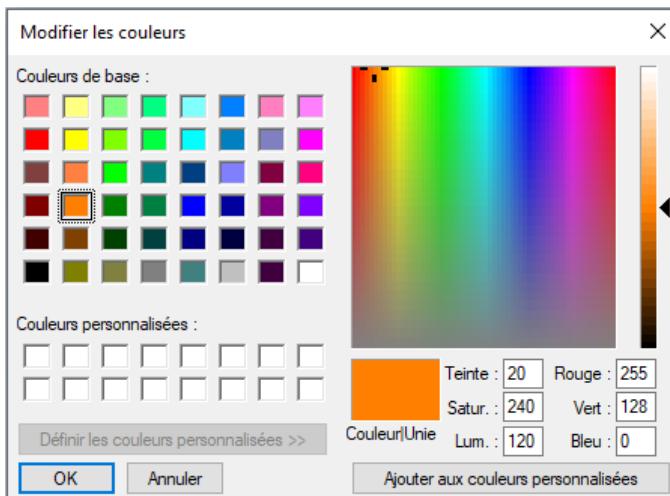
- Noircir quelques carrés sur la figure ci-contre, écrire son code et le proposer à votre voisin. Trouver ce que représente son dessin à partir de son code.



.....

De nos jours chaque pixel n'est plus codé sur un seul bit, mais sur 24, répartis en 3 groupes de 8, chaque groupe représentant l'intensité lumineuse d'une couleur primaire : Rouge, Verte ou Bleue.

- Écrire le codage de la couleur (orange) sélectionnée ci-dessous :



.1111 . 1111 . 1000 . 0000 . 0000 . 0000

- Proposer le codage d'un gris moyen :

.1000 . 0000 . 1000 . 0000 . 1000 . 0000

Tailles et types de fichier

Ordres de grandeur

On vient de voir comment on peut coder un nombre, comment on peut numériser une lettre donc un texte, donc un programme. On a également vu comment numériser une image. Les technologies de prise de vue ou de son permettent encore de numériser ce que l'on voit et ce que l'on entend.

1. Compléter les informations du tableau ci-dessous, donnant quelques valeurs élémentaires du traitement de l'information par un ordinateur.

Information à numériser	Encodage en bit	Nombre de valeurs
	en octet	
Texte (ASCII)	8	1 256
Son (CD)	16	2 65 536
Image (RVB)	24	3 16 777 216

2. Sur le site www.gutenberg.org, on trouve par exemple le roman « Notre-Dame de Paris » de Victor Hugo. La version en texte contient 1 117 109 caractères.

- a. Sur combien de bits ; d'octets ; de mégaoctets est-il codé ?

$$1.117.109 \text{ octets} = 8.936.872 \text{ bits} \approx 1.1 \text{ Mo}$$

- b. Sur le site sont proposés les chargements suivants :

Notre-Dame de Paris by Victor Hugo



Download This eBook

Format ?	Size
Read this book online: HTML	1.3 MB
EPUB (no images)	498 kB
Kindle (no images)	2.1 MB
Plain Text UTF-8	1.1 MB
More Files...	

* Que signifient « MB » et « kB » ? *MB pour MégaByte, donc Mo en français, kB pour kiloByte, donc ko.*

* Pourquoi le fichier HTML est-il plus lourd ? *Il faut rajouter les informations de structure de la page.*

* Pourquoi le fichier EPUB est-il plus léger ? *Il est compressé.*

3. On considère un smartphone dont l'écran comporte 2 560 pixels en largeur sur 1 440 pixels en hauteur.

- a. Combien l'écran contient-il de pixels ? $2.560 \times 1.440 = 3.686.400$
- b. Combien d'octets sont nécessaires pour enregistrer une capture d'écran ?

$$3.686.400 \times 3 = 11.059.200. Soit \approx 11 \text{ Mo}$$

4. Pour enregistrer un son échantillonner sur 16 bits à 44,1kHz (44 100 fois par seconde) en stéréo.

- a. Combien de bits d'informations sont enregistrées en une minute ?

$$60 \text{ (secondes)} \times 44100 \times 2 \text{ (stéréo)} \times 16 \text{ (bits)} = 84.672.000. Soit 10.584 \text{ Mo}$$

- b. Combien de minutes d'enregistrement permet un CD de 550Mo à cette qualité ?

$$550 \div 10.584 \approx 52 \text{ (minutes)}$$

Extensions

- Pour un système d'exploitation, on peut distinguer deux grandes familles de fichiers informatiques :
- * Les fichiers de « **données** ». La plupart des informations qui se trouvent sur un disque dur sont sous la forme de fichiers de données (textes, images, ...). Chaque fichier est identifié par son nom suivi d'une extension qui permet au système d'exploitation de connaître le programme à utiliser pour manipuler le fichier.
 - * Les fichiers « **exécutables** ». Sous Windows, par exemple, les fichiers exécutables sont ceux qui portent les extensions .exe ou .com.

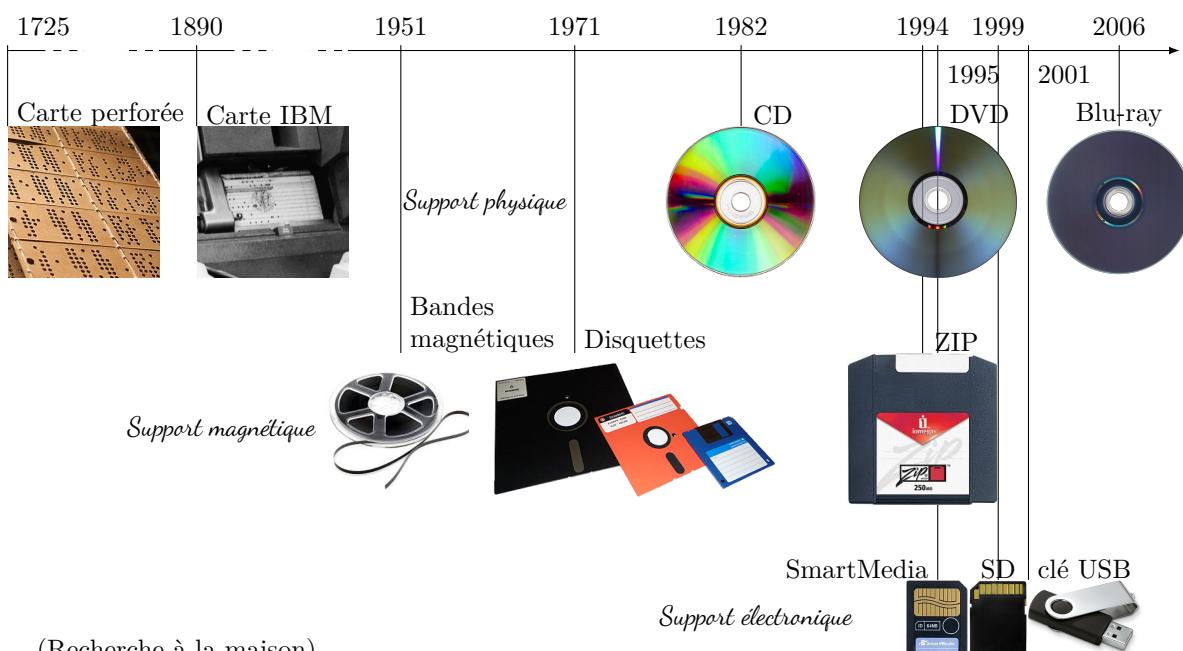
Exemple – Extensions de fichiers

Nom	Taille	Type
fichier.doc	11 Ko	Document Microsoft Word
fichier.htm	1 Ko	HTML Document
fichier.ppt	10 Ko	Présentation Microsoft PowerPoint
fichier.txt	2 Ko	Document texte
fichier.xls	11 Ko	Feuille de calcul Microsoft Excel
fichier.bmp	61 Ko	Image bitmap
fichier.sxw	3 Ko	OpenOffice.org 1.1.0 Texte
fichier.exe	3 Ko	Application
fichier.zip	1 Ko	Filzip file

Citer au moins une extension possible pour chacun des types de fichiers suivants :

Texte ..doc..txt.....	Son ..wav..mp3.....	Image ..bmp..png.....	Exécutable ..exe..com.....	Vidéo ..mp4..avi.....
--------------------------	------------------------	--------------------------	-------------------------------	--------------------------

Les supports de stockage



(Recherche à la maison)

Quelles étaient (sont) les capacités de stockage des supports suivants ?

Carte IBM	Disquette	CD	DVD	Blu-Ray	clé USB
120 octets...	1.44 Mo....	550 à 700 Mo	4.7 à 17 Go	25 à 50 Go	8 Mo à 256 Go...

Programmation et bogue informatique

3

Les programmes informatiques sont écrits par des opérateurs humains dans des langages sous forme de texte ou sous formes de graphiques. Ces programmes sont ensuite traduits en suite d'instructions qu'un processeur doit exécuter. Ils peuvent être stockés, transportés et traité par des ordinateurs.

Citer 5 langages informatiques.

(Il y en a des centaines) (Fortran, Ada, PHP, C#, JavaScript, Java, Python)

Il y a trois fondamentaux dans la programmation :

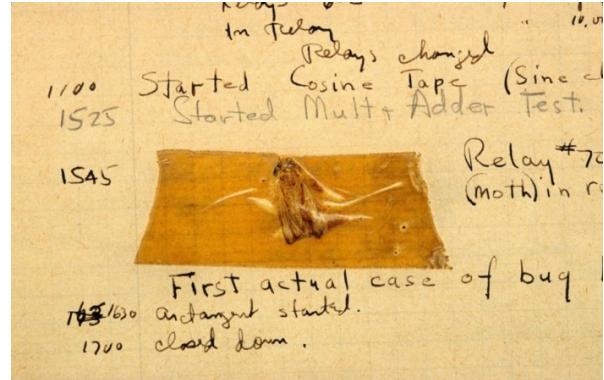
- ★ L'accès en lecture et/ou écriture à la mémoire. Cela peut se faire à l'aide de « variables »
- ★ Les branchements permettant de passer d'un partie du code à l'autre sous certaines conditions. Cela peut se faire à l'aide d'« instructions conditionnelles ».
- ★ La répéter, autant que nécessaire, de certaines parties du code. Cela peut se faire à l'aide de « boucles ».

Bogue

Un programme peut comporter jusqu'à plusieurs centaines de millions de lignes de code, ce qui rend très probable la présence d'erreurs appelées bogues. Ces erreurs peuvent conduire un programme à avoir un comportement inattendu et entraîner des conséquences graves.

Le 9 septembre 1947, à l'université Harvard , le calculateur Mark II défaillie : quelle est la source de la panne ? L'équipe de Grace Hopper (informatienne, mathématicienne et officier supérieure de la marine américaine), chargée de l'appareil, lance ses investigations... Et découvre que c'est un insecte — « » en anglais — qui a provoqué un court-circuit dans les entrailles de l'ordinateur.

Après l'avoir découvert, grillé, entre les flancs du Mark II, l'équipe de Grace Hopper a décidé de scotcher la mite dans le journal de bord de l'ordinateur (le document est conservé au musée Smithsonian de Washington).



Pour comprendre ce qu'est réellement un bogue (ou bug) informatique, écouter la leçon inaugurale de Gérard Berry, au Collège de France, en 2008 : « Pourquoi et comment le monde devient numérique » de 34min à 40min30

<https://www.college-de-france.fr/site/gerard-berry/inaugural-lecture-2008-01-17-18h00.htm>

Étude d'un cas : La fusée Ariane 5. (Documents page suivante.)

1. Quelle a été la cause de l'explosion en vol de la fusée Ariane 5 en 1996 ?

.....

2. Quelles recommandations ont été faites pour éviter que l'accident ne se reproduise ?

.....

Un autres exemples explicités par Gérard Berry en 2019 : « Où va l'informatique ? » de 1h39 à 1h48 : <https://www.college-de-france.fr/site/gerard-berry/course-2019-01-23-16h00.htm>

3. Les bogues de contrôle moteur expliqués dans cette vidéo sont-ils très différents de ceux d'Ariane 5 ?

.....

.....

Document 1 : Explosion d'ariane 5.

En vidéo : <https://www.youtube.com/watch?v=fCnO-UYF3co>

Document 2 : dans Libération le 24 juillet 1996 - Jean-Dominique Merchet.

L'explosion d'Ariane 5, le 4 juin, a été provoquée par une erreur de conception d'un logiciel informatique. C'est la conclusion à laquelle est arrivée la commission d'experts qui a rendu public son rapport, hier à Paris.

Concrètement, le logiciel n'a pas réussi à digérer certaines données concernant la « vitesse horizontale » d'Ariane 5. Une fusée ne s'élève pas sur une trajectoire purement verticale ; peu après le décollage, elle se penche légèrement. La vitesse à laquelle elle quitte la verticale de son point de lancement est sa « vitesse horizontale ». Celle-ci « a dépassé une limite inscrite dans le logiciel du calculateur », affirment les rapporteurs de la commission présidée par Jacques-Louis Lions, de l'Académie des sciences.

« Nous sommes tous coupables », a affirmé Jean-Marie Luton, directeur général de l'Agence spatiale européenne (ESA). Le système de référence inertiel SRI a déjà été utilisé à vingt-trois reprises lors des tirs d'Ariane 4. Mais « on » a oublié que la trajectoire d'Ariane 5 n'était pas la même que celle de sa soeur...

Au passage, les enquêteurs ont découvert d'autres « faiblesses éventuelles de conception au niveau du système de contrôle de vol » et constaté « une anomalie » au niveau de la pression hydraulique des vérins de la tuyère du moteur principal. Le coût des modifications et vérifications est évalué entre 2 et 4% du coût du programme Ariane 5, c'est-à-dire une somme comprise entre 740 millions et 1,48 milliard de francs.

Document 1 : Rapport de la commission d'enquête le 23 juin 1996.**3 CONCLUSIONS**

[...] Les revues et essais approfondis effectués dans le cadre du programme de développement d'Ariane 5 ne comportaient pas les analyses ou essais adéquats du système [...] qui auraient pu mettre en évidence la défaillance potentielle.

4 RECOMMANDATIONS

- 2- Mettre en place une installation d'essais qui réunisse, dans la mesure des possibilités techniques, un maximum d'équipements réels, introduire dans cette installation des données réalistes, et procéder à des essais complets en boucle fermée au niveau systèmes. Procéder à des simulations complètes avant toute mission. Étendre la couverture des essais.
- 11- Revoir la couverture des essais réalisés sur les équipements existants et l'étendre en cas de besoin.
- 12- Traiter les documents de justification avec autant d'attention que le code. Améliorer la technique visant à assurer la cohérence entre le code et ses justifications.

Tests et corrections de programmes

Typeage des variables

En informatique, une variable est un espace mémoire identifié par un nom et pouvant être modifiée. Quand on modifie la mémoire correspondant à une variable, on dit que l'on affecte une valeur à la variable.

Écriture algorithmique :

```
x ← 2
```

Programmation Python :

```
x = 2
```

Cette valeur peut être la numérisation de n'importe quelle information. Pour pouvoir décoder la mémoire correspondant à une variable, il faut donc savoir de quel genre d'information il s'agit : un entier naturel, relatif, un réel, un caractère, un pixel ? C'est ce que l'on appelle le **type** de la variable.

Exemple – Types de variables

Entier	Réel	Chaine de caractères	Booléen
<pre>>>> type(1) <class 'int'></pre>	<pre>>>> type(1.0) <class 'float'></pre>	<pre>>>> type('1.0') <class 'str'></pre>	<pre>>>> type(True) <class 'bool'></pre>

On considère le programme Python ci-contre.

- Quel(s) test(s) pourrait-on mettre en place pour vérifier le comportement normal de ce programme ?

Entrer 12, 18 ou 20.....

Sortie attendue : Vous êtes mineur ; majeur ; majeur

- Quelque soit l'entrée, le programme ne répond pas et affiche le message :

```
Traceback (most recent call last):
  File "C:\Users\Prof\Desktop\EnsSci\majeur.py", line 2, in <module>
    if age < 18 :
TypeError: '<' not supported between instances of 'str' and 'int'
```

Expliquer l'erreur :

Les valeurs entrées sont des chaînes de caractères et ne peuvent donc pas être comparées à l'entier 18.....

Comment la corriger ?

Il faut lire les caractères de la chaîne pour en déduire un entier. Par exemple juste après la ligne 1 : age = int(age).....

- Proposer un autre test qui ferait encore boguer le programme.

La modification age = int(age) ne permet pas de prendre en compte une entrée non entière comme 12.5.....

De manière générale, il peut être très difficile de tenir compte de toutes les entrées possibles d'un utilisateur. Il est parfois plus simple de laisser l'erreur se produire, de l'intercepter et de la traiter, par exemple en indiquant à l'utilisateur que la valeur entrée n'est pas conforme.

Mettre en place un jeu de tests le plus exhaustif possible n'est par ailleurs pas suffisant pour prouver qu'un programme n'est pas boguer. Des preuves plus formelles (et plus complexes à mettre en place) existent.

Instructions conditionnelles

Radar automatique

Sur autoroute, la vitesse pour les véhicules légers (VL) est limitée à 130 km.h^{-1} et pour les poids lourds (PL) à 90 km.h^{-1} . Un radar possède au moins 2 capteurs permettant de mesurer la vitesse et la hauteur des véhicules circulant sur l'autoroute.

On propose le programme suivant :

```

1 def Amende(h,v):
2     if h>2 and v>90:
3         enFaute = True
4     if h<=2 and v>130:
5         enFaute = True
6     else :
7         enFaute = False
8     return enFaute

```



- Que représentent les variables h et v ?

h représente la vitesse et h la hauteur des véhicules.....

- Que représentent les valeurs 2 ; 90 et 130 ?

la valeur 2 représente la limite de hauteur en mètre entre VL et VL 90 et 130 les limites de vitesses.....

- Quelle est le type de la variable « `enFaute` » ?

La variable « enFaute est de type Booléen (attention, `type(enFaute)` renvoie une erreur : variable locale à la fonction).

- Quelles sont les réponses à `Amende(3,100)` et `Amende(1.8,110)` ?

Dans les deux cas : False.....

- Expliquer et corriger l'erreur de programmation qui engendre ce « bug ».

Le `else` ne s'applique qu'au second `if`. Un VL en faute fera donc passer la variable `enFaute` à `True` dans le premier puis à `False` dans le second. On peut corriger en remplaçant le second `if` par `elif`.....

Forfait de ski

Dans la Station de ski de LaRoche, un forfait journée, coûte 35 €. Les enfants de moins de 5 ans ne paient pas pour skier. Ceux de 6 à 17 ans et de plus de 65 ans ont une remise de 20%.

- Proposer un ensemble de tests qui permettront de vérifier le bon fonctionnement d'un programme qui déterminerait le prix du forfait journée à payer en fonction de l'âge.

Il faudrait par exemple tester les âges autour des changements.....

5 ⇒ 0 ; 6 ⇒ 28 ; 17 ⇒ 28 ; 18 ⇒ 35 ; 64 ⇒ 35 ; 65 ⇒ 28.....

- Compléter les programmes suivants, rédigés sous Python, pour déterminer le prix du forfait à payer en fonction de l'âge.

```

1 def prix_forfait_1(age):
2     if age<6:
3         prix = 0
4     else :
5         if age<18 or age>64:
6             prix = 28
7         else :
8             prix = 35
9     return prix

```

```

1 def prix_forfait_1(age):
2     if age<6:
3         prix = 0
4     elif age>17 and age<65 :
5         prix = 35
6     else :
7         prix = 28
8     return prix

```

```

1 def prix_forfait_3(age):
2     prix = 28
3     if 18<=age<=64:
4         prix = 35
5     if age<=5:
6         prix = 0
7     return prix

```

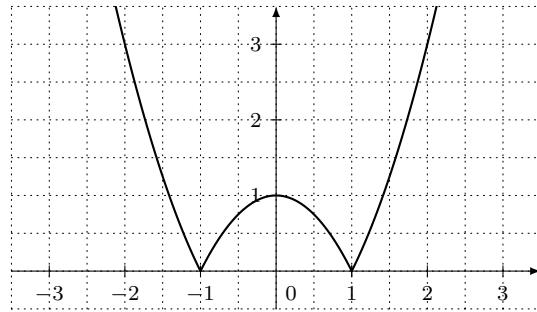
Boucles

Dans cette partie, on considère le programme ci-dessous, d'une fonction mathématique f dont on donne ci-contre une représentation graphique.

```

1 def f(x):
2     y = x*x-1
3     if y < 0 :
4         y = -y
5     return y

```



1. En donner une expression algébrique : $f(x) = \begin{cases} 1 - x^2 & \text{si } x \in [-1; 1] \\ x^2 - 1 & \text{sinon} \end{cases}$ ou $|x^2 - 1|$
2. Étant donné un nombre réel a compris entre 0 et 8, on cherche à déterminer à 0,1 près le plus grand nombre réel x appartenant à $[xmin; xmax]$ et vérifiant $f(x) \leq a$. On propose plusieurs programmes qu'il s'agit de tester et de corriger.

a. Premier programme

Tester le programme ci-contre pour $a = 0,5$, $xmin = -1$ et $xmax = 0$.

* Que constate-t-on ?

La valeur renvoyée x=-0.7 donne f(x)<=a

* Expliquer l'erreur.

La boucle s'arrête quand f(x) n'est plus <=a

On ajoute donc 0.1 de trop

```

1 def solution1(a, xmin, xmax):
2     x = xmin
3     while f(x)<=a :
4         x = x + 0.1
5     return x

```

b. Pour corriger ce programme, on modifie la valeur à renvoyer :

Donner les valeurs (à 0,01 près) renvoyées par :

- * $\text{solution2}(0.5, -1, 0)$ *-0.6*
- * $\text{solution2}(0.5, -1, 1)$ *-0.6*
- * $\text{solution2}(0.5, -1, 3)$ *-0.6*

Expliquer l'erreur de ce programme.

Il s'arrête à la première des valeurs ne donnant pas f(x) <= a

Il devrait s'arrêter à la dernière donnant f(x) <= a

```

1 def solution2(a, xmin, xmax):
2     x = xmin
3     while f(x)<=a :
4         x = x + 0.1
5     return x - 0.1

```

c. Compléter le programme ci-dessous pour qu'il réponde au problème initial.

```

1 def solution3(a, xmin, xmax):
2     x = xmin
3     solution = None
4     while x <= xmax :
5         if f(x)<=a :
6             solution = x
7         x = x + 0.1
8     return solution

```

d. Que penser du programme suivant ?

```

1 def solution4(a, xmin, xmax):
2     x = xmax
3     while f(x)>a :
4         x = x - 0.1
5     return x

```

Il répond correctement au tests précédents, mais il vaut mieux éviter de faire une demande avec un a négatif

Intelligence artificielle

L'intelligence artificielle, en tant que domaine de recherche, est née officiellement en 1956, expression proposée par un chercheur américain, John McCarthy. L'article fondateur est celui de Alan Turing en 1950 « Computing machinery and intelligence ». Il y propose de considérer la question suivante : « Est-ce que les machines peuvent penser ? ». Il propose un test, dit « jeu de l'imitation », passé à la postérité sous le nom de « test de Turing » : un ordinateur passe le test s'il arrive à se faire passer pour un interlocuteur humain pendant quelques minutes d'un jeu de questions-réponses.

Qu'est-ce que l'intelligence artificielle ?

Citer, d'après les documents suivants, les grands enjeux de l'intelligence artificiels et la façon dont ils sont traités.

Reconnaissance d'images, de textes, de sons, de phrases.....

Production de réponses adaptées à un interlocuteur ; dans un jeu ; pour conduire une voiture ; faire un diagnostic médical..

Longtemps basés sur des règles expertes, le Big Data et les cartes graphiques modernes ont favorisé la programmation d'apprentissages.

document 1 - Les enjeux de la recherche en intelligence artificielle - Y. Lecun - 2016

Qu'est-ce que l'intelligence ? Est-ce la capacité à percevoir le monde, à prédire le futur immédiat ou lointain, ou à planifier une série d'actions pour atteindre un but ? Est-ce la capacité d'apprendre, ou celle d'appliquer son savoir à bon escient ? La définition est difficile à cerner. On pourrait dire que l'intelligence artificielle (IA) est un ensemble de techniques permettant à des machines d'accomplir des tâches et de résoudre des problèmes normalement réservés aux humains et à certains animaux.

Les tâches relevant de l'IA sont parfois très simples pour les humains, comme par exemple reconnaître et localiser les objets dans une image, planifier les mouvements d'un robot pour attraper un objet, ou conduire une voiture. Elles requièrent parfois de la planification complexe, comme par exemple pour jouer aux échecs ou au Go. Les tâches les plus compliquées requièrent beaucoup de connaissances et de sens commun, par exemple pour traduire un texte ou conduire un dialogue.

Depuis quelques années, on associe presque toujours l'intelligence aux capacités d'apprentissage. C'est grâce à l'apprentissage qu'un système intelligent capable d'exécuter une tâche peut améliorer ses performances avec l'expérience. C'est grâce à l'apprentissage qu'il pourra apprendre à exécuter de nouvelles tâches et acquérir de nouvelles compétences.

Le domaine de l'IA n'a pas toujours considéré l'apprentissage comme essentiel à l'intelligence. Par le passé, construire un système intelligent consistait à écrire un programme « à la main » pour jouer aux échecs (par recherche arborescente), reconnaître des caractères imprimés (par comparaison avec des images prototypes), ou faire un diagnostic médical à partir des symptômes (par déduction logique à partir de règles écrites par des experts). Mais cette approche « manuelle » a ses limites.

document 2 - Computing machinery and intelligence - A. Turing - 1950

Au lieu de produire un programme qui simule l'esprit d'un adulte, pourquoi ne pas plutôt essayer d'en produire un qui simule celui de l'enfant ? S'il était alors soumis à une éducation appropriée, on aboutirait au cerveau humain. [...] Nous avons en conséquence divisé le problème en deux parties : le programme-enfant et le processus d'éducation.

[...]

Nous pouvons espérer que les machines concurrenceront finalement l'homme dans tous les champs purement intellectuels. Mais par lesquels vaut-il mieux commencer ? Même cette décision est difficile à prendre. Beaucoup de gens pensent qu'une activité très abstraite comme le jeux d'échec serait la meilleure. On peut aussi soutenir qu'il vaut mieux équiper la machine avec les meilleurs organes sensoriels que l'on puisse acheter, puis lui apprendre à comprendre et à parler anglais. Ce processus pourrait se conformer à l'enseignement normal d'un enfant. On lui montrerait et nommerait des objets, etc.

Exemple d'intelligence artificielle sur des règles d'experts

L'algorithme MinMax

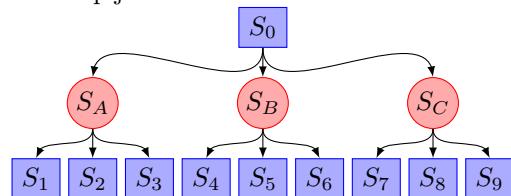
Il est basé sur l'évaluation des situations de jeux en terme de score pour le joueur dont c'est le tour. Il suppose que ce joueur cherchera toujours à maximiser ce score alors que son adversaire cherchera à le minimiser.

Étape 1 : on construit l'arbre des possibles.

situation alors que c'est au tour du joueur.

situation alors que c'est à son adversaire.

→ coup joué amenant à une nouvelle situation.



Étape 2 : On évalue le score des situations de jeux du dernier niveau. (Plus il est élevé plus la situation est favorable au joueur.) Imaginons obtenues les évaluations suivantes :

7 8 3 2 9 1 6 4 5

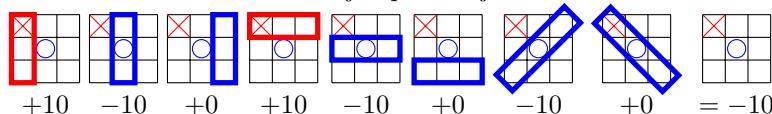
Les étapes 1 et 3 sont les mêmes, quelque soit le jeu choisi. L'étape 2 est plus délicate, car l'évaluation du score peut être subjectif. Par exemple : combien vaut un pion comparé à une dame aux échecs ? On a donc, besoin de l'avis d'un expert.

Application au morpion

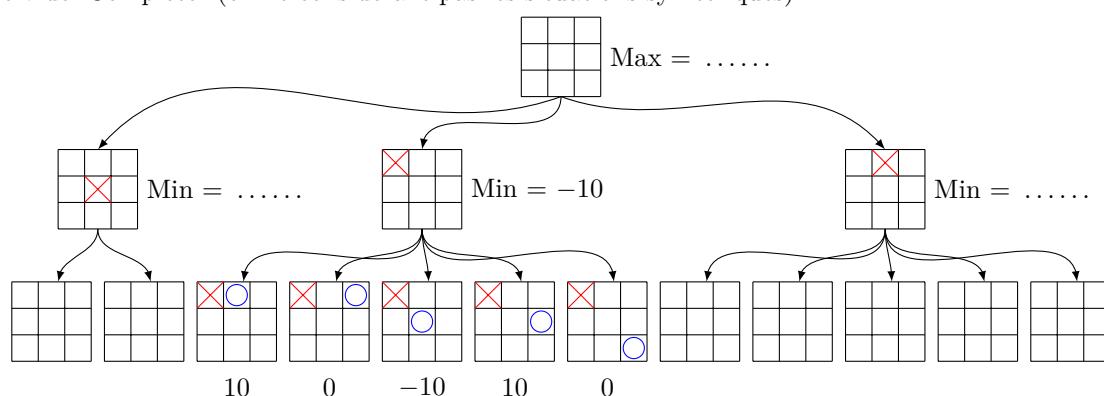
Après consultation d'un expert, on met en place la fonction de score suivante :

- ★ 3 marques identiques alignées valent 1000 points.
- ★ 2 marques identiques alignées avec un case libre valent 100 points.
- ★ 1 marque alignée avec 2 cases libres valent 10 points.
- ★ Les autres alignements valent 0 point.
- ★ Ces points sont comptés positivement si ce sont les marques du joueur, négativement sinon.

Exemple d'une évaluation d'une situation de jeu pour le joueur



On se propose de déterminer ce que serait le premier coup du MinMax avec cette évaluation dans une grille vide. Compléter (en ne considérant pas les situations symétriques) :



L'apprentissage machine

Aussi connu sous le nom d'apprentissage artificiel ou d'apprentissage automatique, il consiste en des programmes capables d'ajuster leur comportement à des données dites d'entraînement. On distingue communément trois modalités principales d'apprentissage.

- ★ L'apprentissage **supervisé**, consiste à apprendre à une machine à catégoriser des données à partir d'un grand nombre de données préalablement étiquetées par l'Homme. Pendant la phase d'apprentissage, on calibre un algorithme en adaptant ses paramètres de catégorisation aux données fournies. On utilise ensuite cet algorithme pour catégoriser de nouvelles données.
- ★ L'apprentissage **non supervisé**, consiste à fournir à la machine un grand nombre données, non étiquetées. La machine y repère des régularités, des proximités, des corrélations pour construire elle-même son algorithme de classification.
- ★ L'apprentissage **par renforcement** consiste à induire le comportement d'un « agent » (par exemple un robot) évoluant dans un « environnement » (par exemple dans une pièce où il doit manipuler des objets) et qui apprend grâce à un système de « récompenses ».

Les exemples développés ici sont issues de l'apprentissage supervisé.

Ajustement linéaire

En mathématiques, un ajustement linéaire est la détermination d'une droite approchant au mieux un nuage de points dans le plan.

Méthode de Mayer

Dans le cas où les abscisses sont différentes deux à deux, elles peuvent être ordonnées et la Méthode de Mayer (parmi d'autres) consiste à partager le nuage de points en deux de tailles égales selon la valeur de leurs abscisses (les petites d'une part, les grandes d'autre part). La droite retenue est alors celle qui passe par le point moyen de chacun des deux nuages.

1. Justifier qu'il s'agit d'un algorithme d'apprentissage supervisé.

Les données fournies sont *les abscisses avec comme « catégories » les ordonnées des points du nuage*.....

Adapter les paramètres, c'est *déterminer le coefficient directeur et l'ordonnée à l'origine de la droite*.....

Catégoriser de nouvelles données, c'est *associée à chaque autre abscisse l'ordonnée correspondante sur la droite*.

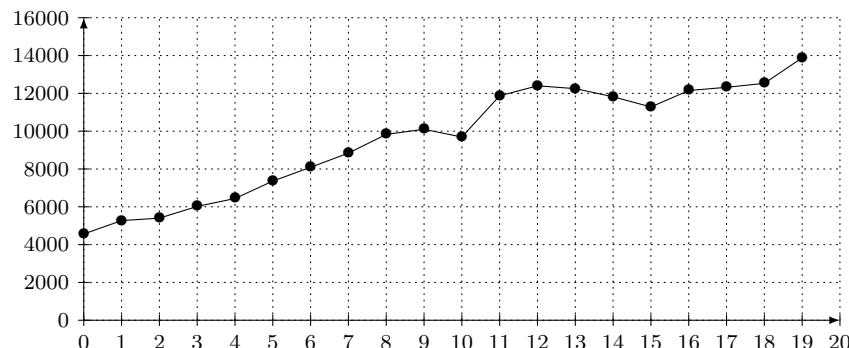
2. Appliquer cette méthode dans le contexte suivant : prix du m² dans le 6^e arrondissement de Paris.

Année 200x _i	0	1	2	3	4	5	6	7	8	9
Prix au m ² y _i	4562	5270	5400	6020	6460	7360	8090	8840	9830	10100

Année 200x _i	10	11	12	13	14	15	16	17	18	19
Prix au m ² y _i	9690	11870	12400	12250	11820	11280	12150	12320	12530	13880

Points moyens : *(2004,5; 7193,2) et (2014,5; 12019)*.....

Tracer la droite correspondante sur la représentation ci-dessous :



Méthode des moindres carrés

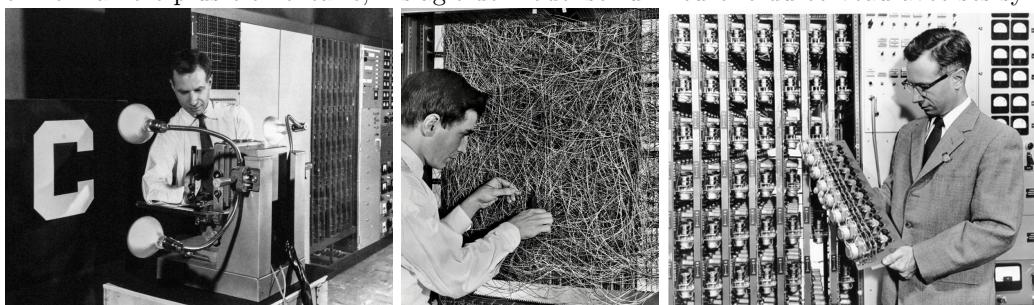
C'est une méthode statistique permettant de trouver la « meilleure » droite, selon certains critères. Dans l'exemple précédent, elle donne : $y = 473,49x + 5\,107,9$.

À quel prix au m² peut-on s'attendre pour 2020 ? *473,49 × 20 + 5\,107,9 ≈ 14578*.....

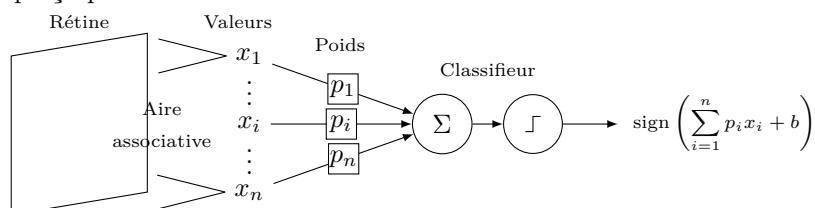
Le Perceptron

La machine

L'invention du perceptron suit l'idée de A. Turing de construire un jeune cerveau que l'on puisse instruire. De manière plus élémentaire, il s'agit de modéliser un neurone du cerveau avec ses synapses.



La machine est constituée d'une rétine pour de percevoir une image simple (à gauche ; très basse résolution : quelques pixels), d'une aire associative permet de fournir les valeurs d'entrée (au milieu ; les synapses) le tout relié à un classifieur (le neurone) qui fait la somme de ces valeurs après les avoir multipliées par des poids (donnés par des potentiomètres motorisés ; à droite) et donnant en sortie la classe de l'image perçue par une fonction en escalier.



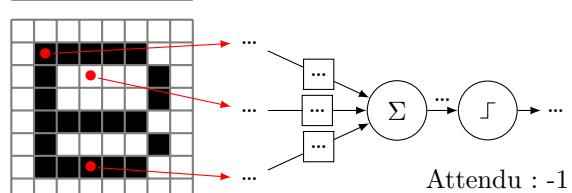
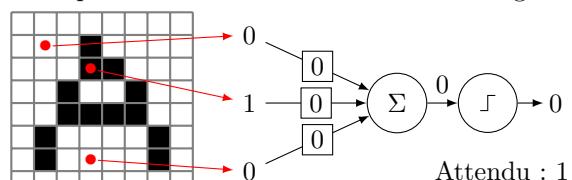
L'apprentissage consiste alors en la modification automatique des poids selon la réponse.

Un exemple : distinguer un « A » d'un « B »

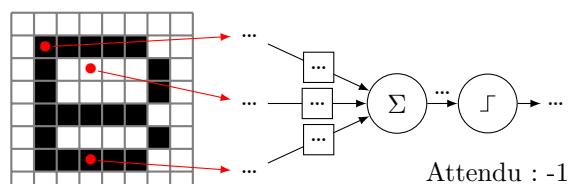
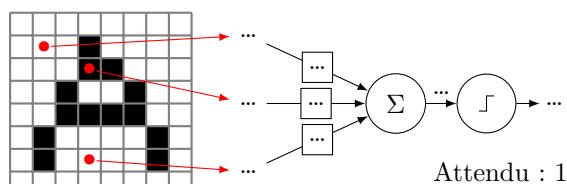
On ne considère que 3 synapses pour simplifier. Voyons comment le Perceptron apprend à reconnaître un « A » d'un « B ». L'entrée « A » doit renvoyer une sortie positive et l'entrée « B » une sortie négative.

On choisit un réglage de départ neutre : chaque poids à 0. On présente l'image d'un « A », pour laquelle on associe blanc à 0 et noir à 1, avec l'indication de la sortie attendue : 1. On obtient le schéma ci-contre.

1. Comme la sortie est inférieure à l'attendu, le Perceptron augmente chacun des poids correspondants à une case noire d'une unité. Reporter les nouveaux poids sur le schéma ci-contre et compléter par rapport à la situation.



2. Comme la sortie est supérieure à l'attendu, le Perceptron diminue chacun des poids correspondants à une case noire d'une unité. Reporter les nouveaux poids sur les schémas ci-dessous et vérifier que les deux tests sont bien validés.



Source de l'activité : Yann Lecun - Lecture inaugurale au Collège de France en 2016.

<https://www.college-de-france.fr/site/yann-lecun/inaugural-lecture-2016-02-04-18h00.htm>

L'inférence bayésienne

L'inférence bayésienne fait référence au révérend Thomas Bayes, mathématicien et pasteur britannique né à Londres aux environs de l'année 1702 et mort en 1761. Ses découvertes en probabilités ont été résumées dans son « Essai sur la manière de résoudre un problème dans la théorie des risques » publié à titre posthume en 1763. On lui doit notamment le théorème de Bayes, très utilisé dans tout ce qui relève du classement automatique (diagnostic médical, filtrage de spams).

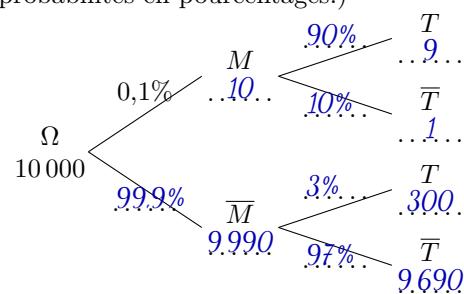
Diagnostic médical

Une personne vient de passer un test de dépistage d'une maladie rare. On sait qu'elle ne touche que 0,1 % de la population. Le médecin lui annonce que le résultat du test est positif. La personne demande alors au médecin si le test est fiable. Sa réponse est sans appel : « Si vous êtes malade, le test est positif dans 90 % des cas et si vous n'êtes pas malade, il est négatif dans 97 % des cas. »

Problème posé : quelle est la probabilité que cette personne soit effectivement malade ?

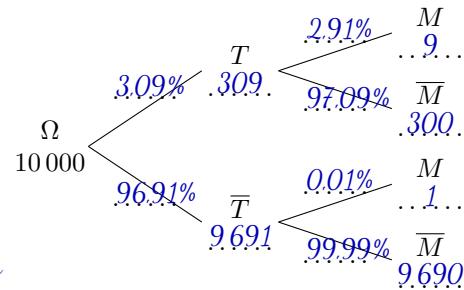
1. Compléter la représentation par un tableau : (pour 10 000 personnes.)
2. Compléter la représentation par un arbre : (probabilités en pourcentages.)

Test →	positif T	négatif \bar{T}	Total
Malade ↓			
oui : M	9.....	1.....	10....
non : \bar{M}	300..	9.690	9.990
Total	309..	9.691	10 000



On constate que l'arbre donne une importance prépondérante au fait d'être malade ou non par rapport au fait d'avoir un test positif ou négatif, alors que le tableau semble plus symétrique.

3. À l'aide des données du tableau, compléter l'arbre inversé ci-contre en arrondissant les probabilités à 10^{-4} près.
4. Quelle est la probabilité qu'une personne positive au test soit effectivement malade ?
environ 3%
5. Pourquoi est-ce contre-intuitif?
Comme un test est positif pour 90 % des malades et négatif pour 97 % des malades, cela laisse penser qu'il était efficace...
6. Pourquoi ce phénomène contre-intuitif se produit-il ?
Parce que la maladie est rare. On pourrait poursuivre l'étude pour déterminer l'influence de cette rareté sur le diagnostic.



Détection de spams

Un des premiers programmes de filtrage bayésien du courrier électronique était le programme iFile de Jason Rennie, publié en 1996. Le principe, analogue à celui du diagnostic médical, repose sur le fait que les mots du dictionnaire ont des probabilités différentes d'apparaître dans les spams et dans les courriers légitimes.

Le filtre de détection des spams ne connaît pas à l'avance les probabilités d'apparition de ces mots, c'est pourquoi il lui faut une phase d'apprentissage pour les évaluer. Il se fait à partir de l'observation du comportement des utilisateurs, qui doivent indiquer manuellement si un message est un spam ou non.

Dans la réalité, on se trompe en faisant l'hypothèse naïve que les mots présents dans un message sont indépendants les uns des autres. Par exemple la probabilité de trouver un adjectif est influencée par celle de trouver un nom. De plus, cette technique, connue sous le nom de filtrage bayésien naïf, ne tient pas compte du sens des mots, alors qu'il a une incidence sur la présence simultanée de certains mots à l'intérieur du message (la présence du mot « anniversaire » n'est pas indépendante de celle du mot « joyeux »).