Rapport Mini Projet C++
HashCode 2018

Hugo MORALI - Damien CHEVALERIAS - Tiphaine BESNARD
Damien FARCE - Alexandre NONNON

Répartition du travail :
Damien FARCE - Alexandre NONNON : Gestion de l'arbitre
Hugo MORALI - Damien CHEVALERIAS - Tiphaine BESNARD : Gestion de la solution

Gestion de la partie "Arbitre":

L'objectif de l'arbitre est de comparer les différents algorithmes produit tout au long du projet. Pour ce faire, l'arbitre appellera chaque solveur individuellement pour chacune des cartes. A chaque exécution d'un solveur, l'arbitre rapportera le temps d'exécution ainsi que le score qu'a obtenu le solveur sur cette carte dans un fichier csv.

Le développement de l'arbitre s'est fait en 2 étapes, la première consistant à calculer le score d'une solution, la seconde consistant à créer le fichier de sortie en appelant depuis le programme principal, les exécutables qui se trouvent dans le dossier.

Pour le calcul du score, la stratégie est de commencer par créer une matrice représentant la carte. Chaque bâtiment du fichier solution est alors ajouté à la matrice tout en vérifiant qu'il respecte bien toutes les contraintes.
Cependant comme le fichier de sortie ne contient que des lignes de la forme :

IDBatiment Ligne Colonne

Il n'est pas possible de construire la carte sans connaître les informations surcelle-ci. Pour cela nous avons créé un parseur qui permet de lire le fichier d'entrée et de stocker les informations dans un objet. Cette objet contient les informations suivantes :

- **Taille de la carte**
- **Distance de marche (pour le calcul du score)**
- **Tableau contenant les bâtiments utilisables, chaque bâtiment contenant :**
    - **Un ensemble de coordonnées représentant le schéma du bâtiment**
    - **Une spécificité, dans le cas d'un bâtiment utilitaire c'est son type (i.e 5 10 115) pour un bâtiment résidentiel c'est un nombre négatif représentant le nombre de résidents.**

Une fois que ces informations sont récupérées, il est possible de parser le fichier solution. Pour cela il suffit d'itérer sur chaque ligne, et pour chaque ligne, prendre le bâtiment

dans la liste de bâtiments utilisables qui correspond à son ID (IDBatiment == place dans la liste).

Une fois que nous avons le bâtiment ainsi que les coordonnées de son futur placement il suffit de parcourir son schéma, ajouter pour chaque coordonnée du schéma la position souhaitée du bâtiment et de changer la valeur dans la matrice en vérifiant quelle n'est pas différente de 0 (0 == vide). La valeur qui est donnée aux cases correspond à la spécificité du bâtiment (soit pour un bâtiment résidentiels -100 par exemple et 2 ou 42 pour un bâtiment utilitaire).

Chaque bâtiment d'habitation est également gardé dans une liste pour éviter de devoir parcourir la matrice pour le calcul du score. Pour chaque bâtiment d'habitation dans la liste, nous générons les coordonnées qu'il occupe en ajoutant sa position avec le schéma du bâtiment. Pour chaque case occupée, nous générons un ensemble de coordonnées en utilisant un hashset, qui contient toutes les coordonnées atteignable en utilisant la walking distance. Enfin une union est faite sur tous les ensembles pour obtenir, à la fin, un ensemble contenant toutes les coordonnées atteignables depuis l'habitation. Cela permet de s'assurer que l'on ne teste pas plusieurs fois la même case atteignable plusieurs fois par le même bâtiment. Une fois que cet ensemble est obtenu, il suffit d'itérer dessus et pour chaque coordonnées vérifier si c'est un bâtiment utilitaire ou résidentiel (dans notre cas cela consiste à vérifier si l'élément de la matrice est positif ou négatif), si le nombre est positif et que ce n'est pas un nombre que nous avons rencontré pour cette habitation nous incrémentons le compteur de score. Il suffit alors de multiplier ce compteur de score par le nombre d'habitant dans ce bâtiment.

La seconde partie du développement de l'arbitre consiste à appeler des programmes externes pour mesurer le temps d'exécution ainsi que leu score pour toutes les cartes proposées, pour finalement renseigner ces données dans un tableau comparatif. Pour appeler des programmes externes, il suffit d'utiliser la fonction system avec la commande à exécuter. Il n'est donc pas compliqué de formater une chaine de caractère pour qu'elle appelle tous les exécutable d'un dossier.

La difficulté est d'obtenir le nom des exécutables dans le dossier. Pour cela nous avons utilisé la librairie filesystem qui est disponible depuis le C++17 et qui a l'avantage d'être cross-platform. Pour mesurer le temps d'exécution, il suffit d'obtenir, avec la librairie time, l'heure avant le lancement du programme et d'y soustraire l'heure à la fin de l'exécution. La création du tableau comparatif est relativement aisée grâce au format csv, il suffit de délimiter chaque champs par une virgule.

Gestion de la partie “Solution” :

Pour ce qui est de la gestion de la solution, nous avons scindé cette partie en plusieurs sous-parties :

- Création des classes pour la solution
- Tri des Bâtiments Résidentiels
- Tri des Bâtiments Utilitaires
- Placement d’un Bâtiment
- Placement des Bâtiments Résidentiels
- Placement des Bâtiments Utilitaire
- Ecriture du fichier de sortie
- Ecriture du fichier de visualisation

La partie “Création des classes pour la solution” a été essentiellement gérée par Hugo MORALI, Damien FARCE et Alexandre NONNON. La stratégie a été de créer les classes :

- InputReader
- Carte
- Bâtiment
- Bâtiment Placé
- Solution

La classe InputReader, permet de lire le fichier d’entrée afin d’y récupérer les différentes informations, tels que la hauteur de la carte, la longueur de la carte, la distance de marche, et le nombre de bâtiments présents. Ces différents informations sont par la suite utiliser afin de créer notre objet Carte.

La classe Carte permet d’interpréter les différentes informations récupérées par la classe InputReader afin de représenter au mieux celle-ci dans notre IDE à l’aide d’une liste de Bâtiments, constituée de tous les Bâtiments disponibles via le fichier d’entrée, une liste de Bâtiments Placés, constituée de tous les Bâtiments Placés sur la carte ou encore une matrice représentant le schéma des Bâtiments Placés de celle-ci afin d’avoir un retour visuel de nos placements. C’est aussi cette classe qui utilise la majorité de nos fonctions permettant de créer le fichier de sortie tel que les fonctions de placement, les fonctions de tri ou encore les fonctions de génération de sortie.

La classe Bâtiment permet de représenter un bâtiment se trouvant sur le fichier d’entrée en un objet utilisable dans notre code. Cet objet est composé d’une hauteur, d’une largeur, d’une spécificité ainsi que d’une ligne (représentant la ligne à laquelle il se trouve dan le fichier d’entrée). La spécificité représentant deux données différentes dans l’énoncé du Google HashCode 2018, la capacité pour un Bâtiment Résidentiel, et le type pour un Bâtiment Utilitaire, nous avons décidé qu’un Bâtiment avec une spécificité négative serait un Bâtiment Résidentiel, tandis qu’un Bâtiment avec une spécificité positive serait un Bâtiment Utilitaire.

La classe Bâtiment Placé permet de représenter un Bâtiment qui a été placé à des coordonnées spécifiques sur une Carte. Cet objet est composé d'un Bâtiment et d'une paire de coordonnées.

La classe Solution fait office de main dans notre architecture. C'est elle qui associe les différentes données lues grâce à la classe InputReader à la classe Carte, ou bien celle qui appelle les différents fonctions de tris, de placements et de génération de sortie, présentes ci-dessous.

La partie "Tri des Bâtiments Résidentiels" a été essentiellement gérée par Tiphaine BESNARD. Les bâtiments résidentiels étant nombreux, nous avons décidé de les trier pour faire ressortir les plus utiles. Avant de trier, un calcul de coefficients d'utilité de chaque bâtiment résidentiel a été effectué selon la formule : "nombre de case prise/(largeur*hauteur) * nombre utilitaire " dans la fonction calculCoeff(). Une fois la liste des Coefficients Résidentiels remplie par la paire <Bâtiment,coefficient> pour chaque bâtiment résidentiel, nous l'avons triée en ordre croissant pour obtenir le bâtiment Résidentiel le plus rentable. Ainsi, la liste des bâtiments Utilitaires est plus facile à utiliser et peut être appelée dans la partie "Placement des Bâtiments Résidentiels".

La partie "Tri des Bâtiments Utilitaires" a été essentiellement gérée par Tiphaine BESNARD. Ce tri n'a pas été réalisé de la même façon que le précédent. Tout d'abord, la fonction calculCoeff() a permis de séparer les bâtiments résidentiels des utilitaires, ce qui a remplie la liste des Bâtiments Utilitaires. Nous avons ensuite utilisé la fonction de tri des bâtiments utilitaires afin d'obtenir une liste de vector<vector< bâtiments>>. Celle-ci regroupe dans une case du premier vector tous les bâtiments d'une même spécificité et cette case est elle-même triée pour que les bâtiments soient triés par largeur. Ainsi, la liste des bâtiments Utilitaires est plus facile à utiliser et peut être appelée dans la partie "Placement des Bâtiments Utilitaires".

La partie "Placement d'un Bâtiment" a été essentiellement gérée par Tiphaine BESNARD et Hugo MORALI. La stratégie a été de de tout d'abord vérifier que l'on ne souhaite pas placer un Bâtiment en dehors des limites de la carte (avec des coordonnées comprises entre 0 et la taille maximale de la carte). Suite à cela, nous vérifions que chaque brique d'un Bâtiment (étant équivalent au "#" dans le fichier d'entrée), n'allait pas être posée sur un autre Bâtiment, ou bien en dehors des limites de la carte. Lorsque cette condition est validée, le placement dudit Bâtiment est alors effectué sur la matrice représentant la Carte. Cette fonction peut alors être appelée pour les parties "Placement des Bâtiments Résidentiels" et "Placement des Bâtiments Utilitaires" afin de placer les différents Bâtiments.

La partie "Placement des Bâtiments Résidentiels" a été essentiellement gérée par Hugo MORALI. La stratégie a été de placer sur la carte le Bâtiment Résidentiel ayant le plus gros coefficient de rentabilité (calculé auparavant grâce aux fonctions de la partie "Tri des Bâtiments Résidentiels"), et de se décaler en largeur d'une distance équivalente à la distance de marche plus la largeur du Bâtiment posé, jusqu'à arriver au bord est de la carte, pour revenir au bord ouest de celle-ci, se décaler en hauteur d'une distance équivalente à la

distance de marche, plus la hauteur du Bâtiment posé, jusqu'à arriver au bord sud de la carte.

Ainsi, nous remplissons nos cartes du Bâtiments Résidentiels disponible le plus "rentable". Suite à cela, nous plaçons les Bâtiments Utilitaires grâce aux fonctions de la partie "Placement des Bâtiments Utilitaire"

La partie "Placement des Bâtiments Utilitaires" a été essentiellement gérée par Damien CHEVALERIAS.

Le placement des utilitaires se fait par complétion de la carte après le placement des bâtiments résidentiels. La carte est parcouru point par point à la recherche de case vide. Une fois une casse vide trouvée, on essaie de placer un bâtiment utilitaire. On test avec tous les bâtiments jusqu'à qu'un bâtiment soit placé ou quand on a essayé avec tous les bâtiments. Les bâtiments utilitaires sont triés par type et par largeur. Chaques bâtiments d'un même type sont essayés, du plus large au moins large, puis on passe au type suivant si aucun n'a été placé. On continue le parcours de la carte à la recherche de case vide. Si on a placé un bâtiment utilitaire, les prochains tests pour placer le bâtiment suivant commenceront avec un type différent de celui placé précédemment.

La stratégie de cet algorithme était de placer le plus possible de bâtiments utilitaires entre deux bâtiments résidentiels (placés sous forme de damiers) mais assez grand pour que les bâtiments utilitaires touchent les bâtiments résidentiels les entourant.

On pourrait améliorer l'algorithme en analysant les bâtiments résidentiels qui seraient affectés lors d'un placement d'un nouveau bâtiment résidentiel. Si ces bâtiments résidentiels ont déjà un bâtiment utilitaire du même type assez proche d'eux, alors on essaie le placement avec un type différent. Si aucun des autres types ne peut être placés, on peut placer un nouveau bâtiment résidentiel à la place.