université PARIS-SACLAY

Travaux Dirigés du projet Fourmis

Cours de programmation modulaire

—Licence MI/IM/MNSI - Info 3—

Analyse descendante du projet

Dans le TD précédent, nous avons effectué une analyse ascendante du projet. En effet, nous avons listé les opérations élémentaires nécessaires à la simulation des fourmis, et nous avons utilisé ces opérations basiques pour construire des opérations plus compliquées, elles-mêmes pouvant êtres combinées à nouveau.

Dans ce TD, nous allons adopter l'approche inverse, c'est à dire effectuer une analyse descendante du problème. Cela consiste à considérer le problème dans son ensemble, et le diviser en plusieurs sous-problèmes plus simples. Ces derniers peuvent être résolus indépendamment les uns des autres (éventuellement par des programmeurs différents) et peuvent être eux-mêmes divisés en sous-problèmes. Cette analyse descendante peut-être menée jusqu'à l'atteinte des sous-problèmes déjà traités dans les TD et TP précédents.

1 Algorithme principal de la simulation

1. Réalisez l'algorithme principal de la simulation en supposant disponibles les procédures spécifiées ci-dessous.

— procédure `initialiserEmplacements` (Résultat lesFourmis : `tabFourmis`,
Résultat leSucre : `ensCoord`,
Résultat leNid : `ensCoord`)
/* Initialise les emplacements des fourmis, du sucre et du nid */

— procédure `dessinerGrille` (Donnée laGrille : `grille`)
/* Dessine une image de la grille dans un fichier */

— procédure `mettreAJourEnsFourmis` (Donnée-Résultat laGrille : `grille`,
Donnée-Résultat lesFourmis : `tabFourmis`)
/* déplace toutes les fourmis en appliquant les règles de leur comportement */

Le problème de la simulation a maintenant été divisé en sous-problèmes dont certains ont été traités dans le TD8, et d'autres sont traités par les trois actions ci-dessus. Dans la suite du TD, nous traitons le sous-problème le plus difficile, à savoir la mise à jour des positions de l'ensemble des fourmis.

2 Déplacement des fourmis

2. Réalisez la procédure `mettreAJourEnsFourmis` utilisée dans l'algorithme principal, en supposant disponible la procédure spécifiée ci-dessous.

— procédure `mettreAJourUneFourmi` (Donnée-Resultat laGrille : `grille`,
Donnée-Résultat uneFourmi : `fourmi`)
/* déplace une fourmi en appliquant les règles de comportement des fourmis */

3. Réalisez la procédure `mettreAJourUneFourmi`, en supposant disponibles la fonction et la procédure spécifiées ci-dessous.

— fonction `condition_n` (Donnée x : entier, Donnée f : `fourmi`,
Donnée p1 : `place`, Donnée p2 : `place`) → booléen
/* retourne Vrai si la condition de déplacement de la règle numéro x s'applique sur la fourmi f, située sur la place p1, pour un déplacement vers la place p2 */
— procédure `action_n` (Donnée x : entier, Donnée-Résultat f : `fourmi`,
Donnée-Résultat p1 : `place`, Donnée-Résultat p2 : `place`)
/* applique l'action de déplacement de la règle numéro x sur la fourmi f, située sur la place p1,
pour un déplacement vers la place p2 */

NB : On suposera aussi disponibles : la fonction `cardinal` qui retourne le nombre d'éléments d'un ensemble de coordonnées `ec` passé en paramètre, ainsi que la fonction `iemeElement` qui prend en paramètres un ensemble de coordonnées `ec` et un entier `i` et retourne le `i`-ème élément de `ec`.

4. Adaptez votre algorithme en prenant en compte la remarque sur la règle 6 dans l'énoncé du projet. On supposera disponible la fonction spécifiée ci-dessous.

fonction `voisinVideAleatoire` (Donnée g : `grille`, Donnée p : `place`) → `coord`
/* retourne les coordonnées dans la grille g d'un voisin vide de p choisi aléatoirement */

3 Règles de déplacement

5. Réalisez la fonction `condition_n` et la procédure `action_n` en supposant disponibles les fonctions `condition1`, ... , `condition6` qui correspondent aux conditions des règles 1 à 6 et les procédures `action1`, ... , `action6` qui correspondent aux actions des règles 1 à 6.

fonction `condition1` (Donnée f : `fourmi`, Donnée p1 : `place`, Donnée p2 : `place`) → booléen
/* retourne Vrai si la condition d'application de la règle de déplacement 1 est vérifiée */

procédure `action1` (Donnée-Résultat f : `fourmi`, Donnée-Résultat p1, p2 : `place`)
/* applique la règle de déplacement 1 */