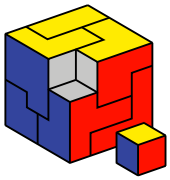
Prologin 2016



Concours national d'informatique

Épreuve écrite d'algorithmique
Paris III

Samedi 12 mars 2016

Un Jeu D'rôle

Dark Dungeons (auteur : Jack T. Chick)

1 Préambule

Bienvenue à **Prologin**. Ce sujet est l'épreuve écrite d'algorithmique et constitue la première des trois parties de votre épreuve régionale. Sa durée est de 3 heures. Par la suite, vous passerez un entretien (20 minutes) et une épreuve de programmation sur machine (4 heures).

Conseils

— Lisez bien tout le sujet avant de commencer.
— **Soignez la présentation** de votre copie.
— N'hésitez pas à poser des questions.
— Si vous avez fini en avance, relisez bien.
— N'oubliez pas de passer une bonne journée.

Remarques

— Le barème est donné à titre indicatif uniquement.
— Indiquez lisiblement vos nom et prénom, la ville où vous passez l'épreuve et la date en haut de votre copie.
— Tous les langages sont autorisés, veuillez néanmoins préciser celui que vous utilisez.
— Ce sont des humains qui lisent vos copies : laissez une marge, aérez votre code, ajoutez des commentaires (**seulement** lorsqu'ils sont nécessaires) et évitez au maximum les fautes d'orthographe.
— Le barème récompense les algorithmes les plus efficaces : écrivez des fonctions qui trouvent la solution le plus rapidement possible.
— Si vous trouvez le sujet trop simple, relisez-le, réfléchissez bien, puis dites-le-nous, nous pouvons ajouter des questions plus difficiles.

2 Sujet

Description

Par une froide soirée hivernale [1], alors que dehors souffle le blizzard [2], l'électricité est coupée, la radio et la télévision ne captent plus [3]. Vous êtes avec quelques amis assis autour d'une table, éclairée à la bougie. Un de vos amis, Gary, avide de jeux de rôles, vous propose de faire une partie. En tant qu'instigateur de la partie, Gary s'autodésigne comme maître de jeu, et obtient ainsi les pleins pouvoirs sur le devenir de votre aventure.

Principe

« Le principe du jeu de rôle sur table, explique Gary, consiste à créer un univers fictif où chacun des joueurs interprête un personnage et son rôle, le tout encadré par certaines contraintes. Que **je** définis. »

Vos amis et vous êtes donc confrontés à différents choix tout au long de la partie, vous menant aux possibilités infinies qui s'ouvrent à vous, et qui font progresser et évoluer votre groupe dans le monde du jeu.

Défaite

« Puisque la mort est inévitable, oublions-la. » — Stendhal

Dans la vie comme dans le jeu, on ne peut que perdre. C'est ainsi la mort des personnages qui marque la fin du jeu : il n'y a pas de concept de victoire, et chaque partie qui n'est pas perdue n'est pas terminée.

Ainsi, votre petit groupe d'aventuriers s'aventura dans un donjon…

Partie 1

Vous êtes finalement entrés dans ce fameux donjon. Ici, trois choix s'offrent à vous : tout droit, à droite, à gauche [4].

Le MJ, qui est omniscient et omnipotent, dispose de toutes les informations résultant de vos choix. À ce tour-ci, il vous dira si vous allez à droite qu'il y a un lapin tueur, à gauche un méchant sorcier, et tout droit un chemin menant à l'objectif de votre quête [5] sans monstre.

Votre but [6] est d'amasser le plus de pièces d'or possible, par exemple :

— Aller tout droit : 0 pièces d'or ;
— Combattre le lapin : 10 pièces d'or ;
— Combattre le méchant sorcier : 100 pièces d'or ;
— Aller en arrière : 0 pièces d'or. Et la partie se termine car vous tombez tous dans les douves.

À chaque tour de la partie, vous aurez une liste d'actions possibles. Cette liste pourra dépendre du tour et des actions précédemment entreprises. Il faudra donc en indiquer une au MJ qui vous annoncera alors, éventuellement après un lancer de dés, les conséquences de votre choix. Celles-ci seront renseignées dans une donnée de type `action`.

1. Comme maintenant.
2. WoW ! s'exclamèrent-ils à la fenêtre.
3. De toute façon, sans électricité…
4. Et en arrière.
5. Si objectif il y a.
6. Pas forcément l'objectif de votre quête, mais un but qui vous tient à coeur.

Question 1 (1 point)

> Proposez une structure de données représentant le déroulement de la partie. Elle doit permettre de connaître, pour chaque tour du jeu, l'action entreprise durant ce tour et ses effets.

Cupide comme vous êtes [7], vous disposez d'une fonction `gain_en_PO` qui prend en argument une action que vous avez effectuée (de type `action`, donc) et renvoie le nombre de pièces d'or qu'elle a rapportées.

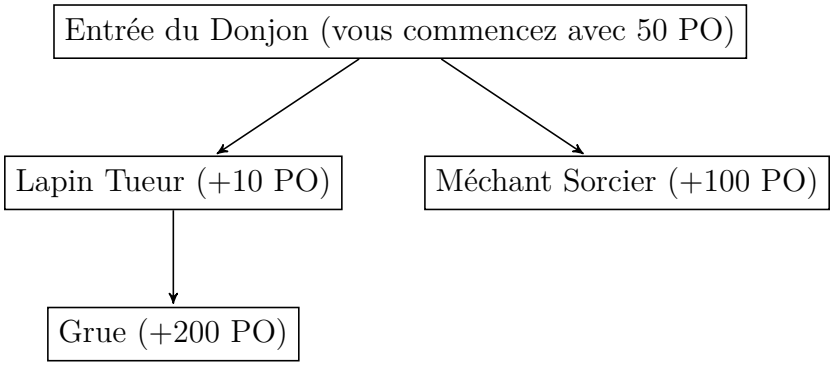
Question 2 (2 points)

> Écrivez une fonction `richesse` qui parcourt la structure de données de la question 1 et renvoie la somme d'or accumulée tout au long de la partie.

En fait, vous vous rendez compte que vous faites beaucoup trop d'erreurs, ce JDR est vraiment trop difficile.

D'ailleurs, le voleur est en train de foncer tête baissée vers un claptor de Marzrok.

Le MJ, qui est gentil, vous autorise à tester plusieurs actions parmi la liste qu'il vous propose. Ainsi, vous pouvez vous retrouver à jouer plusieurs exécutions de la partie en même temps : quand un choix non absurde s'offre à vous, vous pouvez explorer les deux branches. C'est un *branchement*.



Voici un exemple : au tour 0, vous entrez dans le donjon. Vous avez ensuite le choix entre rendre visite au lapin ou au sorcier. Au tour 1 vous décidez d'aller vous frotter au lapin, pour une maigre cagnotte de 10 pièces d'or. Au tour 2, vous regrettez de ne pas être allé voir le sorcier, qui rapporterait sans doute un plus grand butin, et décidez de revenir en arrière, ce que permet le branchement, pour faire ce choix. *Le tour 2 fait donc suite au tour 0 et non au tour 1, ce dernier se situant dans une chronologie différente.* Au tour 3, vous décidez de poursuivre sur la chronologie où vous avez vu le lapin, et vous vous faites manger par une grue dans l'obscurité. *Le tour 3 fait donc suite au tour 1 et non au tour 2.*

Pour représenter tout cela, nous allons renseigner, à chaque tour, à partir de quel point de la partie on continue l'histoire. (Au tour 2 (resp. 3), on continue la partie depuis son état au tour 0 (resp. 1).)

7. Vous avez choisi de jouer un nain.

Question 3 (2 points)

> Expliquez les modifications à apporter à votre structure de données pour coder cette information. Décrivez le contenu de cette structure dans le cas de l'exemple fourni ci-dessus.

Question 4 (4 points)

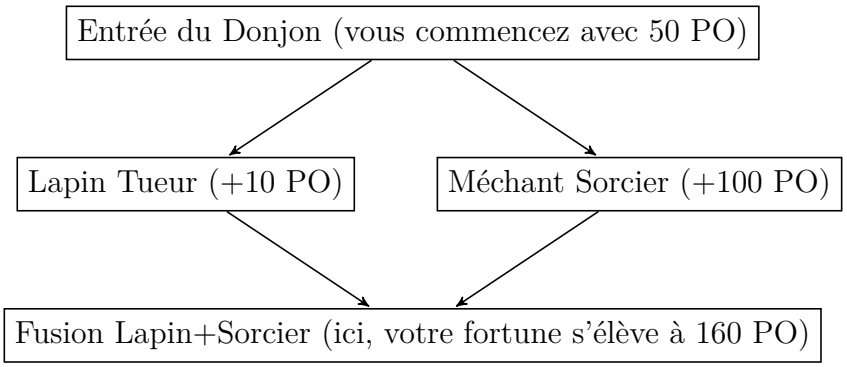
> (a) Écrivez une nouvelle fonction `richesse2` qui calcule, pour un tour donné, le nombre total de pièces d'or que vous avez accumulé jusqu'à ce point dans sa chronologie. Dans l'exemple ci-dessus, les réponses respectives pour les tours 0, 1, 2 et 3 sont 50, 60, 150 et 260 (et non 360 !).
>
> (b) Écrivez une fonction qui calcule pour chaque tour la richesse à ce tour, qui soit plus efficace qu'un appel à `richesse2` sur tous les tours joués jusqu'à présent.

Question 5 (3 points)

> Écrivez une fonction `bifurcation` qui étant donnés deux tours détermine le tour où leurs chronologies ont bifurqué, c'est-à-dire le tour le plus avancé commun aux deux chronologies.

Vous avez brillamment vaincu le lapin tueur dans une branche. Dans l'autre vous avez vaillamment occis le méchant sorcier. Mais c'est dommage d'avoir à choisir entre les deux, aussi demandez-vous au MJ s'il peut vous concocter un monde où le lapin et le sorcier ont été tous deux terrassés.

Le MJ décide que si deux chronologies ne sont pas incompatibles, c'est-à-dire qu'il est plausible que les actions des deux auraient pu avoir lieu dans la même chronologie, vous pouvez les *fusionner*, ce qui compte comme une action. Vous additionnez alors l'argent obtenu dans chacune des deux branches.



Question 6 (1 point)

> Proposez une façon de représenter les fusions dans la structure de données.

Vous vous apprêtez pour la première fois à procéder à une fusion. Les deux chronologies que vous allez fusionner sont données par leurs tours i et j.

Question 7 (3 points)

> (a) Montrez que la richesse après fusion est égale à la somme des richesses aux tours i et j, moins la richesse au tour où les chronologies de i et j ont bifurqué. (Sur l'exemple : $160 = 60 + 150 - 50$.)
>
> (b) Déduisez-en une fonction s'appuyant sur `bifurcation` pour calculer le nombre de pièces d'or que vous aurez en fusionnant i et j.

Cependant, après avoir effectué beaucoup de fusions, la technique précédente ne marche plus.

Question 8 (5 points)

> Écrivez une fonction `richesse3` qui calcule pour un tour donné le nombre de pièces d'or accumulé jusque-là en prenant compte des fusions.

Partie 2

Continuant votre exploration du donjon, vous devez faire régulièrement des pauses, pour attendre la sorcière qui sautille sur une seule jambe.

Quoi ? La sorcière a l'air d'avoir perdu une jambe, et ce dans plusieurs branches d'exploration et vous ne l'aviez pas remarqué ! Comme cela la dérange quelque peu, même si le MJ s'amuse beaucoup, vous cherchez un moyen de vite résoudre ce problème et de retrouver le moment précis où ce fâcheux accident s'est produit, afin de reprendre la partie avant.

Vous disposez d'une fonction `bon_etat` qui prend un tour et renvoie `true` si la sorcière est en bon état et `false` si elle a perdu une jambe. L'exécution de cette fonction consiste à importuner le MJ pour lui demander cette information, ce que vous souhaitez faire le moins possible pour garder la face. Le but sera donc de minimiser les appels à `bon_etat`.

Vous savez de plus que cet accident est survenu précisément une fois au cours de la partie, à un tour présent dans toutes les chronologies où la sorcière a été amputée.

Question 9 (2 points)

> Écrivez une fonction `trouver_accident` qui, à partir d'un mauvais état, trouve le tour exact où la sorcière a perdu sa jambe.
>
> On ne vous demande pas un algorithme optimal.

Cet algorithme n'est pas optimal[8]. Cela vous turlupine, le MJ s'impatiente. Vous avez la furieuse intuition qu'il serait possible d'opérer une sorte de dichotomie sur les états du jeu pour localiser le moment fatidique où la sorcière a perdu sa jambe.

Question 10 (4 points)

> Décrivez l'algorithme qui garantit le moins d'appels possibles à `bon_etat` dans le pire cas, lorsque qu'il n'y a pas de branchements ou fusions au cours du jeu.

8. Ahah, on vous a bien eu !

Question 11 (7 points)

(a) Soit un tour i pour lequel on demande l'état de la sorcière. Montrez que
— si la sorcière dispose de tous ses membres, tous les tours qui précèdent le tour i dans une chronologie peuvent être exclus de la recherche de l'accident ;
— sinon, tous les tours qui ne précèdent *pas* le tour i dans une chronologie peuvent être exclus.
Précision : les tours qui précèdent i sont exactement ceux dont le butin rapporté comptaient dans le calcul de la richesse au tour i dans la partie 1.

(b) Écrivez une fonction pour trouver la requête dont le résultat permettra, dans le pire cas, d'exclure le plus de tours possibles de la recherche. On pourra s'inspirer des fonctions de calcul de richesse.

Question 12 (3 points)

Décrivez un algorithme efficace pour retrouver l'accident avec peu d'appels à `bon_etat`, s'appuyant sur la question précédente.

Partie bonus

Attention, ne traitez cette partie que si vous avez répondu à toutes les autres questions et relu vos réponses.

Question bonus 13 (8 points)

On reprend le problème de la partie 2. Désormais, chaque appel à `bon_etat` produit une certaine irritation quantifiable auprès du MJ, variable selon les tours. Typiquement, plus c'était il y a longtemps, plus le MJ devra fouiller dans ses notes ; quoi qu'il en soit, vous savez pour chaque tour à quel point il sera irrité qu'on l'interroge dessus. Le but est de calculer la stratégie optimale pour retrouver le moment de l'accident en minimisant dans le pire cas la mauvaise humeur induite (qui est la somme des irritations de chaque requête).

(a) Traitez le cas sans branchement ni fusion.

(b) Et dans le cas général ?

Question bonus 14 (1 point)

Vous arrivez devant un vieux sage. Il a quelques questions à vous poser.
— What… is your name ?
— What… is your quest ?
— What… is your favorite colour ?

Fin

Le sujet comporte 6 pages (sans compter la page de garde) et 14 questions, parmi lesquelles 2 questions bonus. Les questions normales sont notées sur 37 points, et les questions bonus rapportent au total 9 points, plus 1 point de présentation.