Projet: Race
Algorithmique et structures de données
Licence 2 Informatique

Julien Bernard

Table des matières

1 Spécifications

Le but de ce projet est d'implémenter un programme qui joue automatiquement au jeu Race. «Automatiquement» signifie que l'utilisateur n'a aucune interaction avec le programme, le programme doit jouer seul, avec les informations dont il dispose.

1.1 Règles du jeu

Le jeu se déroule sur une grille carrée de $\ell \times \ell$ cases. Les lignes sont numérotées de haut en bas de 0 à $\ell - 1$. Les colonnes sont numérotées de gauche à droite de 0 à $\ell - 1$. La figure 1 montre la numérotation des cases de la grille. Chaque case de la grille possède une valeur de bonus/malus qui est représenté par un entier relatif.

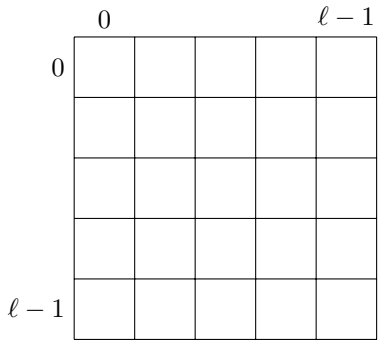

Figure 1 – La grille de jeu

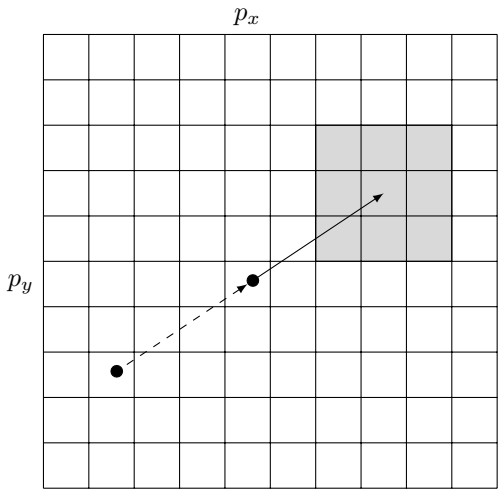

FIGURE 2 – Un déplacement sur la grille avec $(v_x, v_y) = (3, -2)$

Le joueur est placé dans une position initiale (p_x, p_y) et dispose d'un vecteur vitesse (v_x, v_y) nul au départ, et d'un score égal à 0. À chaque tour de jeu, il peut se déplacer d'un vecteur (v'_x, v'_y) de telle sorte que $|v_x - v'_x| \leq 1$ et $|v_y - v'_y| \leq 1$. Dit autrement, le joueur peut choisir d'aller sur la case $(p_x + v_x, p_y + v_y)$ ou une de ses cases adjacentes. Le vecteur vitesse du joueur est alors mis à jour à (v'_x, v'_y). La nouvelle position du joueur doit toujours se situer sur la grille de jeu sinon il est éliminé.

La figure 2 montre un déplacement. La flèche en pointillé montre le déplacement précédent permettant d'arriver à la case (p_x, p_y). Par conséquent, le vecteur vitesse actuel est $(v_x, v_y) = (3, -2)$. Pour le déplacement courant, le joueur peut décider d'aller sur une des cases grisées.

Au départ, le joueur reçoit un objectif qui est un rectangle donné par la position de son coin haut gauche (x, y) et sa taille (w, h). Il doit par déplacements successifs arriver sur une des cases de l'objectif. La valeur de la case atteinte est ajoutée au score du joueur. À ce moment, il reçoit un nouvel objectif qu'il doit atteindre de la même manière. Et ainsi de suite jusqu'à l'objectif final. Le nombre d'objectifs est inconnu au départ.

Le but est de minimiser la somme entre le nombre de déplacements du joueur pour atteindre le dernier objectif et le score du joueur sur l'ensemble du parcours. Si le joueur ne respecte pas les règles du jeu, il est immédiatement éliminé du jeu et il n'a aucun score.

1.2 Protocole

Votre programme échange les données avec le serveur via son entrée et sa sortie standard, sous forme textuelle, ligne par ligne.

1. Au début du jeu :
   (a) Le serveur envoie la taille ℓ de la grille
   (b) Le serveur envoie ℓ^2 valeurs de la grille, en parcourant la grille par colonne d'abord de gauche à droite puis par ligne de haut en bas.

(c) Le serveur envoie la position initiale du joueur : p_x puis p_y

(d) Le serveur envoie le premier objectif : x puis y puis w puis h

2. Puis à chaque tour de jeu :

(a) Le joueur envoie sa nouvelle position x puis y

(b) Le serveur envoie une chaîne d'un caractère pour indiquer le statut du déplacement :
- `OK` : le déplacement est correct
- `ERROR` : le déplacement est erroné, le jeu s'arrête immédiatement pour le joueur
- `FINISH` : le déplacement est correct et le joueur est arrivé à la fin du parcours
- `CHECKPOINT` : le déplacement est correct et le joueur est arrivé sur l'objectif, il reçoit alors le nouvel objectif : x puis y puis w puis h

Le squelette suivant est un exemple d'interaction avec le serveur.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define BUFSIZE 256

int main() {
  setbuf(stdout, NULL);
  char buf[BUFSIZE];

  // get the size
  fgets(buf, BUFSIZE, stdin);
  int size = atoi(buf);

  // get the grid values
  for (int i = 0; i < size * size; ++i) {
    fgets(buf, BUFSIZE, stdin);
    int value = atoi(buf);

    // TODO: do something with value
  }

  // get the position
  fgets(buf, BUFSIZE, stdin);
  int px = atoi(buf);
  fgets(buf, BUFSIZE, stdin);
  int py = atoi(buf);

  // get the objective
  fgets(buf, BUFSIZE, stdin);
  int x = atoi(buf);
  fgets(buf, BUFSIZE, stdin);
  int y = atoi(buf);
  fgets(buf, BUFSIZE, stdin);
```

```
  int w = atoi(buf);
  fgets(buf, BUFSIZE, stdin);
  int h = atoi(buf);

  int vx = 0;
  int vy = 0;

  for (;;) {
    // compute new px and new py

    // TODO

    printf("%i\n%i\n", px, py);

    // get the response
    fgets(buf, BUFSIZE, stdin);

    if (strcmp(buf, "ERROR\n") == 0) {
      break;
    }

    // TODO
  }

  return 0;
}
```

2 Implémentation

Vous êtes totalement libre pour l'implémentation du programme (choix des structures, choix des fonctions, etc). Cependant, voici quelques conseils qui peuvent vous être utiles.

Vous aurez besoin d'un tableau à deux dimensions dont vous ne connaissez pas les dimensions à la compilation. La meilleure façon de faire dans ce cas là est d'utiliser un tableau à une dimension de taille $\ell * \ell$ et de faire comme si toutes les lignes du tableau à deux dimensions avaient été mises bout à bout. Pour des coordonnées (x, y), l'indice du tableau à une dimension correspondant est alors : $x + y * \ell$.

Concernant la stratégie, votre programme doit savoir jouer, ce qui signifie qu'il doit a minima respecter les règles. Des parcours simples vous seront proposés pour permettre de vous évaluer. Une difficulté est de bien calculer les accélérations et freinages. N'oubliez pas la formule suivante qui peut vous aider.

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

Tout ce qui est écrit sur l'erreur standard (`stderr`) n'est pas pris en compte par le serveur et se retrouve dans un fichier de log intitulé `error.log` créé par le serveur. Pour trouver les bugs de votre programme, vous pouvez donc utiliser l'erreur standard pour y écrire des traces que vous pourrez consulter par la suite.

Considérations générales et évaluation

Vous serez évalués sur le choix des structures et des algorithmes utilisés, en particulier leur complexité. Il vous est demandé de commenter votre code *abondamment* de manière à expliquer vos choix.

Un tournoi aura lieu permettant de tester l'efficacité de vos algorithmes. Les trois premiers du tournoi recevront un point bonus sur leur projet. La date et les modalités exactes du tournoi seront précisées ultérieurement.

Le projet est à faire en binôme. Le résultat, sous forme d'une archive `tar.gz` contenant l'ensemble de vos sources et un fichier contenant le nom des deux binômes, est à rendre sur MOODLE avant la date indiquée (aucun retard autorisé).

La note du projet tiendra compte des points suivants (liste non-exhaustive) :

— la pertinence des structures de données utilisées ;
— la complexité optimale des algorithmes proposés ;
— l'absence de fuites mémoire ;
— les commentaires dans le code source ;
— la propreté du code.