Département informatique et réseaux
Première année
Langage C
Micro-projet
Sujet 1

Recherche en temps linéaire du k^e élément d'un ensemble

Niveau : 1

Responsable : Irène Charon

On dispose d'une liste S non ordonnée de n clés distinctes. k étant un entier donné tel que $1 \leq k \leq n$, on cherche à déterminer la k^e clé de la liste, c'est-à-dire la clé ayant exactement $k - 1$ clés plus petites qu'elle. (si k est la partie entière de $\frac{n+1}{2}$, on dit qu'on cherche la médiane).

L'algorithme récursif proposé ci-dessous est linéaire en moyenne ; on demande de programmer en C cet algorithme, **puis de faire une étude statistique pour vérifier, expérimentalement la linéarité de l'algorithme**.

La procédure dépend de deux paramètres S et k.

- si $|S| = 1$, la réponse est l'élément de S
- sinon, on choisit un élément a dans S et on partitionne la liste des (n-1) autres éléments à l'aide de la procédure de partition du tri rapide, en déterminant :
  $S_1 = \{x \in S \mid x < a\}$ $S_2 = \{x \in S \mid x > a\}$

- si $|S_1| \geq k$, on applique la procédure avec S_1 et k ;
- si $|S_1| = k - 1$, la réponse est a ;
- si $|S_1| < k - 1$, on applique la procédure avec S_2 et $k - |S_1| - 1$

Département informatique et réseaux
Première année
Langage C
Micro-projet

Sujet 2

Tri externe

Niveau : 1

Responsable : Annie Danzart

Lorsque le nombre de données à trier en machine est trop grand pour la taille de la mémoire centrale, on doit faire du "tri externe". L'idée générale est d'amener en mémoire centrale p par p les éléments, stockés jusque là sur un fichier, de les trier en constituant ainsi ce que l'on appelle des monotonies, que l'on range dans de nouveaux fichiers, puis de fusionner ces monotonies en monotonies de tailles plus grandes jusqu'à ce que l'ensemble des éléments constitue une seule monotonie : on a alors terminé le tri.
On demande ici d'utiliser le tri "heapsort" pour créer les premières monotonies.
Pour opérer la fusion des monotonies on utilisera l'algorithme de tri équilibré expliqué ci-dessous.
On supposera que la place disponible en mémoire centrale est limitée à 20 éléments pour les données à trier, sans se préoccuper de la place nécessaire pour les variables auxiliaires (variable tampon, variable de boucle…) et que l'on dispose de quatre supports externes (fichiers) S_1, S_2, S_3, S_4. On demandera à l'utilisateur le nombre de données à trier et ces données seront alors choisies par un tirage aléatoire

Explication de la fusion : On utilise deux fichiers en lecture et deux fichiers en écriture.

Au premier passage on crée les monotonies de 20 éléments, que l'on répartit alternativement sur les fichiers S_1 et S_2, comme ci dessous

Sur S_1 on range M_1, M_3, ...

Sur S_2 on range M_2, M_4, ...

Puis on fusionne M_1 et M_2, que l'on écrit sur S_3, M_3 et M_4, que l'on écrit sur S_4, et ainsi de suite en écrivant alternativement sur S_3 et S_4.

On a maintenant des monotonies de longueur 40 sur S_3 et S_4, que l'on fusionne deux à deux et re-écrit alternativement sur S_1 et S_2 et ainsi de suite.

Pendant la fusion, il faudra prendre soin de ne pas avoir plus de 20 éléments à la fois en mémoire centrale.

Le programme devra faire des affichages intermédiaires sur les contenus successifs des quatre fichiers.

Département informatique et réseaux
Première année
Langage C
Micro-projet
Sujet 18

Plus courts chemins

Niveau : 2

Responsable : Irène Charon

On considère un graphe orienté valué ; la fonction de valuation, définie sur les arcs, est à valeurs positives, négatives ou nulles. Le graphe orienté peut posséder des circuits. On cherche à déterminer :

- soit l'existence d'un circuit absorbant ;
- soit, dans le cas où il n'y a pas de circuit absorbant, les distances de tout sommet du graphe à tout autre sommet (éventuellement l'indication qu'il n'existe pas de chemin entre deux sommets donnés).

Il s'agit de programmer en C deux des trois algorithmes résolvant ce problème indiqués dans le livre "Méthodes d'optimisation combinatoire" :

- l'algorithme de Ford (p. 62) (que l'on devra utiliser à partir de chaque sommet)
- l'algorithme général de Ford-Dantzig (p. 63) (que l'on devra utiliser à partir de chaque sommet)
- l'algorithme de Dantzig (p.64)

Après l'exécution de chaque algorithme, l'utilisateur pourra demander, pour un (ou plusieurs) couple(s) de sommets qu'il indiquera, un plus court chemin et la distance de l'un à l'autre.

Les graphes à traiter seront mémorisés dans des fichiers.

Département informatique et réseaux
Première année
Langage C
Micro-projet
Sujet 4

Jeu de NIM

Niveau : 1

Responsable : Irène Charon

Le jeu de NIM, ou sa version simplifiée dite jeu de Marienbad, est mis en œuvre dans l'exemple ci-dessous avec douze allumettes disposées sur trois rangées de la manière suivante :

Le jeu consiste à enlever des allumettes suivant un certain nombre de règles (voir plus bas). Le dernier joueur retirant une ou plusieurs allumettes gagne.

L'ensemble, constitué ici par n_1 allumettes dans la première rangée, par n_2 allumettes dans la deuxième et n_3 dans la troisième, peut être représenté par le triplet (n_1, n_2, n_3). La situation de début du jeu est ici représentée par le triplet (3, 4, 5). Le jeu se joue à deux joueurs. Chacun des deux peut, à son tour, enlever une ou plusieurs allumettes d'une même rangée. Celui qui enlève les dernières allumettes gagne (situation (0, 0, 0)).

Soient deux joueurs X et Y. Une situation (n_1, n_2, n_3) atteinte par X après avoir joué est dite "gagnante" si, quelle que soit l'action de Y, il existe une action de X conduisant à la situation (0, 0, 0). Lorsqu'un joueur X a atteint une situation gagnante, il peut évoluer systématiquement vers la victoire en réagissant à chaque action de son adversaire Y par une action conduisant de nouveau à une situation gagnante. Cette stratégie permet au joueur X d'atteindre, éventuellement, la situation (0, 0, 0) et de gagner la partie. La connaissance des situations gagnantes est donc d'une importance cruciale.

Il est parfaitement possible de déterminer, dans le cas du jeu de NIM, si une situation atteinte après avoir joué est gagnante ou pas.

1. Soit $b_1b_2b_3$ l'équivalent binaire de n_1, $c_1c_2c_3$ celui de n_2 et $d_1d_2d_3$ celui de n_3 (on suppose ici n_1, n_2 et n_3 inférieurs ou égaux à 7).
2. Soient $r_1 = b_1 + c_1 + d_1$, $r_2 = b_2 + c_2 + d_2$, $r_3 = b_3 + c_3 + d_3$.
3. Si, après qu'un joueur ait joué, r_1, r_2 et r_3 sont tous pairs, alors (n_1, n_2, n_3) est une situation gagnante pour ce joueur ; sinon la situation est perdante.

Par exemple, la situation (3, 4, 5) atteinte après avoir joué n'est pas gagnante : en binaire, 3 s'écrit $b_1b_2b_3 = 011$, 4 s'écrit $c_1c_2c_3 = 100$, 5 s'écrit $d_1d_2d_3 = 101$ et $r_2 = 1 + 0 + 0$ est impair.

Au contraire, (1, 4, 5) est une situation gagnante : en binaire, 1 s'écrit $b_1b_2b_3 = 001$, 4 s'écrit $c_1c_2c_3 = 100$, 5 s'écrit $d_1d_2d_3 = 101$ et $r_1 = 0 + 1 + 1 = 2$, $r_2 = 0 + 0 + 0 = 0$ et $r_3 = 1 + 0 + 1$ sont tous les trois pairs.

Si un joueur s'apprête à jouer alors que la situation (n_1, n_2, n_3) est perdante, alors il joue pour faire en sorte qu'à l'issue de son retrait d'allumettes, (n_1, n_2, n_3) soit une situation gagnante.

Écrire un programme qui simule des parties de NIM entre un joueur A (l'utilisateur du programme) et un joueur B (l'ordinateur). Le joueur A entre son choix sous forme de *N* (nombre d'allumettes) à retirer sur la ligne *L*. Dans ce programme, le joueur B essaiera d'atteindre une situation gagnante : si ce n'est pas possible, il enlèvera une allumette dans la première rangée non vide. On vérifiera que B gagne chaque fois qu'il commence, lorsque la position de départ est gagnante.

La configuration de départ (nombre de lignes et nombres d'allumettes pour chaque ligne) pourra être choisie par l'utilisateur. Il n'y aura donc pas forcément trois lignes au départ. Ces deux nombres pourront néanmoins être limités (par exemple, au plus 15 allumettes par ligne et au plus 10 lignes).

Il serait bien d'avoir une représentation graphique.

Département informatique et réseaux
Première année
Langage C
Micro-projet
Sujet 16

Le problème du sac à dos

Niveau : 2

Responsable : Olivier Hudry

L'objectif est d'écrire un programme en C qui résolve le problème du sac à dos par la méthode “par séparation et évaluation” décrite dans le chapitre XI du livre “Méthodes d’optimisation combinatoire”.

Pour programmer l’algorithme, il n'est pas nécessaire de construire réellement l'arbre décrit dans le livre ; celui-ci sert à comprendre la méthode. Il suffira d'utiliser un tableau pour ranger les valeurs des variables et éventuellement d'utiliser la récursivité qui se chargera de faire fonctionner le calcul « selon » l'arbre décrit.

Département informatique et réseaux
Première année
Langage C
Micro-projet
Sujet 6

Génération de plantes : L_systèmes et réécriture de chaînes

Niveau : 1

Responsable : Annie Danzart

La méthode de réécriture de chaînes peut être utilisée pour produire des chaînes de caractères qui pourront être interprétées comme des courbes ou des images.
Cette méthode, qui a été conçue pour modéliser la croissance et la géométrie des plantes, en particulier le branchement des arbres et des buissons, permet de construire les courbes fractales classiques : étoile de von Koch, courbe de Peano, courbe du dragon ...

Au cours de l'algorithme on génère une longue suite de caractères, appartenant à un certain alphabet qui contient des lettres et certains caractères spéciaux tels que "+", "-", "[", "]", ... La correspondance entre une telle chaîne et un tracé est établie par l'interprétation séquentielle, du type de la tortue LOGO, des caractères de commande suivants :

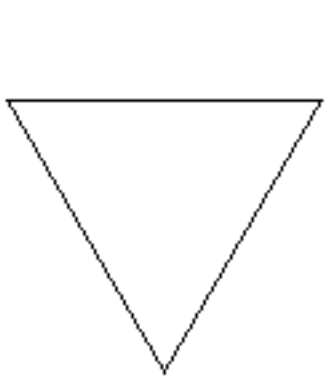
- "F" : Avancer d'un pas, plume baissée, dans la direction courante
- "f" : Avancer d'un pas, plume levée, dans la direction courante
- "+" : Tourner à gauche d'un angle a donné a priori
- "-" : Tourner à droite d'un angle a donné a priori
- " | " : Tourner de Pi (se retourner)
- "[" : Mémoriser l'état courant (position et direction) dans une pile
- "]" : Prendre comme état courant l'état du dessus de la pile et le supprimer de la pile

Cette liste n'est pas exhaustive : on peut rajouter des commandes agissant sur la couleur, le déplacement dans l'espace ... Une première chaîne, appelée axiome, représente la courbe initiale. Chaque caractère de cette chaîne est remplacé par une chaîne suivant une règle de production spécifique à ce caractère. Cette procédure de substitution est itérée N fois pour produire la chaîne finale.

La figure finale est donc déterminée par :
- l'axiome

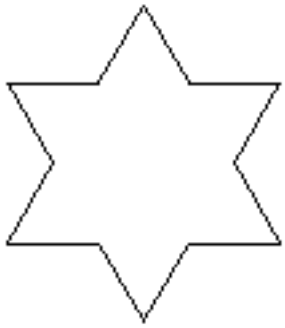
- un angle a priori
- les règles de production
- le nombre d'itérations

EXEMPLE : L'étoile de von Koch
Après une itération la chaîne obtenue est : "F + F - - F + F - - F + F - - F + F - - F + F - - F + F +



Axiome : F - - F - - F
$\alpha = \pi / 3$

Règle : F → F+F - -F + F



Résultat après une itération

F" (s'il n'y a pas d'erreur de frappe)

DÉFINITIONS
Le L_système décrit ci-dessus est un 0L_système car les règles de production ne dépendent pas du contexte : chaque caractère est remplacé par une chaîne qui ne dépend ni des caractères précédents, ni des caractères suivants.

Définition : Soient V un alphabet et V* l'ensemble de tous les mots construits avec cet alphabet. Un 0L_système est un triplet < V, w, P > où w dans V* est l'axiome et P inclus dans V x V* est l'ensemble des règles de production.

Si une paire (C,S) est une règle de production on note : C -> S . Dans un système déterministe il y a, pour chaque lettre C dans V, exactement une et une seule chaîne S dans V*, alors que dans un système probabiliste, on peut associer à C, N chaînes Si, chacune étant appliquée avec une probabilité pi. Un labyrinthe peut aisément se représenter par un graphe non orienté, en représentant chaque carrefour ou cul-de-sac par un sommet et les couloirs par des arêtes reliant ces sommets. On suppose ici que le labyrinthe est représenté par un graphe extrait d'une grille dessinée dans le plan, comme l'illustre le dessin suivant (le graphe est en gras) :

D'autres exemples pourront être consultés à l'adresse suivante :

http://perso.enst.fr/~premiere/micro_projets/exemples_L.html

Département informatique et réseaux
Première année
Langage C
Micro-projet
Sujet 3

Pivot de Gauss

Niveau : 1

Responsable : Bertrand Dupouy

On souhaite avoir un programme capable de résoudre un système de n équations à n inconnues par la méthode du pivot de Gauss. Si le système n'est pas de Cramer, le logiciel indiquera soit qu'il n'existe aucune solution, soit qu'il en existe une infinité.

Ecrire un programme en C pour résoudre ce problème.

Bibliograhie : Introduction à l'analyse numérique matricielle et à l'optimisation – P.G. Ciarlet – MASSON (1982)

Département informatique et réseaux
Première année
Langage C
Micro-projet
Sujet 7

Shellsort

Niveau : 2

Responsable : Christine Potier

Le but de ce projet est de comprendre, programmer et étudier l'algorithme de tri "shellsort" qui est expliqué ci-dessous :

On se donne un tableau C contenant n données entières, que l'on se propose de trier par l'algorithme de "Shellsort".

On se donne une suite strictement décroissante d'entiers $h_1, h_2, ..., h_p$ telle que $h_p = 1$ et $h_1 < n$.

On dit que le tableau C est k-ordonné si :

pour tout i vérifiant $1 \leq i \leq n - k$, $\quad C[i] \leq C[i + k]$

Ecrire en C un programme qui lit les n données, qui lit ou fabrique la suite h_i et qui k-ordonne le tableau C pour les valeurs successives de k : $h_1, h_2, ..., h_p$ (on utilisera une variante du tri insertion) L'algorithme ainsi utilisé est l'algorithme de Shellsort.

En utilisant un compteur qui évalue le nombre de comparaisons effectuées, comparer les différentes valeurs de ce compteur en fin d'exécution pour différent choix de la suite h_i, et le même choix du tableau C initial.(on choisira $n \geq 50$).

Exemples : $p = 1$, $\quad h_1 = 1$ $\quad$ (tri insertion)

$h_i = 2^{p-i+1} - 1$ $\quad$ avec $2^p \leq n$

$\{h_i\} = \{2^q.3^r$ où $q, r \in \mathbb{N}$ et $2^q.3^r < n\}$

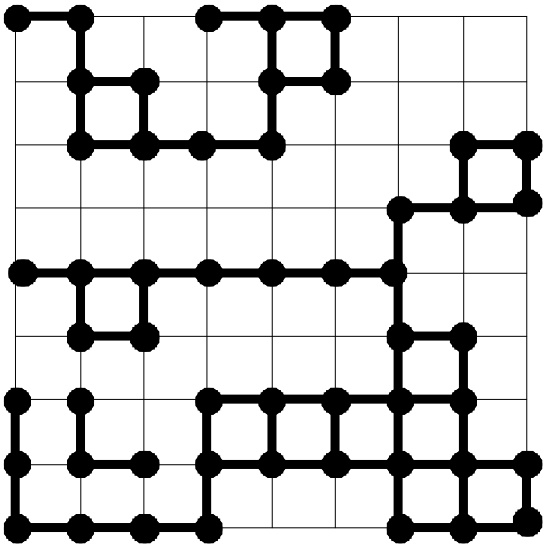
Département informatique et réseaux
Première année
Langage C
Micro-projet
Sujet 8

Le labyrinthe

Niveau : 1

Responsable : Annie Danzart

Un labyrinthe peut aisément se représenter par un graphe non orienté, en représentant chaque carrefour ou cul-de-sac par un sommet et les couloirs par des arêtes reliant ces sommets. On suppose ici que le labyrinthe est représenté par un graphe extrait d'une grille dessinée dans le plan, comme l'illustre le dessin suivant (le graphe est en gras) :



Proposer et programmer un algorithme qui, étant données une entrée et une sortie du labyrinthe (c'est-à-dire deux sommets du graphe), indique un chemin de l'entrée à la sortie (pas nécessairement le plus court chemin) ou, le cas échéant, indique qu'il n'y a pas de chemin de l'entrée à la sortie.

Les labyrinthes seront décrits dans des fichiers.

Il serait bien que le logiciel correspondant donne une représentation graphique du labyrinthe.

Département informatique et réseaux
Première année
Langage C
Micro-projet
Sujet 10

Comparaison d'algorithmes gloutons pour le problème de la couverture minimum d'un graphe

Niveau : 2

Responsable : Olivier Hudry

Le problème de la couverture minimum d'un graphe G = (V,E) consiste à rechercher un sous-ensemble H de sommets de G de cardinal minimum et contenant au moins une extrémité de toute arête de E : pour toute arête {a,b} de E, a est élément de H ou b est élément de H (ou les deux). Ce problème étant NP-difficile, on étudiera quelques algorithmes de type glouton donnant des solutions approchées ; ces algorithmes sont décrits ci-après.

On programmera chacun de ces algorithmes, que l'utilisateur pourra tester sur un même graphe tiré aléatoirement. L'utilsateur donnera l'ordre du graphe et la probabilité d'avoir une arête entre deux sommets donnés. Le programme indiquera le cardinal des couvertures déterminées par chaque algorithme.

Par ailleurs, l'utilisateur devra pouvoir comparer les performances (le nombre de sommets dans la couverture proposée) des différents algorithmes pour des familles de graphes aléatoires construites en donnant la taille de la famille, le nombre de sommets et la probabilité d'existence de chaque arête ; le programme indiquera la moyenne des performances de chaque algorithme.

Algorithme 1 :

- Initialiser H à l'ensemble vide.
- Répéter :
- choisir le sommet restant de plus grand degré et l'ajouter dans l'ensemble H ;
- enlever du sous-graphe ce sommet ainsi que toutes les arêtes incidentes et les sommets de degré nul ;

jusqu'à ce que le sous-graphe restant soit vide.

Algorithme 2 :

- Initialiser H à l'ensemble vide.
- Répéter :
    - s'il y a un sommet de degré 1, choisir son voisin (on gagne toujours à le choisir), sinon choisir le sommet de degré le plus grand ;
    - enlever le sommet choisi, les arêtes incidentes et les sommets de degré nul.

jusqu'à ce que le sous-graphe restant soit vide ;

Algorithme 3 :

- Initialiser H à l'ensemble vide.
- Répéter :
    - s'il y a un sommet de degré 1, choisir son voisin, sinon choisir le sommet x pour lequel le rapport du degré de x sur la somme des degrés des voisins de x est le plus élevé ;
    - enlever le sommet choisi, les arêtes incidentes et les sommets de degré nul ;

jusqu'à ce que le sous-graphe restant soit vide.

Département informatique et réseaux
Première année
Langage C
Micro-projet
Sujet 11

Flot maximum

Niveau : 2

Responsable : Irène Charon

Programmer en C l'algorithme de Ford et Fulkerson. Les graphes seront lus dans des fichiers. Un jeu de test sera élaboré pour tester le programme. On pourra reprendre en particulier les exemples donnés dans le livre d'optimisation combinatoire du cours de SDA.

Département informatique et réseaux
Première année
Langage C
Micro-projet
Sujet 12

Ordonnancement

Niveau : 2

Responsable : Olivier Hudry

Une entreprise doit effectuer des tâches 1, 2, ..., p ; certaines tâches doivent être achevées avant que certaines autres puissent commencer ce qu'on exprime en disant qu'il y a des contraintes de précédence. La tâche i demande un temps t_i pour être effectuée. Le but de ce projet est de déterminer un calendrier optimum (c'est-à-dire qui réduise au minimum la durée totale de l'ensemble des tâches).
Pour résoudre ce problème, on construit un graphe valué G à p + 2 sommets.

Sommets de G :

. une source s qui symbolise la date de début ;
. un puits p qui symbolise la date de fin ;
. pour i = 1, ..., p, le sommet i qui représente le début de la tâche i.

Arcs de G :

. les arcs (s, i) (i = 1, ..., p) de poids nuls ;
. les arcs (i, p) (i = 1, ..., p) de poids t_i ;
. si la tâche i doit précéder la tâche j, l'arc (i, j) de poids t_i.

La première date pour commencer la tâche i est la longueur d'un plus long chemin de s à i dans G.

La date au plus tôt de la fin de l'ensemble des tâches est la longueur d'un plus long chemin de s à p dans G.

Le problème admet une solution si et seulement si G ne contient pas de circuit.

Écrire en C un programme qui donne le calendrier optimum s'il y a une solution, et signale qu'il n'y a pas de solution lorsque G contient un circuit.

L'algorithme utilisé devra nécessairement avoir une complexité de l'ordre du nombre de contraintes de précédence.

Les données du problème seront indiquées à l'aide d'un fichier.

Département informatique et réseaux
Première année
Langage C
Micro-projet
Sujet 13

Attracteurs étranges

Niveau : 2

Responsable : Annie Danzart

En synthèse d'images, la modélisation géométrique d'objets complexes est un problème difficile. Les fractales sont très utilisées pour modéliser la nature : plantes,nuages, paysages
On peut modéliser l'objet comme l'attracteur "étrange" d'un ensemble M de transformations affines contractantes, Rd dans Rd , chacune de ces transformation ayant une probabilité Pi. Cet ensemble de transformations et de probabilité est appelé IFS (Iterated Function System).

Soit un IFS sur défini par M={M1, M2, ...,Mn } et P={P1, P2, ...,Pn } (la somme de ces n probabilités vaut 1).
Pour dessiner une approximation A' de l'attracteur A il existe 2 méthodes classiques . On suppose dans la suite qu'on part d'un point mais il est possible, et parfois préférable de partir d'un ensemble X0 judicieusement choisi.

LE JEU DU CHAOS

On choisit un point x0 de Rd et on calcule une suite de points x1, x2,..., xN en déterminant xk+1=Mi(xk), la transformation Mi ayant été choisie au hasard dans M avec la mesure P.

CONSTRUCTION DÉTERMINISTE :

A partir de x0 on construit l'arbre des transformées où chaque noeud est un point auquel on doit appliquer les n transformations.

QUELQUES IFS ET LEUR ATTRACTEUR :

- Ensemble de Cantor :

M1 x -> x/3 P1=0.5
M2 x -> x/3 + 2/3 P2=0.5

Dragons

M1 z -> sz + 1 (z dans C, s=1/2(1+i)) P1=0.5
M2 z -> sz - 1 (z dans C, s=1/2(1+i)) P2=0.5

Panier de Sierpinski :

M1 [x,y] -> [0.5x , 0.5y] P1=0.333
M2 [x,y] -> [0.5x +0.5, 0.5y] P1=0.333
M3 [x,y] -> [0.5x+0.25 , 0.5y+0.5] P3=0.334

Les deus IFS suivants sont définis par 4 transformations du type

[x,y] -> [ax+by+e, cx+dy+f], p

Fougère :

M1 a=0 b=0 c=0 d=0.16 e=0 f=0 p=0.01
M2 a=0.85 b=0.04 c=-0.04 d=0.85 e=0 f=1.6 p=0.85
M3 a=0.2 b=-0.26 c=0.23 d=0.22 e=0 f=1.6 p=0.07
M4 a=-0.15 b=0.28 c=0.26 d=0.24 e=0 f=0.44 p=0.07

Erable :

M1 a=0.5 b=0 c=0 d=0.5 e=0 f=0.4 p=0.3
M2 a=0.7 b=0 c=0 d=0.7 e=0 f=0 p=0.3
M3 a=0.4 b=0.3 c=-0.3 d=0.4 e=0.22 f=0.15 p=0.2
M4 a=0.4 b=-0.3 c=0.3 d=0.4 e=-0.22 f=0.15 p=0.2

NB: Pour la méthode déterministe de la fougère, il faudrait trouver un critère permettant d'arrêter une branche cf : "The Algorithmic Beauty of Plants" de Prusinkiewicz P. et Lindemayer A., Springer Verlag, 1990.

Des exemples peuvent être trouvés à l'adresse :

http://www.cosy.sbg.ac.at/rec/ifs/

Département informatique et réseaux
Première année
Langage C
Micro-projet
Sujet 14

Recherche d'un motif dans un texte

Niveau : 2

Responsable : Bertrand Dupouy

Le travail que devra faire le logiciel que l'on vous demande d'écrire (en C) est de rechercher dans un texte une chaine de caractères donnée, appelée motif et de savoir trouver la place de la première occurrence de ce motif dans le texte, s'il y figure.

On implémentera deux méthodes différentes : la méthode dite brutale, ou encore intuitive, et l'algorithme dit de Knuth-Morris Pratt.

Le texte pourra être lu soit directement au clavier, soit à l'aide d'un fichier. La chaîne à rechercher sera indiquée au clavier.

Bibliographie : D.E. Knuth, The art of computer programming, Addison-Wesley.

Département informatique et réseaux
Première année
Langage C
Micro-projet
Sujet 15

Les masques

Niveau : 2

Responsable : Bertrand Dupouy

On considère une liste de "mots" séparés par des espaces. Un mot est une chaîne constituée des caractères 'a', 'b', ..., 'z', 'A', 'B', ..., 'Z', '0, '1', ...,'9' et éventuellement de points.

Un masque est une chaîne constituée des caractères 'a', 'b', ..., 'z', 'A', 'B', ..., 'Z', '0, '1', ...,'9', de points, de symboles '∗ 'et '§'. Dans un masque :

- n'importe quel caractère représente lui-même,
- ∗ représente au moins un caractère,
- § représente au plus un caractère.

Il s'agit d'écrire en C un programme qui recherche dans une liste de mots tous les mots correspondant à un masque donné. Un masque peut posséder plusieurs fois les symboles * ou §.

exemple :
liste : .a xx.a 2.a 2.ab
masque : ∗.a
Réponse : xx.a 2.a

Département informatique et réseaux
Première année
Langage C
Micro-projet
Sujet 5

Les mariages

Niveau : 1

Responsable : Olivier Hudry

Soient deux ensembles E1 (hommes) et E2 (femmes) de même cardinalité. Chaque individu de chaque ensemble a ordonné les individus de l'autre ensemble suivant une liste de préférence. Un ensemble de mariages s'obtient en accouplant chaque individu de E1 à un et un seul individu de E2. Un tel ensemble est dit *stable* s'il n'existe pas deux couples tels que le mari du premier couple préfère à la sienne la femme du second et celle-ci préfère également l'homme du premier couple à son mari.

L'algorithme de Gale et Sharpley permet de trouver un ensemble de mariages stables.

- Au début, tous les hommes proposent le mariage à leur premier choix.
- Puis les femmes et les hommes sont consultés à tour de rôle, en commençant par les femmes.
- Quand c'est le tour des femmes, chaque femme sollicitée choisit parmi les hommes qui lui proposent le mariage celui qu'elle préfère et le retient (au moins tant qu'une proposition plus favorable pour elle ne se présente pas).
- Les hommes qui sont retenus maintiennent leur proposition ; ceux qui ne sont retenus par aucune femme proposent le mariage à leur choix suivant.
- On s'arrête lorsque chaque femme a eu au moins une proposition. Alors, chaque femme accepte définitivement le mariage avec l'homme qu'elle a retenu en dernier.

On admettra que cet algorithme mène toujours à une solution convenable et que par conséquent un ensemble stable de couples existe toujours.

Programmer cet algorithme en C avec pour données la taille des ensembles E1 et E2 et les ordres de préférence pour chaque homme et chaque femme.

Département informatique et réseaux
Première année
Langage C
Micro-projet
Sujet 17

Polynômes

Niveau : 2

Responsable : Bertrand Dupouy

Manipulation de polynômes à coefficients réels.

Le but de ce projet est de réaliser un logiciel de manipulation de polynômes qui puisse :

- ajouter deux polynômes ;
- multiplier deux polynômes ;
- ordonner un polynôme suivant les puissances croissantes ou décroissantes ;
- opérer la division euclidienne de deux polynômes.

La structure de données doit être une liste chaînée de structures qui permettra de manipuler des polynômes de degré quelconque ; dans cette liste ne figureront que les monômes à coefficients non nuls.

La chaîne de caractères permettant d'écrire à l'écran le polynôme résultant d'une opération sera choisie pour être aussi proche que possible de la représentation usuelle.

Département informatique et réseaux
Première année
Langage C
Micro-projet
Sujet 9

Jeu de la vie

Niveau : 1

Responsable : Annie Danzart

Créé par John Horton Conway, un mathématicien de l'Université de Cambridge, le "jeu de la vie" est essentiellement une simulation qui permet de décrire l'évolution d'une population soumise à des règles précises. Le "jeu" se déroule sur un damier aux dimensions théoriquement non limitées (les modalités liées à la limitation de fait seront à préciser).

Trois lois "génétiques", prenant en compte les huit cases adjacentes à une case donnée, déterminent l'évolution de la population. Ces lois sont les suivantes :
1) un individu meurt s'il a au moins 4 voisins (surpopulation) ou au plus un voisin (isolement)
2) un individu survit s'il a deux ou trois voisins
3) si une case vide est entourée d'exactement trois cases occupées, il se produit une naissance, et cette case devient occupée à la génération suivante.

Programmer l'évolution d'une telle population en affichant les différentes générations ; on laissera le choix à l'utilisateur entre une population initiale aléatoire, une population initiale de son choix ou une population déjà créée et conservée dans un fichier.

On déterminera des configurations initiales conduisant à des évolutions particulières (disparition totale, oscillation, ...)..

Département informatique et réseaux
Première année
Langage C
Micro-projet
Sujet 33

Répartition de chansons sur des CD

Niveau : 3

Responsable : Olivier Hudry

On souhaite enregistrer n chansons c_1, c_2, …, c_n sur le moins de CD possibles. Chaque chanson c_i possède une durée d_i, et les CD possèdent tous la même durée D (on supposera $D \geq \max d_i$). La méthode suivante, à deux phases, permet de déterminer une solution approchée du problème :

1ère phase (construction d'une solution intiale)
- trier les n chansons par durée décroissante ;
- dans l'ordre défini par ce tri, placer successivement chaque chanson c_i dans le CD K ayant à ce moment-là le plus petit temps résiduel $D - \sum_{c_j \text{ sur } K} d_j$; si un tel CD n'existe pas, en prendre un nouveau.

2nde phase (amélioration de la solution initiale)
Pour essayer de réduire le nombre de CD utilisés, on choisit deux CD K_1 et K_2. S'il existe un transfert d'une chanson de K_1 vers K_2 (ou de K_2 vers K_1) ou un échange entre une chanson de K_1 avec une chanson de K_2 qui ne provoque pas un débordement de K_1 ni de K_2 et qui fait croître le maximum du temps inutilisé sur K_1 et du temps inutilisé sur K_2, on effectue ce transfert ou cet échange. Puis on réitère le même procédé. Dès qu'un transfert vide un CD, on retire celle-ci définitivement.

Programmer en C cette méthode à deux phases. On supposera que les données (longueurs des chansons et durée des CD) sont précisées dans un fichier.

Département informatique et réseaux
Première année
Langage C
Micro-projet
Sujet 20

Tri baquet

Niveau : 2

Responsable : Irène Charon

On considère n entiers compris entre 0 et $10^p - 1$ inclus qu'il s'agit de trier par ordre croissant. Au départ, ces entiers sont placés dans une liste chaînée L_0. Il s'agit de transformer cette liste chaînée en une liste chaînée comprenant les mêmes entiers, mais classés par ordre croissant.

L'algorithme comporte p phases (le nombre de chiffres des nombres traités) ; à l'issue de la $j^{è}$ phase, on a reconstitué une nouvelle liste chaînée L_j comportant les même entiers mais dans un nouvel ordre. La liste L_p est triée par ordre croissant.

On dispose d'un tableau de 10 « baquets » indicé par 0, 1, …, 9.

À chaque phase j, on va distribuer les entiers dans les 10 baquets, en les « chaînant » dans ces baquets ; pour faire cela, on récupère 1 à 1 les maillons de la liste chaînée L_{j-1} et c'est en fait ceux-ci qui sont mis dans les baquets. À la première phase, on met dans la liste chaînée du baquet i (i variant de 0 à 9), les maillons de la liste L_0 dont les entiers ont i en dernière position (comme chiffre des unités). À la deuxième phase, on met dans la liste chaînée du baquet i (i variant de 0 à 9), les maillons de la liste L_1 dont les entiers ont i en avant-dernière position (comme chiffre des dizaines) ; un entier plus petit que 10 est placé dans le baquet 0, ... Plus généralement, à la phase j (j variant de 1 à p), on met dans la liste chaînée du baquet i (i variant de 0 à 9), les maillons de la liste L_{j-1} dont les entiers ont i en (j – 1)^{è} position en partant de la droite (un entier plus petit que 10^{j-1} est mis dans le baquet 0).

Lorsqu'on ajoute un entier dans un baquet, on le met toujours **en fin de liste**.

En ce qui concerne la constitution des listes L_j pour j>0, on les obtient en concaténant les listes du baquet 0, du baquet 1, ..., du baquet 9 à la fin de la phase j. Ces listes ayant été créées, on « vide » les baquets au début de la phase suivante, avant d'y recréer de nouvelles listes. La liste L_p est la solution au problème.

Écrire en C un programme qui saisit les données, les trie à l'aide du tri baquet et affiche la liste triée. Les données pourront être lues dans un fichier et les données triées pourront être écrites dans un second fichier.

Le type de données de base pour constituer les différentes listes chaînées dont il est question dans ce travail est une structure constituée de deux champs : le premier pour un entier, le second pour une pointeur sur un élément de ce type. L'ensemble des baquets sera raisonnablement représenté en mémoire par un tableau de 10 pointeurs sur des éléments de ce même type. Il pourra être utile, tant pour constituer les listes des baquets, que pour concaténer ces mêmes listes, de prévoir aussi des pointeurs sur le dernier élément des listes des baquets.

Département informatique et réseaux
Première année
Langage C
Micro-projet
Sujet 21

Composantes fortement connexes

Niveau : 2

Responsable : Olivier Hudry

Ecrire en C un logiciel permettant de voir si un graphe orienté est ou non fortement connexe, et, dans la négative, donnant les composantes fortement connexes.

Les graphes à traiter seront enregistrés dans des fichiers.

Indications : on pourra utiliser les propositions des pages 78 à 80 du livre “Méthodes d’optimisation combinatoire”

Département informatique et réseaux
Première année
Langage C
Micro-projet
Sujet 22

La calculatrice

Niveau : 2

Responsable : Annie Danzart

On souhaite disposer d'un programme (écrit en C) qui prenne en entrée une expression arithmétique écrite sous forme d'une expresion bien parenthésée et indique à l'utilisateur la valeur de cette expression. L'expression arithmétique ne comportera, en plus des parenthèses, que des constantes numériques et les quatre opérateurs ; la division considérée sera la division réelle.

Si l'expression est mal construite, le programme l'indiquera.

Exemple : ((3 * 5.2) – (4 / (2 + 3)))
ce qui vaut 14,8.

Département informatique et réseaux
Première année
Langage C
Micro-projet
Sujet 23

Grand stable ou grand complet

Niveau : 2

Responsable : Olivier Hudry

On considère un graphe G, simple, d'ordre n et deux entiers p et q strictement positifs tels que : :

$n \geq C_{p+q}^{p}$

On admettra les résultats suivants :
1) Soit x un sommet quelconque de G. On note S les sommets de G adjacents à x, et T les sommets de G non adjacents à x autres que x. On a :

- soit $|S| \geq C_{p+q-1}^{p}$
- soit $|T| \geq C_{p+q-1}^{p-1}$

2) G contient soit un stable à p+1 sommets, soit un complet à q+1 sommets..

Ecrira un programme en C qui :
a) vérifie si l'inégalité entre les paramètres n, p et q est satisfaite.
b) si oui, exhibe dans G, soit un stable à p+1 sommets, soit un complet à q+1 sommets. L'algorithme correspondant ne devra pas être récursif et devra être de complexité polynomiale.

Département informatique et réseaux
Première année
Langage C
Micro-projet
Sujet 24

Tous les stables

Niveau : 2

Responsable : Olivier Hudry

Il s'agit de lister tous les stables d'un graphe non orienté à l'aide d'un programme écrit en C.

On essaiera d'être économe sur la complexité de l'algorithme utilisé ; le mieux est sans doute :

- d'essayer à la main, sur un petit graphe, de lister tous les stables en passant le moins de temps possible tout en étant systématique
- d'expliciter cette méthode sous forme d'un algorithme.

Les graphes étudiés seront mémorisés dans des fichiers.

Département informatique et réseaux
Première année
Langage C
Micro-projet
Sujet 25

Expressions bien parenthésées

Niveau : 2

Responsable : Annie Danzart

On considère les expressions arithmétiques "bien parenthésées" du type suivant :

((B∗B) − (4/(A+C)))

où figurent, en plus des parenthèses et des quatre symboles d'opérations arithmétiques à deux opérandes, des variables (A,B,C ou autres) et des constantes (4 ou autres).

Ecrire un programme en C qui peut :

- transformer une expression arithmétique bien parenthésée en écriture polonaise (l'exemple devient −∗BB/4+AC)
- transformer une expression arithmétique bien parenthésée en écriture polonaise inversée (l'exemple devient BB∗4AC+/ −).
- transformer une expression arithmétique écrite en écriture polonaise en une expression écrite sous forme d'une expression arithmétique bien parenthésée.
- transformer une expression arithmétique écrite en écriture polonaise inversée en une expression écrite sous forme d'une expression arithmétique bien parenthésée
- étant donné une expression arithmétique bien parenthésée en donner la dérivée sous forme bien parenthésée.

Département informatique et réseaux
Première année
Langage C
Micro-projet
Sujet 26

Jeu des chiffres « approché »

Niveau : 2

Responsable : Irène Charon

Ce jeu consiste à approcher du mieux possible un nombre entier donné *N* compris entre 100 et 999 grâce à des additions, soustractions, multiplications ou divisions entières exactes de *p* nombres donnés utilisés chacun au plus une fois et tirés (aléatoirement ou non) dans la liste (avec répétitions) suivante : 1, 2, ..., 9, 10, 25, 50, 75, 100.

Écrire en C un programme simulant ce jeu (pour approcher l'entier *N*, on utilisera une heuristique plutôt qu'une méthode exacte qui risquerait de prendre trop de temps à l'exécution). Le choix est libre pour cette heuristique qu'il faut inventer.

Département informatique et réseaux
Première année
Langage C
Micro-projet
Sujet 28

Carnet d'adresses

Niveau : 2

Responsable : Olivier Hudry

Vous désirez avoir un carnet d'adresses.et numéros de téléphone informatisé. Vous voulez :

- que votre carnet soit rangé, au choix, par ordre alphabétique des noms de famille, des prénoms, ...
- pouvoir rajouter une personne qui se trouve alors automatiquement classé
- pouvoir modifier les coordonnées d'une personne ou la supprimer
- pouvoir retrouver une personne à l'aide de son nom
- pouvoir retrouver une personne à l'aide de son prénom

Programmez en C le logiciel correspondant.

Département informatique et réseaux
Première année
Langage C
Micro-projet
Sujet 29

Enveloppe convexe d'un polygone

Niveau : 3

Responsable Christine Potier

Ecrire en C un programme qui, étant donné un ensemble de points du plan, en donne l'enveloppe convexe.

Les coordonnées des points seront indiquées dans un fichier. L'ensemble des points et l'enveloppe convexe devront être dessinés à l'écran.

Bibliographie : Eléments d'algorithmique de D. Beauquier, J. Berstel, P. Chrétienne.

Département informatique et réseaux
Première année
Langage C
Micro-projet
Sujet 27

Hachage

Niveau : 2

Responsable : Annie Danzart

Il s'agit d'étudier les différentes méthodes de hachage exposées par exemple dans le livre de M.-C. Gaudel, M. Soria et C. Froideveaux, “Types de données et algorithmes” (McGraw Hill ou Collection didactique de l’INRIA) ou, pour partie, dans le polycopié « Elements d’algorithmique ».

Les données à ranger et à rechercher seront des mots écrits dans l'alphabet a, b,..., z.

Le choix de la fonction de hachage est libre.

Seront étudiés :

- le hachage avec chaînage externe,
- le hachage coalescent, avec ou sans zone de débordement,
- le hachage linéaire.

On fera, pour chacun des algorithmes, une étude expérimentle du nombre de comparaisons pour une recherche positive et pour une recherche négative, en fonction de la taille du tableau et du nombre de données (pour le hachage coalescent avec zone de débordement, la taille de cette zone sera aussi prise en compte).

Département informatique et réseaux
Première année
Langage C
Micro-projet
Sujet 30

Jeu des chiffres « exact »

Niveau : 3

Responsable : Olivier Hudry

Le jeu consiste à approcher du mieux possible un nombre entier donné *N* compris entre 100 et 999 grâce à des additions, soustractions, multiplications ou divisions entières exactes de *p* nombres donnés (*p* étant un paramètre valant 3, 4, 5, 6 ou 7) utilisés chacun au plus une fois et tirés (aléatoirement ou non) dans la liste (avec répétitions) suivante : 1, 2, ..., 9, 10, 25, 50, 75, 100.

Écrire un programme en C qui donne systématiquement la solution la plus proche, et donc *N* lorsque cela est possible. Il n'est pas certain que les cas $p = 6$ ou $p = 7$ puissent être traités en un temps raisonnable ; ce sera à voir après programmation.

Le programme devra indiquer les opérations qui permettent d'atteindre ce plus proche nombre.

Département informatique et réseaux
Première année
Langage C
Micro-projet
Sujet 31

Codage de Hamming

Niveau : 3

Responsable: Christine Potier

Pendant la transmission d'informations sur une ligne, le signal peut se dégrader. Il est alors important de donner un codage de l'information qui permette de détecter les erreurs (code autovérificateur) et même de les corriger (code autocorrecteur). Le code de Hamming (voir annexe) qui utilise 11 bits pour représenter 7 bits d'information est un code autovérificateur pour 2 erreurs et autocorrecteur pour une erreur.
Développer un logiciel qui :
- simule l'envoi d'un message non codé (texte en ASCII non étendu).en supposant que chaque bit a une probabilité uniforme d'être erroné, récupère ce message et l'affiche (en remplaçant les caractères non affichables par "?")
- code le message, simule sa transmission, le décode en le corrigeant et l'affiche.
Analyser les résultats obtenus pour divers taux d'erreurs de transfert entre 1% et 10%

CONTRÔLE DE PARITÉ

On emploie souvent un code autovérificateur dit code de parité, à n+1 bits où les n premiers bits sont des bits d'information et le (n+1)ième bit est positionné de telle sorte que le nombre total de bits à 1 soit pair (code à parité paire) ou respectivement impair (code à parité impaire). Ce code n'est autovérificateur que pour une erreur.

CODE DE HAMMING

Ce code est basé sur les tests de parité. Pour un code capable de corriger une seule erreur et d'en détecter 2, on associe aux n bits d'information k bits de parité qui permettront de détecter soit l'absence d'erreur, soit la position du bit erroné dans les n+k bits transmis et de corriger sa valeur. On peut considérer les k bits ($p_k p_{k-1} \ldots p_1$) comme un nombre p en base 2 qui doit pouvoir prendre au moins n+k+1 valeurs (0 pour l'absence d'erreur ou une valeur

entre 1 et n+k pour une position) donc on doit avoir $2^k \geq n+k+1$. Il faut donc 4 bits de parité pour 7 bits d'information.

Les k bits de parité sont imbriqués aux n bits d'information et leur position correspond aux position des puissances de 2 :

position	11	10	9	8	7	6	5	4	3	2	1
	x	x	x	p_4	x	x	x	p_3	x	p_2	p_1

Les bits 1, 2, 4 et 8 sont les bits de Hamming et les bits 3, 5, 6, 7, 9, 10 et 11 les bits d'information.
Le bit 1 contrôle les bits 1, 3, 5, 7, 9 et 11 : en cas d'erreur il indiquera les positions impaires.
Le bit 2 contrôle les bits 2, 3, 6, 7, 10 et 11
Le bit 4 contrôle les bits 4, 5, 6 et 7
Le bit 8 contrôle les bits 8, 9, 10 et 11

Lors de la réception du message on calcule les valeurs des bits de parité correspondant au message reçu et on les compare aux valeurs des bits de parité reçus : pour chaque p_j si les 2 valeurs sont identiques on met p_j à 0, sinon on met p_j à 1. Si $p = (p_k p_{k-1} \ldots p_1)_2$ est nul alors il n'y a probablement pas d'erreur (ou alors une erreur de multiplicité ≥ 3), sinon il y a au moins une erreur qui est à la position p si elle est unique. On corrige donc en conséquence.

Bibliographie:

Andrew Tanenbaum "Architecture de l'ordinateur" , InterEditions Paris, 1987
Meinadier J.P. "Structure et fonctionnement des ordinateurs", Larousse, 1971

Département informatique et réseaux
Première année
Langage C
Micro-projet
Sujet 32

Cycles eulériens

ou

"comment dessiner une petite maison sans repasser deux fois par le même trait?"

Nivau : 3

Responsable : Olivier Hudry

Soit G = (X,E) un graphe simple connexe. On dit que G est eulérien s'il existe un cycle, non nécessairement élémentaire, utilisant une fois et une seule, toute arête de G.

Ecrire en C un programme qui, étant donné un graphe G, examine si le graphe est ou non eulérien (on pourra observer qu'un graphe est eulérien si et seulement si il ne contient pas de sommet de degré impair) et qui, dans le "bon cas", donne un cycle eulérien du graphe. Si G n'est pas connexe, donner les cycles eulériens des composantes connexes eulériennes.

Les graphes à traiter seront enregistrés dans des fichiers.

Département informatique et réseaux
Première année
Langage C
Micro-projet
Sujet 19

Planarité des graphes hamiltoniens

Niveau : 2

Responsable : Irène Charon

Dans un graphe non orienté G, un cycle hamiltonien est un cycle passant une fois et une seule par chacun des sommets de G. Un graphe est dit hamiltonien s'il possède un cycle hamiltonien.

On considère un graphe hamiltonien et on note C un cycle hamiltonien de ce graphe. Ayant représenté le cycle hamiltonien C par un polygone convexe P, on représente les autres arêtes de G par des cordes de P. On construit un graphe auxiliaire G_A dont les sommets représentent les arêtes de G non dans C, deux sommets de G_A étant adjacents si et seulement si les cordes correspondantes se coupent. On admettra que G est planaire si et seulement si G_A est biparti.

Déduire du résultat ci-dessus un algorithme, que l'on programmera en C, pour tester l'éventuelle planarité d'un graphe hamiltonien dont on connaît un cycle hamiltonien.

Dans le cas où le graphe sera planaire, le résultat de l'algorithme sera de préférence représenté graphiquement. Les sommets du cycle hamiltonien seront les sommets d'un polygone régulier. Les autres arêtes que celles du cycle hamiltonien seront tracées avec deux couleurs différentes de façon à mettre en évidence la bipartition déterminée par l'algorithme ; deux arêtes de même couleur ne devront pas s'intersectées.

Les graphes à traiter seront mémorisés dans des fichiers.

Département informatique et réseaux
Première année
Langage C
Micro-projet
Sujet 34

Application du flot maximum aux couplages dans les graphes bipartis

Niveau : 3

Responsable : Irène Charon

Le but de ce projet est d'une part de déterminer si un graphe est biparti, et dans l'affirmative de trouver une bipartition des sommets, d'autre part de trouver un couplage de cardinalité maximum dans un graphe biparti, par application de la théorie des flots, et, plus précisément, de l'algorithme de Ford et Fulkerson.

Les graphes à traiter seront stockés dans des fichiers.

Indications : on pourra utiliser les propositions des pages 101 à 103 du livre “Méthodes d’optimisation combinatoire”.

Département informatique et réseaux
Première année
Langage C
Micro-projet
Sujet 35

Points d'articulation et composantes 2-connexes

Niveau : 3

Responsable : Olivier Hudry

Ecrire en C un programme qui, étant donné un graphe non orienté connexe, en détermine les sommets d'articulation et les composantes 2-connexes.

Les définitions et algorithmes nécessaires pour résoudre ce problème se trouvent entre les pages 80 et 83 du livre “Méthodes d’optimisation combinatoire”, dans le chapitre “Parcours de graphes”.

On utilisera des fichiers pour coder les graphes dont on cherche les points d’articulation et les composantes 2-connexes.

Département informatique et réseaux
Première année
Langage C
Micro-projet
Sujet 36

Stable maximum

Niveau : 3

Responsable : Olivier Hudry

On cherche, dans un graphe non orienté, le cardinal maximum d'un stable (ensemble de sommets deux à deux non adjacents).

On écrira, pour résoudre ce problème, deux algorithmes en C que l'on programmera : l'un sera une méthode approchée (ou heuristique) et l'autre une méthode exacte.

Les graphes à traiter seront mémorisés dans des fichiers. On traitera des graphes suffisamment gros pour que la méthode exacte nécessite un temps « long » pour s'exécuter.

Département informatique et réseaux
Première année
Langage C
Micro-projet
Sujet 37

Le jeu de sudoku

Niveau : 3

Responsable : Irène Charon

Le jeu de sudoku consiste en la donnée d'une grille de neuf carrés sur neuf, divisée en neuf carrés de trois sur trois. Des nombres entre 1 et 9 figurent dans certaines des cases de la grille ; le jeu consiste à compléter la grille de telle sorte que chaque ligne, chaque colonne et chaque petit carré de trois sur trois contiennent chaque chiffre de un à 9. On peut trouver ici un exemple de jeu.

				8	3	4		
3					4	8	2	1
7								
		9	4		1		8	3
4	6		5		7	1		
								7
1	2	5	3					
		7	2	4				

Il s'agit d'écrire un programme en C qui prenne en entrée une grille partiellement remplie et qui indique en sortie soit qu'il est impossible de la compléter, soit qu'il y a une solution unique (et donne alors cette solution), soit qu'il y a au moins deux façons de la compléter.

Département informatique et réseaux
Première année
Langage C
Micro-projet
Sujet 38

L'algorithme de DINIC

Niveau : 4

Responsable : Irène Charon

Programmer en C l'algorithme de Dinic pour déterminer un flot de valeur maximum dans un réseau G .

Les graphes traités seront enregistrés dans des fichiers.

Indications : l'algorithme de Dinic est présenté (rapidement) aux pages 94 et 95 du livre "Méthodes d'optimisation combinatoire".

Département informatique et réseaux
Première année
Langage C
Micro-projet
Sujet 39

Busacker et Gowen

Niveau : 4

Responsable : Irène Charon

Programmer en C l'algorithme de Busacker et Gowen, permettant de trouver un flot maximum de coût minimum dans un graphe orienté valué où les arcs sont munis d'une capacité.

Les graphes à traiter seront stockés dans des fichiers.

Références : page 95 du livre “Méthodes d’optimisation combinatoire.

Département informatique et réseaux
Première année
Langage C
Micro-projet
Sujet 40

Lissage de mesures bruitées

Niveau : 4

Responsable : Christine Potier

On suppose que l'on dispose des valeurs $y_i = f(x_i)+\varepsilon_i$, mesurées en N points $x_1,\ldots,x_N$, où f est une fonction "**régulière**" et ε_i représente une erreur de mesure (erreur gaussienne centrée de variance σ^2). Chercher une fonction d'interpolation à partir de ces données reviendrait à négliger une des caractéristiques du problème: l'erreur de mesure. Il est préférable, dans ce cas, de chercher une fonction qui soit "suffisamment proche" de ces mesures sans passer exactement par ces points. On essaie de garder le maximum d'information sur la fonction et le minimum d'information sur l'erreur de mesure.

Le projet consiste à développer un logiciel permettant, à partir d'une fonction f donnée, d'échantillonner cette fonction en des points x_i équidistants, de simuler l'erreur de mesure et de trouver une approximation de ces mesures par la méthode proposée en annexe.

Définition

Une fonction *spline cubique naturelle s* , de noeuds $x_1,\ldots,x_N \in [a,b]$ est telle que

(i) sur $[x_j, x_{j+1}]$, j =1 à N-1, s est un polynôme de degré ≤ 3,

(ii) $s \in C^2[a,b]$,

(iii) $s''(x_1) = 0$ et $s''(x_N) = 0$.

Une fonction spline cubique *s* est entièrement déterminée dès que l'on connait sa valeur $s(x_i)=a_i$ et la valeur de sa dérivée seconde $s''(x_i) = c_i$ en chaque noeud x_i.

Lissage

On cherche à reconstituer la fonction f en prenant en compte à la fois les mesures y_i aux points x_i et le caractère "régulier" de la fonction. Pour cela, on va chercher une approximation s_p de f telle que s_p minimise la fonctionnelle quadratique suivante:

$$J(v) = \frac{p}{N}\sum_{k=1}^{N}(y_k - v(y_k))^2 + (1-p)\int_a^b (v'')^2\,dx$$

On peut montrer que la fonction qui minimise J(v) est une **spline cubique naturelle** de noeuds $x_1 < x_2 < \ldots < x_N$. On appelle cette fonction $\mathbf{s_p}$ **spline d'ajustement** ou **spline de lissage**.

On peut considérer s_p comme la sortie d'un filtre passe-bas dont la fonction caractéristique est $f(\theta) = 1/(1+\lambda\times\theta^4)$ où $\lambda = (1-p)/p$. Le paramètre p est une constante $\in]0,1[$ qui pondère l'importance donnée à la proximité des mesures représentée par le premier terme et la régularité de la fonction représentée par le second terme (énergie de flexion). Quand p tend vers 0^+, la spline de lissage tend vers la droite des moindres carrés, alors qu'elle tend vers la spline d'interpolation quand p tend vers 1^-.

Calcul de la spline de lissage

Soient $s(x_i) = a_i$ et $s''(x_i) = c_i$ les valeurs de la fonction et de sa dérivée seconde en chaque noeud.

Les propriétés de continuité des dérivées premières et secondes et les conditions "naturelles" aux limites donnent les relations suivantes entre les vecteurs **a** et **c**, en posant $h_i=x_{i+1}-x_i$:

$$\begin{cases} c_1 = 0 \\ c_{i-1}\,h_{i-1} + 2c_i(h_{i-1}+h_i) + c_{i+1}h_i = 6\,[\,(a_{i+1}-a_i)/h_i - (a_i-a_{i-1})/h_{i-1}] \\ c_N = 0 \end{cases}$$

que l'on peut écrire sous forme matricielle

$$\mathbf{Rc}=6\,{}^t\mathbf{Qa} \qquad (1)$$

où **R** est une matrice (N-2,N-2), symétrique définie positive (si $h_i\neq 0$ pour tout i) et ${}^t\mathbf{Q}$ une matrice (N-2,N).

On écrit:

$$\int_a^b (s'')^2\,dx = \sum_{i=1}^{N-1} \int_{x_i}^{x_{i+1}} (s'')^2\,dx$$

et sur chaque $[x_i,x_{i+1}]$, on utilise une formule d'intégration exacte pour toute fonction l linéaire:

$$\int_0^h (l(x))^2\,dx = \frac{h}{3}\,[\,l(0)^2 + l(0)l(h) + l(h)^2]\,.$$

En exprimant s en fonction de **a** et **c**, on obtient l'expression suivante pour la fonctionelle J:

$$J(s)= p/N\ {}^t(\mathbf{y-a})(\mathbf{y-a}) +(1-p)/6\ ({}^t\mathbf{c\ R\ c}).$$

La matrice R étant inversible, (1) s'écrit $\mathbf{c}=6\,\mathbf{R}^{-1}\,{}^t\mathbf{Q\ a}$, et on obtient :

$$J(s)=p/N\ {}^t(\mathbf{y-a})(\mathbf{y-a}) + 6(1-p)^t(\mathbf{R}^{-1}\,{}^t\mathbf{Q\ a})\mathbf{R}(\mathbf{R}^{-1}\,{}^t\mathbf{Q\ a})\,.$$

Comme ${}^t(\mathbf{R}^{-1}\,{}^t\mathbf{Q})\mathbf{R}(\mathbf{R}^{-1}\,{}^t\mathbf{Q})=\mathbf{Q}\,\mathbf{R}^{-1}\,{}^t\mathbf{Q}$ est une matrice symétrique définie positive J(s) est minimum quand $-p/N(\mathbf{y-a})+6(1-p)\mathbf{QR}^{-1}\,{}^t\mathbf{Qa}=0$ et donc après multiplication par ${}^t\mathbf{Q}$:

$$p/N\ \mathbf{Rc} + 6(1-p)^t\mathbf{QQc} = 6p/N\ {}^t\mathbf{Qy} \qquad (2)$$

ou en posant c=6pu, on résoud

$$\boxed{\ p\ \mathbf{Ru} + 6N(1-p)\ {}^t\mathbf{QQ\ u} = {}^t\mathbf{Qy}\ } \qquad (3)$$

Après avoir déterminé **u** et **c** , on déduit **a** par $\mathbf{a}=\mathbf{y}-6N(1-p)\ \mathbf{Qu}$ (3')

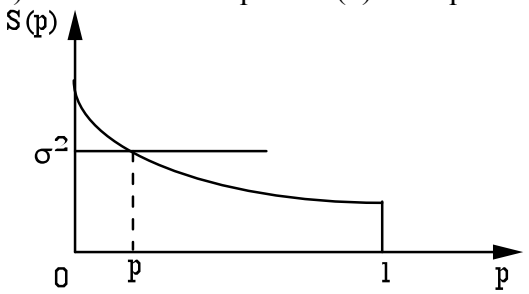
Résolution du système linéaire

La matrice du système linéaire (3) à résoudre est une matrice symétrique définie positive avec seulement 5 termes non nuls sur chaque ligne répartis autour de la diagonale. Pour mémoriser cette matrice on utilise au maximum 5 vecteurs et on résoud le système par la méthode d'élimination de Gauss adaptée à cette matrice.

Calcul du paramètre p

Dans le cas où la valeur σ^2 est connue, on peut déterminer p en imposant la valeur σ^2 à la variance "empirique" $S(p) = \frac{1}{N}\Sigma(s_p(x_i)-y_i)^2$.

La fonction S(p) est une fonction décroissante de p. La méthode de la sécante est bien adaptée pour déterminer la solution de $S(p)=\sigma^2$, en prenant comme valeurs initiales $p_0=1$ et $p_1=0$. Bien que les splines de lissage ne soient définies que pour $0<p<1$, S_0 et S_1 peuvent être calculées en utilisant les équations (3) et (3'), alors qu'il est nécessaire de faire un calcul particulier pour p=0 (droite des moindres carrés) si l'on utilise l'équation (2) et l'équation (2'): **a**=**y**-N(1-p)/p **Qc.**



Simulation d'une erreur de mesure

On veut simuler une variable aléatoire qui suit une loi normale $N(0,\sigma^2)$ pour une valeur σ donnée.

Soit X une variable aléatoire uniformément répartie sur [a,b]:

$E(X) = (b+a)/2$ et $var(X) = E(X^2) - E(X)^2 = (b-a)^2/12$.

Théorème centrale limite

Soient $X_1,\ldots,X_n$ une suite de variables aléatoires indépendantes de même loi centrée: $E(X_i) = 0$ et $var(X_i) = \alpha$ (constante). La suite de variables aléatoires

$$Y_n = \sum_{i=1}^{N} X_i \longrightarrow N(0, n\,\alpha)$$

Application:

On dispose de la fonction Random en Turbo-Pascal qui génère des nombres "pseudo uniformément répartis" dans [0,1].

Soient $X_i - \frac{1}{2}$ des tirages de cette v.a. $X_i - \frac{1}{2} \in [-\frac{1}{2}, \frac{1}{2}]$

$\Rightarrow E(X_i - \frac{1}{2}) = 0$ et $var(X_i - \frac{1}{2}) = 1/12$.

On fait n tirages de cette variable $\Rightarrow Y_n = \sum_{i=1}^{N}\left(X_i - \frac{1}{2}\right) \longrightarrow N(0, n/12)$

et $\sigma Y_n \longrightarrow N(0, n\,\sigma/12)$.

Si on prend **n = 12** $\Rightarrow$ $\sigma Y_{12} = \sigma \sum_{i=1}^{12}\left(X_i - \frac{1}{2}\right) \longrightarrow N(0, \sigma^2)$

Références bibliographiques

De Boor C. : A Practical Guide to Spline, Springer-Verlag, 1979.

Département informatique et réseaux
Première année
Langage C
Micro-projet
Sujet 41

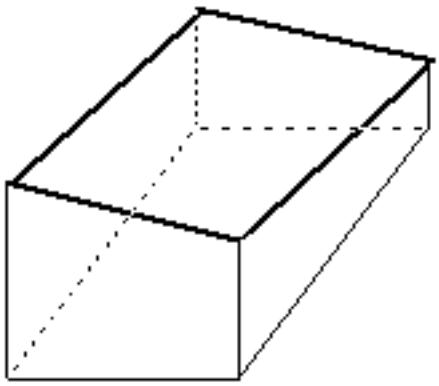
Montagnes fractales

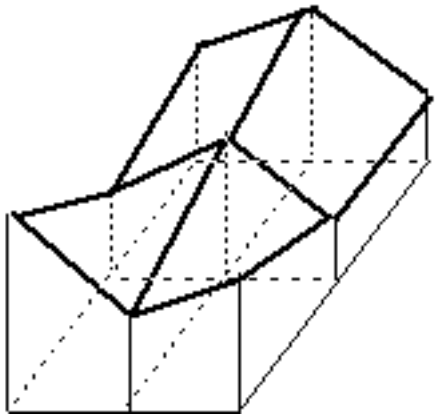
Niveau : 4

Responsable : Christine Potier

Principe :

Pour réaliser un paysage montagneux on utilisera une méthode de subdivision récursive (cf l'étoile de von Koch) mais au lieu de prendre un découpage régulier on ajoute une composante aléatoire (variable normale centrée) pour mieux rendre l'aspect "naturel".





On réitère le processus, un certain nombre de fois pour chaque carreau obtenu. Pour que la surface ainsi obtenue ne soit pas uniquement du bruit (pour qu'elle ressemble réellement à un terrain) on associe à chaque itération une hauteur qui décroit..

Visualisation :

Les coordonnées des carreaux sont calculées dans un repère propre au terrain (espace objet). On choisit pour la visualisation une position de l'observateur qui permettra de projeter les facettes à l'écran (espace image).

On choisira une source lumineuse (position du soleil) et on coloriera les facettes en considérant :

- la couleur du terrain suivant l'altitude (lacs, plaines, neiges éternelles .....)
- l'orientation des facettes par rapport " au soleil ".

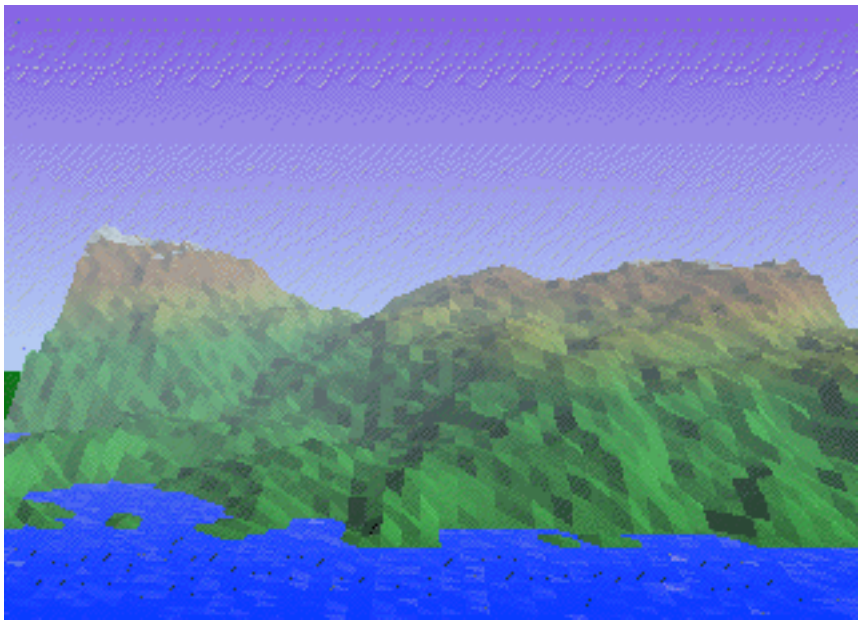
On pourra utiliser l'algorithme " du peintre " qui trie les facettes par distance décroissante à l'observateur et les affiche dans cet ordre.

Bibliographie :

" Principles of Interactive Computer Graphics ", Newman et Sproull, McGraw-Hill, 1984.

THE SPACE-TIME TRAVEL MACHINE
http://blanche.polytechnique.fr/lactamme/Mosaic/descripteurs/demo_14.html

Fractal Gallery: Scott Fuqua
http://mirror.wwa.com/mirror/gallerie/fracgall/fg950526.htm



Département informatique et réseaux
Première année
Langage C
Micro-projet
Sujet 42

Construction d'une surface lisse

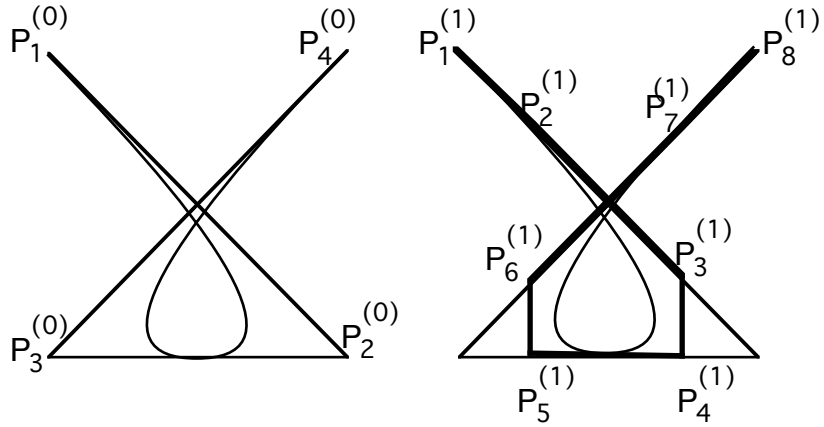
Niveau : 4

Responsable : Christine Potier

En Conception Assistée par Ordinateur, les surfaces sont très souvent représentées à partir d'un réseau de points de contrôle. Cette représentation est une généralisation à deux dimensions des méthodes utilisées dans les ateliers de dessin pour le tracé des courbes planes. Elle offre de nombreux avantages et en particulier celui de pouvoir "prévoir" la forme de la surface en voyant le réseau de points (avec un peu d'habitude).
On utilise ici une de ces nombreuses méthodes et on se place dans le cas particulier d'un réseau de points de contrôle $P_{i,j} =(u_i,v_{j,},z_{i,j})$ pour i=1 à N et j=1 à M. À ces points de contrôle, on associe une surface biquadratique ouverte S.

Pour tracer la surface, on calcule une approximation de cette surface par facettes polygonales puis on utilise une méthode de visualisation à partir d'une position de l'observateur. Pour cela on peut utiliser l'algorithme de **Catmull et Clark** qui est une généralisation (par produit tensoriel) à deux dimensions de l'algorithme de **Chaikin**. Cet algorithme est basé sur la subdivision du réseau de points de contrôle et cette subdivision sera répétée jusqu'à ce que l'approximation par facettes polygonales soit suffisamment "lisse" et proche de la surface.

Algorithme de Chaikin TRACÉ D'UNE COURBE
Soient P_1, P_2, …, P_N , N points du plan et le polygone ouvert $\mathbf{P}^{(0)}$ défini par { P_1, P_2, …, P_N }.
On associe à ce polygone $\mathbf{P}^{(0)}$ une courbe spline quadratique ouverte S (cf. figure) que l'on approche par une suite de polygones $\mathbf{P}^{(k)}$ qui convergent (?) vers la courbe S .



aux extrêmités

$$P_1^{(1)} = P_1^{(0)}$$
$$P_8^{(1)} = P_4^{(0)}$$

et pour i=1 à N-1

$$P_{2i}^{(1)} = \frac{3}{4} P_i^{(0)} + \frac{1}{4} P_{i+1}^{(0)}$$
$$P_{2i+1}^{(1)} = \frac{1}{4} P_i^{(0)} + \frac{3}{4} P_{i+1}^{(0)}$$

Soit $\mathbf{P}^{(k)}$ défini par { $P_1^{(k)}$, $P_2^{(k)}$, …, $P_r^{(k)}$ }, alors $\mathbf{P}^{(k+1)}$ est obtenu à partir de $\mathbf{P}^{(k)}$ par subdivision :

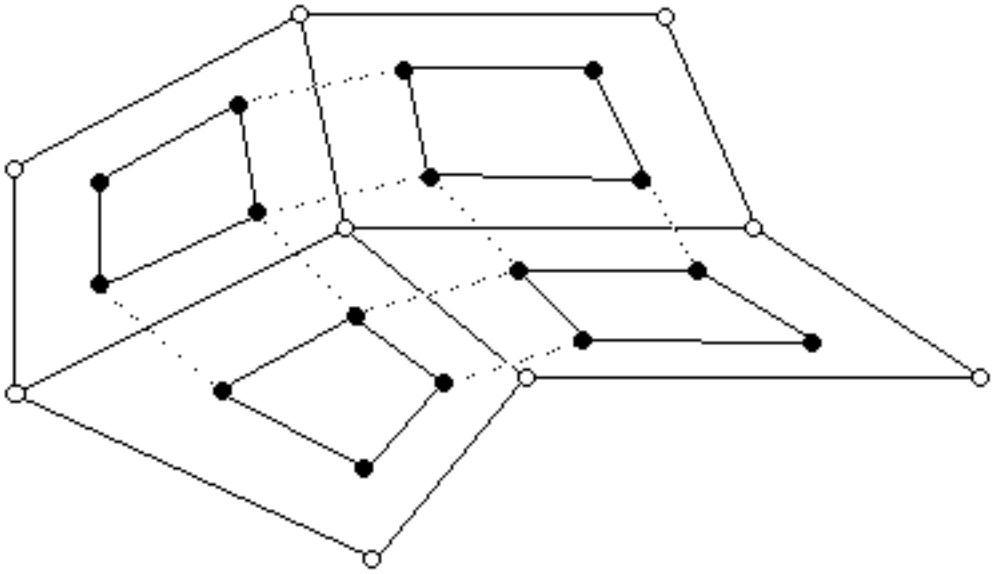
- $P_1^{(k+1)} = P_1^{(k)}$
- $P_{2i}^{(k+1)} = 3/4\ P_i^{(k)} + 1/4\ P_{i+1}^{(k)}$ pour i = 1 à r-1
- $P_{2i+1}^{(k+1)} = 1/4\ P_i^{(k)} + 3/4\ P_{i+1}^{(k)}$ pour i = 1 à r-1
- $P_{2r}^{(k+1)} = P_r^{(k)}$

Algorithme de Catmull et Clark

On passe du réseau $P_{i,j}$ au réseau plus fin $Q_{i',j'}$ (pour les points intérieurs) par les formules suivantes:

$$Q_{i',j'} = 1/16\ (\ 9\ P_{i-1,j-1} + 3\ P_{i-1,j} + 3\ P_{i,j-1} + P_{i,j}\ )$$
$$Q_{i',j'+1} = 1/16\ (\ 3\ P_{i-1,j-1} + 9\ P_{i-1,j} + P_{i,j-1} + 3\ P_{i,j}\ )$$
$$Q_{i'+1,j'} = 1/16\ (\ 3\ P_{i-1,j-1} + P_{i-1,j} + 9\ P_{i,j-1} + 3\ P_{i,j}\ )$$
$$Q_{i'+1,j'+1} = 1/16\ (\ P_{i-1,j-1} + 3\ P_{i-1,j} + 3\ P_{i,j-1} + 9\ P_{i,j}\ )$$

La restriction de la surface à chaque bord est une courbe spline que l'on approche par une ligne polygonale en utilisant l'algorithme de Chaikin (la surface passe par les 4 coins). On calculera les formules permettant d'obtenir les points du bord, comme dans l'algorithme de Chaikin. A chaque subdivision le nombre de sommets est multiplié par 4. En effet soit NxM le nombre de sommets avant subdivision. Le nombre de facettes est (N-1)x(M-1) donc on calcule 4x(N-1)x(M-1) points intérieurs, + 2 N +2 N +2 M + 2 M points sur les bords, - 4 coins (car les coins ont été comptés deux fois). Donc au total 4 NxM points.



Subdivision d'une surface biquadratique C1 par l'algorithme de Catmull et Clark

BIBLIOGRAPHIE :

BOEHM W., FARIN G et KAHMANN J : "A survey of curve and surface methods in CAGD", Computer Aided Geometric Design 1,1984 pp1-60.

MICCHELLI C.: Subdivision algorithms for curves and surfaces, dans "Extensions of B-spline curve algorithms to surfaces", SIGGRAPH 86, Course 5.

Département informatique et réseaux
Première année
Langage C
Micro-projet
Sujet 43

Contours d'une image noir et blanc

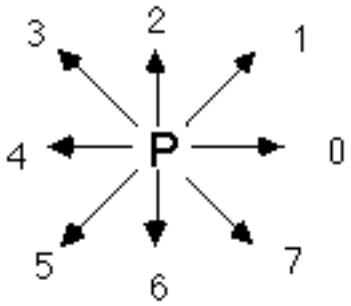
Niveau : 4

Responsable : Christine Potier

Un scanner permet de numériser un document sous forme d'image binaire. Cette représentation permet l'archivage et la consultation (en utilisant des algorithmes de compression) mais ne permet pas la modification car elle ne contient aucune information sur la structure du document. Quelle que soit la nature du document (texte ou graphique) il est nécessaire d'en déterminer les composantes connexes afin d'étudier chaque composante séparément (reconnaissance de caractères ou description géométrique). Chaque composante sera décrite par un contour extérieur et, éventuellement, plusieurs contours intérieurs.

QUELQUES DÉFINITIONS

Nous supposons qu'une image binaire est une matrice M*N de pixels ("picture elements") valant chacun 0 s'il appartient au fond (blanc) ou 1 s'il appartient à un tracé (noir).



P_3	P_2	P_1
P_4	**P**	P_0
P_5	P_6	P_7

Le pixel P, s'il n'est pas sur un bord de la grille a des voisins dans les 8 directions cardinales numérotées de 0 à 7 comme ci-dessus. Par la suite, comme nous nous intéressons uniquement aux pixels noirs, nous supposerons qu'il n'y en a pas sur les bords de la grille.et donc que chacun de ces pixels a huit voisins.

Le pixel P admet les pixels P0, P2, P4 et P6 comme voisins directs et les pixels P1, P3, P5 et P7 comme voisins indirects. On appelle voisin , un voisin direct ou indirect.

1) CONNEXIONS

- 2 pixels blancs sont connectés s'ils sont voisins directs
- 2 pixels noirs sont connectés s'ils sont voisins (directs ou indirects)
- un pixel noir et un pixel blanc ne sont pas connectés

L'image peut alors être considérée comme un graphe où les sommets sont les pixels et les arêtes les connexions entre pixels.

2) CONTOUR

Le contour est l'homologue discret de la frontière d'un ensemble du plan.

Soit F une composante connexe noire :

- P appartient à Contour(F) s'il a au moins un voisin direct blanc
- Le contour de F peut être décrit par un ou plusieurs cycles : le contour extérieur et, éventuellement, les contours intérieurs.

PARCOURS DU CONTOUR EXTÉRIEUR

On détermine un premier point P0 du contour en balayant le tableau ligne par ligne, de haut en bas, et pour chaque ligne en examinant les colonnes de gauche à droite : le premier pixel noir rencontré est P0 qui a pour voisins des pixel blancs dans les directions 2, 3 et 4.

L'algorithme de construction du contour peut être décrit comme la marche d'un observateur, au départ situé dans un pixel P0 du contour dont le 4-voisin est blanc et orienté vers le bas (direction 6), qui chemine dans les pixels de F en choisissant à chaque fois le pixel le plus à sa droite. La marche est terminé quand il est revenu en P0.

-Lorsque l'observateur arrive sur un pixel avec la direction D, le choix du pixel suivant est fait en examinant successivement la présence d'un pixel de F dans les directions : D-2, D-1, D, D+1, ..., D+4 (ces valeurs étant prises modulo 8) et en choisissant le premier rencontré (attention au cas particulier d'un pixel noir isolé !) .

- Pour reconnaître les pixels du contour dans le tableau initial, on incrémente de 1 la valeur d'un pixel (du contour) chaque fois qu'il est traversé.

- Chaque contour peut être décrit par P0 puis par la suite des directions empruntées (chaincode de Freeman).

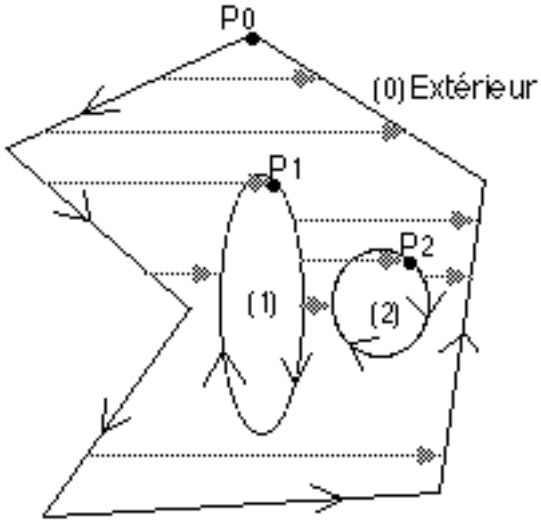
CONTOURS INTÉRIEURS

On applique le même algorithme pour déterminer les contours intérieurs. Le point de départ est choisi par un balayage horizontal de gauche à droite de l'intérieur de F, jusqu'à rencontrer soit le bord droit d'un trou non traité soit un contour déjà parcouru. De cette façon le contour extérieur est parcouru dans le sens trigonométrique direct et les contours intérieurs dans le sens inverse.

Pour le balayage on utilise une file (d'attente) Q de contours qui au départ est vide, puis contient le contour extérieur (qui existe toujours) et dans laquelle on met chaque contour intérieur successivement trouvé .

Définition : un pixel P du contour est sur une chaîne descendante si, dans le circuit, P est atteint par une direction D, 4<= D<= 7 et le successeur de P par une direction D', 5 <= D' <=7

On balaye uniquement l'intérieur de F, pour déterminer la présence éventuelle de trous, de la façon suivante (attention au 1er et au dernier points de chaque contour) :



Algorithme :

```
- Tant que Q est non vide faire.
     - Initialiser Pt-courant comme le 1er élément de Q
     - Si Pt-courant est sur une chaîne descendante (attention au 1er pt ) alors
          A <-- Pt-courant
          - Répéter {recherche vers la droite }
               Examen des triplets de pixels ABC dans la direction 6
               Si (A=0 et B=1) ou (A=0, B=2 et C=0 ) alors
                    Déterminer un contour intérieur à partir de B
                    Ajouter ce contour à Q et Terminé <-- vrai
               Si (A=0 , B >2 et C=0) ou (A=1, B=2 et C=0) alors
                    Terminé <-- vrai {on a atteint un contour tracé}
               Si on est au bord droit de la ligne
                    Terminé <-- vrai
          Jusqu'à terminé
     - Supprimer Pt-courant de Q
```

FICHIERS :

Pour vos tests, vous copierez des fichiers(~premiere/public_html/fichiers/image1.pbm ...image7.pbm) utilisant le format de représentation suivant:

P4
commentaires
256 264
suite de caractères correspondant à une suite de 0 et de 1 rassemblés 8 par 8
où
- P4 correspond au format de représentation,
- les lignes suivantes sont des commentaires personnels à condition de commencer par un dièse,
- la ligne suivante contient, en clair et dans l'ordre, le nombre de pixels par ligne et le nombre de lignes de l'image,
- les caractères suivants correspondent au codage des suites de 0 et de 1 représentant l'image (0=blanc, 1=noir).

BIBLIOGRAPHIE :

PAVLIDIS T. "Algorithms for Graphics and Image Processing" ,Computer Science Press, 1982

Première année
Département informatique et réseaux
Langage C
Micro-projet
Sujet 44

Intersection de courbes de Bézier

Niveau : 4

Responsable : Christine Potier

Soit une suite ordonnée de N+1 points du plan: {Pi=(xi,yi) }i=0 à N , qui définit un polygone à N côtés appelé polygone de contrôle. On appelle approximation de Bernstein-Bézier de cet ensemble de points, la courbe paramétrée:
Les polynômes p i,N(t) ont en particulier les propriétés suivantes :

$$B_N(t) = B_{(P_0,\dots P_N)}(t) = \sum_{i=0}^{N} P_i p_{i,N}(t) \quad \text{où} \, p_{i,N}(t) = C_N^i t^i (1-t)^{N-i} \quad \text{pour} \, t \in [0,1]$$

La dernière propriété entraîne que tout point P de la courbe est une combinaison linéaire

$$p_{0,N}(0) = 1 \text{ et } p_{i,N}(0) = 0 \text{ pour } i > 0 \text{ donc la courbe passe par } P_0$$

$$p_{N,N}(1) = 1 \text{ et } p_{i,N}(1) = 0 \text{ pour } i < N \text{ donc la courbe passe par } P_N$$

$$B'_N(0) = N(P_1 - P_0) \quad \{\text{resp. } B'_N(1) = N(P_N - P_{N-1})\}$$

$$\sum_{i=0}^{N} p_{i,N}(t) \equiv 1 \ \forall t \in [0,1]$$

convexe de P0, ..., PN et est à l'intérieur du polygone fermé défini par la frontière convexe de P0, ..., PN.

Les polynômes p i,N(t) vérifient la relation de récurrence :

$$p_{i,N}(t) = (1-t) p_{i,N-1}(t) + t\, p_{i-1,N-1}(t)$$

Cette propriété est utilisée dans l'algorithme de Casteljau pour la construction d'un point de la courbe correspondant à une valeur de t dans [0,1].

- On part du polygone à N côtés défini par Pi(0)=Pi pour i=0 à N.
- On construit les N points Pi(1)=(1-t) Pi(0) +t Pi+1(0) pour i=0 à N-1 qui forment un polygone à N-1 côtés (le point Pi(1) appartient au côté Pi(0)Pi+1(0)).
- A partir de ce polygone à N-1 côtés P0(1) à PN-1(1) on recommence l'opération précédente et on obtient N-1 points P0(2) à PN-2(2) qui définissent un polygone à N-2 côtés.
- Après N itérations, on obtient un point P0(N) qui est le point cherché.

SUBDIVISION
Soient C1 la courbe définie par le polygone { P0(0), P0(1), P0(N) } (diagonale supérieure du tableau de Casteljau) et C2 la courbe définie par le polygone {P0(N), P1(N-1), PN(0) } (diagonale inférieure du tableau de Casteljau). On montre que la courbe C composée des arcs de courbes C1 et C2 (segments) est identique à la courbe B définie par P0 = {P0, P1, ..., PN }. D'après la propriété de la frontière convexe les polygones P1 et P2 sont plus proches de la courbe que le polygone initial P0 donc constituent une meilleure approximation polygonale de la courbe initiale.
En répétant k fois cette opération de subdivision, sur chaque segment, on obtient 2k segments de courbe, dont la réunion est la courbe initiale. La réunion des polygones associés converge vers la courbe lorsque k -> 00 .

INTERSECTION de DEUX COURBES de BÉZIER

Soient deux courbes de Bézier BN et BM. Un point d'intersection de ces deux courbes (s'il existe) est solution de l'équation BN(t) = BM(z) pour t et z dans [0,1].
Or la détermination de t et z vérifiant cette équation suppose la transformation sous forme implicite des équations et conduit à des problèmes de degré élevé dans le cas général. On utilise les propriétés de la frontière convexe du polygone et de la subdivision pour déterminer l'intersection de deux courbes: deux courbes ne peuvent se couper que si les enveloppes convexes des polygones s'intersectent.

Algorithme :

- Si les enveloppes convexes de chacune des courbes sont disjointes il n'y a pas intersection,
- Sinon on subdivise chacune des courbes en deux et on compare les polygones ainsi obtenus deux à deux.

L'algorithme s'arrête soit parce qu'il n'y a pas d'intersection, soit quand les polygones obtenus par subdivisions successives sont suffisamment "petits".
L'approximation du point d'intersection des deux courbes sera un point (le barycentre par exemple) de l'intersection des deux enveloppes convexes finales.

Cette méthode se programme facilement par une procédure récursive. Pour chaque segment, on gardera les points de contrôle du polygone et l'intervalle de variation du paramètre initial. Le résultat de l'algorithme sera d'une part le point A=(x,y), d'autre part les valeurs t et z du paramètre pour chaque courbe. On peut évaluer l'algorithme en comparant les points BN(t) , BM(z) et A=(x,y).

Le test pour savoir si deux polygones convexes se coupent est assez difficile à mettre en oeuvre dans le cas quelconque et peut être un peu long. On peut le remplacer par le test de la boîte min-max qui ne donne le même résultat que dans le cas où il n'y a pas intersection. On appelle boîte min-max le plus petit rectangle parallèle aux axes qui contient le polygone convexe et au lieu de comparer les deux polygones on compare les deux boîtes min-max. La boîte min-max peut aussi être utilisée pour savoir si un polygone est suffisamment "petit".

BIBLIOGRAPHIE :

B.A. BARSKY : " Arbitrary subdivision of Bézier curves " Technical report UCB/CSD 85/265, University of California Berkeley, California 94720

J.M. LANE et R.F. RIESENFELD : " A Theoretical Development for the Computer Generation and display of Piecewise Polynomial Surfaces " , IEEE Trans. on PAMI, Vol. PAMI-2, No 1, Janvier 1980, pp. 35-46