**Algorithmique Avancée – Solo Noble**

DEMANGE Alessi – NICOL Benoît

**1 – Explication Générale du code**

**Classe Grille :**

Attribut Grille :

* nbLignes, nbColonnes : entier | Ces attributs correspondent à la taille de la grille en fonction de ses lignes et de ses colonnes.
* grille : String[][] | Cet attribut correspond à la grille en question celle ou va évoluer les billes

Constructeurs Grille :

* Avec paramètres File : Le constructeur prend un objet de type « File » en paramètre qui correspond au tablier que l’on veut résoudre. Il lit le fichier et il convertit les charactères en String dans un tableau à double dimension. De plus, il initialise les autres attributs.
* Sans paramètre : Même chose que pour le constructeur avec attributs mais la grille est initialisée sous la première forme (celle de base du tablier1).

Méthode calculerNbBilles :

* Permet de calculer le nombre de bille dans la grille.

Méthode deplacementValide :

* Méthode qui renvoie un booléen, si on peut déplacer une bille à des coordonnées données et à une direction donnée alors on renvoie vrai sinon on renvoie faux

Méthode deplacerBille :

* Méthode qui change de valeurs les cases lors d’un déplacement d’une bille

Méthode retourArriere :

* Méthode qui fait la même chose que deplacerBille mais en arrière (lors du BackTracking)

+ Getteur + Setteur

**Classe soloNoble :**

Attribut soloNoble :

* tablier : Grille | Cet attribut correspond à l’objet Grille qui va être rempli car un fichier dans le constructeur.
* solutions : TreeMap<Integer, String[][] | Cet attribut correspond au stockage des solutions pour ensuite afficher le tout.
* nombreDeplacement : entier | Cet attribut correspond au nombre de déplacement total lors d’une résolution d’un tablier
* nombreAppelsResoudreSoloNoble : entier | Cet attribut correspond au nombre de fois où la méthode resoudreSoloNoble a été appelé.

Méthode resoudreSoloNoble :

* Méthode principale qui utilise la méthode déplacer bille. Elle permet, en utilisant le BackTracking, de trouver la solution parfaite pour résoudre le tablier qui est inscrit.

Méthode ecrireToutesSolution :

* Méthode qui affiche toutes les solutions les unes après les autres en visant dans la TreeMap solutions.

Méthode Main :

* Méthode qui nous permet d’exécuter le programme principal.

+ Guetteur

**2 – Rapport sur le projet**

Au commencement du projet soloNoble, il a fallu préparer un algorithme papier à l’aide du BackTracking pour pouvoir résoudre un tablier donné. Nous avons fait, tous les deux, un algorithme différent pour pouvoir mettre en commun notre vision d’un algorithme permettant de résoudre le problème. Ce choix nous a beaucoup aidé car nous avons obtenu beaucoup d’éléments pour commencer notre algorithme java.

La programmation java est devenu plus compliquée que l’algorithme papier. Notre plus grand problème était l’affichage des grilles lors qu’une erreur et d’un retour en BackTracking. Nous avons stocké nos solutions dans une variable de type TreeMap pour pouvoir afficher nos solutions les unes après les autres et enfin d’aller en arrière lors d’une erreur. Après avoir réglé le problème, un autre arriva un plus tard. Certains mouvements de billes n’étaient en joie de se produire. Nous avons donc cherché à faire le moins de mouvements possibles afin d’avoir un temps d’exécution assez rapide (pour optimiser le code).

**Test de rapidité d’exécution :**

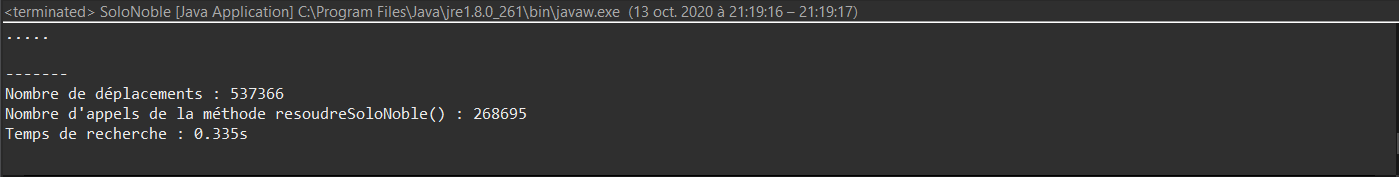
Pour le tablier1.txt, on a testé de nombreuses combinaisons de déplacements et seule une poignée nous permet d’obtenir une solution (en moins d’une seconde), le reste ne donnant pas de solutions en plus d’une minute... Cependant nous ne savons pas si les autres combinaisons ne permettent pas de trouver de solutions (ce qui serait étrange) ou sont seulement extrêmement inefficaces. Par ailleurs, nous avons remarqué que pour les combinaisons acceptables, la dernière bille finit toujours au milieu de la grille !

La meilleure combinaison que nous ayons trouvée pour tablier1.txt est « 1.haut 2.gauche 3.droite 4.bas » :

Une image contenant texte

Description générée automatiquement

Pour le tablier2.txt, nous avons testé certaines combinaisons et à priori toutes les combinaisons sont fonctionnelles. Cependant, il s’est avéré que certaines combinaisons étaient bien plus efficaces que d’autres. Par exemple, avec la meilleure combinaison pour le tablier1.txt que nous ayons trouvée (« 1.haut 2.gauche 3.droite 4.bas » ), le temps pour trouver une solution est de 9 secondes. Cependant nous avons remarqué que la combinaison (« 1.droite 2.bas 3.haut 4.gauche » ) était plus efficace que d’autres :



On remarque que sa durée de recherche est inférieure à celle pour le tablier1.txt mais elle effectue plus de déplacements… Peut-être y en a-t-il une meilleure ?

Enfin, pour le tablier3.txt, nous avons testé quelques combinaisons seulement et elles fonctionnent à priori toutes et sont également très rapides (moins de 0,1 seconde). La meilleure que nous ayons trouvée est la suivante :

