UNIVERSITÉ D'ANGERS

Presentation du projet de programmation fonctionnelle en OCAML

Par

ZENINE Hamid

Année Universitaire 2023-2024

Idée generale:

Le but generale du projet est de pouvoir gerer des grille (carrées) de taille donnée et de trouver des chemins de differnetes manieres (efficaçités differentes) entre deux grilles.

Implementation:

Types:

Direction : Droite | Bas | Gauche | Haut

Fonctions:

I. position grille x : retourne la position d'un entier "x" dans une grille "grille". Retourne une erreur si l'entier n'est pas present.

II. valeur grille p : retourne la valeur d'une position "p" (indiçage a partir de 0) dans une grille "grille". Retourne une erreur si la grille est trop petite.

III. modifier_valeur_indice grille p v : Modifie la valeur d'une position "p" (indiçage a partir de 0) dans une grille "grille" en y plaçant la valeur "v"

IV. echange grille v1 v2 : echange les positions de deux valeurs "v1" et "v2" dans une grille "grille"

V. deplacement_possible position tailleGrille direction : Verifie si le deplacement (explicité par direction de type Direction) dans d'une position "position" grille de taille "tailleGrille"
   - Haut => la case n'est pas dans la premiere ligne (position>=tailleGrille)
   - Bas => la case n'est pas dans la derniere ligne (position < tailleGrille*(tailleGrille-1)
   - Gauche => la case n'est pas dans la colonne tout a gauche (position mod tailleGrille != 0)

- Droite => la case n'est pas dans la colonne tout a droite (position mod tailleGrille != (tailleGrille-1))

VI. nouvelle_position tailleGrille oldPos direction : Retourne la nouvelle position apres deplacement (explicité par direction de type Direction) dans une grille de taille "tailleGrille" depuis "oldPos"
- Haut => On fait remonter d'une ligne (oldPos – tailleGrille)
- Bas => On fait descendre d'une ligne (oldPos + tailleGrille)
- Gauche => (oldPos – 1)
- Droite => (oldPos + 1)

VII. deplacer puzzle direction: deplace le 0 dans un puzzle (puzzle = (grille, tailleGrille)) dan sune direction donnée (explicité par direction de type Direction) utiisant ainsi deplacement_possible pour decider si le deplacement est valable, et aussi nouvelle_position pour avoir la nouvelle position du 0 apres deplaçement

VIII. melange puzzle iterations : effectue un nombre de mouvements aléatoires correspondant au nombre d'itérations. Les mouvements sont ainsi pris au hasard (avec Random.int 4) et le nombre d'iterations est egal au nombre de mouvements réellement effectués sur la grille initiale (les mouvement impossibles ne sont pas comptabilisés)

IX. testerChemin puzzle chemin : "Suit" le chemin "chemin" (direction list) en effectuant recursivement les deplacement correspondant au directions du chemin et renvoie la grille resultante.

X. liste_grilles_chemins (grilleInit, chemin) taille : Calcule toutes les grilles resultantes d'un seul deplacement (Droite | Bas | Gauche | Haut) a partir d'un grille "grilleInit" (Attention: ne garde que les deplacements valables) et retourne une liste de grille avec leurs chemin mis a jour (chemin auquel on ajoute la direction prise)

XI. parcours1_grille_finale_taille_prochaines_grilles : Trouve et retourne le chemin entre une grille initiale et une grille finale en utilisant un traitement “naif” (test de toutes les possibilités avec la fonction liste_grilles_chemins sans enregistrer les parcours precedents)

Resultat tests:

[3;1;2;4;7;5;0;6;8] => [0;1;2;3;4;5;6;7;8] -> 220ms

[6;3;1;4;0;2;7;8;5] => [0;1;2;3;4;5;6;7;8] -> 256ms

[1;4;3;6;2;8;7;5;0] => [0;1;2;3;4;5;6;7;8] -> Stack overflow apres 17min

XII. parcours2_grille_finale_taille_prochaines_grilles_grilles_parcourues : Trouve le chemin entre une grille initiale et une grille finale en utilisant un traitement “naif amelioré” (test de toutes les possibilités avec la fonction liste_grilles_chemins en enregistrant les grilles parcourues pour eviter les traitement repetitifs)

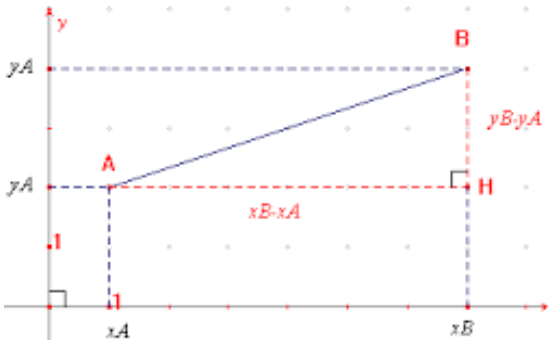
Resultat tests:

[3;1;2;4;7;5;0;6;8] => [0;1;2;3;4;5;6;7;8] -> 250ms

[6;3;1;4;0;2;7;8;5] => [0;1;2;3;4;5;6;7;8] -> 3min

[1;4;3;6;2;8;7;5;0] => [0;1;2;3;4;5;6;7;8] -> 6.7s

XIII. distance_grille_initiale_grille_finale_taille : retourne la distance (nombre de mouvements minimums) entre deux grilles de meme taille (utilise la formule mathematique pour calculer la distance entre deux points d’un repere –voir figure- et retourne ainsi la somme (int) des distance des points entre la position dans la grille initiale et celle dans la grille finale)



XIV. liste_grilles_proches___(grilleInit,_chemin)_grille_finale_taille : meme fonctionnement/traitement que liste_grilles_chemins mais retourne une liste de

(grille, chemin, distance) (distance represente la distance de chaque grille de la grille finale, chemin represente le chemin de chaque grille depuis la grille initale)

XV. sort l : trie une liste de triplés (grille, chemin, distance) (int list * Direction list * int) selon le champs distance

XVI. grille_parcourue grille liste_grilles : Verifie si une grille a ete parcourue (presentes dans une list de (grille, chemin, distance)) et ainsi ne prends pas en compte les champs chemin ni distance)

XVII. parcours3_grille_finale_taille_prochaines_grilles_grilles_parcourues : Trouve et retourne le chemin entre une grille initiale et une grille finale en utilisant un traitement "intelligent" (choix des prochaines grilles en fonction de la distance de la grille finale et verification du precedent parcours d'une grille)

XVIII. resoudre grille_init grille_finale taille parcours : Fonction qui ne fait qu'appeler les differents parcours en fonction du parcours choisi en donnant les bon parametres pour prochaines_grilles et grilles_parcourues

Compilation et execution:

Le projet est divisé en 3 "modules"

- Base.ml qui contient les fonctions de la premiere partie (manipulation de grille)
- Melange.ml qui contient les fonctions de la seconde partie (generation de grilles, test de chemins)
- Resolution.ml qui contient les fonctions de la derniere partie (recherche de chemins)

Ce qui rend la compilation ocaml "classique" (ocamlc ...) sans modification du code impossible. Donc l'execution se passe en toplevel (cat resolution.ml | ocaml)