Propositions pour le futur programme de mathématiques du lycée

Groupe de travail des
sociétés savantes de mathématiques et d'informatique.

Société française de statistique, SFdS
Société informatique de France, SIF
Société de mathématiques appliquées et industrielles, SMAI
Société mathématique de France, SMF.

21 octobre 2016

Quatre sociétés savantes de mathématiques et d'informatique font des propositions pour le futur programme de mathématiques du lycée

Les différentes sciences que l'on enseigne dans le premier et le second degré (les mathématiques, l'informatique, la physique, la biologie, etc.) sont autonomes, car elles étudient des objets différents, par des méthodes également différentes. Mais ces différentes sciences ne sont pas indépendantes : leur histoire est largement commune, faite de continuels emprunts des unes aux autres. Dans leurs développements contemporains, les recherches interdisciplinaires (physique mathématique, bioinformatique, etc.) jouent un rôle central.

Pour rendre l'enseignement des sciences fidèle à ce qu'elles sont devenues aujourd'hui, il est important de montrer la richesse de ces liens que le nécessaire découpage en disciplines peut parfois trop masquer (avant le bac, comme après).

Les sociétés savantes ont un rôle essentiel à jouer dans la proposition de contenus d'enseignement qui montrent l'unité des sciences. C'est dans cet esprit que la SFdS – Société française de statistique, la SIF – Société informatique de France, la SMAI – Société de mathématiques appliquées et industrielles, et la SMF – Société mathématique de France, ont constitué un groupe de travail pour réfléchir à ce que pourrait être un programme de mathématiques liées à l'informatique au sein du programme de mathématiques du lycée.

Mathématiciens et informaticiens sont naturellement concernés par ces questions : il s'agit bien de définir des contenus d'enseignement qui concernent à la fois les mathématiques et l'informatique. Il ne s'agit nullement de chercher à définir une frontière entre les deux disciplines mathématiques et informatique, mais à identifier un périmètre ouvert à l'intersection des deux disciplines dont l'enseignement sera profitable en tant que tel aux lycéens, et sera une base pour des prolongements en mathématiques, en informatique, et plus généralement en sciences.
Les programmes de mathématiques pourraient par exemple intégrer des notions telles que des graphes, de la combinatoire ou de la logique, qui relèvent d'abord des mathématiques et sont aussi des notions fondamentales en informatique. La modélisation, déjà abordée au collège, devra être approfondie au lycée. Il faudra également sensibiliser les lycéens aux problèmes mathématiques et informatiques liés aux calculs approchés sur machine (complexité et coût, stockage, compression de données...).

Aujourd'hui, la situation est la suivante

- le nouveau programme du collège en vigueur à la rentrée 2016 intègre un enseignement de l'algorithmique et de la programmation au sein du programme de mathématiques ;
- programme de mathématiques du lycée aborde lola notion d'algorithme dans un périmètre bien restreint et qui doit donc évoluer ;
- enseignement d'ISN – Informatique et science du numérique, spécialité de terminale S, introduit il y a quelques années dans certains lycées, est majoritairement enseigné par des professeurs de mathématiques ;
- un enseignement exploratoire d'informatique et création numérique (ICN) est instauré en classe de seconde depuis la rentrée 2015, et un enseignement facultatif d'informatique et création numérique en classes de première (sections S, ES et L) et en classes de terminale (sections L et ES) est proposé dans certains lycées à tous les élèves depuis la rentrée 2016.

Les connaissances et compétences en science informatique des lycéens vont rendre possible l'enseignement de mathématiques jusqu'alors absentes des programmes en s'appuyant sur des savoirs et savoir-faire informatiques, et vont aider à l'enseignement de ces mathématiques et plus généralement des mathématiques, dans une approche expérimentale.

Cette proposition, résultat de travaux menés pendant un an par le groupe de travail, s'articule

autour de quatre thèmes : logique, graphes, combinatoire, représentation et modélisation de l'information. Sur chacun de ces thèmes, nous avons repéré les notions mathématiques nécessaires pour l'informatique qu'il convient d'enseigner au lycée. Cette proposition vise aussi à montrer les apports de la discipline informatique aux mathématiques. Tout cela sans perdre de vue que les mathématiques « liées » à l'informatique sont des mathématiques et peuvent être étudiées pour elles-mêmes. Des exemples d'exercices sur chacun des thèmes sont proposés.

Notons que pour une mise en place de cette proposition, dans le cadre d'une réforme des programmes de lycée, il sera essentiel que soit organisée la formation initiale et continue des enseignants aux mathématiques requises, mathématiques qu'ils n'ont souvent pas abordées pendant leurs études.

Les programmes de lycée vont changer puisque les nouveaux programmes de collège sont appliqués depuis la rentrée 2016. Les changements à apporter doivent tenir compte de l'arrivée officielle de l'informatique au collège et au lycée (ICN, ISN).

Les quatre sociétés savantes de mathématiques et d'informatique sont concernées [1] au premier chef par les questions sur l'enseignement de ces disciplines. Elles proposent donc ce texte comme une réflexion en amont sur les programmes de mathématiques au lycée pour contribuer utilement à l'évolution de ces programmes.

Société française de statistique, SFdS
Société informatique de France, SIF
Société de mathématiques appliquées et industrielles, SMAI
Société mathématique de France, SMF.

21 octobre 2016

1. Ces 4 sociétés savantes ont élaboré ensemble avec Femmes & Mathématiques et l'ONISEP (et le soutien du CNRS, INSMI et INS2I, d'INRIA...) le « Zoom des métiers des mathématiques et de l'informatique », qu'on peut télécharger à http ://metiers-mathsinfo.fr/

Résumé des propositions pour le futur programme de mathématiques du lycée

Le présent document élaboré par le groupe de travail présente, pour chacun des quatre domaines identifiés, les notions clés qui lui semblent pouvoir et devoir être enseignées dès le lycée, ainsi que quelques exercices permettant de les illustrer, ceci indépendamment de la série et du devenir des lycéens. La démarche envisagée vise à introduire des notions fondamentales à partir d'exemples simples et d'expérimentations.

Sur le thème de la logique, l'objectif est que les élèves aient compris à la fin du lycée les notions de démonstration – notion fondamentale à la base de tout raisonnement scientifique –, et aussi d'énoncé logique (propositionnel et avec quantificateurs) et de validité dans un modèle. Ces notions seront introduites à partir d'exemples concrets adaptés à la section des élèves, en montrant leur intérêt en informatique (par exemple, connecteurs utilisés dans les instructions conditionnelles). Nous proposons de commencer par la notion de démonstration, en utilisant des exemples de règles de déduction portant sur des objets divers, par exemple, en utilisant des règles de recettes de cuisine pour montrer qu'à partir de certains ingrédients on peut obtenir certains gâteaux, et aussi que certains gâteaux sont impossibles à obtenir sans certains ingrédients, ou encore que certains scores ne peuvent jamais être obtenus au rugby. Cette démarche permettra d'aborder la notion de démonstration, et les notions corollaires de démontrabilité et de non-démontrabilité, dans toute leur généralité, sans avoir à traiter en même temps les difficultés introduites par la notion de proposition.

Sur le thème des graphes, notre proposition est que l'enseignement puisse s'appuyer sur des exemples de problèmes à modéliser, puis introduise des algorithmiques « simples » sur les graphes à l'occasion de la résolution de problèmes mis en évidence par ces exemples, et enfin passe à une implémentation à l'aide de bibliothèques ad hoc. Il convient de travailler sur des jeux de données de taille conséquente pour mettre en évidence la « complexité » des algorithmes (faire la différence entre les problèmes qui sont réellement de forte complexité et pour lesquels les machines n'arrivent pas au bout des calculs en un temps raisonnable et ceux pour lesquels on a trouvé de mauvaises solutions). Pour les lycéens qui se destineraient a priori vers des études scientifiques, des définitions de structures de données ad hoc pour la représentation de graphes peuvent être envisagées. De manière générale, le programme actuel de la spécialité mathématique(s) de terminale ES a servi de base de départ pour cette proposition.

Sur le thème de la combinatoire, l'objectif est d'introduire les notions et savoir-faire permettant de compter les éléments d'ensembles finis, et de les transférer au calcul élémentaire de complexité en temps d'algorithmes. Les notions mathématiques introduites couvrent les couples et n-uplets, leur dénombrement, la fonction factorielle et le triangle de Pascal. Il sera ainsi possible de traiter d'identités remarquables et de probabilités, de compter des objets informatiques, et de distinguer complexité en temps d'un algorithme et complexité intrinsèque d'un problème. Ces enseignements reposent largement sur une approche mathématique expérimentale dans laquelle le développement de programmes informatiques amène la découverte des notions, leur acquisition, en confrontant résultats expérimentaux et concepts mathématiques.

Sur le thème de la représentation et modélisation de l'information, notre proposition est de traiter de problématiques au cœur du monde numérique. Ces problématiques permettent de relier des notions mathématiques importantes (représentation des nombres, combinatoire,

probabilité, algèbre linéaire) aux applications concrètes de la compression de données et des codes correcteurs d'erreurs. La représentation approchée des nombres réels et son impact sur la stabilité des calculs numériques seront également abordés. L'informatique permet d'appliquer ces concepts sur des données de grande taille, afin de mieux saisir leur importance (à travers le gain en taille du fichier ou en robustesse aux erreurs).

Groupe de travail

Le groupe de travail constitué de : Olivier Baudon (SIF), Jean-Pierre Borel (SMF), Gilles Dowek (SIF), Christine Froidevaux (SIF), Camelia Goga (SfdS), François Jouve (SMAI), Pierre Loidreau (SMF), Roger Mansuy (SMF), Philippe Marquet (SIF), Gabriel Peyré (SMAI), Aviva Szpirglas (SMF) s'est réuni une dizaine de fois depuis septembre 2015.

Il est co-animé par Aviva Szpirglas <`Aviva.Szpirglas@math.univ-poitiers.fr`> et Philippe Marquet <`Philippe.Marquet@univ-lille1.fr`>.

Table des matières

Chapitre 1

Logique

Sur le thème de la logique, l'objectif est que les élèves aient compris à la fin du lycée les notions de démonstration – notion fondamentale à la base de tout raisonnement scientifique –, et aussi d'énoncé logique (propositionnel et avec quantificateurs) et de validité dans un modèle. Ces notions seront introduites à partir d'exemples concrets adaptés à la section des élèves, en montrant leur intérêt en informatique (par exemple, connecteurs utilisés dans les instructions conditionnelles).

Nous proposons de commencer par la notion de démonstration, en utilisant des exemples de règles de déduction portant sur des objets divers, par exemple, en utilisant des règles de recettes de cuisine pour montrer qu'à partir de certains ingrédients on peut obtenir certains gâteaux, et aussi que certains gâteaux sont impossibles à obtenir sans certains ingrédients, ou encore que certains scores ne peuvent jamais être obtenus au rugby. Cette démarche permettra d'aborder la notion de démonstration, et les notions corollaires de démontrabilité et de non-démontrabilité, dans toute leur généralité, sans avoir à traiter en même temps les difficultés introduites par la notion de proposition.

1.1 Notions mathématiques

1.1.1 Démonstration (classe de Seconde)

Ce premier chapitre permet d'aborder la notion de démonstration d'un point de vue relativement intuitif en se reposant sur les structure sous-jacente (arbre) et sans utiliser le formalisme du calcul propositionnel.

- Notions d'axiome, de règles d'inférence
- Démonstration, représentation sous forme d'arbres
  Analogie avec la notion mathématique de démonstration : formalisation de quelques exercices de géométrie de quatrième.
- Chaînage avant et arrière
- Exemple de non démontrabilité
- Fonctions booléennes, construction de circuits booléens
- Application : évaluation des propriétés vraies à la sortie de boucles

1.1.2 Proposition (classe de Première)

— Justifier l'importance de la formalisation des propositions avec l'ambiguïté d'un énoncé en langage naturel
— Énoncés en logique des prédicats (avec quantificateurs mais dans un langage contraint et limité)
Par exemple, exprimer, dans le langage 0, 1, $+$,$\times$, $=$, $\leq$ *les propositions suivantes*
  — *Il existe un nombre supérieur à tout nombre*
  — *tout nombre admet un opposé*
  — *tout nombre admet un opposé unique*
  — *s'il existe un nombre* x *tel que* $x^2 - 1515x + 1 = 0$*, alors* x *est positif*
— Exemple de démonstrations dont les nœuds sont des propositions de la logique des prédicats

1.1.3 Modèle (classe de Terminale)

— Modèle, évaluation, vérité dans un modèle
Par exemple, la proposition "pour tout x*, il existe* y*, tel que* xRy*" est vraie et la proposition "il existe* y*, pour tout* x*, tel que* xRy*" est fausse dans le modèle* $(\mathbf{N}, \leq)$*; de même, la proposition "pour tout* x*, il existe* y*, tel que* $x * yR0$*" est vraie dans* $(\mathbf{Z}, 0, +, =)$ *mais pas dans* $(\mathbf{N}, 0, +, =)$ *ni dans* $(\mathbf{Z}, 1, \times, =)$
— Distinguer "démontrable" et "vrai dans un modèle"
— Application : évaluation d'une proposition dans un modèle fini (par exemple, le calcul booléen rentre dans ce cadre).

1.2 Pertinence

L'objectif de ces trois chapitres est de permettre de mettre en évidence la notion de démonstration aussi bien en informatique (avec pour objectif des idées sous-jacentes la correction/terminaison de programme) qu'en mathématique.

1.3 Exercices

1.3.1 Exercice : Démonstration

1. On cherche à étudier les scores au rugby où les points sont marqués par 3, 5 et 7 points. Pour cela on introduit quatre axiomes $R(0)$, $R(3)$, $R(5)$, $R(7)$ et une règle d'inférence :

$$\frac{R(a) \qquad R(b)}{R(a+b)}$$

   (a) Donner une preuve de $R(18)$ (c'est-à-dire, l'équipe a marqué 18 points). Cette preuve est-elle unique ?
   (b) Peut-on démontrer $R(17)$?
2. Reprendre la question avec les scores 4, 6, 8.

1.3.2 Exercice : chaînage avant et arrière

Pour cuisiner, on dispose d'ingrédients (les axiomes) et de recettes de cuisine (les règles d'inférence).

Voici la liste des recettes :
— Si farine et beurre alors pate-tarte
— Si pate-tarte et poires alors tarte-poires
— Si pate-tarte et pommes alors tarte-pommes
— Si pate-tarte et chocolat et crème-fraîche et œufs alors tarte-chocolat
— Si tarte-chocolat et poires alors tartes-poires-chocolat
— Si farine et beurre et sucre alors crumble
— Si crumble et pommes alors crumble-pommes
— crumble et poires alors crumble-poires
— Si farine et œufs et lait alors crêpes
— Si crêpes et chocolat alors crêpes-chocolat
— Si crêpes et pommes alors crêpes-pommes
— Si crêpes et poires alors crêpes-poires
— Si crêpes-poires et chocolat alors crêpes-poires-chocolat
— Si farine et sucre et œufs et beurre et chocolat alors brownie
— Si farine et sucre et œufs et beurre et levure alors cake
— Si cake et chocolat et poudre-amande alors cake-chocolat
— Si cake et poires alors cake-poires
— Si cake et pommes alors cake-pommes
— Si cake-poires et chocolat alors cakes-poires-chocolat

1. Indiquer la liste des gâteaux que je peux réaliser si je dispose de : farine, œufs, lait, beurre, sucre, chocolat ;
2. Je veux faire un cake aux poires chocolat, de quoi ai-je besoin ?
3. Démontrer que je ne peux réaliser de brownie sans chocolat.

1.3.3 Exercice : Démonstration de proposition

Dans cet exercice, on considère un triangle non plat ABC. On cherche à écrire des propositions en utilisant les points du plan et l'égalité des distances. Par exemple, la proposition "le point M est égal à A" s'écrit $AM = AA$.

1. Écrire la proposition : "le point M appartient à la médiatrice de $[A, B]$".
2. Écrire la proposition : "les médiatrices du triangle ABC sont concourantes".
3. Démontrer cette dernière proposition avec la règle d'inférence : "si $OM = ON$ et $ON = OP$, alors $OM = OP$" et l'axiome d'Euclide sur les droites sécantes.

1.3.4 Exercice : Vérité d'une proposition

Trois personnes retrouvées sur le lieu d'un cambriolage font les déclarations suivantes :

Julia : Léo est coupable et Dominique est innocent.

Léo : Si Julia est coupable, alors Domonique l'est aussi.

Dominique : Je suis innocent, mais au moins l'un des deux autres est coupable.

On utilise les notations suivantes pour exprimer les formules :
JC : Julia est coupable ; LC : Léo est coupable ; DC : Dominique est coupable.

1. Traduire les déclarations en formules du calcul propositionnel.
2. Évaluer le nombre d'interprétations (situations possibles) pour l'ensemble de formules.
3. Si tous sont innocents, qui a menti ?
4. Si tous disent vrai, qui est coupable ?

Chapitre 2

Graphes

Sur le thème des graphes, notre proposition est que l'enseignement puisse s'appuyer sur des exemples de problèmes à modéliser, puis introduise des algorithmiques « simples » sur les graphes à l'occasion de la résolution de problèmes mis en évidence par ces exemples, et enfin passe à une implémentation à l'aide de bibliothèques *ad hoc*. Il convient de travailler sur des jeux de données de taille conséquente pour mettre en évidence la « complexité » des algorithmes (faire la différence entre les problèmes qui sont réellement de forte complexité et pour lesquels les machines n'arrivent pas au bout des calculs en un temps raisonnable et ceux pour lesquels on a trouvé de mauvaises solutions).

Pour les lycéens qui se destineraient *a priori* vers des études scientifiques, des définitions de structures de données ad hoc pour la représentation de graphes peuvent être envisagées.

De manière générale, le programme actuel de la spécialité mathématique(s) de terminale ES a servi de base de départ pour cette proposition.

2.1 Introduction

Travailler avec des graphes est un bon moyen pour aborder la notion de modélisation de problèmes. On s'attachera à partir de problèmes concrets pour aller de la modélisation jusqu'à l'algorithme, son implémentation et l'exécution du programme. Des exemples d'algorithmes sur des graphes permettent d'introduire la notion intuitive de complexité temporelle (lorsque les machines n'arrivent pas au bout des calculs en un temps raisonnable).

Les graphes ne seraient abordés qu'à partir de la classe de Première ; ce qui suit concerne deux années (Première et Terminale).

2.2 Introduction des graphes sur des exemples simples de modélisation

On repérera à travers des exemples aussi bien physiques que dématérialisés la notion abstraite de graphe et l'on reconnaitra que des mêmes questions se retrouvent dans des problèmes a priori très différents.

Voici une liste indicative d'exemples.

— Recherche d'un chemin dans un réseau physique (carte de métro, réseau des rivières, réseaux d'alimentation électrique...) et graphe sous-jacent.

— Problème du loup, de la chèvre et du chou.

— Jeu de dominos (comportant toutes les paires de chiffres de 0 à n une et une seule fois) : rechercher si il est possible de les aligner. Plus généralement, graphes de mots (au sens de de Bruijn) avec des applications pour le traitement de texte, la génomique...
— Combien de créneaux pour organiser une série d'examens, à partir de la liste des examens ayant des élèves en commun ?
— Combien de couleurs pour colorier une carte géographique.
— Ordonnancement de tâches.
— Dans un groupe de personnes, il y en a toujours au moins deux qui ont le même nombre d'amis dans le groupe.
— Étude du graphe des amis sur Facebook : densité, diamètre, sommet central.
— Automates simples : reconnaître un nombre multiple de 10, multiple de 3, un mot ayant un nombre pair de lettres 'a'.
— Exemple de graphe probabiliste (l'allumeur de réverbère : au jour 0, le réverbère est éteint et chaque jour, l'allumeur change l'état du réverbère avec un probabilité de 0,75 ; à l'aide d'un arbre, calculer la probabilité que le réverbère soit allumé le 20ème jour.).
— N élèves doivent choisir parmi k stages ceux qui leur conviennent. Existe-t-il une attribution des stages de telle façon que chaque élève se voit attribuer un stage parmi ceux qu'il a sélectionné ?

Dans cette partie « activités », on introduit des graphes pour chaque exemple. On reprendra ces exemples pour les traiter (pour les S et les ES) au fur et à mesure que les outils nécessaires seront introduits.

2.3 Exemples de résolution de problèmes

Dans chacun des exemples, qui doivent rester simples, on « résoudra »les problèmes jusqu'au bout avec ou sans machine. Si c'est avec machine le professeur est à la manœuvre. Ainsi

— pour le problème des dominos où il faut déterminer s'il existe ou non une chaîne eulérienne dans le graphe construit ce qu'on fait après avoir recherché une chaîne eulérienne en utilisant les outils existants sur machine en faisant varier n (par exemple graphtool) ;
— pour le problème des examens ou du coloriage des cartes géographiques, utiliser des algorithmes connus de coloration de graphe ;
— pour les problèmes d'ordonnancement, de même ;
— montrer, sur des exemples simples, certaines propriétés classiques.
  — Graphe biparti : on peut toujours colorier un graphe avec deux couleurs (répartir les sommets en deux ensembles disjoints A et B tels que toute arête soit entre un sommet de A et un sommet de B) si et seulement si tous les cycles du graphe sont de longueur paire.
  — Couplage dans un graphe biparti au travers de la procédure Admission PostBac : soit $G = (A \cup B, E)$ un graphe biparti. Si pour tout sous-ensemble S de A, le nombre de sommets de B voisins d'au moins un sommet de S est toujours au moins égal à $|S|$, alors il existe un couplage de taille $|A|$ dans G. Cette condition est également nécessaire.

2.4 Principaux concepts à maîtriser

Définitions de base sur les graphes (on reprend le programme de la spécialité mathématiques de la série ES). Les notions seront introduites au fur et à mesure à partir des exemples de modélisation.

— graphes isomorphes
— sommets, sommets adjacents,
— arêtes,
— degré d'un sommet,
— ordre et taille d'un graphe,
— graphe simple (sans boucle ni arête multiple),
— chaîne, cycle, longueur d'une chaîne, d'un cycle,
— graphes orientés : arc, chemin, circuit, degré entrant et sortant,
— listes d'adjacence, de successeurs,
— listes d'incidence,
— graphe complet, graphe connexe,
— chaîne eulérienne (voir l'exemple des dominos),
— matrice d'adjacence associée à un graphe.

2.5 Introduction d'algorithmes simples

On confrontera les élèves aux algorithmes suivants en les reliant aux différents exemples de la partie 3.1 consacré à la modélisation ; ils seront mis en œuvre « à la main »par les élèves.

— Parcours en largeur : vérification de la connexité, calcul du diamètre dans un réseau d'amis comme Facebook, recherche de la solution la plus courte dans le problème du loup, la chèvre et le chou.
— Recherche d'un arbre couvrant minimal (Prim) : réseaux de distibution.
— Recherche du plus court chemin sur un graphe pondéré (Dijkstra) : réseaux de transport.
— Recherche d'une chaîne eulérienne (dominos, ponts de Koenisberg).

2.6 Complexité

On sensibilisera les élèves au fait que deux problèmes en apparence de même nature n'admettent pas forcément des solutions algorithmiques aussi simples.

— coloration d'un graphe avec 2 couleurs ou avec 3 couleurs,
— cycle eulérien ou cycle hamiltonien.

A travers quelques exemples, on expérimente les changements entraînés par le choix de la représentation des graphes.

— Comparer les complexités de questions simples en fonction de la représentation : matrice d'adjacence ou listes d'adjacences (les voisins de chaque sommet sont stockés dans un tableau).
  — Deux sommets sont-ils voisins ?
  — Afficher les voisins d'un sommet.
  — Quel est le degré d'un sommet ?
— Chercher un élément « puits » dans un graphe (degré entrant égal à n-1 et degré sortant 0) en n appels à la matrice d'adjacence (calcul de complexité pour liste de voisins).

2.7 Propositions d'exercices

2.7.1 Exercice pour le paragraphe « graphes sur des exemples simples de modélisation »

Exercice 1 *Dans un tout petit pays, il n'y a que 15 villes. On peut aller de chaque ville à au moins 7 autres villes du pays par une autoroute. Peut-on se rendre, par autoroute, de la capitale du pays à chacune des autres villes ?*

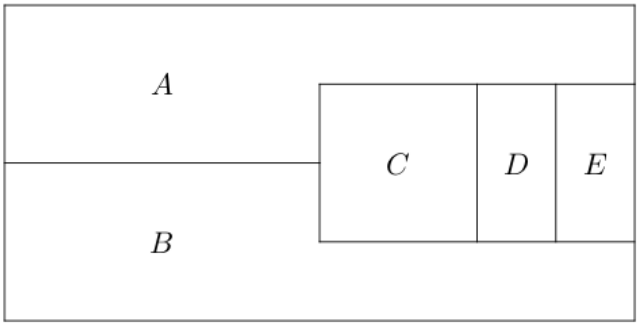
Exercice 2 *Un code d'entrée d'un immeuble est composé de trois chiffres valant 0 ou 1. Trouver un mot, de longueur aussi petite que possible, qui contienne toutes les suites de trois chiffres 0 ou 1 ; un tel mot contiendra alors toujours le code d'entrée.*

Exercice 3 *Pendant un festival, on veut organiser des tournois de scrable (S), échecs (E), go (G), dames (D), tarot (T) et master-mind (M). Plusieurs personnes se sont inscrites à la fois pour les tournois E, S, G, d'autres personnes pour les tournois G, D, M, et enfin d'autres personnes pour les tournois M, T, S. Il est entendu qu'une participation simultanée à plusieurs tournois est impossible et que les organisateurs veulent satisfaire tout le monde.*

a) Quel est le nombre maximum de tournois qui pourraient se dérouler en même temps ?

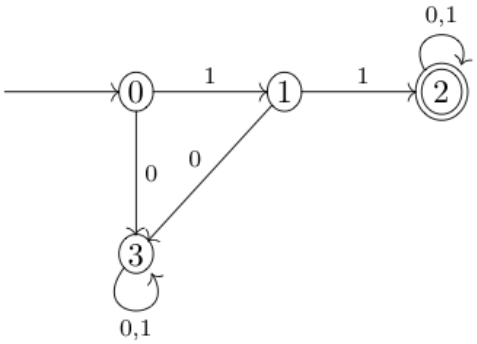
b) En sachant que chaque tournoi doit durer au maximum 3 heures, proposer un horaire des tournois nécessitant une durée minimale et respectant bien sûr les choix des participants.

Exercice 4 *Voici ci-dessous la carte du pays d'Ovalie avec ses 5 régions A, B, C, D et E.*



On veut colorier cette carte de telle manière que deux régions frontalières aient des couleurs différentes. Combien de couleurs au minimum faudra-t-il prévoir ?

Exercice 5 *Soit l'automate :*



— *Les mots "11", "101", "110", "1011" sont-ils reconnus par cet automate ?*
— *Donner la liste des mots de quatre lettres reconnus par celui-ci.*
— *Caractériser les mots reconnus.*

Exercice 6 *Une évolution de population.*

Deux villes X et Y totalisent une population d'un million d'habitants. La ville X est plus agréable, mais la ville Y offre de meilleurs salaires. 20% des habitants de Y partent chaque année habiter X pour avoir un meilleur cadre de vie, et 5% des habitants de X partent chaque année habiter Y pour augmenter leur niveau de vie. Sachant qu'en l'année 0 , un quart des habitants sont en X, calculer la population de X et de Y au bout de 1,2,5,10 ans.

Que se passe-t-il si l'on suppose que 99% des habitants sont initialement en X ou en Y ? Ou que la population est également répartie entre les deux villes en l'année zéro ? Que constate-ton ?

Exercice 7 *L'allumeur de réverbère (exercice du document d'accompagnement).*

Chaque matin, l'allumeur de réverbère du Petit Prince change l'état du réverbère de sa planète avec une probabilité de $0,75$.

1) Qu'observe-t-on en simulant une grande population de réverbères régis par le même système probabiliste de changements d'états.

2) Au jour 0*, le réverbère est éteint. Faire un arbre permettant de trouver l'état probabiliste du réverbère au deuxième jour.*

3) Décrire cette situation à l'aide d'un graphe probabiliste. Soit M *la matrice de transition associée à ce graphe.*

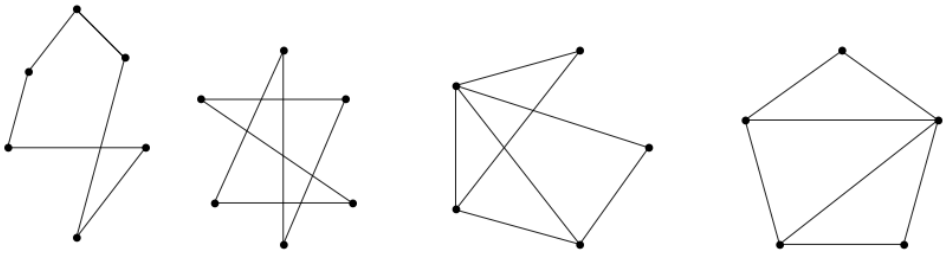
4) Vérifier que l'on a : $M = N - \frac{1}{2}R$*, où* $N = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}$ *et* $R = \begin{pmatrix} \frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{pmatrix}$.

Calculer N^2, R^2, NR *et* RN*. En déduire* M^n*, pour* n *entier naturel supérieur ou égal à* 1.

5) Au jour 0*, le réverbère est allumé (resp. éteint). Calculer la probabilité* v_n*(resp.* u_n*) que le réverbère soit allumé (resp. éteint) au* n^{ieme} *matin. Déterminer la limite de* v_n*(resp.* u_n*) lorsque* n *tend vers l'infini.*

2.7.2 Pour le paragraphe « maîtrise des principaux concepts »

Exercice 8 *Décider si les dessins suivants représentent les mêmes graphes.*



G_1 G_2 G_3 G_4

Exercice 9 *Dessiner les graphes complets* K_n*, pour* $n = 2, 3, 4, 5$*. Combien ont ils d'arêtes ?*

Exercice 10 *Dessiner les graphes simples d'ordre 3, 4, 5 et 6 dont tous les sommets sont de degré 2.*

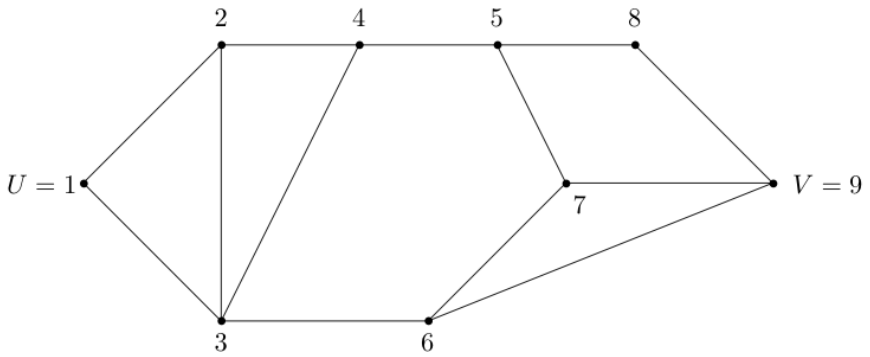
Exercice 11 *Le conseil municipal d'une ville comprend 7 commissions, qui obéissent aux règles suivantes :*

Régle 1 : tout conseiller municipal fait partie de 2 commissions exactement.

Régle 2 : deux commissions quelconques ont exactement un conseiller en commun ;

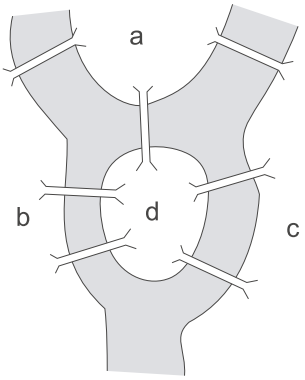
Combien y a-t-il de membres dans le conseil municipal ?

Exercice 12 *Sébastien se rend régulièrement en train de la ville U à la ville V, dans un réseau donné par le graphe ci-dessous. Il fait toujours le trajet en 5 étapes, et veut faire à chaque fois un chemin différent. Combien de trajets pourra-t-il faire ?*



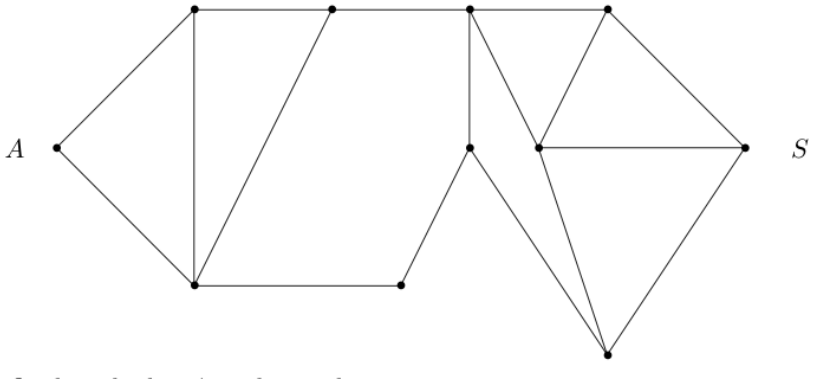
2.7.3 Pour le paragraphe « algorithmes simples »

Exercice 13 *Problème des ponts de Königsberg : Au XVIIIe siècle, la ville de Koenisberg comprenait 2 îles et 7 ponts suivant le plan ci-dessous :*

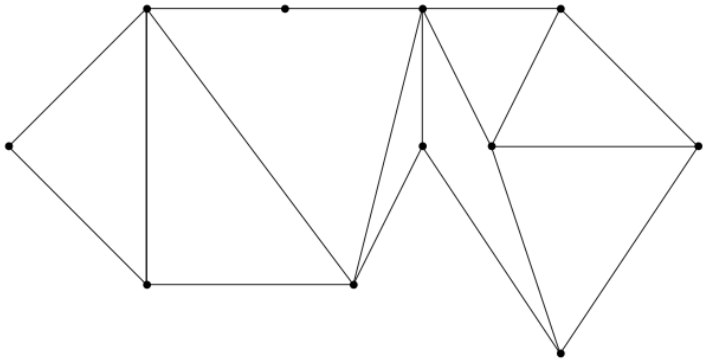


Les habitants souhaitaient faire une promenade passant une et une seule fois sur chaque pont. Y sont-ils arrivés ?

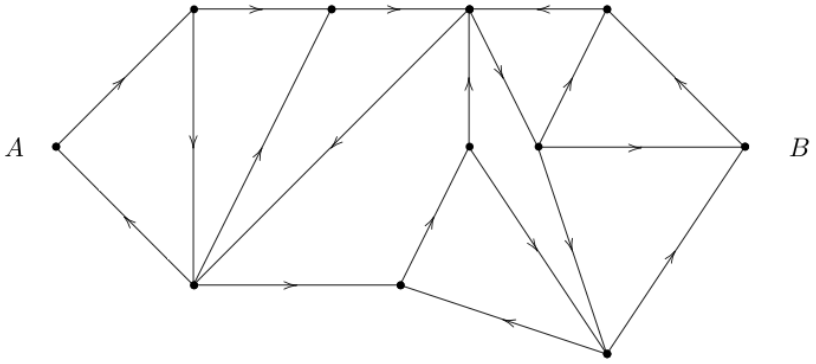
Exercice 14 *Quelle est la distance entre les sommets A et S dans le graphe suivant :*



Exercice 15 *Quel est le diamètre du graphe suivant :*

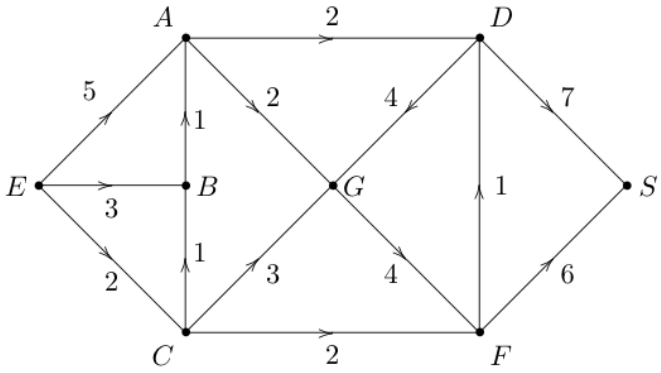


Exercice 16 *Le graphe suivant représente une partie d'une ville où toutes les rues sont à sens unique.*



Quel est le nombre de manières d'aller en voiture, en 5 étapes, de A à B ? quel est le nombre de manières de faire le même itinéraire à pied (on n'est donc plus obligé de respecter les sens interdits) ?

Exercice 17 *Le graphe suivant représente un réseau routier (avec des sens interdits) ; quel est l'itinéraire le plus court qui relie E à S ?*



Exercice 18 *Le prince est parti à la recherche du trésor ; il peut accomplir les actions suivantes :*

- *Aller à la ville du marché, en contournant la rivière par un gué :4 jours.*
- *Traverser la forêt : 1 jour.*
- *Depuis la forêt, abattre des arbres pour traverser la rivière, et se rendre à la ville du marché : 2 jours.*
- *Depuis la forêt, se rendre à la capitale provinciale en traversant les marais : 7 jours.*
- *S'équiper chaudement au marché, et partir pour le col du nord : 5 jours.*
- *Trouver un bon cheval au marché, et se rendre à la capitale provinciale par la grand-route : 3 jours.*
- *Depuis le col du nord, se rendre au refuge du devin : 3 jours.*
- *Depuis la capitale provinciale, se rendre au refuge du devin : 4 jours.*
- *Se rendre de la capitale provinciale au palais du roi, en étant retardé par des contrôles : 10 jours.*
- *Au sortir du devin, partir directement chercher l'épée, et la trouver après s'être perdu par manque de carte : 20 jours.*
- *Au sortir de chez le devin, au mépris de ses avis, se rendre directement à la grotte et tuer le dragon avec un canif : 32 jours (il faut du temps pour le tuer avec un canif).*
- *Bien conseillé par le devin, prendre un raccourci pour le palais du roi : 5 jours.*
- *Un fois arrivé au palais du roi, séduire la bibliothécaire, puis trouver les cartes qui expliquent l'emplacement de l'épée et du trésor : 6 jours.*
- *En utilisant les cartes trouvées dans la bibliothèque, faire tout le tour de la montagne, et traverser un labyrinthe qui mène directement au trésor : 30 jours.*
- *En utilisant les cartes, aller chercher l'épée pour combattre le dragon : 7 jours.*
- *S'entraîner à l'épée, puis tuer le dragon : 8 jours.*
- *Une fois l'épée trouvée, au lieu d'affronter le dragon, utiliser l'épée pour creuser un tunnel par dessous, et déboucher directement dans la cachette du trésor : 18 jours.*
- *Une fois le dragon tué, résoudre l'énigme qui ouvre la cachette du trésor : 9 jours.*

Comment doit-il faire pour récupérer le trésor le plus vite possible ? Quel temps lui faudra-t-il ?

Ce chapitre s'inspire en partie du texte du groupe IREM de Luminy de 2002, **Graphes pour la terminale ES**, *par Pierre Arnoux, Fernand Didier, Catherine Dufossé, Nicolas Lichiardopol, Christian Mauduit, Dominique Proudhon et Christiane Rambaud*

Chapitre 3

Combinatoire

Sur le thème de la combinatoire, l'objectif est d'introduire les notions et savoir-faire permettant de compter les éléments d'ensembles finis, et de les transférer au calcul élémentaire de complexité en temps d'algorithmes. Les notions mathématiques introduites couvrent les couples et n-uplets, leur dénombrement, la fonction factorielle et le triangle de Pascal. Il sera ainsi possible de traiter d'identités remarquables et de probabilités, de compter des objets informatiques, et de distinguer complexité en temps d'un algorithme et complexité intrinsèque d'un problème. Ces enseignements reposent largement sur une approche mathématique expérimentale dans laquelle le développement de programmes informatiques amène la découverte des notions, leur acquisition, en confrontant résultats expérimentaux et concepts mathématiques.

3.1 Notions mathématiques

Les notions de mathématiques nécessaires ou introduites incluent
- Cardinal du produit cartésien (fini d'ensembles finis).
  Remarques
  - la terminologie *produit cartésien* ne sera pas nécessairement introduite ;
  - on introduira les notions de couples, n-uplets ;
  - les règles (de la somme et) du produit suffisent au dénombrement des couples et n-uplets.
- Fonction factorielle.
  - définition, manipulation simple ;
  - la notion de permutation n'est peut-être pas utile ici.
- Triangle de Pascal.
  - construction du triangle de Pascal ;

Nous n'introduirions pas les notions suivantes :
- permutation ; arrangement ; combinaison, et donc coefficients du binôme ;

L'introduction de la notion d'arbre des possibilités comme outil (didactique ?), comme support pour les probabilités peut être envisagée. Dans ce cas, lien entre factorielle et n tirages successifs de n objets décrit par un arbre équilibré. (On ne fera pas le lien avec les graphes.)

3.2 Pertinence

Quelques usages en maths,

— pour le calcul : identité remarquable $(a+b)^n$;
— pour les probabilités : loi binomiale (définition, diagramme en bâtons, tirage sans remise) ;
— probabilité conditionnelle intuitive (proba de « truc » si « machin » vu sur l'arbre des possibilités ou des probabilités) ; item aller vers le concept de permutations (en spécialité maths seulement ?) comme bijections d'un ensemble sur lui-même.

et quelques usages en informatique

— savoir compter des objets ;
— calcul de complexité élémentaire : boucles, récursion ;
— distinguer complexité en temps / complexité intrinsèque ;
— dénombrer des objets informatiques, des strutures.

3.3 Support

Comme cela est illustré dans quelques exercices ci-après, il est par exemple possibe de demander à écrire un programme qui génère les éléments d'ensemble que l'on désire dénombrer. Cette programmation impliquant une compréhension du dénombrement. Ainsi, l'enseignement de ces notions de mathématiques s'appuie sur les savoirs et savoir-faire informatiques des élèves.

3.4 Proposition d'exercices

3.4.1 Les règles de la somme et du produit

Empruts à la médiathèque La médiathèque municipale détient 2362 romans, 1723 BD, et 623 essais.

— Combien de livres différents puis-je emprunter ?

La médiathèque municipale détient aussi 472 CD, 287 livres audios.

— Combien de documents sonores différents puis-je emprunter ?
— Si je peux emprunter un livre et un document sonore, combien d'emprunts différents puis-je faire ?

Couples d'acteurs Le club théâtre d'un lycée fait des essais pour la pièce de printemps. Six hommes et huit femmes passent des auditions pour les premiers rôles masculin et féminin.

— Combien de couples possibles peut-on former ?

Jeu de formes géométriques On veut construire un jeu de formes géométriques en bois peint. Une forme peut être un carré, un triangle, ou un disque. Une forme peut être de couleur rouge, jaune, bleue, ou verte.

— Écrivez un programme Python qui affiche l'ensemble des formes différentes que je peux produire.
— Combien de formes différentes puis-je produire ?

Affichage Python Soit le programme Python suivant :

```
for i in range(1,13):
    for j in range(5,11):
        print (i-j)
```

— Combien de fois la fonction `print` est elle appelée ?

Soit le second programme :

```
for i in range(1,13):
    for j in range(5,11):
        for k in reversed(range(8,16)):
            print((i-j)*k)
```

Note : on rappelle que (en Python) `range(a,b)` est l'intervalle des entiers supérieurs ou égaux à a et inférieurs strictement à b.

3.4.2 Factorielle

Définition et notation La factorielle d'un entier naturel n est le produit des entiers strictement positifs inférieurs ou égaux à n. On note $n!$ cette opération.

On a donc, 1re définition,

$$n! = \prod_{i=1}^{n} i$$

On peut aussi définir fatorielle par récurrence, 2e définition,

— $0! = 1$

— pour tout entier $n \geq 1$, $n! = (n-1)! \times n$

Calculer des factorielles

— Calculer 5!, la factorielle de 5.

— Calculer 6!, la factorielle de 6.

— Calculer 7!, la factorielle de 7. (L'idée est d'amener à l'utilisation de la seconde définition de factorielle).

Programmer la factorielle Écrire une fonction Python qui retourne la factorielle d'un entier donné.

— Considérer la première définition de factorielle pour proposer une première implémentation.

— Considérer la seconde définition de factorielle pour proposer une seconde implémentation.

3.4.3 On compte !

En rang pour la photo Dans un groupe de dix étudiants, quatre ont décidé de s'asseoir sur une rangée pour prendre une photo. De combien de manières peuvent-ils s'installer ?

10 (première position) x 9 (2e position) x 8 (3e position) x 7 (4e position)

Soit 10!/6!

La résolution de cet exercice est une illustration de factorielle. Il s'agit plus de découvrir que d'appliquer une connaissance a priori du nombre d'arrangements.

Nombre de chemins Soit un quadrillage surlequel on fait des pas soit vers la droite soit vers le haut. C'est-à-dire que les déplacements sont tels que

— l'abscisse soit croissante,

— l'ordonnée soit croissante,

— à chaque pas, on conserve l'une des coordonnées.

— Combien de chemins différents lient le point $(0, 0)$ au point $(3, 2)$

— le point $(3, 2)$ au point $(7, 4)$

La fonction Python `chemins` paramétrée par les coordonnées des deux points (soit quatre entiers) renvoie la liste de tous les chemins, par exemple

```
>>> chemins(2,1,3,4)
        ['DHHH', 'HDHH', 'HHDH', 'HHHD']
```

— Donnez le code de la fonction `chemins`

3.4.4 Complexité

Recherche d'un élément dans un ensemble, d'une carte dans un jeu On dispose d'un jeu de 52 cartes à jouer. On désire recherche une carte donnée dans ce jeu, par exemple le valet de carreau.

— Combien d'essais pour trouver la bonne carte ? Quel algorithme utiliser ?
  — tirer une carte au hasard (sans remise)...
  — regarder les cartes les unes après les autres
  sont bien entendu la même chose.
— Combien d'essais...
  — Combien d'essais au mieux ?
  — Combien d'essais au pire ?
  — Combien d'essai en moyenne ?
— Combien d'essais pour un jeu de tarot de 78 cartes ?

On trie le jeu de cartes. Les quatre enseignes dans l'ordre, par exemple pique, cœur, carreau et trèfle. Les valeurs de 2, 3, 4... valet, dame, roi, as.

— Quel nouvel algorithme peut être mis en place ?
— Combien d'essais... au mieux, au pire, en moyenne ? (On peut supposer 32 ou 64 cartes pour simplifier)

Exponentiation rapide (Calcul rapide d'élévation à la puissance) Le calcul "naïf" de 5^{13} soit $5 \times 5 \times 5 \times 5 ... \times 5$ nécessite 13 multiplications.

Un possible calcul "rapide" de cette expontielle considère :

— le codage en base 2 de 13 : $13 = 1101b$
— soit $13 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$
— et donc $5^{13} = 5^{(2^3+2^2+2^0)}$
  ou $5^{13} = 5^{2^3} \times 5^{2^2} \times 5^{2^0}$

On réalise alors les calculs suivants :

$$\begin{array}{lcl} & - & 5^1 \\ 5^1 \times 5^1 = 5^2 & - & +0 \times 5^2 \\ 5^2 \times 5^2 = 5^4 & - & +1 \times 5^4 \\ 5^4 \times 5^4 = 5^8 & - & +1 \times 5^8 \end{array}$$

soit 3 multiplications et 4 additions

Considérons le calcul de i^{13}

— Combien de multiplication sont nécessaires pour le calcul naïf ?
— Combien de muliplications et additions sont nécessaires pour le calcul rapide ?

Considérons le calcul de i^n

— Combien de multiplication sont nécessaires pour le calcul naïf ?
— Combien de muliplications et additions sont nécessaires pour le calcul rapide ?

On parle de complexité

— nombre d'opérations d'une exécution de l'algorithme
— ordre de grandeur *vs* nombre d'opération
— complexité de l'algorithme
— complexité de l'algorihme de calcul de la puissance n-ième

Écrire deux fonctions Python qui retournent i^n

— une première fonction qui implémente la version naïve de l'algorithme de calcul ;
— une seconde fonction qui implémente la version rapide ;
— comparer les temps d'exécution des deux fonctions pour le calcul de ... (donner des valeurs de i et n pour lesquelles on pourra *observer* une différence du temsp d'exécution)

3.4.5 Autres pistes pour la complexité

Retour sur la somme et produit Reconsidérer les règles de la somme et du produit en relation avec l'exécution des programmes, la complexité.

Complexité des constructions
— séquence d'instructions `i1 ; i2`
— alternative `if (p1) i1` et `if (p1) i1 else i2`
— itérations `for e in E : i1`
Retour sur les exercises "Combien d'appels à la fonction `print`".

Tri Pourquoi trier
Plusieurs algorithme de tri. Comparaison complexité.
Complexité intrinsèque ?

Fibonacci Versions récursive et itérative. Programamtion. Mesure temps de calcul. Complexité (des éléments de...), comparaison.

Chapitre 4

Représentation et modélisation de l'information

Sur le thème de la représentation et modélisation de l'information, notre proposition est de traiter de problématiques au cœur du monde numérique. Ces problématiques permettent de relier des notions mathématiques importantes (représentation des nombres, combinatoire, probabilité, algèbre linéaire) aux applications concrètes de la compression de données et des codes correcteurs d'erreurs. La représentation approchée des nombres réels et son impact sur la stabilité des calculs numériques seront également abordés. L'informatique permet d'appliquer ces concepts sur des données de grande taille, afin de mieux saisir leur importance (à travers le gain en taille du fichier ou en robustesse aux erreurs).

4.1 Introduction

Dans les médias actuels, le contenu est systématiquement numérisé, car les ordinateurs, processeurs, téléphones manipulent des structures finies qui sont des vecteurs de 0 et de 1 (bits). C'est ce que l'on peut appeler, par raccourci, l'information numérique.

Ensuite cette information est traitée pour plusieurs raisons : afin d'économiser de la mémoire grâce à des techniques de compressions, ou bien pour la protéger contre un bruit aléatoire provenant du média sur lequel transite cette information – par des techniques de codage correcteur d'erreur –, ou encore pour la chiffrer par des techniques cryptographiques afin qu'elle soit inintelligible pour des destinataires non-légitimes.

Toutes ces techniques et procédés utilisent des notions mathématiques variées fondées sur de l'arithmétique, de l'algèbre linéaire, de la géométrie algébrique, des probabilités et statistiques ainsi que de la théorie des graphes.

Nous présentons dans cette partie les notions de base qui nous semblent fondamentales pour commencer à toucher à la compréhension des techniques de traitement de l'information.

4.2 Notions mathématiques

Les notions minimales de mathématiques à connaître et à maîtriser sont donc :
— représentation des entiers : numération en base 2 et en base 16 au moins,
— arithmétique : calcul modulaire, division euclidienne d'entiers et de polynômes,
— algèbre linéaire de base : vecteurs, espaces vectoriels, matrices, bases, produit matrice/vecteur,

— probabilités et statistiques : espérance, variance, loi normale.

Certaines de ces notions sont enseignées en spécialité en série S, mais il serait bon de les aborder aussi dans les autres séries, dans la mesure du possible.

4.3 Pertinence

Cette section décrit comment les notions de mathématiques introduites précédemment s'inscrivent dans la pratique de l'informatique.

4.3.1 Numération en base N quelconque

Au niveau informatique, la notion de base de numération est fondamentale dans la représentation et le calcul. Il faut aussi aborder la théorie des graphes pour construire un arbre de compression même simple, les probabilités et statistiques, afin, par exemple de calculer les fréquences de lettre et la manière de construire l'arbre de compression, pour attaquer des systèmes cryptographiques construits ci-dessous.

Il est important de faire comprendre à l'élève que tout va être codé, pas seulement les nombres. On peut par exemple donner comme exemples les textes (chaque caractère va être codé) ou les images (chaque pixel va être codé).

Il faut savoir exécuter les opérations élémentaires en base N (avec une priorité à la base 2), puis expliquer que, quelle que soit l'information codée, il faudra ensuite utiliser ces opérations élémentaires astucieusement, algorithmiquement, pour effectuer des calculs.

Par exemple, pour une image, on peut calculer son négatif, l'assombrir, la flouter (pour enlever du "bruit") [1]. On peut également cacher un message par des opérations sur les codes (comme modifier les bits de poids faibles dans une représentation binaire).

Enfin, il faut s'intéresser aux enjeux de stockage, donc de compression (mais on peut aussi faire de la compression pour extraire des motifs, pour comprendre) aborder des problématiques de chiffrement.

4.3.2 Probabilités et statistiques

Si l'on fixe les données à coder, le codage source (i.e. compresser "sans perte" l'information) met en jeu des notions simples de calcul de fréquence d'apparition des symboles (par exemple des lettres dans un texte, ou bien les niveaux de gris dans une image en N&B). On donne plus bas un exercice (section 4.5.3) pour illustrer ceci. Cet exercice propose également de réaliser le codage sur machine, afin de pouvoir aborder des données de grandes tailles. Ceci permet de se rendre compte du gain en terme de stockage d'un fichier.

Si l'on souhaite étudier plus en détail ces questions, on peut aborder la notion de modélisation statistique de l'information, qui consiste à supposer que les symboles sont indépendants (hypothèse simplificatrice mais en générale fausse) et suivent une certaine loi de probabilité. La théorie de Shannon permet de définir des codes optimaux pour compresser sans perte. Ceci est trop compliqué pour être abordé au lycée, mais on peut faire toucher du doigt l'impact de la distribution de probabilité des symboles (certains textes se compressent mieux que d'autres). Voir à ce sujet un exercice proposé plus bas.

1. Voir l'article "Le traitement numérique des images" `http://images.math.cnrs.fr/Le-traitement-numerique-des-images.html`.

4.3.3 Arithmétique et calcul modulaire

La notion de division euclidienne et de calcul modulaire dans les entiers est fondamentale dans les aspects de traitement numérique de l'information. Les codes BCH, de Hamming, les codes de Reed-Solomon cycliques, le code CRC (cyclic redundancy check) qui sont très présents dans toutes les chaînes de communications font grand usage de la division euclidienne pour détecter ou corriger des erreurs.

On pourra introduire le calcul modulaire, d'un point de vue algorithmique (ex : algorithme d'*exponentiation modulaire* qui relie le calcul modulaire et la représentation en base 2 par exemple) ainsi que pour de la cryptographie de base comme le chiffrement de César, Scytale ou Vigenère, sur l'alphabet français de 26 lettres. Concernant des application plus récentes, les systèmes de chiffrement de type RSA, reposent également sur des opération d'arithmétique modulaire.

4.3.4 Algèbre linéaire de base

Un grand nombre de codes correcteurs d'erreurs sont des espaces vectoriels sur des corps finis. L'idée étant qu'un espace vectoriel se représente plus facilement en machine qu'un ensemble quelconque.

De même le nouveaux systèmes de chiffrements dits *Post-Quantiques*, font également grand usage d'algèbre linéaire sur les réels et sur les corps finis. Egalement, en conception de primitives symétriques (AES, fonctions de Hachage SHA-3), on utilise des produits matrice-vecteurs sur des octets pour assurer le principe de la diffusion.

4.3.5 Approximation des nombres

La représentation approchée des nombres réels correspond au problème concret de la mesure de l'information depuis le monde qui nous entoure. La quantification de l'information revient à choisir une représentation des réels dans laquelle on arrondit les nombres. La section "support" ci-dessous propose de réaliser une expérience ("scanner" une image à la main) pour aborder cette question. Cette quantification est aussi présente lors de la compression avec perte des données multimédia (son, image et vidéo correspondant aux fichiers MP3, JPEG et MPEG).

Il conviendra également d'aborder le problème, souvent ignoré du grand public, du passage de la modélisation mathématique d'un phénomène à sa résolution numérique et approchée sur ordinateur. Les aspects de complexité algorithmique en lien avec d'autres sections de ce document seront soulignés. Les problèmes liés à l'approximation et au conditionnement (qui relie la précision du résultat à la précision des données d'entrée) pourront être introduits au travers d'exemples simples comme la résolution d'un système linéaire 2×2.

4.4 Support

En ce qui concerne la compression de données (ici le codage source), l'utilisation de l'outil informatique (par exemple lors de TP) est important pour pouvoir donner des exemples de plus grande taille, qui seront plus "impressionnants" et convaincants. En effet, dans de nombreux cas, le gain n'est vraiment visible que sur des grosses masse de données (par exemple un livre entier ou bien une image assez grande) que l'on ne peut pas traiter à la main.

De même, en ce qui concerne les aspects du calcul modulaire comme l'exponentiation modulaire, la représentation du nombre dans une base permet de réduire drastiquement la complexité d'un calcul. Ce ne peut être visible que sur des très grands nombres.

Un exemple concret, pour faire toucher du doigt l'importance du problème de représentation des données, est de réaliser un scanner d'image "à la main" : on demande à chaque élève de remplir un tableau de pixels de taille 8×8, avec des nombres entre 1 et 10, chaque nombre étant censé être proportionnel au niveau de gris d'une petite image. On peut ensuite afficher l'image sur l'ordinateur, et la modifier (négatif, assombrir, etc.).

4.5 Proposition d'exercices

4.5.1 Preuve par 9 et/ou $N-1$

La preuve par neuf est l'exemple d'un exercice qui permet d'introduire la notion de code détecteur d'erreur. L'objectif de l'exercice est que l'élève se rende compte que cet algorithme renvoie deux cas :

- *ou bien les sommes des chiffres modulo 9 de A et de $QB+R$ sont différentes et il y a forcément une erreur ;*
- *ou bien les sommes des chiffres modulo 9 de A et de $QB+R$ sont égales. Cela ne signifie pas qu'il n'y a pas d'erreur, mais qu'on a 90% de chances de ne pas s'être trompé, sous de bonnes propriétés de distribution des chiffres de A et B.*

Dans le cas de la base 2, on peut relier cet algorithme à la détection d'erreur pour le code dit "de parité", qui était implémenté dans les mémoire des ordinateurs dans les années 90-2000 (on rajoute à la fin du vecteur un bit qui est la somme modulo 2 de tous les bits précédents, ce qui fait qu'on a toujours un nombre pair de bits non nuls)

Soit $A = a_{n-1} \cdots a_0$ un nombre à n chiffres écrit en base 10 : $A = \sum_{i=0}^{n-1} a_i 10^i$.

1. Montrer que $A \bmod 9 = \sum_{i=0}^{n} a_i \bmod 9$, c'est-à-dire que $A \bmod 9$ est la somme modulo 9 des chiffres de A
2. Soit $A = 10012102123$ et $B = 3521$. Calculer $A \bmod 9$ et $B \bmod 9$
3. Soit A et B deux nombres et Q et R respectivement le quotient et le reste de la division euclidienne de A par B, *i.e.*
$$A = QB + R,\ R < B$$
Donnez la valeur de $A \bmod 9$ en fonction de $Q \bmod 9$, $B \bmod 9$ et $R \bmod 9$
4. En déduire une méthode permettant de détecter une erreur effectuée lors de la division euclidienne de A par B
5. Est-ce que $10012102003 \stackrel{?}{=} 320003 \times 3221 + 240$?

Preuve par $N-1$ (extension de la preuve par 9) sur le même modèle.

4.5.2 Euclide en binaire

C'est un exemple simple d'algorithme qui a un contenu mathématique basique sans être trivial, qui montre l'un des intérêts de l'écriture en base 2, et dont on peut facilement prouver la correction et majorer explicitement le temps de travail.

But de l'algorithme : calculer le PGCD de deux nombres A, B écrits en binaire.

Exercice introductif : montrer qu'il est très facile de voir si un nombre A écrit en binaire est pair, et dans ce cas, de le diviser par 2 ; plus généralement, montrer qu'on peut immédiatement donner la plus grande puissance de 2 qui divise A

Algorithme : On donne deux entiers A, B.

Initialisation : Calculer le plus grand entier n_A (resp. n_B) qui divise A (resp. B). Diviser A par 2^{n_A} et B par 2^{n_B}. On se retrouve donc avec deux nombres impairs, et on met en mémoire $n = min(n_A, n_B)$. On réordonne la paire pour que $A > B$.

Coeur de l'algorithme : Tant que $B \neq 0$, remplacer A par $A - B$, diviser A par la plus grande puissance possible de 2 (c'est-à-dire éliminer les 0 à la fin), et réordonner la paire (A, B).

Sortie : A la sortie de la boucle, $B = 0$ et $2^n A$ est le PGCD.

Il est facile de démontrer ce résultat, et de prouver que si A est un nombre de p bits et B un nombre de q bits, alors la boucle est effectuée au plus $p + q$ fois, car on perd au moins un bit à chaque passage, ce qui implique aussi que l'algorithme se termine en temps fini. On peut aussi majorer explicitement le nombre d'opérations pour une boucle (mais il faut alors préciser les opérations élémentaires).

L'algorithme d'Euclide usuel fait intervenir des divisions euclidiennes, ce qui est relativement coûteux en temps de calcul ; le nombre de boucle sera à peu près le même que pour l'algorithme binaire présenté ici, mais chaque boucle est au mieux quadratique en p (division) au lieu d'être linéaire comme ici (soustraction). Il existe un algorithme additif, parfois appelé algorithme de Farey, mais celui-ci peut être extrêmement lent dans le pire des cas.

4.5.3 Codage source

On considère des documents textuels composés uniquement des lettres 'A', 'B', 'C', 'D'. Il y a donc quatre symboles.

1. Si l'on souhaite coder chaque lettre sur le même nombre de bits, combien de bits faut-il par symbole ?

On considère le codage suivant, qui utilise un nombre de bits variable par symbole :

$$A \to 0, \quad B \to 10, \quad C \to 110, \quad D \to 111.$$

2. Coder le message '$ABAA$' à l'aide de ce code.
3. Décoder le message codé '101101110010'. Comment avez-vous fait ? Expliquer pourquoi on peut toujours décoder un message codé à l'aide d'un tel système de codage.
4. On souhaite coder un texte avec un nombre de bits variable suivant la fréquence d'apparition de chaque symbole, en choisissant le codage le plus court pour le symbole le plus fréquent : 'A' présent 220 fois, 'B' présent 80 fois, 'C' présent 90 fois et 'D' présent 20 fois. Calculer le nombre total de bits nécessaire pour coder ce texte à l'aide du premier système de codage (avec nombre de bits fixe par symbole). Refaire le calcul, mais cette fois à l'aide du deuxième système de codage (avec nombre de bits variable).
5. À la place d'un texte, on pourra consider une image en niveaux de gris contenant quatre valeurs de pixels. Les élèves pourront programmer sur machine les algorithmes de codage et de décodage. Ils pourront utiliser un code uniforme (question 1) ou bien adapter le code de la question 2, et constater la réduction de la taille du fichier codé obtenu.

Afin de conforter les observations menées sur le codage de textes de la première partie de l'exercice, on fournira aux élèves [2] un programme Python de codage d'images en niveau de gris contenant quatre valeurs de pixels pour les deux systèmes de codage.

Il s'agira pour les élèves

1. Afin de s'approprier ce programme, de proposer des fonctions de décodage d'images.
2. De réaliser des expérimentations de codage d'images de grande taille, d'observer les tailles des fichiers codés obtenus, et de produire une analyse de ces expérimentations.

2. Annexe à venir.

4.5.4 Codage des nombres

Les nombres flottants sont une représentation informatique et approchée des nombres réels. En Python, on utilise le type `float`.

Observons.

1. Les nombres flottants ne sont pas des réels :

```
>>> 0.1 + 0.2
0.30000000000000004

>>> (0.1 + 0.2) - 0.3 == 0
False
```

2. L'addition des nombres flottants n'est pas associative :

```
>>> (1 + 2**(-53))-1
0.0
>>> 1 + (2**(-53)-1)
1.1102230246251565e-16
```

3. Les grandes valeurs sont «absorbantes»

```
>>> 1.0 + (2**53 - 2**53)
1.0
>>> (1.0 + 2**53) - 2**53
0.0
```

Expliquons Le principe de codage des flottants [3].

Comprenons Maintenant que nous savons comment sont codés les nombres flottants, comprenons le pourquoi des résultats observés initialement [4].

4.5.5 Chiffrement de César

L'objectif de cet exercice est de chiffrer un message simple avec un chiffrement de César et d'utiliser une technique statistique simple pour le cryptanalyser.

On considère le codage des 26 lettres de l'alphabet comme des entiers modulo 26.

$$A \leftrightarrow 0,\ B \leftrightarrow 1,\ \cdots,\ Z \leftrightarrow 25$$

On considère un entier $\mathcal{K}$ entre 0 et 25 appelé la "clé". Le chiffrement de César consiste à ajouter $\mathcal{K}$ à chaque lettre de l'alphabet. Par exemple si $\mathcal{K} = J = 9$ (J étant la dixième lettre de l'alphabet), et si l'on a le texte clair $BONJOUR$ que l'on veut chiffrer, on procède de la manière suivante :

— On transforme $BONJOUR$ en suite de nombres entiers :

$$BONJOUR \leftrightarrow 1\ 14\ 13\ 9\ 14\ 20\ 17$$

— on additionne $\mathcal{K} = 9$ modulo 26, ce qui donne le texte chiffré suivant.

$$10\ 23\ 22\ 18\ 23\ 3\ 0 \leftrightarrow KYXSYDA$$

— Plus généralement, écrire une procédure qui, à partir d'un texte en clair, produit le texte codé par le chiffrement de César et inversement.

Applications :

3. Cette partie sera complétée dans la version à venir du document.
4. Cette partie sera complétée dans la version à venir du document.

— Étant donné le texte clair $AUREVOIR$ et la clé $\mathcal{K} = 12$, quel est le texte chiffré correspondant ?
— Étant donné le texte chiffré :

$$VUAXWAOYUTZIKYYKXVKTZY$$

et sachant que la lettre la plus fréquente du texte clair est le S retrouver le texte clair ainsi que la clé.