



CONCOURS CENTRALE•SUPÉLEC

Sujet 04 — MPI

Jeu de Scrabble




Durée : 3h

Centrale-Supélec

Préambule

Ce sujet d'oral d'informatique est à traiter, sauf mention contraire, en respectant l'ordre du document. Votre examinatrice ou votre examinateur peut vous proposer en cours d'épreuve de traiter une autre partie, afin d'évaluer au mieux vos compétences.

Le sujet comporte plusieurs types de questions. Les questions sont différenciées par une icône au début de leur intitulé :

- les questions marquées avec  nécessitent d'écrire un programme dans le langage demandé. Le jury sera attentif à la clarté du style de programmation, à la qualité du code produit et au fait qu'il compile et s'exécute correctement ;
- les questions marquées avec  sont des questions à préparer pour présenter la réponse à l'oral lors d'un passage de l'examinatrice ou l'examinateur. Sauf indication contraire, elles ne nécessitent pas d'appeler immédiatement l'examinatrice ou l'examinateur. Une fois la réponse préparée, vous pouvez aborder les questions suivantes ;
- les questions marquées avec  sont à rédiger sur une feuille, qui sera remise au jury en fin d'épreuve.

Votre examinatrice ou votre examinateur effectuera au cours de l'épreuve des passages fréquents pour suivre votre avancement. En cas de besoin, vous pouvez signaler que vous sollicitez explicitement son passage. Cette demande sera satisfaite en tenant également compte des contraintes d'évaluation des autres candidates et candidats.

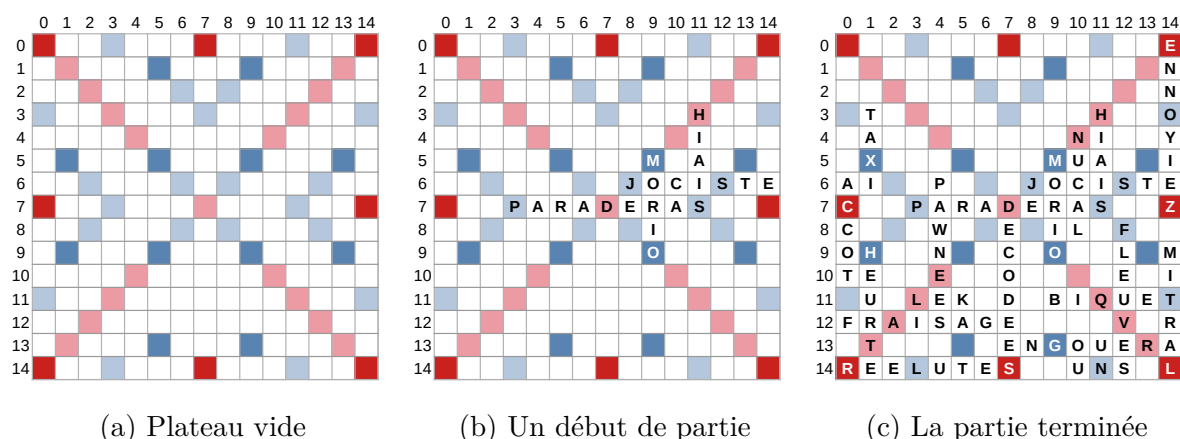


FIGURE 1 – Des plateaux de Scrabble

1 Introduction

Au jeu de Scrabble, on dispose d'un plateau qui est constitué d'une grille de cases. À chaque tour, un joueur tire au hasard une liste de lettres¹. Il doit, en utilisant toutes ces lettres ou seulement une partie, trouver un mot qui existe dans le lexique. Il doit ensuite le former sur le plateau en le plaçant à côté d'un mot déjà posé lors des coups précédents, ou bien en croisant un tel mot. Pour que le placement d'un mot soit valable, le mot lui-même et tous les autres mots adjacents, dans le sens vertical ou le sens horizontal, doivent former des mots appartenant au lexique.

Dans la figure 1b, les premiers coups ont par exemple été joués dans l'ordre suivant :

- avec son premier tirage de lettres, le joueur a placé le mot **PARADERAS** en recouvrant la case centrale,
- avec son deuxième tirage, le joueur a composé le mot **MORIO**, perpendiculairement au mot précédent en réutilisant la lettre R,
- au troisième tirage, le joueur a tiré des lettres qui lui ont permis de former le mot **HIAIS** en réutilisant le S final du mot **PARADERAS**,
- au quatrième tirage, le joueur tire des lettres qui permettent de former le mot **JOCISTE**. Le placement de ce mot est possible car, verticalement, cela crée les mots **JE** et **CA** verticaux qui sont bien dans le lexique. Si l'un d'eux n'avait pas été un mot correct, le coup n'aurait pas été permis.

Chaque mot est posé dans une direction, horizontale ou verticale.

Les lettres de l'alphabet possèdent un poids qui permet de calculer les scores. Pour calculer le score associé au placement d'un mot :

- dans les cas simples, on somme les poids des lettres qui se trouvent dans la direction du mot que l'on vient d'ajouter, y compris les lettres du mot déjà présentes sur le plateau,
- cependant, certaines cases du plateau apportent des bonus, uniquement lors du coup qui les recouvre. Ces bonus sont des coefficients multiplicateurs : certaines cases doublent ou triplent les points accordés à la seule lettre posée sur la case, d'autres cases doublent ou triplent les points accordés à tout le mot. La case centrale double

1. Si vous connaissez déjà ce jeu, dans notre sujet, il n'existera pas de lettre joker.

les points du premier mot. Quand il y a plusieurs bonus, on tient compte d'abord du bonus sur les lettres individuelles, puis on applique le plus grand bonus (double ou triple) de mot,

- dans la direction perpendiculaire au mot, on somme les poids des *nouveaux* mots verticaux (ceci se produit quand on ajoute un mot parallèlement à un autre). On compte, pour un tel mot, les bonus qui se trouvent sur ce mot.

Par exemple, le coup **JOCISTE** dans la figure 1c était particulièrement intéressant au score car la lettre **J** a compté double pour le calcul des points de **JOCISTE**, mais elle a également compté car on a créé le mot vertical **JE**. Ainsi, ce coup a rapporté :

- la somme des lettres de **JOCISTE** en doublant la valeur du **J** et du **S**, ce qui rapporte 25 points,
- la somme des lettres de **JE** en doublant le **J**, ce qui rapporte 17 points,
- la somme des lettres de **CA**, qui rapporte 4 points.

Ce coup vaut au total 46 points.

Plusieurs fichiers vous sont fournis pour traiter ce sujet :

- un fichier **alphabet.ml** contient quelques fonctions utilitaires pour manipuler l'alphabet du jeu, vous n'aurez probablement pas à modifier ce fichier,
- un fichier **gaddag.ml** contient des fonctions pour manipuler la structure de **GAD-DAG** définie en partie 3,
- un fichier **scrabble.ml** contient des fonctions qui implémentent les règles et le déroulement du jeu,
- un fichier **lexique.txt** contient la listes des mots valides,
- un fichier **scrabble_games.sqlite** contient une base de données SQL utile pour la partie 6.

On rappelle qu'un fichier source OCaml peut être compilé en code objet avec l'appel : `ocamlc -c bar.ml`, qui produit `bar.cmo`. Ceci définit un module **Bar** qui peut être utilisé dans d'autres fichiers sources. Il est également possible d'utiliser ce module depuis **utop** en lançant par exemple `utop bar.cmo` depuis un terminal.


Vous disposez d'un script `./compile` à exécuter dans un terminal depuis le répertoire qui contient vos sources. Ce script compile les trois fichiers OCaml fournis par le sujet et les lie ensemble dans un exécutable nommé `./scrabble`.

2 Gestion du plateau

Les plateaux manipulés dans tout le sujet seront des grilles carrées. Une case peut contenir une lettre, correspondant à un pion déjà posé dans un tour précédent, ou bien être vide et dans ce cas on indique le bonus de points accordé par cette case.


Les cases sont représentées par des valeurs de type **case**. Un plateau est une valeur de type **plateau**, qui est une matrice que l'on suppose carrée de cases. Les définitions de ces types sont dans le fichier **scrabble.ml**, prenez le temps de les lire et de vous familiariser avec les fonctions déjà écrites dans ce fichier.

▷ **Question 1.** ♣ En utilisant les fonctions fournies, afficher un plateau sans aucun mot, c'est-à-dire dans son état initial au début du jeu. Afficher également le plateau de la figure 1c.

▷ **Question 2.**  Écrire une fonction `lettres_necessaires_horizontal` qui prend en paramètres :

- un plateau,
- un mot à former sur le plateau,
- un couple (i, j) qui indique les coordonnées sur le plateau de la case dans laquelle la première lettre du mot doit être placée,

et qui détermine la liste des lettres qu'il faut ajouter sur le plateau pour former ce mot, c'est-à-dire la liste des lettres du mot privée des lettres déjà en place sur le plateau.


▷ **Question 3.**  Proposer une modification simple de la fonction précédente afin d'obtenir une fonction `lettres_necessaires` qui prend en paramètres :


- un plateau,
- un mot à former sur le plateau,
- un couple (i, j) qui indique les coordonnées sur le plateau de la case dans laquelle la première lettre du mot doit être placée,
- un booléen qui est vrai si le mot est placé verticalement et faux s'il est placé horizontalement,

et qui détermine la liste des lettres qu'il faut ajouter sur le plateau pour former ce mot, c'est-à-dire la liste des lettres du mot privée des lettres déjà en place sur le plateau.

Une solution qui évite de dupliquer des parts significatives de code sera appréciée.


Le but des questions suivantes est de calculer le score obtenu en plaçant un mot sur le plateau.


▷ **Question 4.**  Écrire une fonction `somme_lettres` : `string -> int` qui renvoie la somme des valeurs des lettres d'un mot.


▷ **Question 5.**  Écrire une fonction `valeur_mot_seul` qui prend en paramètres :

- un plateau,
- un mot à former sur le plateau,
- un couple (i, j) qui indique les coordonnées sur le plateau de la case dans laquelle la première lettre du mot doit être placée,
- un booléen qui est vrai si le mot est placé verticalement et faux s'il est placé horizontalement,

et qui renvoie la valeur de ce mot (et uniquement lui, sans compter les mots perpendiculaires) en le formant sur le plateau. On supposera (et votre fonction ne le vérifiera pas) que le placement du mot est possible. Dans le cas contraire, le comportement n'est pas défini.

▷ **Question 6.**  Combien de points rapporte le mot NUCAL ajouté verticalement à partir de la case en ligne 4 colonne 10 sur la figure 1b ?

▷ **Question 7.**  Expliquer votre stratégie pour calculer les points des mots perpendiculaires au mot ajouté.


▷ **Question 8.**  Écrire une fonction `valeur_mot` qui prend en paramètres :


- un plateau,

- un mot à former sur le plateau,
- un couple (i, j) qui indique les coordonnées sur le plateau de la case dans laquelle la première lettre du mot doit être placée,
- un booléen qui est vrai si le mot est placé verticalement et faux s'il est placé horizontalement,

et qui renvoie le nombre de points rapportés par ce mot. On supposera (et votre fonction ne le vérifiera pas) que le placement du mot est possible. Dans le cas contraire, le comportement n'est pas défini.

Le premier mot doit être placé de sorte qu'une de ses lettres doit être placée sur la case centrale. En cours de partie, les mots suivants doivent avoir au moins une lettre adjacente à un mot déjà posé. La case centrale et les cases adjacentes à un mot déjà posé sont appelées des *ancres*.

▷ **Question 9.**  Écrire une fonction `ancres_plateau` qui prend en paramètre un plateau et qui renvoie la liste de tous les couples (i, j) tels que la case de coordonnées (i, j) est une ancre. On notera que la case centrale n'est une ancre que si elle n'a pas encore été recouverte par une lettre.

▷ **Question 10.**  Combien d'ancres possède le plateau donné en figure 1c ?

3 Manipulation du lexique


L'ensemble des mots que l'on peut poser sur un plateau est fixé par un lexique de mots, donné dans le fichier `lexique.txt`. Ce fichier contient un mot par ligne. Afin de stocker ce lexique d'une manière efficace pour trouver les coups autorisés sur le plateau, on utilise une structure appelée *GADDAG*, dont l'implémentation est déjà fournie.

▷ **Question 11.**  Rappeler pourquoi l'ensemble des mots du lexique forme un langage régulier.

Étant donné un mot u dont les lettres sont $u_1 u_2 \cdots u_n$, on définit son *mot miroir* \bar{u} par $\bar{u} = u_n u_{n-1} \cdots u_2 u_1$.

On se donne un caractère spécial, $!$, que l'on suppose distinct de toute lettre apparaissant dans un mot du lexique. Étant donné un mot u , on définit l'ensemble des *ancres de u* comme étant l'ensemble $\{\bar{v}!w \mid vw = u \text{ et } v \neq \epsilon\}$.

Le GADDAG est un automate fini qui reconnaît l'ensemble de tous les ancres de tous les mots du lexique.

▷ **Question 12.**  Écrire dans `gaddag.ml` la fonction `construit_gaddag` qui prend en paramètre le nom de fichier contenant un lexique et qui renvoie le GADDAG associé. On pourra utiliser la fonction, déjà écrite, `ajoute_conjugues` qui prend en paramètre un mot et un GADDAG, et ajoute tous les ancres de ce mot au GADDAG.

4 Trouver un coup valide

Pour déterminer un coup valide, on doit d'abord :

- choisir une ancre,

- choisir un sens (vertical ou horizontal) de placement du mot.

On considère d'abord le cas d'un mot placé horizontalement. On tente de former sur l'ancre une lettre du tirage actuel. On note q_0 l'état initial du GADDAG et q_1 l'état obtenu après avoir lu la lettre que l'on vient de placer. Si la case à gauche de l'ancre est vide, on place une nouvelle lettre du tirage parmi les transitions du GADDAG de source q_1 . Si la case à gauche de l'ancre contient déjà une lettre, on ne place aucune lettre du tirage et on suit la transition étiquetée par cette lettre depuis q_1 . On itère ce processus jusqu'à ce que le GADDAG se trouve dans un état q_n tel qu'il existe une transition étiquetée par ! depuis q_n . On se replace alors sur le plateau à droite de l'ancre et à chaque étape :

- si la case est libre, on ajoute une lettre du tirage parmi celles autorisées dans l'état actuel par le GADDAG,
- si la case est occupée, on suit la transition correspondante dans le GADDAG.

Un mot est placé correctement si le GADDAG est dans un état final et que la case suivante à droite est vide ou bien se trouve sur le bord du plateau.

À chaque étape, il peut y avoir de nombreux choix possibles :

- plusieurs lettres peuvent être valablement placées sur la case en cours d'examen,
- lors du déplacement vers la gauche, on peut soit placer une lettre de plus soit lire une transition ! et repartir à droite de l'ancre.

De plus, il est possible que certains choix ne mènent pas à un mot valide (le GADDAG n'a aucune transition valide, le tirage est épuisé et le GADDAG n'est pas dans un état final).

▷ **Question 13.** 📖 Écrire une fonction `trouver_coup` qui prend en paramètres :

- un plateau,
- la liste des lettres du tirage,
- un GADDAG,
- les coordonnées d'une ancre,
- un booléen qui indique le sens de placement souhaité,

et qui renvoie, s'il existe, un mot pouvant être placé autour de cette ancre.

5 Jeu en mode dupliqué

En mode dupliqué, tous les joueurs possèdent le même tirage et ils doivent trouver le mot qui réalise le score maximal pour un plateau donné.

▷ **Question 14.** 🧑 Formuler l'algorithme proposé pour résoudre ce problème.

▷ **Question 15.** 📖 Écrire une fonction `meilleur_coup` qui détermine le meilleur coup pour un plateau donné. Cette fonction prend en paramètres un plateau, un tirage et un GADDAG. Elle renvoie la position d'une ancre, le mot placé, un booléen qui donne le sens de placement et le score de ce coup.

6 Statistiques des tournois

Le fichier `scrabble_games.sqlite` contient une base de données SQL qui possède une table `games` qui décrit les résultats de parties qui ont été jouées en tournois. En voici les principales colonnes :

- `gameid` est un identifiant numérique unique à chaque partie,
- `tourneyid` est l'identifiant du tournoi dans lequel la partie a eu lieu,
- `winnerid` est l'identifiant du joueur qui a gagné la partie,
- `loserid` est l'identifiant du joueur qui a perdu la partie,
- `date` est la date à laquelle s'est jouée la partie,
- `winneroldrating` est le rang de classement international, avant la partie, du gagnant de la partie,
- `winnernewrating` est son rang mis à jour après la partie,
- `loseroldrating` est le rang du perdant avant la partie,
- `losernewrating` est son rang mis à jour après la partie.

Dans un terminal, la base de données peut être manipulée grâce à la commande `sqlite3 scrabble_games.sqlite`, qui permet en suite de saisir directement des requêtes SQL.

▷ **Question 16.** 🚩 La table des joueurs ayant été égarée, indiquer comment reconstituer la liste des identifiants des joueurs à partir des parties.

▷ **Question 17.** 📄 Pour chaque joueur, écrire une requête qui détermine son nombre de victoires total.

▷ **Question 18.** 🚩 Quel est l'identifiant du joueur qui totalise le plus grand nombre de victoires ?

▷ **Question 19.** 🚩 Écrire une requête qui permet d'obtenir, pour chaque joueur, son rang de classement obtenu suite à la dernière partie qu'il a jouée.

▷ **Question 20.** 🚩 Proposer une requête qui détermine, pour un joueur fixé, le plus grand nombre de victoires obtenues sans aucune défaite entre elles. (Indication : cette requête est difficile, il peut être pertinent d'y réfléchir pendant que vous traitez la suite du sujet et d'y revenir plus tard. On pourra commencer par déterminer la durée entre ces deux victoires.)

7 Jeu en mode classique

En mode classique, chaque joueur joue à tour de rôle. Initialement, un joueur tire uniformément 7 lettres à partir d'un sac de lettres dont le contenu est fixé au départ du jeu, puis il place un mot sur le plateau et c'est au deuxième joueur de jouer sur le même principe.

▷ **Question 21.** ♠ Justifier que cette situation est modélisable par un jeu d'accessibilité à deux joueur.

▷ **Question 22.** ♠ Proposer une méthode *précise, complète* de recherche de stratégie gagnante.

▷ **Question 23.** 📄 Implémenter la méthode proposée. Comparer le résultat obtenu sur quelques instances en opposant un joueur qui suit cette stratégie et un joueur qui jouerait chacun de ces coups à l'optimal trouvé en mode dupliqué.