SUJET SAE 2.02 : Exploration algorithmique d'un problème

Introduction

Afin d'évaluer la durée de vie d'un jeu vidéo du type jeu de rôle (RPG) en monde ouvert, on a réalisé un modèle simplifié du jeu. Le jeu simplifié se présente comme une carte rectangulaire en deux dimensions sur lesquels les lieux sont repérés par deux coordonnées (x,y).

Fonctionnement

Déplacements

Le point (0,0) se trouve en bas à gauche de la carte c'est-à-dire au sud-ouest et **c'est le point de départ du personnage**.

On suppose que les déplacements sur la carte se font uniquement horizontalement et verticalement. Chaque déplacement d'une case prend **exactement une unité de temps**. On peut imaginer qu'une unité de temps prend entre 2 et 5 minutes à un joueur réel mais c'est sans incidence sur le projet. On supposera toujours qu'on se déplace d'un lieu à un autre par un plus court chemin dans notre modélisation. Par exemple, se déplacer de (4,1) à (2,5) prend toujours exactement 2 + 3 = 5 unités de temps.

Quêtes

Le principe du jeu est de faire évoluer un personnage dans le monde et de réaliser des quêtes afin de le faire monter en expérience, puis de réaliser la quête finale quand elle est disponible et que le personnage a assez d'expérience. Les **points d'expérience** commencent à 0 et sont accumulés au fur et à mesure que les quêtes sont accomplies.

Il y a schématiquement 3 types de quêtes dans le véritable jeu :

- les explorations, qui consistent à se rendre sur un lieu et le parcourir
- les combats, qui consistent à vaincre un ou des adversaires majeurs
- les dialogues, qui consistent à parlementer avec un ou des personnages.

Du point de vue de la modélisation, ces trois types de quêtes sont équivalents et sont regroupés sous l'intitulé "quête". On donnera plusieurs **scénarios** qui sont des listes de quêtes correspondant à plusieurs versions du jeu.

Les fichiers scenario_i.txt donnent une liste des quêtes, une par ligne, sous la forme suivante:

```
56|(13,7)|((25,),(12,45))|3|200|Combattre l'Aigle Démoniaque
```

Les informations, séparées par des pipes "|", sont dans l'ordre :

- **numéro de quête** (entier) : 56. Le numéro de la quête est 56, ceci permet de l'identifier facilement.

- **position de la quête** (tuple de deux entiers) : (13,7). La quête se trouve aux coordonnées (13,7) sur la carte.
- **précondition** (tuple de un ou deux tuples) : ((25,),(12,45)). Cette quête devient disponible quand la quête 25, ainsi qu'une des deux quêtes 12 ou 45, ont été accomplies.
- **durée** (entier) : 3. Accomplir cette quête prend 3 unités de temps.
- **expérience** (entier) : 200. La quête rapporte 200 points d'expérience (attention, pour la quête 0 l'interprétation de l'expérience est différente, voir plus loin).
- **intitulé** (string) : Combattre l'Aigle Démoniaque. La quête consiste à combattre un piaf venu des enfers. L'intitulé est là pour le fun et n'a aucune incidence sur l'algorithmique.

Préconditions

La précondition, qui est une condition à satisfaire pour qu'une quête soit disponible (condition à satisfaire avant d'où le "pré"), est sous une forme conjonctive. Elle donne une liste de conditions qui doivent toutes être validées/ effectuées, ces conditions étant elles-mêmes des disjonctions (de la forme 'ou'). Voici quelques exemples

- ((2,3),(5,7)) : avoir validé la quête 2 OU 3, ET avoir validé la quête 5 OU 7
- ((3,4),) : avoir validé la quête 3 OU la quête 4
- ((2,),(1,5)) : avoir validé la quête 2 ainsi que la 1 OU la 5
- ((3,),) : avoir validé la quête 3
- () : aucune condition à valider

En pratique, il y a 0, 1 ou 2 disjonctions elles-mêmes de taille 1 ou 2. Si la précondition est vide, égale à (), cela signifie que la quête peut être effectuée n'importe quand. Ces conditions permettent de modéliser le fait de débloquer les quêtes au fur et à mesure et de les enchaîner, et le 'ou' permet de modéliser le fait que certaines quêtes nécessitent d'obtenir un objet ou un autre afin d'être de puissance suffisante, ou par exemple que plusieurs personnages différents peuvent révéler l'existence de ce lieu.

Quête finale

Finalement, la dernière ligne du fichier de quêtes a une interprétation légèrement différente. Son numéro est 0 qui indique qu'il s'agit de la quête finale (boss de fin). La différence principale est que l'expérience indiquée n'est pas l'expérience que rapporte le fait de vaincre le boss, mais l'expérience nécessaire pour pouvoir le faire. *Afin d'effectuer la quête finale, il faut donc à la fois valider sa précondition, et avoir l'expérience nécessaire indiquée.*

Le but du jeu est donc d'accumuler de l'expérience en réalisant des quêtes, puis quand on a débloqué la quête finale et qu'on a l'expérience nécessaire pour la réaliser, de le faire.

Solutions

Une solution du jeu est une suite de quêtes de la forme

```
25 56 45 125 56 ... 0
```

qui indique un ordre dans lequel les quêtes peuvent être accomplies. On commence obligatoirement par une quête sans précondition et on termine obligatoirement par la quête 0, et pour chaque quête de la liste,

sa précondition doit être satisfaite par les quêtes précédentes. Toutes les quêtes n'ont pas besoin d'être dans la liste, il suffit juste d'avoir assez d'expérience et la précondition finale pour entreprendre la quête 0.

Une solution pourra être affichée sous forme d'une telle suite. Il sera également envisageable de l'afficher proprement sous la forme suivante (par exemple) :

- se déplacer en (3,4)
- vaincre le zombie des marais
- se déplacer en (2,8)
- parler avec le troll bavard
- ...

de sorte à avoir une liste des actions à effectuer. On pourra également envisager une représentation graphique.

Durée

La **durée** d'une solution est la somme des durées des quêtes qui la composent *ainsi que la somme des durées de tous les déplacements depuis le point de départ et de quête en quête jusqu'à la quête finale*. Cette durée sert à évaluer la durée de vie du jeu quand il sera commercialisé, que ce soient des runspeeds (durée minimale) jusqu'aux joueurs les plus lents ou ceux qui veulent faire le jeu à 100%.

Solutions efficaces et exhaustives

On va distinguer deux types de solutions :

- les solutions qui réalisent la quête finale 0 dès que c'est possible, que l'on appelera **solutions efficaces**
- les solutions qui réalisent toutes les quêtes avant de faire la quête finale, que l'on appelera **solutions exhaustives**

On ne considèrera pas d'autres types de solutions (qui feraient plus de quêtes que nécessaires mais pas tout...). On ne considère pas non plus de solutions qui feraient des déplacements inutiles, on va toujours de quête en quête par un plus court chemin.

Objectifs et niveaux de réalisation

Voici différents niveaux de réalisation du point de vue algorithmique pour votre programme. Faites ce que vous pouvez dans chaque niveau.

Niveau 1

- Donner **une** *solution efficace* qui soit "gloutonne", c'est-à-dire qui va toujours réaliser une des quêtes disponibles **les plus proches** à chaque moment. Ce genre de solutions correspond au comportement d'un joueur moyen qui optimise un peu mais pas trop.
- Donner **une** *solution exhaustive* utilisant aussi la tactique gloutonne. Ce genre de solution correspond au comportement d'un joueur qui veut finir le jeu à 100%.

Niveau 2

- Donner une solution efficace optimale en termes de durée. Ce genre de solution correspond à un speedrun.
- Vous pouvez également proposer pour aller plus loin:
    - une solution efficace optimale exhaustive en termes de nombre de quêtes et non de durée
    - une solution efficace optimale en termes de déplacement total
    - même chose que les cas précédentes mais avec une solution exhaustive optimale en termes de durée, nombre de quêtes ou de déplacements
    - étendre les cas précédents traités en proposant les N meilleures solutions au lieu de la première
    - étendre les cas précédents en proposant les pires solution plutôt que les meilleures (on suppose toujours quand même que les déplacements se font en plus court chemin)

Niveau 3

Si vous avez traité tous les cas précédents, vous pouvez écrire votre programme de façon unifiée. On sélectionne :

- le nombre de solutions souhaitées (N, qui par défaut vaut 1)
- si on veut des solutions efficaces ou exhaustives
- la fonction objectif étudiée (durée / nombre de quêtes / déplacements)
- si on veut le meilleur / le pire

et le programme affiche la ou les réponses et permet de les visualiser.

On peut avoir d'autres idées encore, par exemple afficher une solution qui minimise le nombre de points d'expériences (on a obtenu le nombre requis mais si possible pas plus !), ou afficher la durée moyenne sur toutes les solutions, etc. Vous aurez peut-être d'autres idées qui seront les bienvenues.

Exigences de qualité

Le code doit être aussi simple et clair que possible. Par exemple, il sera apprécié d'unifier les différents calculs dans un même endroit plutôt que d'avoir des redondances. Votre code doit bien entendu être commenté, avoir des noms de classes/méthodes/variables intelligemment choisis et des spécifications détaillées.

Vous devez pouvoir expliquer clairement vos algorithmes ainsi que leurs optimisations.

Exemple de scénario

Le scénario 0 est de petite taille et nous allons le résoudre ci-dessus "à la main". Le scénario contient les quêtes suivantes :

```
1|(4, 3)|()|2|100|explorer pic de Bhanborim
2|(3, 1)|((1,),)|1|150|dialoguer avec Kaela la chaman des esprits
3|(0, 4)|((2,),)|3|200|explorer palais de Ahehona
4|(3, 2)|((2,),)|6|100|vaincre Loup Géant
0|(1,1)|((3,4),)|4|350|vaincre Araignée lunaire
```

Pour accomplir la quête finale (l'Araignée lunaire), il faut accomplir la quête 3 ou la quête 4, et accumuler 350 points d'expérience (xp).

On doit obligatoirement commencer par la quête 1. Ensuite, seule la quête 2 sera disponible. On a ensuite le choix entre les quêtes 3 et 4. Les solutions commencent donc par `1, 2, 3` ou `1, 2, 4`.

Deux solutions **exhaustives** sont donc `1, 2, 3, 4, 0` et `1, 2, 4, 3, 0`. Calculons précisément la durée de la solution 1, 2, 3, 4, 0 :

- 7 : déplacement de (0,0) à (4,3)
- +2 : quête 1 (total xp : 100)
- +3 : déplacement de (4,3) à (3,1)
- +1 : quête 2 (total xp : 250)
- +6 : déplacement de (3,1) à (0,4)
- +3 : quête 3 (total xp : 450)
- +5 : déplacement de (0,4) à (3,2)
- +6 : quête 4 (total xp : 550)
- +3 : déplacement de (3,2) à (1,1)
- +4 : quête 0

soit une durée totale de 40 unités de temps. Si l'on souhaite une solution **efficace**, on aurait pu ne pas faire la quête 4 car après 1,2,3 la condition et les xp nécessaires pour la quête 0 étaient acquis. On obtient ainsi une solution `1,2,3,0` qui a une durée de `7 + 2 + 3 + 1 + 6 + 3 + 4 + 4 = 30`.

L'autre solution non exhaustive est `1,2,4,0`, qui a une durée totale de 27. Elle est donc optimale en terme de durée.