Propositions pour la spécialité ISN en terminale S

Une délégation du réseau des IREM, composée d'Alex Esbelin, Michel Myara et Denis Pinsard, a été reçue le 21 janvier 2011, à la demande du conseil scientifique des IREM, par le groupe d'experts chargé de l'élaboration du programme de la future spécialité *Informatique et sciences du numérique* en terminale S.

Les réflexions et les échanges que nous avons eus avant, pendant et après cette réunion se trouvent synthétisés dans le texte ci-dessous.

Nous espérons par ce document contribuer modestement à la réussite de l'introduction des sciences du numérique au lycée.

❑ Quels objectifs pour cette spécialité ?

Le terme « sciences » qui apparait dans l'intitulé de la spécialité devrait être pleinement justifié par le programme, en particulier dans l'énoncé de ses objectifs. Pour beaucoup l'informatique se résume à manipuler des objets numériques, un clavier, une souris, à naviguer avec dextérité dans des menus complexes. Certains en deviennent accrocs alors que d'autres en font un rejet. La spécialité devrait être l'occasion de montrer que les sciences du numérique n'ont pas grand chose à voir avec cela. Dans ce sens l'intitulé *Informatique et sciences du numérique* relève peut-être davantage de l'oxymore que du pléonasme. Cette meilleure compréhension de ce que sont les sciences du numérique permettra aux lycéens de mieux apprécier leur motivation pour cette discipline.

Quels bénéfices les élèves devraient-ils retirer de cet enseignement ?

a) Une ouverture culturelle vers les sciences du numérique :

Les élèves prennent conscience

- que les objets numériques qu'ils côtoient quotidiennement, sous leur aspect banal et familier, cachent tout un monde qu'ils ne soupçonnaient pas auparavant,
- que ce monde repose sur des principes, des concepts qui sont intelligibles et qu'ils peuvent comprendre s'ils veulent s'en donner les moyens,
- que ces concepts forment une science, qui possède une histoire récente forgée par des hommes, et qui est source de métiers attrayants,
- que ces métiers ne se réduisent pas à manipuler des outils logiciels, pas plus que dans les autres métiers.

Les élèves prennent conscience aussi bien des potentialités que des limites des sciences et technologies numériques.

Les outils issus de la technologie numérique évoluent sans cesse. Apprendre à utiliser les outils qui existent aujourd'hui ne peut suffire à la formation d'un individu pleinement autonome dans la société numérique de demain. La compréhension des principes généraux de cette technologie nous parait devoir faire partie de la culture générale de tout citoyen.

b) Des compétences scientifiques :

Les élèves mobilisent et étendent leurs compétences intellectuelles pour comprendre et résoudre

- des problèmes de façon structurée,
- des problèmes d'une complexité limitée mais réelle,
- des problèmes qui ont un lien avec leur monde quotidien.

Les élèves s'initient aux modes de pensées et de raisonnements propres aux sciences du numérique.

L'épistémologie et la didactique des sciences du numérique diffèrent de celles des mathématiques, mais nous pensons que les compétences intellectuelles que ces deux disciplines cherchent à développer chez les élèves se recoupent dans une large mesure.

❑ Quelle forme donner à cet enseignement ?

Il nous semble que la spécificité de l'informatique est de posséder à la fois une dimension mathématique et une dimension technologique et que ces deux dimensions devraient être en permanence à l'esprit des enseignants de cette spécialité. Les activités que l'on peut proposer aux élèves peuvent être très diverses : étude d'applications et d'objets numériques, réalisation de petits programmes, études de problèmes algorithmiques, réalisation d'une petite application de la phase de conception à la phase de test. Le contenu individuel de ces activités compte moins que leur cohérence entre elles : les applications étudiées permettent de faire émerger des problèmes algorithmiques intéressants et formateurs qui pourront pour certains être programmés pour comparer leurs performances et motiver ainsi la notion de complexité. Vu l'horaire restreint il importe d'éviter les activités « gratuites » ou à objectifs isolés, en particulier pour les activités chronophages. Chacune doit contribuer à forger les concepts fondamentaux de l'informatique : machine programmable, langage, information, algorithme.

▹ L'approche par les applications

L'étude d'objets numériques familiers comme par exemple un téléphone portable permet de faire émerger des questionnements, des problématiques et d'introduire à partir de là les concepts fondamentaux de l'informatique. L'annexe *A* à la fin de ce document donne un exemple de questionnement possible autour du téléphone portable. Il nous semble préférable que les élèves consacrent du temps à approfondir un ou deux thèmes plutôt que de vouloir couvrir un vaste domaine. Ces thèmes pourraient être étudiés par des groupes d'élèves et présentés sous forme d'exposés au reste de la classe. Ces thèmes peuvent également être la source de projets qui pourraient compter pour l'évaluation au baccalauréat. Si cette approche par les applications nous parait pertinente, il convient toutefois de bien garder à l'esprit qu'il s'agit au bout du compte d'amener les élèves à percevoir l'universalité des concepts informatiques au travers de la diversité des applications.

L'idée de numérisation est un des concepts clés à transmettre aux élèves. Une entrée motivante est de commencer par manipuler des images ou des sons à l'aide d'outils logiciels pour ensuite se poser les questions : comment un fichier peut-il contenir une image ou un son ? qu'y a-t-il exactement dans ce fichier ? comment le logiciel modifie-t-il ce fichier ? pourquoi existe-t-il plusieurs formats ? Ces questionnements font émerger les concepts d'information, de structure de données puis conduisent à se poser les problèmes d'efficacité et donc à comparer les formats et à étudier les algorithmes associés.

▹ L'approche par la programmation

Une pratique modeste de la programmation nous parait nécessaire pour que les élèves puissent véritablement s'approprier les concepts de machine, d'information, de langage et d'algorithme. Cela soulève toutefois des difficultés :

- Nous constatons actuellement que la pratique de la programmation en seconde dans le cadre de l'algorithmique ne peut rester qu'à un niveau extrêmement modeste, très loin de pouvoir amener les élèves à une réelle compréhension de ce qu'est l'informatique.
- L'apprentissage de la syntaxe d'un langage, la familiarisation avec son environnement de programmation, l'utilisation des bibliothèques de fonctions standards, tout cela peut être très chronophage (l'horaire n'est que de deux petites heures par semaine).
- De plus, même une bonne pratique d'un langage peut d'une certaine façon rester superficielle et ne garantit pas que l'élève ait réellement compris les processus mis en jeu par son activité.
- Par ailleurs une trop grande place accordée au travail sur ordinateur risque de renforcer chez les élèves l'identification réductrice entre sciences du numérique et manipulation d'outils logiciels.

L'IREM de Marseille et l'Institut de Mathématiques de Luminy proposent lors de leurs stages Hippocampe-Maths un travail autour de machines à registres. Cela nous semble être une idée à reprendre pour quelques séances de travaux pratiques, comme alternative à des séances de programmation. L'apprentissage d'un langage informatique par un élève de terminale n'est pas une fin en soi mais le moyen de comprendre et se représenter correctement les concepts informatiques et tout particulièrement la notion universelle de machine programmable.

Les élèves devraient d'un côté comprendre ce qu'est une machine programmable et voir qu'un langage informatique est le moyen que nous avons de communiquer avec une telle machine. Dans cet esprit, cela parait judicieux de faire réfléchir les élèves au processus d'interprétation et d'aborder éventuellement les notions d'analyse lexicale, grammaticale, sémantique. De réfléchir également à la représentation des données dans la machine.

De l'autre côté il est intéressant de montrer aux élèves qu'au delà de la syntaxe, un langage informatique nous sert à exprimer des concepts. Pour pouvoir faire réaliser à une machine des tâches complexes il est nécessaire d'imaginer des concepts, des niveaux d'abstraction, des structures de données. Par exemple, l'implémentation dans un programme de structures telles que des piles ou des arbres pourraient être formateur. La programmation est aussi le moyen d'analyser expérimentalement les performances de certains algorithmes étudiés en classe.

La question du langage

Ces objectifs que nous assignons à la pratique d'un langage sont très (trop ?) ambitieux pour les élèves mais devraient être dans la ligne de mire des enseignants. Dans cette perspective, le choix du langage de programmation n'est pas neutre et nous semble même être un choix crucial. Les éléments suivants nous semble devoir être pris en compte :

- Un langage avec une syntaxe simple et uniforme rend l'apprentissage plus rapide. Cela présente également le très grand avantage de rendre intelligible et plus concret le processus accompli par l'interpréteur. L'élève peut simuler mentalement le travail de l'interpréteur.
- Un langage interprété avec un mode ligne de commande permet à l'élève d'avoir un contact plus immédiat avec la machine programmable.
- Une machine programmable et son langage qui implémentent des concepts simples et en petit nombre peuvent procurer aux élèves la satisfaction de maîtriser le fonctionnement de cette machine. Par ailleurs, la démarche qui consiste

à créer des concepts de plus haut niveau, que le langage n'inclut pas mais rend possible, prend alors toute sa signification.

- Un langage qui permet les appels récursifs car beaucoup d'algorithmes s'expriment naturellement sous cette forme.

Et de façon moins essentielle :

- Un langage qui offre les deux paradigmes fonctionnel et impératif permet de programmer un algorithme dans le style où il a été pensé. De comparer les styles de programmation et de faire réfléchir les élèves au choix le plus pertinent en terme d'efficacité, de lisibilité, d'élégance, etc.
- Un environnement de programmation très dépouillé permet de ne pas perdre de temps et de se concentrer sur l'essentiel.
- Un langage qui donne un accès facile (sans API complexes) à des interfaces et périphériques variés (écran graphique, clavier, souris, lecture/écriture de fichiers, réseau, carte son).

▹ L'approche algorithmique

C'est en classe de seconde que les élèves entendent pour la première fois prononcer le mot « algorithme » de façon récurrente. L'écriture et l'exécution de petits programmes sont alors indispensables pour étayer et donner corps à cette notion abstraite. Nous pensons qu'un algorithme est d'abord une idée. Une idée qui fournit un mode de résolution à un problème donné. S'il est vrai que le premier contact avec une notion marque durablement la perception que nous en avons, nous aimerions que ce soit cette idée là que les élèves retiennent. Les élèves doivent faire l'expérience qu'il ne suffit pas qu'un programme accomplisse correctement sa tâche mais qu'il doit aussi l'accomplir rapidement et que la solution efficace à ce problème consiste à imaginer des nouvelles manières d'accomplir cette tâche, à inventer de nouveaux algorithmes. Il est bon de faire remarquer aux élèves que nombre de progrès technologiques actuels reposent uniquement sur un algorithme, c'est-à-dire sur une idée. L'exemple du PageRank devrait marquer les esprits et générer des vocations en algorithmique. Lors de l'étude d'un nouvel algorithme, les élèves devraient être amenés à se poser d'eux mêmes la question de son efficacité. Les algorithmes de tri sont intéressants à étudier à cet égard. Les élèves peuvent proposer eux mêmes des méthodes « naïves » avant d'étudier des algorithmes plus efficaces.

> **Un exemple : Le crible d'Érathosthène**
> *Établir la liste des nombres premiers inférieurs à n.*
> Les élèves programment une version de base sous la forme de deux boucles imbriquées « Pour i de 2 à n » et « Pour j de 2 à n » puis effectuent des mesures de temps d'exécution pour diverses valeurs de n. L'évolution en n^2 permet de prédire un temps astronomique pour $n = 10^9$. Des améliorations successives de l'algorithme, conduisant à des modifications mineures du programme, permettent de réduire ce temps à une poignée de secondes. Les couples (i,j) forment un carré de côté n. Les améliorations consistent à ne considérer qu'une partie de ce carré. La complexité peut être établie en évaluant l'aire de cette partie grâce à un calcul intégral.

Il est intéressant de montrer aux élèves que certaines idées (par exemple diviser pour régner) se retrouvent dans divers algorithmes et que par ailleurs un même algorithme peut être appliqué dans des situations très variées, y compris hors du champ de l'informatique (comme sortir d'un labyrinthe ou remplir son sac à dos).

Un autre objectif est de faire prendre conscience aux élèves que la machine fonctionne à une autre échelle que la nôtre. Pour programmer cette machine il est nécessaire de se placer à son échelle de temps et d'espace. Quoi de plus évident que de tracer un cercle ? Un compas suffit et nul besoin d'algorithme ingénieux pour cela. Quoi de plus évident pour une machine que d'extraire une racine carrée ? Une simple calculatrice ne nous donne-t-elle pas instantanément la réponse ? Cependant, lorsque l'on zoome fortement sur l'image on prend conscience qu'à l'échelle de la machine il n'y a pas de cercle. Il n'est question que d'allumer ou d'éteindre des pixels. Oui mais lesquels ? De plus, dans des scènes de jeu vidéo en 3D la carte graphique doit dessiner des millions de courbes en une fraction de seconde. À cette échelle de temps l'extraction d'une racine carrée dure un siècle. L'invention d'un algorithme qui permet d'éviter cette extraction transfigure la carte graphique. Pour finir, il nous paraît utile d'évoquer avec les élèves la notion de preuve d'un algorithme en reliant cette question aux conséquences potentielles d'un bug. Peut-on se satisfaire d'une vérification sur quelques données quand il s'agit d'envoyer une navette dans l'espace ? Étudier deux ou trois exemples de preuve d'algorithme serait aussi une occasion pour les élèves d'appliquer une démarche mathématique hors du cours de maths ce qui serait une grande première.

▹ L'approche par la conception de logiciel

Un exemple :

Une machine programmable lit une chaine de caractères sensée représenter un polynôme et affiche le polynôme dérivé. Écrire le programme.

Il s'agit de montrer aux élèves qu'une réflexion approfondie et une mise en place de concepts permettent de réaliser un programme modulaire, élégant, intelligible, évolutif :

- Représentation interne du polynôme dans la machine en concevant une structure de données idoine
- Conception d'un analyseur syntaxique
- Conception d'un module d'affichage de polynôme
- Conception d'une fonction de dérivation de polynôme

Il nous parait formateur de montrer que la complexité est tout de suite présente en informatique et que c'est notre aptitude à abstraire, à conceptualiser qui nous permet de maîtriser cette complexité. Il n'y a pas de place pour le bricolage.

La réalisation d'un programme informatique met en jeu de nombreuses compétences. Il nous semble intéressant qu'au travers des diverses activités proposées au cours de l'année, les élèves aient été confrontés à chacune des tâches et questionnements suivants : Étudier un problème et le découper en sous-problèmes. Définir des interfaces et se répartir le travail à l'intérieur d'un groupe. Concevoir une organisation du logiciel. Rechercher des solutions algorithmiques. Élaborer une solution et la rédiger dans un langage informel. Confronter cette solution à d'autres solutions possibles. Est-elle meilleure ? Dans quelle sens est-elle meilleure ? Plus efficace, plus économique, plus intelligible, plus élégante ? Rédiger cette solution dans un pseudo-code. Écrire et mettre au point le programme. Comment être certain qu'il ne reste aucun bug ? Écrire des programmes de test. Élaborer des jeux de données d'essais.

Cette expérience globale de la conception d'un logiciel peut être sensiblement enrichie par des activités de robotique. La réalisation et la programmation d'un robot de type LEGO-MINDSTORMS nous semblent présenter de nombreux intérêts :

- La grande motivation des élèves pour ce type de projet (testé actuellement à Toulouse dans le cadre d'ateliers scientifiques).
- La matérialisation de la notion de machine programmable.
- La programmation ne nécessite pas la connaissance d'un langage particulier car une interface graphique permet la définition de l'algorithme de fonctionnement du robot.
- C'est un bon prétexte pour étudier les notions de base du fonctionnement d'un ordinateur et de la circulation de l'information.
- Les élèves ont accès aux programmes sources et peuvent s'en inspirer pour réaliser leurs propres programmes.
- Un logiciel de simulation permet de tester une conception avant de charger un programme sur le robot.
- Une grande place est faite à l'imagination et à la créativité.

❑ Comment évaluer ?

▹ L'évaluation au baccalauréat

L'évaluation dans le cadre de l'examen du baccalauréat doit prendre en compte la nature particulière de cet enseignement possédant des aspects aussi bien conceptuels que pratiques. Cette spécialité étant enseignée l'année de terminale, les modalités de l'épreuve au baccalauréat auront un impact fort sur l'enseignement lui-même. Il convient donc de trouver des modalités qui découragent le bachotage.

On peut imaginer une épreuve orale en deux parties inspirée des TIPE en classes préparatoires. D'une part les candidats font un exposé oral sur un projet (type TPE) qu'ils ont réalisé au cours de l'année, d'autre part ils doivent étudier un sujet proposé par le jury. Quelques questions complémentaires permettent d'évaluer l'étendue et la solidité des connaissances des candidats.

Nous pensons que cette forme d'évaluation peut mettre les élèves dans une situation de défi très motivante pour avancer vers un objectif.

▹ L'évaluation au cours de l'année

Il nous semble que les évaluations au cours de l'année doivent être très limitées afin de ne pas dénaturer l'enseignement en un bachotage stérile.

Les travaux des élèves, évalués ou non, peuvent prendre de nombreuses formes. Ce peut être des exercices, des problèmes algorithmiques à résoudre, dans le style de ce que l'on propose habituellement en mathématiques. Ce peut être également des exposés oraux ou des compositions écrites sur un sujet donné. Ce peut être enfin des travaux pratiques de programmation ou des analyses de petits programmes ou d'algorithmes en lien avec les concepts étudiés.

❑ Les enseignants de la spécialité

▹ La formation des enseignants

Les professeurs doivent bénéficier d'une formation consistante en informatique afin qu'ils puissent enseigner cette discipline avec le recul nécessaire et ne pas réduire leur pratique au centre d'intérêt qui a motivé initialement leur engagement. Une formation purement académique ne nous semble pas être une réponse satisfaisante. Il serait bon que les enseignants qui suivent cette formation disposent également d'une structure pour échanger entre eux, partager leurs compétences complémentaires, réfléchir ensemble aux activités à mettre en place, élaborer des ressources pédagogiques, initier une véritable didactique de l'informatique dans l'enseignement secondaire. Les IREM pourraient fournir une telle structure. Il pourra alors se créer un réseau d'enseignants se connaissant bien et une dynamique pourra s'initier et perdurer au delà de cette année de formation.

▹ Un programme ouvert et évolutif

Cette spécialité n'a pas vocation à constituer un pré-requis pour des études ultérieures. L'esprit du programme compte davantage que son contenu particulier. Figer un contenu trop précis peut brider les enseignants sans pour autant garantir l'esprit. Le programme pourrait définir des objectifs clairs et laisser une grande autonomie et une large responsabilité aux enseignants. En contrepartie, ceux-ci pourraient remettre un bilan annuel, motivant leurs choix pédagogiques, les réussites et les difficultés rencontrées, les évolutions envisagées, etc. Un tel mode de fonctionnement permettrait une plus forte appropriation par chacun de son propre enseignement, favoriserait les réflexions et échanges de pratiques entre les enseignants et serait un moyen de faire évoluer avec souplesse ces pratiques, au travers de documents d'accompagnement de qualité régulièrement actualisés.

❑ Annexe A : Introduire les concepts de l'informatique à partir d'un questionnement autour du téléphone portable

Comment transmettre des sms, de la voix, des images ?	Principe de la numérisation. Convertisseurs analogique/numérique et numérique/analogique. Liens possibles avec le cours de physique sur les ondes sonores et électromagnétiques.
Comment augmenter le débit tout en préservant la bande passante ?	Algorithmes de compression (génériques ou non, avec ou sans perte d'information)
Comment se prémunir des erreurs de transmission ?	Détection et correction des erreurs Thème d'approfondissement : Code de Hamming
Comment être sûr que l'interlocuteur distant est bien celui qu'il prétend être et comment se parler en toute discrétion ?	Algorithmes d'authentification et de chiffrement de l'information. Thème d'approfondissement : Étude des principes généraux des algorithmes A3, A5 et A8 du GSM.
Que se passe-t-il concrètement au moment où on valide le numéro de l'appelé sur notre téléphone ? Et où cela se passe-t-il ?	Faire percevoir la très grande complexité des processus qui entrent alors en jeu et le nombre important de machines programmables concernées. Mais fondamentalement ce ne sont que des machines à registres rudimentaires. L'informatique consiste à maîtriser la complexité de ces milliards d'opérations élémentaires.
Qu'y a-t-il à l'intérieur d'un téléphone portable ?	Démontage. Qu'est-ce qu'un circuit ? À quoi cela sert-il ? Comment est-ce fabriqué ? La partie invisible du portable : le logiciel.
Comment mon téléphone fait-il pour choisir une antenne GSM ? Comment le réseau fait-il pour savoir où je suis ?	Rien n'est magique, il faut bien qu'il y ait des méthodes de localisation, que ces méthodes soient décrites par des algorithmes précis et que ceux-ci se traduisent par des logiciels embarqués dans le portable. Il peut être intéressant de laisser les élèves concevoir par eux-mêmes des procédures de localisation.
Comment le logiciel est-il organisé pour gérer la complexité des processus ?	Modèles en couches de protocole . API entre les couches. Thème d'approfondissement : Étudier les échanges de messages entre le portable et la station de base permettant l'établissement d'une communication
Comment est conçu le logiciel d'un téléphone portable ?	Gérer la complexité, méthodes de conception, des logiciels pour fabriquer des logiciels. Les conséquences des bugs. Méthodes de test. Des logiciels pour tester des logiciels.
Est-il suffisant qu'un algorithme soit correct ?	Les contraintes temps réel nécessitent des algorithmes non seulement corrects mais efficaces. Exemple : chiffrer ou déchiffrer la communication consiste à réaliser un simple OU exclusif entre deux informations. Complexité d'un algorithme.

❑ Annexe B : Quelques applications possibles

▹ Exemple 1

Étudier les différents modes de stockage numérique des images à travers les types de fichiers JPEG, TIFF, BMP ou Post-Script et réaliser une application qui permet de convertir une image couleur en tons de gris ou d'obtenir une image négative. À travers cette application, on peut aborder la structure des données, leur organisation, les algorithmes de compression, les notions de programmation structurée de base (conditions, boucles, procédures, fonctions, paramètres, récursivité). On peut également aborder le problème de la transmission par Internet qui nécessite des fichiers de faible volume d'où le problème de la compression d'image (fichiers JPEG).

▹ Exemple 2

Étudier la numérisation du son et le traitement qui permet, en combinant deux tableaux de coller deux morceaux de musique, de produire une image animée à partir d'un morceau de musique ou encore de mixer des sons.

▹ Exemple 3

À l'aide de logiciels standards comme Paint, étudier la représentation numérique des images et étudier les différents algorithmes de représentation des images géométriques. En déduire une application permettant de dessiner une image fractale.

❑ Annexe C : Quelques thèmes en lien avec les mathématiques

▹ Objets géométriques discrets

Problème : quelle définition donner à un ensemble de pixels rectiligne ?

Quelques définitions : droites diophantiennes, droites de recouvrement, droites de Réveilles ; propriétés de chacunes d'entre elles (connexité, intersection). Algorithmes de tracé et de reconnaissance des droites de Réveilles.

On cherche à faire comprendre que les mathématiques ne produisent pas seulement des théorèmes, mais aussi des concepts permettant de décrire la réalité et des définitions permettant d'obtenir des propriétés et des procédures de calcul. Les notions mathématiques investies relèvent de la géométrie très élémentaire (adjacence des pixels) et de l'arithmétique de TS (algorithme d'Euclide).

▹ Tomographie

Problème : reconstituer un objet 3D (en pratique, les activités seront en 2D) à partir d'informations sur des projections.

Activités : pratique de reconstitution dans des cas simples, programmation d'algorithmes exhaustifs, pratique et programmation d'algorithmes gloutons.

Ce problème conduit à la programmation d'algorithmes exhaustifs et permet de comprendre les limites de tels algorithmes. Il permet la pratique de problèmes ayant plusieurs solutions, et pose une question de conditions d'unicité des solutions.

▹ Codes correcteurs d'erreurs

Problème : repérer et corriger des erreurs dans les messages binaires.

Codes de parité, codes de Hamming, codes de Hamming étendus.

Activités : pratique de codage, de détection et de correction d'erreurs.

La description de ce problème et de ses solutions engage vers la nécessité d'une description précise des objets manipulés : distinguer « message non bruités » de « message correct » ; comprendre que les codes correcteurs ne donnent des affirmations correctes que pour des ensembles d'erreurs définis (de poids 1 pour les exemples considérés).

Les notions mathématiques investies relèvent de l'algèbre linéaire sur un corps à deux éléments qui devra être présentée de manière ad hoc et non sous une forme générale.

▹ Complexité des algorithmes

Problème : comparer l'efficacité d'algorithmes.

Descrition des paramètres intervenant dans la vitesse d'exécution d'un programme (machine, données, modalités de l'implémentation) ; définition de la complexité au pire et de la complexité en moyenne d'un algorithme, exemple de calculs.

Activités : algorithmes de primalité, de calcul de pgcd, d'exponentiation de matrices.

Prolongement dans le cadre des cours de mathématiques : vitesse de croissance de suites et de fonctions, suites définies par récurrence.

▹ Terminaison des algorithmes

Problème : comment s'assurer de la terminaison et de la correction d'un algorithme ?

Distinction entre expérimentation et raisonnement comme moyens d'obtenir des éléments de réponse. Utilisation d'invariants de boucle et de témoins.

Ce problème renforce l'intérêt des démonstrations en mathématiques.

▹ Cryptographie

Problème : comment protéger les données binaires contre des erreurs ?

Codes par permutation, code de Hill, . . .

Activités : pratique de codage, de décodage et de décryptage.

Ce problème classique permet d'illustrer des méthodes mathématiques dans des applications contemporaines : statistiques, complexité des algorithmes, arithmétique.

▹ Arithmétique des ordinateurs

Problème : comment les ordinateurs effectuent les calculs ? comment contrôler les erreurs ?

Arithmétique des entiers : représentations binaire et hexadécimale ; pratique et programmation d'algorithmes de calcul en entiers courts et en entiers longs ;

Arithmétique modulaire : représentation des entiers en systèmes de résidus ; problème de la détection d'over-flow.

Arithmétique des réels flottants: gestion des erreurs.

Le problème de la programmation re-motive l'étude de questions classiques (algorithmes de calcul et gestion des erreurs). Les notions mathématiques investies sont élémentaires excepté dans les algorithmes de résolution de systèmes d'équations modulaires. Les problèmes posés par la difficulté de contrôle des erreurs permettent des prolongements sur le rôle social de l'informatique (voir les exemples classiques chez J.M. Muller, F. Morain, . . .)