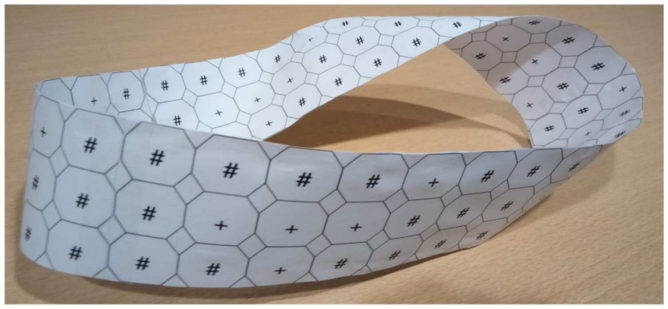
UNIVERSITÉ PARIS DESCARTES

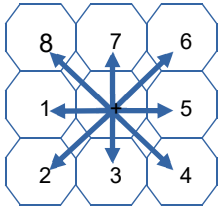
DUT Informatique - 1ère année
SDA - Structures de Données et Algorithmes
Equipe pédagogique
Marie-José Caraty, Denis Poitrenaud,
Julien Rossit, Camille Kurtz,
Jacques Alès-Bianchetti, Denis Jeanneau

PROJET

UE - STRUCTURE DE DONNEES ET ALGORITHMES

Dédale sur un ruban de Möbius à pavage octogonal





Un Dragon ailé est missionné pour la recherche des Plans du Monde, à l'abri des convoitises dans un labyrinthe plus redoutable encore que celui de Dédale : ce labyrinthe est un ruban de Möbius à pavage octogonal. Notre dragon est heureusement doué d'une mémoire phénoménale ; il est en effet capable de mémoriser tous ses déplacements d'une façon aussi sûre que le fil d'Ariane. C'est cette faculté exceptionnelle qui lui permettra de sortir du labyrinthe par l'entrée qu'il avait emprunté dès qu'il aura accompli sa mission.

Le projet consiste à simuler la mission du Dragon : i) pénétrer par l'entrée du labyrinthe, ii) rechercher les Plans du Monde jusqu'à leur découverte, iii) tracer au cours de la recherche le chemin que mémorise le Dragon : de la position de l'entrée du labyrinthe à sa position. Des algorithmes choisis vont nous permettre de résoudre le problème de la recherche des Plans du Monde et de la sortie du labyrinthe par une méthode de planification qui vous aidera aux prises de décision. Le labyrinthe sur *un ruban de* Möbius à *pavage octogonal* peut être matérialisé par deux damiers, un pour chaque face. Chaque damier comporte (m*n) cases avec m le nombre de colonnes et n le nombre de lignes. Les colonnes vont d'ouest en est et les lignes du nord vers le sud. L'extrémité est de chaque damier est jointive de l'extrémité ouest de l'autre damier après une torsion d'un demi-tour. Le ruban ainsi obtenu est sans fin et n'a ni intérieur, ni extérieur. Chaque case (autre que les cases aux extrémités sud et nord du ruban) est voisine de 8 autres cases. Chaque case porte un symbole : « # » représente un mur, « + »

un espace permettant un passage possible, « D » la position du Dragon et « P » la position des Plans du Monde.

Ce labyrinthe est décrit dans un fichier texte. La première ligne du fichier indique les valeurs m, n des dimensions x (colonne) et y (ligne). A partir de la deuxième ligne du fichier texte, les deux damiers du labyrinthe (séparés par une ligne blanche) sont donnés. Par convention, la case (0,0) du 1er damier correspond au 1er symbole de sa 1ère ligne (idem pour le second damier). Vous trouverez ci-dessous un exemple de labyrinthe décrit dans le fichier texte.

Les 8 déplacements autorisés dans le labyrinthe sont les suivants dans l'ordre d'exploration : ouest (1), sud-ouest (2), sud (3), sud-est (4), est (5), nord-est (6), nord (7) et nord-ouest (8). Le Dragon qui occupe une case originellement « + » ne peut se déplacer que sur l'une des cases voisines, à condition qu'il s'agisse d'une case de symbole « + ». Le passage d'un damier à l'autre se fait à partir des extrémités « ouest » et « est ». Par exemple, si les cases (m-1, y) du damier 1 et (0, n-2-y) du damier 2 ont comme symbole « + », il est possible de passer de la première à la seconde par un déplacement sud-est (4).

Pour trouver les Plans du Monde, on programmera l'algorithme donné en annexe. Cet algorithme est fondé sur une pile qui permet de mémoriser les index des positions correspondant à des cases qui ne sont pas encore visitées et à prospecter dans le futur (lorsque les déplacements conduisent à une impasse). Pour sortir du labyrinthe, on utilisera un autre type abstrait de données (que vous aurez à déterminer) pour mémoriser le chemin « utile » parcouru depuis l'entrée du labyrinthe aux Plans du Monde et qui simule la mémoire du Dragon.

```
30 4
#######D######################
+#++#++++++#####++++##########++++          Face 1
#+###+#+###++#######++####+##+#++
##############################

##############################
+#+########++++++#####++########+          Face 2
+++++P+++++++++#+++++#+++++++++
##############################
```

Exemple d'un labyrinthe à deux faces comportant chacune 4 lignes et 30 colonnes

Tests et recette de l'application

Vous trouverez sous COMMUN le labyrinthe `labMobius.txt` qui vous servira aux phases de développement et de test. Le développement sera incrémental à base de Sprints (5 au total). Au Sprint 5, votre application devra évidemment donner des résultats valides sur ce labyrinthe qui admet un chemin allant de l'entrée du labyrinthe aux Plans du Monde. Vous aurez également à tester une configuration de labyrinthe sans chemin d'accès au Plans en modifiant, par exemple, le labyrinthe fourni. Les positions du Dragon à l'entrée du labyrinthe et des Plans sont quelconques. Ainsi, au moins deux tests seront exigés à chaque Sprint.

La semaine du 6 janvier 2020, vous passerez la recette de votre application lors de votre séance de SDA3. Il s'agira de tester votre programme sur un labyrinthe de grande taille en présence de votre enseignant. Vous aurez à compiler et à exécuter votre programme sous Eclipse. Vous vérifierez la validité de votre solution par comparaison à la solution attendue qui vous sera communiquée par votre enseignant par la fonction compare d'Eclipse. Si les deux fichiers sont identiques votre application est 0-défaut sur le jeu de test (JDT) c'est-à-dire le labyrinthe de test, sinon le programme sera testé sur le sprint inférieur. Pour la comparaison des solutions, vous devez créer le fichier texte correspondant à la trace d'exécution demandée pour chacun des 5 sprints.

Cadre du développement logiciel

Le projet est à réaliser en binôme. Les membres d'un binôme ne doivent pas nécessairement être du même groupe de TD. Vous devez programmer et tester l'application demandée.
L'un des objectifs de ce projet est la réutilisation de composants. Vous devez choisir er réutiliser les composants techniques (pile, file ou liste) vus en cours et/ou développés en TP, pour la résolution des problèmes posés par l'application.
Votre travail sera évalué à partir i) des sources de votre application, ii) des traces d'exécution des tests effectués et iii) de la rédaction d'un dossier de développement logiciel.

Dossier du développement logiciel

Vous devez porter une attention particulière à la rédaction de votre dossier. Sa qualité est déterminante pour l'évaluation de votre travail. La composition de votre dossier doit être la suivante :

Une page de garde indiquant le **nom** et le **groupe** des membres du binôme, l'objet du dossier. Une table des matières paginée sur l'ensemble du dossier incluant les annexes de code. Une introduction d'une page du projet. Le graphe de dépendance des fichiers sources de votre application. Tous les composants (réutilisés ou développés) de l'application devront figurer sur le graphe (cf. Cours 4). L'organisation des tests de l'application. Le résultat des tests. Pour les tests que vous concevrez, vous donnerez les jeux de données de test (JDT) et les solutions en annexe. Un bilan du projet (les difficultés rencontrées, ce qui est réussi, ce qui peut être amélioré).
En annexe au dossier :
1) le listing complet de vos sources. La présentation doit suivre l'ordre suivant : i) le point d'entrée de l'application (programme principal), ii) la liste des composants utilisés par l'application (tous les fichiers de spécification (.h), suivi de tous les fichiers corps (.cpp) dans le même ordre).

2) chacun des jeux de données de test (JDT) que vous avez conçu pour votre application (fichier texte du labyrinthe testé) et sa solution (fichier texte de la solution). Tout JDT doit être commenté de manière à illustrer le test effectué.

Spécifications de programmation

Le code source doit être structuré en composants pour une compilation séparée. Les codes-sources doivent être commentés (cartouche, structures de données et

fonctions) suivant le formalisme Javadoc. Les conventions de nommage données en cours doivent être respectées. Les préconditions des fonctions doivent être dans la documentation et être testées par assertion dans le code.nVous veillerez à ce que votre programme ne présente aucune fuite mémoire. Suivez toutes les spécifications données sous peine de pénalisation : i) convention de référentiel, ii) ordre d'exploration des chemins(cf. algorithme), iii) format du fichier texte de la solution.

Date limite de remise de projet

La date limite de remise du dossier de programmation (version papier) est fixée le **7 janvier 2020** (au secrétariat). Vous déposerez également dans le puits (SDA/Gr?) vos **sources et version électronique de votre dossier** (archive *zip*) de format de nom « *Nom1Gr?Nom2Gr?* » le **7 janvier 2020** (au plus tard à 23h59).

Annexe. Algorithme de recherche dans le labyrinthe

```
Pile utilise Positions, Booléen
Pile P ←PileVide

empiler l'index de position (iD, jD) de l'entrée du labyrinthe sur la pile p
// Rem : l'index correspond également à la position du Dragon

tant_que (le Dragon n'a pas trouvé les Plans du Monde et que p n'est pas vide)
      (i, j) ← sommet(p) // index de position courant
      // Rem : au premier passage, (i, j) est l'index de position du dragon
      // à l'entrée du labyrinthe
      dépiler l'index de position (i, j) de la pile p
      mettre à jour le « chemin connexe » de l'entrée du labyrinthe au Dragon
      si(l'index de position (i, j) n'est pas celui des Plans du Monde)
           empiler dans la pile p chacune des positions autorisées
           pour le déplacement
           // Rem : parmi les 8 cases autorisées : celles qui sont des cases
           // du labyrinthe symbolisées par «+» et « non visitées »
           // empiler impérativement dans l'ordre relatif au référentiel défini
           // ouest (x-1, y), sud-ouest(x-1, y+1), sud (x, y+1), sud-est (x+1, y+1),
           // est (x+1, y), nord-est (x+1, y-1), nord (x, y-1) et nord-ouest(x-1, y-1)
           marquer « visitée » la case (i, j) du labyrinthe
      fin_si
fin_tant_que
si (l'index de position est celui des Plans du Monde)
      // le Dragon a trouvé les Plans, il est heureux !!!
      afficher le bonheur du Dragon…
      éditer la solution     // suite des index de position du chemin
                             // de l'entrée du labyrinthe aux Plans du Monde
sinon
      // tout le labyrinthe a été exploré (p est vide)// Plans inaccessibles
      afficher le feu du mécontentement du Dragon…
fin_si
```