PROJETS EN PROGRAMMATION ORIENTEE OBJET

Les niveaux de difficultés des projets vont de 1 à 10. Chaque niveau de difficulté se décompose en

1. Difficulté conceptuelle du projet /5. Certains projets comprennent une forte composante exploratoire
2. Difficulté d'implémentation /5. Celle-ci est généralement assez déconnectée de la première. Elle concerne à la fois le volume du code minimum à produire et la difficulté des techniques à maîtriser.

Pour un projet sa difficulté est donnée sous la forme (a,b) : une difficulté conceptuelle de a, une difficulté d'implémentation b. La difficulté d'un projet est a+b.

Pour tous les projets, ils sont librement implémentables en java ou en C++. Comme au final le cours aura beauoup moins porté sur C++, les projets codés dans ce langage seront regardés avec beaucoup plus de complaisance.

Pour chaque projet, l'évaluation se fera sur la base :

- Du code rendu et d'un **petit rapport** expliquant le fonctionnement du code, des classes produites et illustrant le fonctionnement du programme (copie d'écran pour les applications graphiques, résultats de simulations etc...)
- Chaque projet fera l'objet d'une **soutenance individuelle** de chacun des membres qui lui sont attachés. Cette soutenance comprendra 5 questions sur le code produit, 5 questions d'ordre général sur la programmation orientée objet.

Le volume de code produit pour les projets VBA a été plus que satisfaisant et il ne vous est pas nécessairement demandé de produire autant de code. N'hésitez pas à faire des points sur vos projets, ce que vous pensez rendre au final, ce que vous voulez implémenter, les difficultés que vous rencontrez, voire à demande de l'aide au démarrage, la spécification d'étapes intermédiaires. Pour chaque projet, il est en effet possible de le décomposer en étapes. Enfin, deux personnes m'ont soumis des idées de projets personnels que j'ai validées après en avoir parlé avec eux, je le signale par soucis d'équité et j'offre la même possibilité à d'autre sous résèrve de l'intérêt de la proposition.

PROJETS-FINANCE

Projet A.1 Simulation de comportements d'investissement sur un marché boursier (3,3)

A partir d'un certain nombre de comportements automatiques d'achat-vente (stratégies disponibles sous forme de classes) : création de code pour faire de la simulation massive sur différents historiques financiers : on regarde pour différents intervalles de temps quelles sont les comportements les plus efficaces.

Matériel fourni : des fichiers d'historiques d'actifs et une classe de récupération des données depuis les fichiers.

Projet A.2 SimulationEcologique sur des populations de stratégies d'investissement (5,5)

On étudie des populations d'agents sur un marché. Les agents sont susceptibles d'adopter différents de comportements. Dans une démarche écologique, les comportements les plus performants se diffusent au sein de la population, les comportements les moins performants disparaissent : quels sont les comportements sélectionés ? Le projet consiste à créée r des populations d'agents aléatoirement et à voir leur évolution au cours du temps.

Matériel fourni : des fichiers d'historiques d'actifs et une classe de récupération des données depuis les fichiers.

Pour les deux projets suivants, ils sont à demi ouverts. Pour le premier, il est très prospectif, pour le second, beaucoup de choses ont déjà été faites et il s'agit de trouver un nouveau point de vue intéressant.

Projet A.3 Valeur de la couverture du risque de change par une méthode Monte Carlo(5,2)

Projet A.4 Pricing d'option par simulation monte carlo (5,3)

Projet A.5 Algorithme génétique pour la sélection de portefeuille (5,3)

Les algorithmes génétiques constituent des méthodes de recherche de solutions à un problème donné. Généralement ils sont appliqués pour des problèmes où l'espace des solutions est trop grand pour être entièrement parcouru en testant chaque solution indépendamment afin de trouver la meilleure. L'une des applications les plus citées est la résolution du problème du voyageur de commerce : soient N villes, on donne une liste des distances entre chaque couple de villes. Dans quel ordre les villes doivent être parcourues pour minimiser la distance totale ? Ce problème est dit difficile en informatique (NP-complet). Une des solutions est de générer une population de solutions, soit une population d'ordres sur les villes : V=(7, 6, 3, 4, 1, 2, 5) correspond au fait de d'abord passer par la ville 7, puis la ville 6 etc... Il est possible d'associer une distance totale à chaque solution, pour V, la distance totale est la distance entre la ville 7 et la ville 6, entre la ville 6 et la ville 3, entre la ville 3 et la ville 4 etc. A chaque génération, on récupère les M solutions qui ont les meilleures performances, ie celles qui minimisent la distance totale. On crée M/2 couples à partir des M plus performantes et pour chaque couple, on crée une nouvelle stratégie à partir des deux stratégies du couple. On a ainsi M/2 nouvelles solution, on substitue ces M/2 nouvelles solutions aux M/2 moins performantes de la population. On obtient ainsi une nouvelle génération. On répète les étapes suivantes un certain nombre de fois :

- Evaluation de la performance des solutions de la population
- Selection et croisement des individus les plus performants
- Remplacement des individus les moins performants par les individus issus des croisements.

Cette démarche reproduit une démarche de selection Darwinienne. La performance des solutions est censée augmenter au cours du temps. On se propose de faire la même démarche pour la selection de portefeuilles d'actifs. Un lien décrivant la démarche : http://lsis.univ-tln.fr/~tollari/TER/AlgoGen1/node2.html**.**

On veut choisir un portefeuille d'actifs au temps t. On dispose de données sur ces actifs de t-n à t-1. Une solution est une distribution du portefeuille entre les différentes actifs : (0, 0.5, 0.05, 0 ,0, 0.45). La fonction de performance est libre, si on se cantonne à des données d'un marché actions, on utilise comme performance la rentabilité et le risque. Si on considère d'autres actifs, on peut rajouter la liquidité comme élément de performance. Le projet doit implémenter un algorithme génétique (les choix des données, des méthodes d'évaluation de la performance etc. sont libres) et montrer l'évolution de la performance moyenne de la population en fonction des générations successives.

Matériel fourni : des fichiers d'historiques d'actifs et une classe de récupération des données depuis les fichiers.

AUTRES PROJETS

Projet B.1 Premiers essais de métrologie du Web (4,3)

Ce projet demande l'acquisition de très légères connaissances en html

On dispose d'un outil pour récupérer des pages Web sur une machine locale (tous les langages le permettent). Cette page Web est traitable comme un fichier texte, il est notamment possible d'identifier les liens vers d'autres pages Web. Chaque page Web est reliée à d'autres pages par des liens. Ces pages et ces liens forment donc un graphe. La première étape du projet consiste à créer un code qui permet de récupérer toutes les pages qui sont référencées par une page Web. Le projet consiste ensuite à calculer des statistiques sur le graphe formé par les pages Webs.

Quel intérêt ? On peut par exemple chercher à analyser les réseaux de sites identifiables à partir des premières pages des sites d'informations : ces sites pointent ils vers de même sources d'informations ? Pointent ils vers des blogs ?

Combien d'images sont intégrées à ces premières pages ? Quelle évolution des pages référencées d'un jour sur l'autre ?

Matériel fourni : l'outil de récupération de pages Web en ligne et une classe de récupération des données depuis les fichiers.

Projet B.2 Animation pour montrer la diminution de la complexité en fonction des algorithme de tri (1,2)

Le projet consiste à implémenter trois algorithmes de tri de tableau (ces algorithmes sont donnés) et à voir l'évolution du temps de tri moyen en fonction de la taille du tableau, pour chacun des algorithmes.

Matériel fourni : java fournit des éléments de gestion des dates.

Projet B.3 Animation pour montrer la diminution de la complexité en fonction des algorithme de tri Version 2 (3,5)

Idem que le Projet B.2, mais on rajoute la création d'une interface graphique qui montre le tableau au fur et à mesure qu'il est trié.

Projet B.4 Etudes textuelles (1,2)

Projet d'analyse de texte. Il s'agit d'étudier les réseaux sémantiques au sein d'un texte. Une première partie consiste dans la proposition d'une série de retaitements qui permettent d'obtenir des textes sous un format standard : gestion des caractères accentués, gestion des mots non sémantiques "et", "à" etc... Puis on peut proposer diverses statistiques comme la co-occurence des mots dans un texte, différents indicateurs sont possibles : pondération d'un couple de mots en fonction de la distance entre les deux mots, prise en compte d'un couple de mots si ils sont à une distance inférieure à un seuil n etc...

Matériel fourni : Une classe de récupération des données depuis les fichiers.

Projet B.5 Parours de labyrinthe (1,3)

Dans un labyrinthe en deux dimensions L*H, un mobile se déplace d'une case vers une case adjacente prise au hasard. Combien de temps faut il pour que le mobile ait parcouru l'ensemble de l'espace ? Et si on interdit le parcours de certaines cases (obstacles) ? Et si on rajoute une dimension ?

Projet B.6 Parcours de labyrinthe Version 2 (1,5)

Le même projet que le précédent mais avec un affichage graphique en plus

Projet B.7 Programmation graphique (1,4)

A partir de codes minimaux, il est possible d'obtenir des motifs très complexes analysés en théorie du chaos (dimensions non entières etc...). On se propose de programmer un certain nombre de fractales à l'aide des outils graphiques de Java. La complexité du projet peut très vite augmenter : en commençant à prendre le redimensionnement des fenêtres etc...

Matériel fourni : Un premier exemple graphique et les algorithmes de dessin à implémenter

Projet B.8 Projet de description d'un projet Open source (4,4)

Ce projet consiste à récupérer le code d'une application open source (le choix en est libre : http://sourceforge.net/) et de commenter l'ensemble du code qui a été produit. Le travail n'est pas un travail d'implémentation en soi, mais la compréhension du code d'autrui peut s'avérer extrêmement complexe.

Projet B.9 Creation d'un réveil-agenda-ordonanceur de tache (3,3)

On crée un programme de réveil et d'ordonancement de tache sur la machine. Ce programme se déclenche lors du lancement de la machine, il affiche des messages, lance des fichiers audio et vidéo ou tout type de programme aux heures souhaitées. Ce programme se lance notamment à l'ouverture de la machine. Ce projet demande de faire 5 lignes de C++ pour créer un .exe qui sera appelé au démarrage., d'apprendre à lancer l'exécution d'un programme au démarrage de la machine, de rentrer et de stocker des évenements.

Matériel fourni : Une classe Date, un méthode pour lancer des commandes depuis java. Une classe de récupération des données depuis les fichiers.

Projet B.10 Test du chi-deux (3,3)

Implémentation du test du chi-deux : on passe une série statistique en entrée, on choisit une loi de probabilité, on fait un test du chi deux pour cette série statistique et cette loi de probabilité

Matériel fourni : Un fichier texte contenant la table pour la loi du chi-deux. Une classe de récupération des données depuis les fichiers.