Projets algorithmique et complexité M1 info

Pr. Dominique Michelucci, Université de Bourgogne, Dijon
LE2I, UMR CNRS 5158
Dominique.Michelucci@u-bourgogne.fr

Introduction

Le rapport écrit doit être réalisé en Latex ; vous pouvez utiliser Lyx. Ce texte est rédigé en Latex. Le fichier latex et le fichier bib (pour la bibliographie) sont disponibles sur le site web du M1 informatique de Dijon.

Le rapport doit être écrit en français, et compter entre 25 et 35 pages. Il est inutile d'inclure les sources dans votre rapport.

Rédigez votre rapport avec soin. La langue naturelle est le premier langage de programmation et de modèlisation, aussi un rapport confus et criblé d'erreurs d'orthographe ou de français augure-t-il mal de votre talent d'informaticien.

Le programme ispell vérifie l'orthographe ; attention, il ne gère pas du tout les accords.

Donnez les références bibliographiques, les adresses des sites web que vous avez utilisés.

Pensez à commenter vos sources, sans excès.

Vous créerez une archive de votre projet, contenant les sources, le makefile, les fichiers d'exemples, votre rapport, le fichier de votre présentation orale, avec la commande

```
tar cvzf VOTRENOM.tgz votrerepertoire
```

que vous envoierez par courriel à `dmichel@u-bourgogne.fr` et `jjchab@u-bourgogne.fr` L'archive peut ne pas contenir les fichiers supports de votre présentation orale : vous pourrez la préparer après les vacances.

Pourquoi Ocaml est il imposé pour certains projets ?

- rien ne vous interdit de programmer d'abord votre projet avec votre langage favori.

- vous pouvez utiliser le style procédural en Ocaml.

- Vous maîtrisez (ou vous devriez) le C, C++, Java, ou d'autres langages procéduraux. C'est donc l'occasion d'apprendre un langage vraiment différent, et une façon différente de penser et de programmer.

- Une fois que vous saurez faire, programmer en ocaml est plus rapide, plus sûr (un programme Ocaml qui compile a plus de chances de fonctionner qu'un programme C++ qui compile...), plus simple (vous n'avez pas à gérer la mémoire avec des delete plus ou moins hasardeux), et plus amusant ; d'ailleurs les prouveurs actuels comme Coq sont développés en Ocaml, ainsi que des programmes critiques.

- il y a un seul compilateur de Ocaml, celui de l'INRIA (donc pas de problème d'incompatibilité).

- les programmes en Ocaml sont plus courts que les programmes en Java, en C++, etc

- ils sont davantage génériques, sans avoir à utiliser la grosse artillerie (template en C++, par exemple, qui sont, ou ont été interdits dans plusieurs entreprises).

- à l'exécution, les programmes en Ocaml (compilés) sont pratiquement aussi rapides que les programmes en C++, et plus rapides que les programmes Java.

- Microsoft et Dassault Systems font partie du consortium Ocaml. Le langage F# de Microsoft est directement inspiré de Ocaml : Ocaml est donc une excellente introduction à F#.

- De nombreux cours, tutoriels, livres sur ocaml, ou ocaml et OpenGl [CMP00, CMP98, Har07, Hic08] sont disponibles sur internet. Voir par exemple :
`http://www.freebookcentre.net/Language/Free-OCaml-Books-Download.html`

Cela va sans dire : n'attendez pas la dernière semaine pour commencer votre projet.

Voici une liste de projets possibles. Vous pouvez proposer vos propres projets.

1 Eternity II. 1 étudiant. Ocaml.

Vous trouverez sur Internet la description du jeu Eternity II. Vous générerez des puzzles aléatoires, vous en mélangerez les pièces, et vous résoudrez par une méthode de recherche arborescente avec retour en arrière (*backtrack*). Votre programme doit pouvoir résoudre des puzzles de taille 12 par 12, avec une dizaine de couleurs.

Dernière nouvelle : voir dans le répertoire UTILITES les fichiers relatifs à

Eternity.

2 Eternity II et SAT. 2 étudiants. Ocaml.

Vous trouverez sur Internet la description du jeu Eternity II. Vous récupérerez sur internet un programme de satisfaction de contraintes booléennes (le problème s'appelle SAT, ou 3-SAT ; des solveurs sont minisat ou picosat). Vous chercherez sur internet "eternity II SAT" pour trouver des articles présentant comment formuler le puzzle d'Eternity II comme un problème de satisfaction de contraintes booléennes ([Heu08, ABFM08] par exemple). Vous générerez des puzzles aléatoires du jeu Eternity II. Vous générerez le problème de contraintes booléennes correspondant, que vous sauverez dans un fichier ; vous appelerez le solveur de contraintes booléennes ; vous lirez le fichier solution. Vous testerez quelles tailles de puzzle sont solubles en un temps raisonnable (moins d'une 1/2 heure). Vous pouvez essayer plusieurs représentations du problème sous forme de contraintes booléennes, ainsi que plusieurs solveurs booléens.

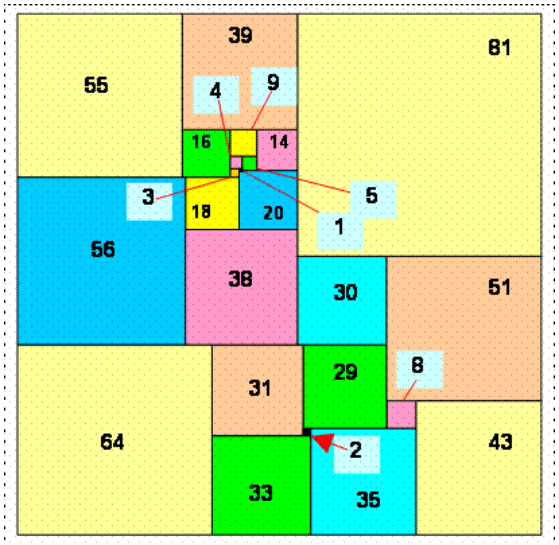
Dernière nouvelle : il existe un logiciel en ocaml qui résout le problème SAT : SAT-MICRO, "petit mais costaud", qui est dû à Sylvain Conchon, Johannes Kanig, Stéphane Lescuyer. Il est téléchargeable sur internet. Voir aussi les fichiers "conchon*" dans le répertoire UTILITES.

Dernière nouvelle : voir dans le répertoire UTILITES les fichiers relatifs à Eternity.

3 Pavage en carrés. 1 étudiant. Ocaml.

Soit un carré de coté n unités. Pour certaines valeurs de n, il est possible de paver le carré par des carrés plus petits, de côtés entiers et tous différents. Vous chercherez d'abord si n^2 peut s'écrire comme une somme de carrés d'entiers tous distincts. On sait que ce problème ne peut pas se généraliser en 3D et au delà.

Voir `http://villemin.gerard.free.fr/Pavage/CarrParf.htm`, d'où vient cette image :



4 Solitaire. 1 étudiant. Ocaml

Un carré de 2×2 cases est enlevé aux quatre coins d'un damier 7×7. Il reste donc $72 - 4 \times 4 = 33$ cases au damier. La case centrale est vide ; les 32 autres cases sont occupées par 32 pions, à raison d'un pion par case. Le but du jeu est d'enlever tous les pions, sauf un. Vous avez le droit de déplacer un pion vers une case vide, en le faisant sauter au dessus d'un autre pion que vous devez enlever ; les 3 cases concernées sont consécutives sur une rangée ou une colonne du plateau de jeu de 33 cases.

Attention : il y a beaucoup de configurations possibles, si bien qu'il faut vraisemblablement adapter l'algorithme de recherche standard (backtrack) ; par exemple, il peut être nécessaire d'encoder le damier sur un entier (mais il y a 33 cases).

Vous essaierez ensuite d'agrandir le damier ; jusqu'où pouvez vous aller ?

5 Calcul de π. 1 ou 2 étudiants. Ocaml

Empruntez le livre de Jörg Arndt et Christoph Haenel : "A la poursuite de Pi", éditions Vuibert, à la BU (cote 510/1232). Pour un étudiant, programmez 4 algorithmes parmi les 9 du livre. Pour 2 étudiants, programmez en 7. Vous téléchargerez les décimales de π pour vérifier vos résultats.

6 Taquin. 1 étudiant. Ocaml

Faire des mouvements aléatoires sur un taquin. Puis résolvez le par programme (et sans tricher). Jusqu'à quelle taille de taquin pouvez vous aller avant que votre programme ne devienne trop lent ?

7 Cube de Rubik. 2 étudiants. Ocaml

Appliquez des rotations aléatoires sur le cube de Rubik. Puis remontez le, par exemple avec la méthode des 3 étages.

Vous utiliserez lablgl ou lablglut.

8 Sudoku. 1 étudiant. Ocaml

Le programme de DM sur le site convertit le problème du Sudoku en un problème de coloriage de graphe. Vous pouvez faire mieux ! Vous utiliserez le fait que le 1 (ou tout autre numéro de 1 à 9) d'une ligne, d'une colonne, ou d'un bloc de 3^2 cases doit bien se trouver quelque part, et que parfois il n'y a plus qu'une case possible (ou aucune case possible, auquel cas la branche courante dans l'arbre de recherche est une impasse). Ce principe permet le plus souvent d'éliminer tout retour en arrière. Votre programme expliquera ses choix, en français.

Votre programme doit toujours être capable d'effectuer des tentatives et des retours en arrière, pour les sudokus les plus difficiles.

Vous comparerez votre programme (en mesurant le nombre de retour en arrière effectués) avec celui de DM disponible sur le site.

Vous testerez des sudokus de taille arbitraire. Jusqu'où pouvez vous aller ? Dans une seconde partie, vous essaierez de générer des jeux de Sudoku, avec juste les données suffisantes (un minimum de cases remplies) pour que la solution soit unique.

9 Recherche d'une chaîne de caractères. 1 étudiant. Ocaml

Programmer toutes les méthodes suivantes de recherche d'une chaine de caractères dans un fichier : la méthode naïve, Rabin-Karp (pas besoin d'utiliser

la librairie nums !), par automate fini déterministe, Knuth-Morris-Pratt. Ces méthodes sont présentées dans [CLRS01].

10 Algorithme de Johnson. 1 étudiant. Ocaml

Programmez l'algorithme de Johnson [CLRS01], qui calcule tous les plus courts chemins dans un graphe sans circuit négatif. Tracez la courbe donnant le temps de calcul en fonction de la taille du graphe, pour prouver que votre implantation est correcte.

11 Flot maximum. 2 étudiants. Ocaml

Programmez la recherche du flot maximum dans un graphe. Utilisez la méthode de Ford-Fulkerson modifiée par Karp, et deux méthodes de préflots. Ces méthodes sont présentées dans [CLRS01]. Vous pouvez utiliser le logiciel libre dotty pour visualiser le graphe. On ne demande pas d'interface interactive pour entrer ou modifier le graphe. Prévoyez un format de fichier commode pour décrire les données du problème (le graphe et les capacités minimales et maximales des arcs). Prévoyez aussi une interface (pas au sens IHM !) par appel de fonctions : flotmax legraphe, et les fonctions nécessaires pour construire un graphe et les capacités des arcs.

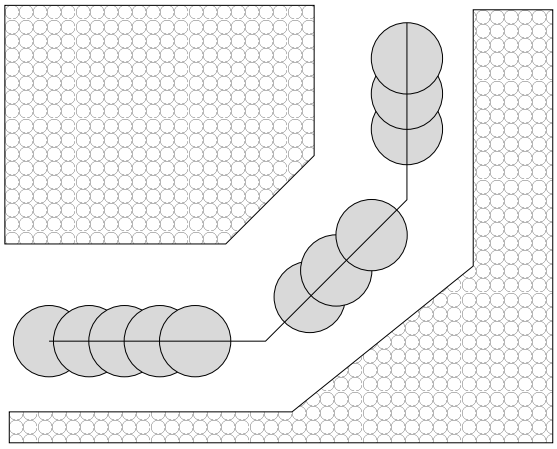
12 Couplage optimal. 2 étudiants. Ocaml

n étudiants doivent se répartir n projets dans le module Algorithmique et Complexité. Chaque étudiant classe les projets en $1, 2..., n$, par priorité décroissante. Trouver une correspondance un-un (biunivoque, ou bijective) qui minimise le mécontentement des étudiants est un problème de couplage optimal. Programmez l'algorithme hongrois [PS98], ou une autre méthode en temps polynomial. Toute méthode polynomiale calculant le flot de coût minimum convient.

13 Planification de trajectoire. 1 étudiant. Ocaml

Un robot est décrit par un ensemble de tiges articulées, chaque tige est décrite par un ensemble de disques. Les obstacles sont décrits, eux aussi, par un ensemble de disques. Trouver une trajectoire pour le robot d'une configuration à une autre. Une configuration est donnée par la position de la "tête" du robot (un disque particulier) et les angles entre les tiges. Une configuration est licite ssi le robot ne touche pas les obstacles et si les tiges ne se heurtent pas entre

elles. Méthode : échantillonner l'espace des configurations, avec une résolution de plus en plus fine. Dans les premières étapes, de petits heurts sont tolérables (par exemple en réduisant le rayon des disques du robot) ; quand une trajectoire grossière a été trouvé, elle est améliorée en échantillonnant davantage l'espace des configurations près de cette trajectoire. Vous utiliserez la notion d'écart : l'écart d'un point M par rapport au plus court chemin de A à B est $AM + MB - AB$ (où AM, MB, AB sont trois plus courts chemins).



14 Triangulation de surfaces implicites $f(x, y, z) = 0$. 2 étudiants. Ocaml

Vous généraliserez au 3D la méthode vue en 2D pour dessiner des courbes implicites données par une équation $f(x, y) = 0$. Pour chaque cube (voxel) élémentaire traversé par la surface d'équation $f(x, y, z) = 0$, vous partitionnerez le cube en 5 (ou 6) tétraèdres, et vous approcherez l'intersection de la surface avec un tétraèdre par un triangle ou un quadrilatère.

Vous utiliserez une arithmétique d'intervalles.

Ocaml fournit un lex et un yacc : ocamllex, ocamlyacc. Vous pouvez les utiliser pour analyser des fichiers de description de surface.

Pour la visualisation d'un ensmble de faces, vous pouvez utiliser, ou modifier, les programmes de DM dans
`http://math.u-bourgogne.fr/michelucci/OCAML/FIG/`

Voir aussi :

`http://www.opengl.org/resources/faq/technical/miscellaneous.htm`

15 Triangulation de surfaces algébriques implicites. 2 étudiants. Ocaml

C'est le même projet que le précédent, mais les surfaces sont algébriques et vous utiliserez les bases de Bernstein et la méthode de Paul Faget de Casteljau, qui est bien expliqué dans Wikipedia, dans le cas des fonctions à 1 et 2 variables ; l'extension au cas de 3 variables est immédiate. Vous utiliserez cet algorithme pour calculer les encadrements de $f(x, y, z)$, avec x, y, z dans une boite donnée.

16 OpenGL. 1 étudiant. Ocaml

Pour les passionnés de graphique. Améliorez le programme de modèlisation et de visualisation de DM, sur :
`http://math.u-bourgogne.fr/michelucci/OCAML/FIG/`
Enrichissez les objets primitifs disponibles. Définissez un langage de description de scènes (utilisez soit ocamlyacc et ocamllex, soit camlp5). Proposez quelques fichiers d'exemples.

17 Editeur interactif de courbes 2D. ocaml, 2 étudiants

Reprenez et améliorer les programmes disponibles sur le site `math.u-bourgogne.fr/michelucci/OCAML/BEZIER_INTERACTIF`. Ajouter d'autres types de courbes, par exemple les Bézier rationnelles, les Bsplines rationnelles. Demandez le livre de David Salomon : "Curves and surfaces for Computer Graphics" à D. Michelucci, pour trouver les définitions des courbes.

Votre programme doit permettre à l'utilisateur d'éditer ses propres courbes, et de s'initier aux courbes paramétrées ; vous permettrez à l'utilisateur de modifier interactivement les poids pour les courbes rationnelles.

La librairie graphics.cma est suffisante, mais vous pouvez aussi vous inspirer du programme ugly disponible sur `http://math.u-bourgogne.fr/michelucci/OCAML/FIG/`.

18 Editeur de fontes. Ocaml. 4 étudiants

Ce projet étend le projet précédent. Vous utiliserez l'éditeur pour créer votre propre police de caractères. Bien sûr, il faudra pouvoir colorier l'intérieur de chaque lettre, et sauvegarder des images bitmap avec diverses résolutions. Avec

cet éditeur, vous pourrez créer votre propre police de caractères. Il n'est pas demandé d'utiliser cette police pour votre rapport, mais vous pouvez étudier la question.

Bibliographie : le livre Metafont de Donald Knuth.

19 Métamorphose (*morphing*) entre 2 photos de visage. 2 ou 3 étudiants. ocaml

Votre programme permettra à l'utilisateur d'ajuster interactivement des sommets caractéristiques sur chaque visage (les coins des yeux, des sourcils, de la bouche, etc). Eventuellement, votre logiciel essaiera de retrouver automatiquement ces ajustements. Ces sommets sont ceux d'un maillage constitués de triangles, ou de triangles et de quadrilatère. Après prétraitement des deux visages, vous déformerez le premier maillage pour qu'il coincide avec le deuxième ; par exemple si le sommet P est en P_1 sur le premier maillage et en P_2 sur le deuxième, son mouvement sera décrit par $P(t) = P_1 + (1-t)P_2$. Bien sûr, il faut aussi interpoler les pixels à l'intérieur de chaque triangle (ou quadrilatère).

Le site `http://math.u-bourgogne.fr/michelucci/OCAML/RAY_ML_BERNSTEIN_MIGS_4/` contient des fichiers de manipulation d'images : image.ml, diaporama.ml, que vous pouvez utiliser et modifier. Vous pouvez aussi utiliser la librairie camlimages, disponible sur la toile.

Si vous êtes 3 étudiants sur ce projet, votre logiciel devra tenter de trouver automatiquement les points caractéristiques du visage.

20 Rotation de visages. 2 ou 3 étudiants. ocaml

Variante du projet précédent. A partir d'une photo de visage, vous essaierez de retrouver des coordonnées 3D plausibles pour les points caractéristiques de ce visage : l'image ne vous en donne que deux coordonnées par sommet. L'idée est d'ajuster le maillage sur la photo du visage, comme dans le projet précédent ; mais les sommets de ce maillage portent aussi une coordonnée de profondeur. Ensuite, vous texturerez ou vous peindrez chaque triangle du maillage 3D avec la partie correspondante dans l'image du visage. Soit vous utiliserez des textures openGl, soit vous subdiviserez les (assez gros) triangles du maillage du visage en triangles plus petits : par exemple, vous subdivisez chaque triangle du maillage en 4 triangles, et vous appliquez 2 fois cette subdivision ; et vous coloriez chaque sommet de ce maillage avec la couleur correspondante dans la photographie du visage. Vous visualiserez le visage en 3D avec openGl. Des programmes graphiques 3D en ocaml sont disponibles sur `http://math.u-bourgogne.fr/michelucci/OCAML/FIG/`.

Si vous êtes 3 sur ce projet, vous permettrez de copier une expression (neutre, sourire, peur, colère) d'un visage sur un autre. Pour cela, il faut numériser le visage d'une même personne (l'un d'entre vous) avec plusieurs expressions, en déduire les modifications sur le maillage canonique. Vous effectuerez ensuite les mêmes modifications sur le maillage ajusté à un autre visage. Plutôt que copier une expression, vous pouvez copier un mouvement (tirer la langue, hausser les sourcils, etc), le principe est le même.

21 Lancer de rayons. 2 étudiants. Ocaml

Vous visualiserez au moins : des cubes, des sphères, des cylindres, des cônes, et leurs opérations booléennes (union, intersection, différence). Bien évidemment, il faut pouvoir appliquer des transformations affines (rotations, translations, changements d'échelle sur les objets ou les groupes d'objets).

22 Tampon de profondeur. 1 étudiant. Ocaml

Programmer la méthode du tampon de profondeur (*Z buffer*) ; c'est la méthode utilisée par OpenGL. Essentiellement, il faut écrire une procédure d'affichage de polygones, qu'on supposera convexe.

23 Percolation. 1 étudiant. Ocaml

Une grille 2D est partitionnée en $n \times n$ carrés. Chaque carré est blanc avec une probabilité p, ou noir avec une probabilité $1 - p$. Il y a percolation quand il existe un chemin de cases noires d'une case dans la ligne du haut de la grille vers une case de la ligne en bas de la grille. Faites un programme pour mesurer empiriquement la probabilité de percolation en fonction de p et de n. Que constatez vous ? On peut se poser d'autres questions, telles que la probabilité d'un chemin blanc entre la colonne gauche et la colonne droite, l'influence de la connexité (un carré a-t-il 4 ou 8 voisins ?). Des applications de la théorie de la percolation sont l'analyse des risques de la propagation des incendies, épidémies, rumeurs, etc.

24 Animation de volutes de fumées. 1 étudiant. Ocaml

Pour les passionnés de graphique. De la fumée monte d'une cheminée. Modélisez et animez les volutes, représentées (par exemple) par un ensemble de particules. Par exemple, vous pourrez décrire chaque particule par sa durée de vie, sa position, un vecteur vitesse, peut-être un vecteur accélération, une masse, une couleur, une densité, une transparence... Ces particules sont plongées dans un champ de déplacements de l'air, représenté par une grille 3D cubique, avec un vecteur vitesse en chaque sommet de la grille. La fumée ne doit pas pouvoir passer à travers les obstacles (vous pouvez les représenter par un tableau de voxels). Utilisez la fonction bruit ou turbulence de Ken Perlin. Vous pouvez proposer un petit langage de scripts pour que l'utilisateur puisse faire facilement des essais.

Remarque : le plan (le point de vue) est fixe. Vous pouvez plaquer une ou des images en fond.

25 Textures 3D. 1 étudiant. Ocaml

Pour les passionnés de graphique. Modélisez des textures 3D.

26 Programmation linéaire. 1 étudiant. Ocaml

Empruntez le livre "Recipes in C" [PTVF92], et traduisez le programme du simplexe qu'il contient en Ocaml. Vos indices dans les tableaux commenceront à 0 (ils commencent à 1 dans le livre).

27 Multiplication de Strassen. 1 étudiant. Ocaml

La méthode de Strassen permet de multiplier des matrices carrées n par n en moins de temps que $O(n^3)$. Vous programmerez cette mltiplication, ainsi que l'inversion basée sur la mutiplication rapide de Strassen [CLRS01]. Vous comparerez les vitesses de la multiplication de Strassen et de la multiplication usuelle pour différentes valeurs de n. Vous testerez aussi la précision du calcul de l'inverse (la différence entre MM^{-1} et l'identité).

Remarque : votre programme ne doit pas supposer que n est une puissance de 2.

28 Interpolation. 1 étudiant. Ocaml

Vous calculerez la courbe algébrique $f(x, y) = \sum_i \sum_j a_{ij} x^i y^j = 0$ $(i + j \leq d)$ de degré d donné qui passe, ou approche, un ensemble donnés de points dans le plan. Vous utiliserez une résolution aux moindres carrés [CLRS01, PTVF92]. Vous afficherez la courbe et les points. Les points peuvent être désignés avec la souris. Vous pouvez aussi générer des points sur une courbe connue (cercle, ellipse), pour vérifier que vous retrouviez bien la courbe en question. Vous chercherez ensuite un autre type d'interpolation, par exemple par noyau gaussien (gaussian kernel). Mots clefs pour la recherche sur internet : gaussian kernel, radial basis function.

29 Algèbre linéaire. 1 étudiant. Ocaml

Empruntez [PTVF92] à la BU ; il est aussi disponible sur internet. Vous traduirez en Ocaml les fonctions effectuant les décompositions LUP et SVD, le calcul de l'inverse, la résolution de systèmes linéaires. Vos tableaux doivent avoir 0 en premier indice (et non 1, comme dans [PTVF92]). Votre module Ocaml doit pouvoir être utilisé facilement par un programmeur en Ocaml.

30 Programmation linéaire. 1 étudiant. Ocaml

Programmez la méthode du simplexe en 2 passes qui est décrite dans la dernière version du livre "Introduction à l'algorithmique", empruntable à la BU. Attention, il vous faudra corriger une petite erreur.

31 Compression de fichiers. 1 étudiant. Ocaml

Programmer 2 méthodes de compression de fichiers, ainsi que les méthodes de décompression correspondantes, par exemple la compression de Huffman, ainsi qu'un codage arithmétique.

Voir `http://en.wikipedia.org/wiki/Huffman_coding`

32 Méthode de Dijkstra. 1 étudiant. Ocaml

Programmez la recherche du plus court chemin par la méthode de Dijkstra, en testant plusieurs structures de données pour la queue de priorité : un tas (heap), tel que les sommets connaissent leur position dans le tas (le tas est mis

à jour quand la distance du sommet à la source diminue), un tas de Fibonacci [CLRS01], un tas binomial [CLRS01].

Vous implanterez un module avec foncteur pour chacune des structures de données. Il faut pouvoir passer en paramètre un module qui fournit une fonction de comparaison (et peut-être d'autres petites choses). Pour l'interface (au sens de signature), vous pouvez vous inspirer des modules Set, Hashtabl de Ocaml.

33 Stéganographie. 1 étudiant. Ocaml

Vous utiliserez le logiciel convert, ou xv, sous linux pour convertir les images dans des formats de fichiers facilement lisibles. Vous pouvez utilisez la librairie camlimages.

Dans une image RVB, il est souvent possible de modifier, sans effet visible, le bit de poids faible de l'octet des composantes rouge, verte, bleue. Une méthode de stéganographie consiste à sacrifier ces bits et à les replacer par ceux d'un texte encrypté (ou d'une image binaire encryptée) par la méthode RSA, ou n'importe quelle autre méthodevue en cours de cryptographie. Ocaml fournit une arithmétique sur des entiers (ou des rationnels) de longueur arbitraire ; vous l'utiliserez.

Ecrire le programme d'encryptage et de décryptage. Les clefs publiques et privées seront contenues dans des fichiers.

Vous aurez besoin de calculer des grands entiers premiers p et q. La clef publique est le produit $n = pq$. La clef secrète est le couple (p, q). Vous utiliserez le test probabiliste de primalité donné en [CLRS01].

Erreur à ne pas commettre : encrypter chaque bit (ou chaque octet) séparément... Si le message à encrypter est court, il faut le compléter avec du bruit.

34 FFT et multiplication de polynômes. 1 étudiant. Ocaml

Lire le chapitre consacré dans [CLRS01], et programmer cette méthode. Programmer 2 variantes : les coefficients sont des nombres flottants, et les coefficients appartiennent à un corps fini, les calculs sont effectués modulo un nombre premier. Dans ce dernier cas, vous pouvez utiliser la librairie num de Ocaml.

35 Arithmétique sur de grands entiers (1 étudiant)

Encadrant : JJC

Programmer une arithmétique $(+, -, \times, :, \mathrm{mod})$ sur de grands entiers. Programmer aussi le pgcd.

Pour la multiplication, utiliser la méthode de Karatsuba.

Avec cette arithmétique, programmer le test probabiliste de primalité (vous trouverez les détails par vous mêmes, par exemple dans : Introduction à l'algorithmique, de Cormen, Leiserson, Rivest.).

Programmer la méthode de factorisation rho de Pollard.

Programmer la recherche de la racine carrée modulo p (livre de Naudin et Quitté : algorithmique géométrique, à la bibliothèque : 518.4–1099), et le critère d'Euler.

S'il reste du temps : programmer la multiplication avec la transformée de Fourier (Introduction à l'algorithmique, de Cormen, Leiserson, Rivest.). Comparer les performances avec la méthode de Karatsuba.

Ces notions sont très utilisées en cryptographie (de nombreux livres y sont consacrés).

36 Coloration de graphes (2 étudiants)

Encadrant : J.-J. Chabrier.

Programmer et optimiser votre propre méthode de coloration de graphe. Testez la sur différents graphes ; par exemple, sur divers graphes aléatoires (par exemple, toute arête a une probabilité p d'exister ; évaluer le nombre minimal de couleurs, et les performances de votre méthode, en fonction de p et du nombre n de sommets), de complexité n croissante.

Applications : sudoku ou autres casse-têtes, allocation de ressources, par exemple allocation de registres en compilation.

37 Tracer de rayons sur des fractales (1 étudiant)

Encadrant : Christian Gentil.

Vous décrirez des fractales par des IFS ; un IFS est un ensemble de transformations affines contractantes (le déterminant, non nul, a une valeur absolue

inférieure à 1). Visualisez la fractale par lancer de rayons, et (pour comparer et corriger vos erreurs) par la méthode de marche aléatoire en openGl.

Pour le lancer de rayons, il faut d'abord calculer une sphère englobant l'attracteur ; ensuite l'intersection entre un rayon et une sphère S est l'ensemble vide si le rayon ne coupe pas la sphère, sinon c'est le centre de la sphère si la sphère est suffisamment petite, et sinon c'est la réunion des intersections entre le rayon et $t_1(S), \dots t_k(S)$ où les t_i sont les transformations de l'IFS. Vous pouvez faire de petits films d'animation, avec les fractales classiques (Sierpinski, Menger, fougère, arbre). On ne vous demande pas d'écrire un modeleur interactif de fractales. Un rendu élémentaire suffit : aucun effet spécial (réflexion, réfraction, ombre...) n'est demandé.

38 Factorisation d'entiers. 1 étudiant. Ocaml

Programmer 2 méthodes de factorisation des entiers : la méthode Rho de Pollard [CLRS01], et une réduction au problème SAT (satisfiabilité d'une formule logique).

Pour cette réduction, il faut multiplier, en base 2, $x = x_k \dots x_0$ par $y = y_k \dots y_0$, pour que $xy = n$, où n est le nombre à factoriser, et où les x_i et y_i sont des booléens inconnus. Vous avez le droit d'utiliser des variables auxilliaires (des retenues pour les additions, par exemple), pour trouver la formule logique correspondante. Bien sûr, vous devez utiliser un nombre raisonnable d'inconnues, polynomial en $\log n$. Vous utiliserez un solveur de contraintes booléennes tels que minisat ou picsat, téléchargeable gratuitement sur internet. Tous ces solveurs utilisent le même format de fichier. Vous pouvez demander de l'aide à Olivier Bailleux. Il doit être possible de factoriser un nombre inférieur à 1000.

Dernière nouvelle : il existe un logiciel en ocaml qui résout le problème SAT : SAT-MICRO, "petit mais costaud", qui est dû à Sylvain Conchon, Johannes Kanig, Stéphane Lescuyer. Il est téléchargeable sur internet. Les articles sont dans le répertoire UTILITES.

39 Métamorphose 2D (*morphing*). 1 étudiant. Ocaml.

Faites votre propre logiciel de métamorphose *"morphing"*.

Vous utiliserez le logiciel convert, ou xv, sous linux pour convertir les images dans des formats de fichiers facilement lisibles. Vous pouvez utilisez la librairie camlimages.

Votre logiciel doit permettre de désigner des points clés (A_i et B_i) en correspondance sur les deux images, par exemple les coins des yeux ou de la bouche

pour des images de visage. Ces points clés sont les sommets de deux maillages en triangles (ou en quadrilatéres), un maillage par image. La topologie des 2 maillages est identique, seules changent les coordonnées des sommets A_i et B_i. Vous générerez ensuite une animation, c'est à dire une séquence d'images qui passera de la première image A à la seconde B ; cette séquence devra donner l'illusion d'une transformation continue. Vous tenterez plusieurs types d'interpolation. L'interpolation la plus simple est l'interpolation linéaire : le sommet du maillage intermédiaire $P_i(t) = (1-t)A_i + tB_i$ vaut A_i pour $t = 0$ et B_i pour $t = 1$, et t passe de la valeur 0 à la valeur 1 par pas de $1/20$, par exemple. Au temps t, la couleur d'un point M à l'intérieur d'un triangle $P_1(t), P_2(t), P_3(t)$ est déterminée en calculant les coordonnées barycentriques de M : si $M = u_1P_1(t) + u_2P_2(t) + u_3P_3(t), u_1 + u_2 + u_3 = 1$, alors la couleur de M est obtenue en interpolant entre la couleur de $M(0) = u_1A_1 + u_2A_2 + u_3A_3$ et de $M(1) = u_1B_1 + u_2B_2 + u_3B_3$. Vous essairez des interpolations plus sophistiquées, qui sont expliquées dans des articles sur la métamorphose *morphing* entre images, que vous trouverez sur internet.

40 Puissance 4. 1 étudiant. Ocaml

Programmer le jeu Puissance 4 ; l'utilisateur doit pouvoir jouer contre l'ordinateur ; *ne perdez pas votre temps avec l'interface graphique : ce n'est pas le problème.* Programmer un minimax avec élagage alpha-beta. Le principe est donné dans wikipedia. Un module implantant le minimax est fourni dans [CMP00, CMP98], disponible librement sur internet (ou à la BU).

41 Morpion. 1 étudiant. Ocaml

Programmer le jeu du morpion sur une grille de 8×8 sommets. L'utilisateur doit pouvoir jouer contre l'ordinateur ; *ne perdez pas votre temps avec l'interface graphique : ce n'est pas le problème.* Programmer un minimax avec élagage alpha-beta. Le principe est donné dans wikipedia. Un module implantant le minimax est fourni dans [CMP00, CMP98], disponible librement sur internet (ou à la BU).

42 Othello (Reversi). 1 étudiant. Ocaml

Programmer le jeu Othello, aussi appelé Reversi. L'utilisateur doit pouvoir jouer contre l'ordinateur ; *ne perdez pas votre temps avec l'interface graphique : ce n'est pas le problème.* Programmer un minimax avec élagage alpha-beta. Le principe est donné dans wikipedia. Un module implantant le minimax est fourni

dans [CMP00, CMP98], disponible librement sur internet (ou à la BU).

43 Complétion de Knuth-Bendix. 1 étudiant. Ocaml

Programmer l'algorithme de complétion de Knuth-Bendix, uniquement sur les mots (et pas des arbres, des expressions). *Aucune interface graphique n'est demandée.*

Voir `http://www.vermi.fr/blog/files/RapportKB.pdf`, wikipedia, etc. Vous utiliserez l'exemple de wikipedia.

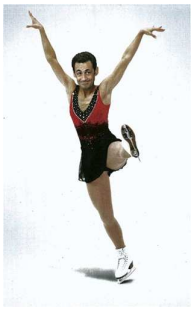
Vous générerez ensuite tous les mots non réductibles, d'une longueur inférieure à un seuil donné.

Remarque : les "lettres" des mots peuvent être des chaines telles que `x'` ou `x''`.

Par curiosité, vous essaierez les règles du cube de Rubik. Il y a 6 rotations (éventuellement 12). Outre que toutes les rotations sont d'ordre 4, chaque rotation peut être remplacée par les rotations des autres faces. Vraisemblablement, l'algorithme doit saturer la mémoire.

44 Photomontage. 1 étudiant. Ocaml

Faites votre logiciel de photomontage. Par exemple, votre logiciel doit pouvoir extraire une région polygonale d'une image (par exemple un visage) et la coller sur une autre image, comme sur l'image ci-dessous (dont je ne connais pas la provenance). Amusez vous, mais dans les limites de la décence, et ne mettez pas en danger la carrière de vos enseignants.

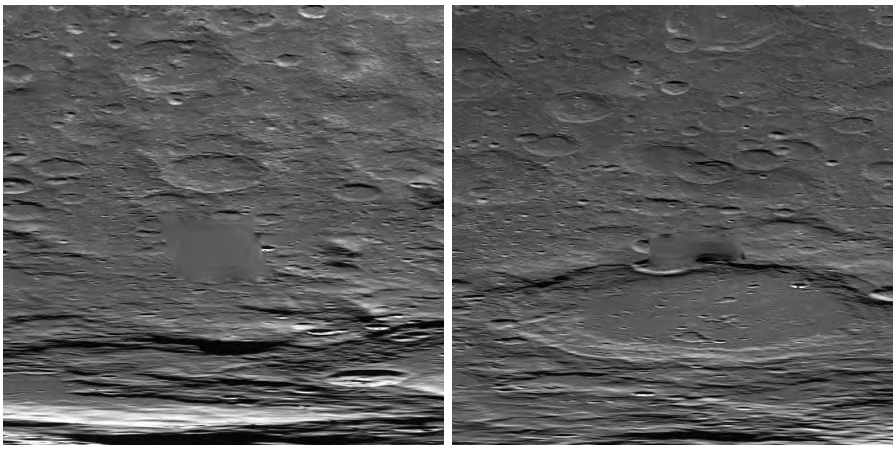


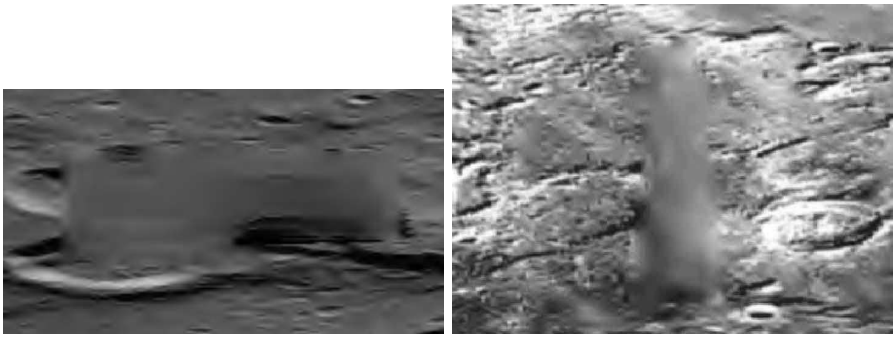
45 Traitement d'images. 1 étudiant. Ocaml

Vous utiliserez le logiciel convert, ou bien xv, sous linux pour convertir les images dans des formats de fichiers facilement lisibles. Vous pouvez utilisez la librairie camlimages.

Empruntez un livre de traitement d'images à la bibliothèque. Programmez les filtres usuels de traitement d'images, par exemple le filtre de Sobel, Laplace, etc pour reconnaître les contours. Application : tentez de reconnaître les zones floues, ou floutées, comme sur ces images du sol lunaire dues à la NASA.

Remarque 1 : les photos récentes de Mars sont elles aussi floutées. C'est moins évident que pour les anciennes photos lunaires. D'un autre coté, des méthodes de "déflouage" (en fait de déconvolution) peuvent être appliquées sur les photos de Mars.





46 Traitement d'images. 1 étudiant. Ocaml

Vous utiliserez les logiciels convert ou xv sous linux pour convertir les images dans des formats de fichiers facilement lisibles. Vous pouvez utilisez la librairie camlimages.

Empruntez un livre de traitement d'images à la bibliothèque. Programmez les filtres usuels de traitement d'images, par exemple le filtre de Sobel, Laplace, etc pour reconnaître les contours, ainsi que la transformée de Hough, qui détecte les segments de droite. Application : détectez et surlignez les segments de droite sur les images (par exemple dans les façades des immeubles), et les trainées des avions dans le ciel, comme sur les images suivantes (région de Saint-Etienne, ou Dijon, septembre 2009) :

Les 2 images ci-dessous proviennent du site `http://chemtrails-france.com`. Voir aussi `strange days strange skies`, le site du physicien Jean-Pierre Petit, [Bar02]...

47 Preuve combinatoire de théorèmes géométriques. 2 étudiants. Ocaml

Lisez la page `http://hexamys.free.fr/` sur les hexamys. Votre programme cherchera les hexamys pour prouver les théorèmes géométriques en 2D prouvés par R. Pouzergues. Vous pourrez ajouter la règle : si $ABCD$ sont 4 points distincts, où aucun triplet n'est aligné, alors les 3 points $AB \cap CD$, $AC \cap BD$, $AD \cap BC$ sont distincts et non alignés et ils sont distincts de $ABCD$). Vous utiliserez du chaînage avant, la règle précédente, et la règle de l'hexamys. Selon cette dernière, si on appelle hexamys un hexagone dont les cotés opposés se coupent selon 3 points alignés, alors toute permutation de l'hexamys est aussi un hexamys ; on peut donc déduire des alignements d'autres alignements. Les faits gérés par votre moteur de chainage avant seront : tel triplet de points est aligné / inconnu / non aligné. Vous aurez aussi besoin, vraisemblablement, d'une liste de points initiaux dont vous savez qu'ils sont tous différents. Vous pourrez gérer un dessin exemple du théorème, et éventuellement vous en servir pour élaguer la recherche. La librairie graphx de M. Quercia permet de sauvegarder des graphiques sous différents formats, tel postscript ou xfig, que vous pourrez ensuite insérer dans votre rapport.

48 Géométrie dynamique. 1 étudiant. Ocaml

Téléchargez GeoGebra (ou un autre lgiciel de géométrie dynamique), jouez un peu avec, avec les théorèmes de Pappus, Desargues, Pascal (soient 6 points sur un cercle ou une autre conique ; alors les cotés opposés se coupent selon 3 points alignés ; cette propriété reste vraie lorque l'on modifie la position d'un point sur le cercle). Ensuite, programmez votre propre logiciel de géométrie dynamique, et utilisez le pour prouver une dizaine de théorèmes géométriques, et tracer quelques lieux géométriques (ex : la position d'un point d'intersection quand un point "de base" se déplace le long d'une droite ou d'un cercle).

Il vous faudra représenter par un arbre les relations de dépendances entre les éléments géométriques (points, droites, cercles). Les éléments aux feuilles de l'arbre sont des éléments de base, qui ne dépendent d'aucun autre. Les noeuds représentent les droites par 2 points (2 fils du noeud), les cercles par 3 points (3 fils du noeud), les points d'intersection entre 2 éléments fils du noeud, etc.

Un élément de base peuvent être sélectionnés et déplacés ; les éléments de l'arbre sont mis à jour.

49 Grapheur 2D. 2 ou 3 étudiants. ocaml

Ce projet généralise le grapheur vu avec Lex et Yacc en licence 3. Cette fois ci, les courbes 2D dans le plan xy sont décrites par un système d'équations et d'inéquations, par exemple $x(1+t^2)-(1-t^2)=y(1+t^2)-2t=0, -1 \leq t \leq 1$ décrit un arc de cercle (certes, il y a plus simple !). Il y a 1 équation de moins que d'inconnues. Vous afficherez ces courbes, plus précisément la projection de ces courbes sur un plan, par exemple le plan x, y. Les 2 variables sur lesquelles vous projetez doivent pouvoir être choisies par l'utilisateur.

Dans un premier temps, la méthode utilisée sera la subdivision de boites de $\mathbb{R}^n$, s'il y a n variables. Vous essaierez ensuite d'optimiser cette méthode. Par exemple, si la courbe est proche d'un segment dans une boite, vous cesserez la subdivision et afficherez le segment correspondant. Nous avons vu en TP comment dériver un dag et l'évaluer par intervalles.

Pour 2 étudiants, les équations seront écrites en ocaml, en définissant les opérations `+&, -&`, etc, comme il a été fait en TP. Pour 3 étudiants. vous écrirez un analyseur syntaxique, avec soit ocamlyacc et ocamllex, soit avec camlp5 (lire la documentation et les exemples).

50 Contraintes géométriques. 1 ou étudiants. Ocaml

Vous résoudrez des systèmes de contraintes géométriques en 2D. Elles spécifient des incidences, des distances ou des angles entre des éléments géométriques (points, droites, cercles). L'utilisateur doit pouvoir positionner approximativement des éléments géométriques : c'est l'esquisse. L'utilisateur dit quels éléments ou coordonnées sont modifiables, ou non. Votre logiciel convertira les contraintes en équations (utilisez des DAGs). Vous partirez de l'esquisse pour effectuer des itérations de Newton. La méthode de Newton se généralise quand le jacobien n'est pas inversible, en utilisant une décomposiion en valeurs singulières (SVD : singular Value Decomposition) [PTVF92].

51 Dessin de fractales et d'attracteurs étranges. 1 étudiant. Ocaml

Dessiner les attracteurs de Rossler, de Hénon, de Lorentz. Dessiner les fractales de Mandelbrot, Julia. Votre logiciel doit permettre de faire des agrandissements. Faites une animation avec ces dernières, par exemple un zoom sur la bordure de l'ensemble de Mandelbrot, ou bien une animation sur l'ensemble de Julia : ce dernier est paramétré par un nombre complexe, qui peut parcourir une courbe dans le plan complexe (par exemple suivre –plus ou moins– le contour de l'ensemble de Mandelbrot). Internet est très riche dans ce domaine, mais lisez aussi quelques livres de la BU sur le sujet [PR86, PSF+89].

52 Dessin de plantes. 1 étudiant. Ocaml

Dessinez quelques plantes. Les livres [PL04, DL05] sont disponibles sur internet.

53 Isomorphisme de graphes. 1 étudiant. Ocaml

Ecrire une fonction qui décide si 2 graphes non orientés sont isomorphes. De plus la fonction retourne une liste des isomorphismes (il peut y en avoir plusieurs). Un isomorphisme est un appariement des sommets des deux graphes. Vous testerez pour de grands graphes réguliers (hypercube, hypertore).

54 Coloriage de graphes. 1 étudiant. Ocaml

Programmez 2 ou 3 algorithmes classiques de coloriage des sommets d'un graphe, pour un graphe et un nombre de couleurs donnés. On rappelle que deux sommets liés par une arête dans le graphe doivent avoir 2 couleurs distinctes. Vous programmerez des algorithmes exacts, et des méthodes approximatives (gloutonne, ou autre). Votre rapport devra aussi présenter des applications de ce problème (affectation).

55 Calcul des nombres de Ramsey. 1 étudiant. Ocaml

K_n est le graphe complet (avec toutes les arêtes possibles) avec n sommets. Quel que soit le coloriage en 2 couleurs des arêtes de K_6, il contient un K_3 monochrome. Par contre il y a des coloriages en 2 couleurs des arêtes de K_5 où aucun K_3 n'est monochrome. On dit que le nombre de Ramsey de 3 est 6 : un K_3 monochrome devient inévitable à partir K_6. Votre programme calculera les nombres de Ramsey de $3, 4, \ldots n$. Jusqu'où pouvez vous aller ? J-J. Chabrier encadre ce projet.

56 Lancer de photons. Ocaml. 2 ou 3 étudiants.

Pour les passionnés de synthèse d'images, uniquement...

Suivre les flots de photons à partir des sources lumineuses dans une scène décrite par un arbre de construction [Jen05, MJ03, CWH93, Gla94], ou, plus modestement, un ensemble de facettes planes ou triangulaires. Compter les photons passant sur chaque surface (attacher un ensemble de tiroirs à chaque surface). En déduire une image.

Certains des livres cités sont disponibles à la BU, ou sont disponibles (partiellement) sur internet.

Mot clef pour une recherche sur internet : "photon mapping", "radiosity", "wikipedia Radiosity Image Synthesis".

On ne demande pas d'interface graphique pour le modeleur géométrique : vous pouvez vous contenter de toujours la même scène.

Le but de ce projet n'est pas de produire des images réalistes, mais d'écrire une introduction sur ce problème et le principe de la méthode, et de faire des images illustrant cette méthode : par exemple, les impacts des photons seront visualisés par des points dans des images en niveau de gris, ou le chemin des photon sera visualisé par des segments. Programmer la méthode vous permettra de bien comprendre les difficultés du problème, et la méthode. Encore une fois, il ne vous est pas demandé de rivaliser avec les professionnels.

57 Yacc. 2 étudiants. Ocaml

Réécrire un programme d'analyse syntaxique. Les grammaires sont décrites dans des fichiers, par exemple :

```
E : empty
E : F PlusOuMoins E
E : '(' E ')'
F : nombre
F : nombre MultOuDiv F
PlusOuMoins: '+'
PlusOuMoins: '-'
MultOuDiv: '*'
MultOuDiv: '/'
```

Vous devez engendrer un programme qui, quand il reçoit l'entrée : nombre * nombre * nombre + nombre * nombre + nombre * (nombre + nombre) doit rendre (et afficher) l'arbre syntaxique correspondant ; les feuilles sont des lexèmes ; les noeuds sont étiquetés par des numéros de règles et des noms de symboles non terminaux. Les méthodes nécessaires sont (très bien) expliquées dans [ALSU06]. Vous pouvez utiliser la librairie dotty pour afficher les arbres syntaxiques produits. La compréhension de dotty n'est pas requise.

Les algorithmes sont présentés dans les livres [ALSU06] traitant de la compilation et des compilateurs ; voir le rayon sur ce thème à la BU.

Dernière nouvelle : comme le projet réalisé en 2009–2010 est disponible sur internet, vous ne pouvez plus choisir ce projet.

58 Prolongation de suites. Ocaml. 1 étudiant

Une suite entière $a_i, i \in \mathbb{N}$ est donnée par ses 10 ou 20 premiers termes. On l'encode par une série formelle $\sum_i a_i x^i$. Ecrire un algorithme qui trouve la loi de formation de la série, et qui calcule les termes de rang quelconque.

Par exemple, à partir des premiers termes de la suite de Fibonacci : $1, 1, 2, 3, 5, 8, 13, 21$, cette heuristique doit trouver la relation $\psi(x)(1 - x - x^2) = 1$. Cette dernière série $\phi(x)$ a pour coefficient du monome x^i le i ème terme de la suite de Fibonacci. Remarquez que :

$$\begin{array}{rcccccccc}
\phi(x) & = & 1x & +1x^2 & +2x^3 & +3x^4 & +5x^5 & +8x^6 & +\dots \\
x\phi(x) & = & & 1x^2 & +1x^3 & +2x^4 & +3x^5 & +5x^6 & +\dots \\
x^2\phi(x) & = & & & 1x^3 & +1x^4 & +2x^5 & +3x^6 & +\dots \\
(1-x-x^2)\phi(x) & = & 1 & & & & & &
\end{array}$$

Il faut donc deviner une relation linéaire entre les vecteurs colonnes des coefficients de la suite et ses décalages successifs. Ici il y a la même relation dans tous les vecteurs colonnes : $(1, 0, 0)$, $(1, 1, 0)$, $(2, 1, 1)$, etc. 3 est le plus petit nombre pour lequel il y a une telle relation, et vous essaierez dans l'ordre $k = 1, 2, 3, \dots$ jusqu'à trouver la valeur du degré.

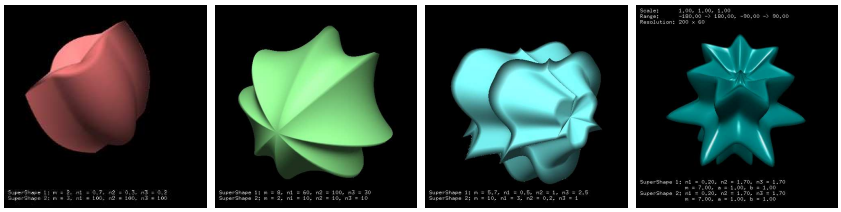


FIG. 1 – Images de "supershapes", extraites du site de Paul Bourke

Plus généralement, votre méthode doit trouver une relation $\psi(x)q(x) = p(x)$ où p et q sont des polynômes en x. Ensuite, en utilisant la méthode d'exposant rapide (cf IA), vous calculerez les termes quelconques de la suite.

Cette méthode fonctionne sur les suites polynomiales, exponentielles, périodiques (la période doit être plus petite que le nombre de termes disponibles, bien évidemment).

Mots clefs pour une recherche sur internet : Sloane, inverseur de Simon Plouffe...

59 "Supershape". 1 étudiant. Ocaml, lablgl

Pour un passionné de graphique. Comme ce projet est facile, l'étudiant devra soigner l'interface graphique de son projet. En s'inspirant des fichiers ugly.ml, il réalisera sa propre version du programme "supershape" de Paul Bourke, sur le site :

`http://local.wasp.uwa.edu.au/~pbourke/geometry/supershape3d/`

La figure 1 montre des images de "supershapes", sur le site de Paul Bourke.

60 Graphes d'intervalles

Tout ensemble d'intervalles $S_i = [u_i, v_i]$ génère un graphe d'intervalles ; chaque intervalle génère un sommet, et une arête lie deux sommets quand les intervalles correspondants se chevauchent.

Tous les graphes ne sont pas des graphes d'intervalles. Vous programmerez l'algorithme qui décide si un graphe est ou non un graphe d'intervalles.

La recherche d'une clique maximale et d'un stable maximal dans un graphe est un problème difficile en général. Ces problèmes sont polynomiaux pour les graphes d'intervalles. Vous programmerez ces méthodes.

Application : vous gérez la programmation d'un gymnase (ou d'un stade, ou d'un théâtre...). Des évènements ont lieu pendant des intervalles fixés, connus à l'avance. Le gymnase ne peut accueillir qu'un seul événement en même temps. Quels évènements accueillir pour maximiser l'occupation du gymnase ? ou, autre problème, comment accueillir le maximum d'évènements ?

Vous pouvez demander conseil à Olivier Togni.

61 Problème de ramassage dans une ville virtuelle - Algorithme des Fourmis

Pour 4 étudiants. En Java.

Voir fichier word ou PDF dans le même répertoire.

Ce projet est encadré par Mr J-J. Chabrier.

62 Plus longue séquence monotone croissante. 1 étudiant. ocaml.

Comme le titre l'indique : vous calculerez la plus longue séquence monotone croissante $a_{i_1} \leq a_{i_2} \leq a_{i_L}, i_1 < i_2 \dots i_L$ dans une séquence $a_1, a_2 \dots a_n$.

63 Résoudre $x^2 + Ny^2 = P$ dans $\mathbb{N}$ (1 personne)

Encadrant : DM.

Ecrire un programme Ocaml pour résoudre $x^2 + Ny^2 = P$ dans $\mathbb{N}$ avec P un nombre premier. De plus $0 < N < P$. La méthode nécessite le calcul des racines carrées modulo P et la réduction des bases de réseaux de $\mathbb{Z}^2$ par l'algorithme de Gauss.

Un cas particulier célèbre est $N = 1$, ce qui donne l'équation $x^2 + y^2 = P$. Selon le théorème de Noël dû à Fermat, tout nombre premier P est somme de 2 carrés ssi $P = 2$ ou si P est égal à 1 modulo 4 (25-12-1640, lettre de Fermat à Mersenne). Les preuves de ce théorème ne sont pas toujours constructives, ou alors trop lentes, comme la méthode triviale (essayer toutes les valeurs possibles, elles sont en nombre fini).

Exemple : pour le petit nombre de Mersenne $P = 2^{31} - 1$, et $N = 6$, la solution est $(x, y) = (44029, 5901)$. Quelle est elle pour $N = 3, P = 2^{521} - 1$?

Algorithme L'équation est d'abord résolue dans $\mathbb{Z}/P\mathbb{Z}$, autrement dit on résout : $x^2 + Ny^2 = 0 \bmod P \Rightarrow x^2 = -Ny^2 \bmod P$. Si N n'est pas un carré dans $\mathbb{Z}/P\mathbb{Z}$, il n'y a pas de solution dans le corps fini $\mathbb{Z}/P\mathbb{Z}$, et a fortiori il n'y en a pas dans $\mathbb{N}$. Supposons que R soit une racine carrée de $-N$, modulo P. Alors les couples (x, y) solutions modulo P forment deux droites discrètes d'équation : $x = Ry$ et $x = -Ry$. Considèrons la droite $x = Ry$, contenant les points : $(0, 0), (R \bmod P, 1), (2R \bmod P, 2) \ldots, (kR \bmod P, k)$ pour $k \in \mathbb{Z}/P\mathbb{Z}$. La figure montre la droite pour $P = 13, N = 1, R = 5$. Ces points forment une partie d'un réseau de $\mathbb{Z}^2$. Ce réseau est obtenu en prolongeant le dessin hors du carré $[0, P]^2$. Chacun des points (a, b) de ce réseau est par construction tel que $a^2+Nb^2 = mP$, avec m entier positif ou nul (car N est positif). On sait résoudre l'équation dans $\mathbb{N}$ quand on sait trouver (a, b) tel que $m = 1$. Il suffit donc de trouver le sommet du réseau le plus proche de l'origine, et différent de l'origine. Soit ce point est tel que $m = 1$ et alors ce point est la solution cherchée ; soit ce point est tel que $m > 1$ et alors l'équation n'a pas de solution dans $\mathbb{N}$.

Or trouver le point d'un réseau (on dit aussi lattice, ou treillis, ou module) de $\mathbb{Z}^2$ le plus proche de l'origine est trivial quand on connait une base minimale (U, V) du réseau ; alors $U = (U_x, U_y)$ est le vecteur le plus court, donc (U_x, U_y) est le sommet le plus proche de l'origine, et $V = (V_x, V_y)$ est le plus court des vecteurs indépendants de U. Cette base est unique (aux signes près de U_x, U_y, V_x, V_y). Elle est calculée rapidement par une méthode due à Gauss. Il ne reste ensuite plus qu'à vérifier que $U_x^2 + NU_y^2 = P$ (et pas un multiple de P).

Résumons l'algorithme. Trouver R une racine carrée de $-N$ modulo P. Si $-N$ n'est pas carré, alors l'équation est sans solution dans $\mathbb{N}$. Calculer une base (pas forcément la plus courte) du réseau ; un premier vecteur de la base est $(R, 1)$; le second est le premier vecteur $(kR \bmod P, k)$ différent de (kR, k), pour $k \in \mathbb{N}$. Réduire cette base par la méthode de Gauss. Soit (U, V) la base minimale avec $||U|| \leq ||V||$. Si l'équat ion a une solution, c'est (U_x, U_y).

Réseaux de $\mathbb{Z}^2$, bases minimales, réduction de Gauss Un réseau de $\mathbb{Z}^2$ est l'ensemble des combinaisons linéaires à coefficients entiers relatifs d'une base de 2 vecteurs de $\mathbb{Z}^2$. Deux bases sont équivalentes ssi elles engendrent le même réseau. Pour les réseaux 2D, il existe une base minimale, contenant le vecteur le plus court, et le "second plus court" : le vecteur le plus court indépendant du premier (cela ne se généralise pas en dimension 4 et au delà). La méthode de Gauss la calcule ainsi. Soit (U, V) la base initiale, avec V le vecteur le plus long. Soit W la projection de V sur U. Soit $kU, k \in \mathbb{Z}$ le vecteur le plus près de W. Si $k = 0$, alors la base ne peut être réduite et est minimale. Sinon $W - kU$ est plus court que V et toujours indépendant de U. Recommencer sur la base $U, W - kU$).

Calcul d'inverse modulo P Théorème de Fermat : si P est premier, alors $x^{P-1} = 1 \bmod P$, pour tout x non nul. Donc l'inverse de x modulo P est

$x^{-1} = x^{P-2} \bmod P$. Utiliser l'exponentiation rapide, qui nécessite $O(\log P)$ opérations, sur des nombres entiers inférieurs à P^2

Autre méthode possible,de même complexité : utiliser l'algorithme d'Euclide étendu. Pour (a, b) donnés, il calcule (u, v) tel que $au + bv = \text{pgcd}(a, b)$. L'appliquer pour $(a, b) = (x, P)$. On obtient (u, v) tels que $xu + Pv = 1$ donc $x^{-1} = u$ modulo P.

Critère d'Euler a est un carré modulo P (P premier) ssi $a^{(P-1)/2} = 1 \bmod P$. Preuve : s'il existe x tel que $x^2 = a \bmod P$, alors $x^{P-1} = 1$ par Fermat ; or $x^{P-1} = a^{(P-1)/2}$.

Racine carrée dans $Z/P\mathbb{Z}$ Si $P = 3 \bmod 4$ et que a est un carré, alors sa racine carrée est $a^{(P+1)/4}$ modulo P. Si $P = 1 \bmod 4$, on utilise la méthode de Zassenhauss-Cantor (expliquée dans le livre de Naudin et Quitté : Algorithmique géométrique, empruntable à la bibliothèque en 51 8.4–1099). Voici un exemple. $P = 13$ et on veut α tel que $\alpha^2 = a = 10$. Pour tous les nombres $t + \alpha$ dans $\mathbb{Z}/P\mathbb{Z}$, on doit avoir, d'après fermat : $(t + \alpha)^{(P-1)/2} = \pm 1$. La méthode choisit des t au hasard dans $Z/P\mathbb{Z}$ jusqu'à en trouver un qui convienne, et calcule "formellement" $(t + \alpha)^{(P-1)/2} = u + v\alpha$: ils sont considérés comme des polynômes en α, dont les coefficients sont dans $\mathbb{Z}/P\mathbb{Z}$, et en tenant compte de : $\alpha^2 = a = 10$. On obtient dans cet exemple, selon les valeurs de t :

$t = 1 \Rightarrow (1 + \alpha)^6 = 12 + 0\alpha = -1$

$t = 2 \Rightarrow (2 + \alpha)^6 = 0 + 2\alpha = \pm 1$

$t = 3 \Rightarrow (3 + \alpha)^6 = 1 + 0\alpha = 1$

$t = 4 \Rightarrow (4 + \alpha)^6 = 0 + 11\alpha = \pm 1$

$t = 5 \Rightarrow (5 + \alpha)^6 = 0 + 2\alpha = \pm 1$

$t = 6 \Rightarrow (6 + \alpha)^6 = 7 + 12\alpha = 1$ ou 0 selon que $\alpha = 6$ ou 7.

$t = 7 \Rightarrow (7 + \alpha)^6 = 7 + 1\alpha = 1$ ou 0 selon que $\alpha = 7$ ou 6.

$t = 8 \Rightarrow (8 + \alpha)^6 = 0 + 11\alpha = \pm 1$

$t = 9 \Rightarrow (9 + \alpha)^6 = 0 + 2\alpha = \pm 1$

$t = 10 \Rightarrow (10 + \alpha)^6 = 1 + 0\alpha = 1$

$t = 11 \Rightarrow (11 + \alpha)^6 = 0 + 11\alpha = \pm 1$

$t = 12 \Rightarrow (11 + \alpha)^6 = 12 + 0\alpha = -1$

Il y a 3 cas. Le premier cas $u = 0, v \neq 0$ survient pour beaucoup de valeurs de t, ici $2, 4, 5, 8, 9, 11$; alors $v\alpha = 1$ (ou -1), donc $\alpha = \pm v^{-1} \bmod P$. Dans le deuxième cas, pour t dans 1, 3, 10, 12, il y a échec : $v = 0, u = \pm 1$ et on ne

peut calculer α. Enfin dans le troisième cas, $t = 6, 7$ égale α, mais ce cas ne se produit que 2 fois sur $P - 1$. En moyenne, 2 ou 3 essais de t suffisent pour se trouver dans le premier cas. Voir pg 316-321 du livre de Naudin-Quitté de 1992.

Référence Voir le livre de Ian Stewart : L'univers des nombres, chez Belin, Pour la Science, empruntable à la bibliothèque en 513–1006, pages 101–105, chapitre : Le théorème de Noël.

64 Arithmétique quadratique (1 personne)

Encadrant : DM

Programmer en Ocaml une arithmétique quadratique : elle permet de calculer de façon exacte avec des racines carrées de nombres positifs. Elle permet de prouver que $\sqrt{3 + 2\sqrt{2}} = 1 + \sqrt{2}$ par exemple. Elle permet de faire des calculs exacts avec les constructions géométriques à la règle et au compas. Vous utiliserez la librairie nums.cma d'Ocaml, qui fournit une arithmétique sur les rationnels et entiers longs.

65 Prolongation de suites (1 personne)

Encadrant : DM

Une suite entière $a_i, i \in \mathbb{N}$ est donnée par ses 10 ou 20 premiers termes. On l'encode par une série formelle $\sum_i a_i x^i$. "Formelle" signifie qu'on ne s'intéresse pas à la convergence de cette série, ni aux valeurs que peut prendre cette série. Une série formelle est pour un matheux ce qu'est une liste pour les informaticiens. Ecrire un algorithme qui trouve la loi de formation de la série, et qui calcule les termes de rang quelconque.

Par exemple, à partir des premiers termes de la suite de Fibonacci : $1, 1, 2, 3, 5, 8, 13, 21$, cette heuristique doit trouver la relation $\psi(x)(1 - x - x^2) = 1$. Cette dernière série $\psi(x)$ a pour coefficient du monome x^i le i ème terme de la suite de Fibonacci. Remarquez que :

$$\begin{array}{rcccccccc}
\psi(x) & = & 1x & +1x^2 & +2x^3 & +3x^4 & +5x^5 & +8x^6 & +\ldots \\
x\psi(x) & = & & 1x^2 & +1x^3 & +2x^4 & +3x^5 & +5x^6 & +\ldots \\
x^2\psi(x) & = & & & 1x^3 & +1x^4 & +2x^5 & +3x^6 & +\ldots \\
(1 - x - x^2)\psi(x) & = & 1 & & & & & &
\end{array}$$

Il faut donc deviner une relation linéaire entre les vecteurs colonnes des coefficients de la suite et ses décalages successifs. Ici il y a la même relation dans tous les vecteurs colonnes : $(1, 0, 0)$, $(1, 1, 0)$, $(2, 1, 1)$, etc. 3 éléments est le plus

petit nombre pour lequel il y a une telle relation, et vous essaierez dans l'ordre $k = 1, 2, 3 \ldots$ jusqu'à trouver la valeur du degré.

Plus généralement, votre méthode doit trouver une relation $\psi(x)q(x) = p(x)$ où p et q sont des polynômes en x. Ensuite, en utilisant la méthode d'exposant rapide (cf IA), vous calculerez les termes quelconques de la suite.

Cette méthode fonctionne sur les suites polynomiales, exponentielles, périodiques. Elle ne marche pas pour la suite des factorielles qui augmente "trop vite", plus vite que l'exponentielle. Variante : vous chercherez une relation entre les suites des dérivées successives : $\psi(x)$, $\psi'(x)$, $\psi''(x)$, etc. Cela permet de trouver une équation algébrique dont ψ est solution (à vérifier...).

66 Tas binomial (2 personnes)

Encadrant : JJC

Programmer en C (ou C++) ET en Ocaml une librairie de tas binomial (chap. 19 de IA). Attention : votre version doit être fonctionnelle, *i.e.* l'insertion ou la destruction ne doit pas modifier la structure de données mais en rendre une nouvelle, qui partage avec l'ancienne un maximum d'informations. Quelle est la difficulté en C (C++) ?

Utiliser votre librairie pour programmer la méthode de Dijkstra (en C (ou C++) ET en Ocaml.

Vous programmerez aussi la diminution d'une clef, et le test : un élément est il dans la structure, si vous en avez besoin dans votre programme de plus court chemin.

En Ocaml, vous utiliserez des foncteurs, et le type des éléments et la fonction de comparaison sera passée en paramètre. En C++, l'usage des templates en C++ n'est pas demandé ; il est même déconseillé,

67 Arbres équilibrés (1 personne)

Encadrant : JJ Chabrier

Programmer en C (ou C++) ET en Ocaml une librairie de gestion des arbres équilibrés (chap. de IA).

En Ocaml, vous utiliserez des foncteurs, et le type des éléments et la fonction de comparaison sera passée en paramètre. Les templates en C++ ne sont pas demandés.

68 Homographies et agroglyphes (2 personnes)

Encadrant : D. Michelucci.

Sur la toile, récupérez des images d'agroglyphes ("crop cercles"), ou numérisez des photos que vous emprunterez à DM. Le but du projet est de redresser ces photographies, de telle façon que les cercles soient bien des cercles (et pas des ellipses), et les carrés des carrés (et pas des quadrilatères). Vous pouvez tout d'abord plaquer l'image sur une face en opengl et demander à l'utilisateur de la bouger... Ensuite, vous calculerez les paramètres de la transformation (une homographie) à appliquer sur l'image pour ce redressement ; cette transformation s'écrit : $x' = (ax+by+c)/(ux+vy+w)$, $y' = (a'x+b'y+c')/(ux+vy+w)$. Bien sûr, si $(u, v, w, a, b, c, a', b', c')$ est solution, tout multiple est aussi solution. Vous pouvez trouver $(u, v, w, a, b, c, a', b', c')$ en résolvant un système linéaire, ou un problème au moindres carrés (chap. 28 de IA). Indications : il faut désigner 5 points sur une ellipse pour la définir. Une homographie est définie par 4 points (3 ne sont jamais alignés) et leurs 4 points images. Vous pouvez utiliser la GSL pour l'algèbre linéaire, ou le chap. 28 de IA.

68.1 Générateur aléatoire

Encadrant : O. Bailleux.

Trouver dans la littérature des méthodes de génération de suites aléatoires d'entiers (pseudo aléatoires serait plus juste, ces méthodes étant déterministes), ainsi que des méthodes pour tester la qualité statistique de ce pseudo hasard, et l'imprédictibilité des suites générées. Programmer les, et tester les. Attention : ce n'est pas si facile que vous le croyez.

Soignez votre rapport : il doit pouvoir être lu comme un cours d'introduction à ce problème.

Si possible, soumettez les suites que vous avez engendrées au programme de prolongation des suites mis au point par votre collègue.

Applications : cryptographie, sécurité, confidentialité.

Dernière nouvelle : C'est par erreur que cette section est une sous-section. Ceci fausse la numérotation. Ce projet a été déplacé plus loin das le texte....

69 Recherche arborescente (1 personne)

Encadrant : JJC

Programmez en C (ou C++) ET en Ocaml la bibliothèque de recherche arborescente générique et définitive ! Pour valider cette bibliothèque, joignez une dizaine de programmes utilisant cette bibliothèque. La brièveté de ces programmes et leur nombre prouveront l'intérêt de votre bibliothèque.

En Ocaml, vous utiliserez des foncteurs ; les types des sommets, les fonctions donnant les voisins d'un sommet, etc seront passées en paramètre. (Pour les questions relatives à Ocaml, consultez DM).

70 Optimisation par essaim, et optimisation par simplexe (ocaml, 1 ou 2 étudiants)

Encadrant : JJC ou DM

Vous résoudrez des systèmes de contraintes géométriques en les transformant en un problème d'optimisation : par exemple vous minimiserez la somme des carrés des équations. Lisez le sujet sur les contraintes géométriques 50 pour une définition de ces contraintes.

Vous utiliserez une méthodes en 2 phases : vous utiliserez une méthode d'optimisation par essaims particulaires (cherchez dans Wikipedia, ou bien : `http://www.particleswarm.info/`) pour explorer l'espace de recherche et trouver de bonnes solutions initiales. Vous optimiserez ensuite ces solutions initiales (par exemple les 20 meilleures de ces solutions) avec l'algorithme du simplexe de Nelder-Mead (rien à voir avec la programmation linéaire). Il existe beaucoup d'articles sur cette méthode. Je crois (à vérifier...) qu'il y en a une description dans Numerical Recipes in C [PTVF92]. Bien sûr, votre rapport devra décrire ces 2 méthodes.

Pour la génération des contraintes et des équations, vous pouvez travailler en commun avec le ou les étudiants qui résolvent des contraintes géométriques par des itérations de Newton.

71 Génération d'anagrammes plausibles de patronymes. ocaml. 1 étudiant

Encadrant : JJC ou DM

A partir d'un prénom P et d'un nom N (en fait des lettres présentes dans le prénom et le nom : L), vous générerez un anagramme plausible : un autre prénom P' et un autre nom N'. Vous utiliserez un dictionnaire de prénoms. La principale difficulté est de générer un nom plausible avec les lettres restantes (non utilisées dans le prénom). Jean Véronis propose un tel logiciel sur son blog.

Pour générer un nom plausible, vous pouvez calculer, à partir de la liste de tous les prénoms, une chaîne de Markov : M_{ij} est la probabilité que la lettre i soit suivie de la lettre j. Vous pouvez ajouter 2 lettres virtuelles, pour le début et la fin du nom. Vous choisissez ensuite la permutation des lettres restantes pour le nom qui a la probabilité la plus grande.

Le programme de Jean VERONIS génère, à partir de : "jean veronis", les anagrammes : "Jenna VOIRES", "Jonis AVENER, "Joane RIVENS, "Javier NONSE, "Joann VEISER", etc.

72 Problème de ramassage dans une ville virtuelle - Algorithmes génétiques

Pour 4 étudiants. Java.

Voir fichier word ou PDF dans le même répertoire.

Ce projet est encadré par Mr J-J. Chabrier.

73 Problème de ramassage dans une ville virtuelle- Algorithme du Recuit Simulé

Pour 3 étudiants. Java.

Voir fichier word ou PDF dans le même répertoire.

Ce projet est encadré par Mr J-J. Chabrier.

74 Problème de ramassage dans une ville virtuelle - Algorithme "Tabu Search"

Pour 3 étudiants. Java.

Voir fichier word ou PDF dans le même répertoire.

Ce projet est encadré par Mr J-J. Chabrier.

75 Problème du sac à dos. Méthodes heuristiques : algorithmes génétiques et algorithmes de fourmis

Pour 3 étudiants. Java.

Voir fichier word ou PDF dans le même répertoire.

Ce projet est encadré par Mr J-J. Chabrier.

76 Recherche stochastique pour le problème des N reines

Pour 2 étudiants. Java.

Voir fichier word ou PDF dans le même répertoire.

Ce projet est encadré par Mr J-J. Chabrier.

77 Pavage de cercles. ocaml. 1 étudiant

Encadré par DM. Un pavage de cercles est un ensemble de cercles, dont certains sont extérieurement tangents. C'est une façon de représenter géométriquement les graphes planaires : chaque sommet est représenté par un cercle, et une aête ab du graphe spécifie la tangence entre les cercles représentant les sommets a et b. Vous programmerez la méthode de Collins et Stephenson, présentée dans l'article "A circle packing algorithm" (accessible sur la toile) (voir le fichier ACPA.ps ou ACPA.pdf dans le répertoire UTILITES). Il n'est pas nécessaire de comprendre toute la théorie pour programmer la méthode. Pour générer des graphes planires, vous triangulerez un ensemble de points aléatoires dans le plan ; la triangulation sera obtenue avec une variante d'un algorithme de calcul d'enveloppe convexe de points dans le plan. Vous permettrez à l'utilisateur de bouger interactivement la figure, en déplaçant certains sommets. Vous permettrez aussi d'appliquer des inversions sur le pavage (une inversion transforme un pavage en un autre pavage : les cercles sont transformés en d'autres cercles (ou droites) et les tangences sont conservées).

78 PSLQ. 2 étudiants. Ocaml

Encadré par DM. L'algorithme PSLQ permet de découvrir des relations entières entre des nombres réels décrits par des approximations : soit $x_1, \ldots x_n$ n nombres réels, approchés par des flottants ou des rationnels ; une relation entière est un ensemble d'entiers a_i tels que $\sum a_i x_i$ est un petit nombre en valeur absolue (et les a_i ne sont pas tous nuls !). Un exemple trivial est : $x_1 = 0.6666667$ et $x_2 = 1$; alors $a_1 = 3$, $a_2 = -2$ est une solution, qui permet de voir que "vraisemblablement", $x_1 = 2/3$. Autre exemple : $x_1 = 1.6180339887498949$, $x_2 = x_1^2 = 2.6180339887498949$, $x3 = 1.$; alors la relation entière $a1, a2, a3 = -1, 1, -1$ suggère que x_1 est en fait la racine de $x^2 - x - 1 = 0$, autrement dit $x_1 = \frac{1+\sqrt{5}}{2}$. Pour fabriquer des exemples, vous résoudrez des équations algébriques à coefficients entiers, et vous chercherez des relations entières entre les puissances des racines ; vous pourrez ainsi vérifier que vous retrouvez bien les équations algébriques initiales. Parfois, si le polynôme est factorisable, vous trouverez des relations plus simples (des équations algébriques de degré plus bas) : c'est une façon de factoriser les polynômes. Vous tenterez de généraliser votre programme aux nombres complexes. Il y a beaucoup de documents sur la toile à propos de la méthode PSLQ et ses variantes (par exemple "A gentle introduction to PSLQ" de Armin Straub, et "PSLQ : an algorithm to discover integer relations" par David Bailey et J.M. Borwein).

79 Ensembles de Julia. Ocaml. 1 étudiant

Pour $c \in \mathbb{C}$, l'ensemble de Julia J_c est l'ensemble des points z du plan complexe tels que l'orbite de z : $\{z, f(z), f(f(z)), f(f(f(z))), \ldots\}$, où $f(z) = z^2 + c$, ne va pas à l'infini. Vous afficherez ces ensembles avec la méthode décrite par Afonso Paiva, Jorge Stolfi et Luis Henrique de Figueredo dans l'article "Robust visualization of strange attractors using affine arithmetic" Computers & Graphics 30 (6), 2006 pages 1020–1026 [PdFS06]. Vous n'êtes pas obligé d'utiliser une arithmétique affne, une arithmétique d'intervalles est suffisante. Le calcul des composantes fortement connexes d'un graphe orienté est expliqué dans le livre "Introduction à l'algorithmique" de Cormen, Leiserson, Rivest et Stein [CLRS01]. Des valeurs de c donnant de "beaux" ensembles de Julia sont mentionnés dans wikipedia. Voir le fichier figueiredo.pdf dans le répertoire UTILITES.

80 SVD, GSVD, et compression d'images de visages. ocaml. 1 ou 2 étudiants

Vous programmerez la méthode présentée par Hervé Abdi dans "Singular Value Decomposition (SVD) and Generalized Singular Value Decomposition (GSVD)" pour compresser des fichiers d'images de visage. (voir fichier SVD.pdf dans le répertoire UTILITES). Vous pouvez utiliser la librairie ocamlgsl, qui est une interface avec la GNU Scientific Library.

81 Générateur aléatoire

Encadrant : O. Bailleux. Trouver dans la littérature des méthodes de génération de suites aléatoires d'entiers (pseudo aléatoires serait plus juste, ces méthodes étant déterministes), ainsi que des méthodes pour tester la qualité statistique de ce pseudo hasard, et l'imprédictibilité des suites générées. Programmer les, et tester les. Attention : ce n'est pas si facile que vous le croyez.

Soignez votre rapport : il doit pouvoir être lu comme un cours d'introduction à ce problème.

Si possible, soumettez les suites que vous avez engendrées au programme de prolongation des suites mis au point par votre collègue.

Applications : cryptographie, sécurité, confidentialité.

Références

[ABFM08] Carlos Ansótegui, Ramón Béjar, César Fernàndez, and Carles Mateu. Edge matching puzzles as hard sat/csp benchmarks. In *CP '08 : Proceedings of the 14th international conference on Principles and Practice of Constraint Programming*, pages 560–565, Berlin, Heidelberg, 2008. Springer-Verlag. http ://www.springerlink.com/content/m574272363128136/.

[ALSU06] Alfred V. Aho, Monica S. Lam, Ravi Sethi, and Jeffrey D. Ullman. *Compilers : Principles, Techniques, and Tools (2nd Edition)*. Addison Wesley, August 2006.

[Bar02] Antony Barnett. Millions were in germ war tests : Much of britain was exposed to bacteria sprayed in secret trials. *The Guardian*, Sunday April 21, 2002. http ://www.guardian.co.uk/Archive/Article/0,4273,4398507,00.html.

[CLRS01] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms.* MIT Press, 2001.

[CMP98] Emmanuel Chailloux, Pascal Manoury, and Bruno Pagano. *Développement d'applications avec Objective Caml*. O'Reilly France, 1998. http ://caml.inria.fr/pub/docs/oreilly-book/.

[CMP00] Emmanuel Chailloux, Pascal Manoury, and Bruno Pagano. *Developping Applications with OCAML*. Oreilly book, 2000. http ://caml.inria.fr/pub/docs/oreilly-book/.

[CWH93] Michael F. Cohen, John Wallace, and Pat Hanrahan. *Radiosity and realistic image synthesis*. Academic Press Professional, Inc., San Diego, CA, USA, 1993.

[DL05] O. Deussen and B. Lintermann. *Digital design of nature, Computer Generated Plants and Organics*. Springer Verlag, 2005. http ://www.scribd.com/doc/18636115/Digital-Design-of-Nature.

[Gla94] Andrew S. Glassner. *Principles of Digital Image Synthesis*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1994.

[Har07] Jon D. Harrop. *OCaml for Scientists*. May 2007. http ://www.ffconsultancy.com/products/ocaml_for_scientists/.

[Heu08] Marijn J.H. Heule. Solving edge-matching problems with satisfiability solvers. pages 88–102. University of Leuven, 2008. http ://www.st.ewi.tudelft.nl/~marijn/publications/eternity.pdf.

[Hic08] Jason Hickey. *Introduction to Objective Caml*. 2008.

[Jen05] Henrik Wann Jensen. *Realistic Image Synthesis Using Photon Mapping*. AK Peters, 2005.

[MJ03] John Marlon and Marlon John. *Focus on Photon Mapping*. Premier Press, 2003.

[PdFS06] Afonso Paiva, Luiz Henrique de Figueiredo, and Jorge Stolfic. Robust visualization of strange attractors using affine arithmetic. *Computers & Graphics*, 30(6) :1020–1026, december 2006. http ://www.tecgraf.puc-rio.br/ lhf/ftp/doc/sa.pdf.

[PL04] P. Pruiunkiewicz and A. Lindenmayer. *The algorithmic beauty of plants*. Springer Verlag, 2004. http ://www.scribd.com/doc/18634929/The-Algorithmic-Beauty-of-Plants.

[PR86] Heinz-Otto Peitgen and Peter H. Richter. *The Beauty of Fractals : Images of Complex Dynamical Systems*. Springer Verlag, 1986.

[PS98] C. H. Papadimitriou and K. Steiglitz. *Combinatorial optimization : algorithms and complexity*. Dover, 1998.

[PSF+89] Heinz-Otto Peitgen, Dietmar Saupe, Yuval Fisher, Michael McGuire, Richard F. Voss, Michael F. Barnsley, Robert L. Devaney, and Benoit B. Mandelbrot. *The Science of Fractal Images*. Springer Verlag, 1989.

[PTVF92] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C, the Art of Scientific Computing*. Cambridge University Press, 1992.