IUT de Paris - Rives de Seine
Université Paris Cité

BUT1 – SAÈ S1.02

Comparaison d'approches algorithmiques

– Lexicon –

Le but du projet est de développer un logiciel permettant à un ensemble de joueurs de disputer une partie de Lexicon. L'application doit veiller au respect des règles du jeu et gérer la totalité du déroulement de la partie jusqu'à l'annonce du gagnant.

Le Lexicon est un jeu de lettres créé dans les années 1930. Il a été breveté en 1925 par le britannique David Whitelaw et c'est un des premiers jeux de ce type. Il a été distribué en France par Miro Company à partir de 1936. Les règles du jeu officielles sont accessibles ici `https://regle.escaleajeux.fr/lexic_rg.pdf`. La présentation qui suit est fortement inspirée de ce document.

1 Règles du jeu

Le matériel de jeu est constitué d'un paquet de 52 cartes. Chaque carte porte une lettre de l'alphabet à l'exception d'un *joker* (appelé carte "Maîtresse" dans les règles officielles). À chaque carte est associé un nombre de points. La distribution des cartes (et le nombre de point associé) est donnée en annexe.

Une partie voit s'affronter 2 à 4 joueurs et 10 cartes sont distribuées à chacun d'entre eux. Les cartes restantes forment *le talon* et sont posées face cachée sur la table. La première carte du talon est retournée (face visible). Elle forme *la pile des cartes exposées*.

Les joueurs joue à tour de rôle. Lorsque c'est son tour, le joueur doit nécessairement réaliser une (et une seule) des actions suivantes :

1. Le joueur pioche la première carte exposée ou la première carte du talon et dépose une de ses cartes sur la pile des cartes exposées.

2. Le joueur forme un mot complet à partir de cartes qu'il détient et le pose sur la table.

3. Si un mot a déjà été posé sur la table, le joueur le modifie pour former un nouveau mot en remplaçant au moins une des lettres du mot par des lettres qu'il détient. Par exemple, le mot FARINE peut être transformé et FAMINE en remplaçant le R par un M. Chacune des lettres retirées du mot doit être remplacée par une lettre du joueur. Les lettres retirées sont ramassées par le joueur.

4. Si un mot a déjà été posé sur la table, le joueur le complète pour former un nouveau mot. Toutefois, il ne peut pas changer l'ordre des lettres mais uniquement insérer de nouvelles lettres (en plusieurs positions si nécessaire) qu'il détient. Par exemple, le mot MER peut être complété au début et à la fin pour donner AMERTUME. De même, le mot DOS peut être changé en DROITES.

La carte joker remplace n'importe quelle lettre de l'alphabet au choix du joueur qui la détient.

L'objectif du jeu est de poser toutes ses cartes le plus rapidement possible. Dès qu'un des joueurs y parvient, le tour de jeu est terminé. Les autres joueurs voient leur score augmenter du nombre de points correspondant aux cartes qu'ils ont encore en main. Les cartes sont réunies, mélangées et un nouveau tour débute. Toutefois, tout joueur atteignant 100 points est exclu du tour à venir. Le dernier joueur encore en jeu (ou celui ayant le moins de points si tous ont atteint 100 points) est le gagnant de la partie.

Les mots posés ou formés doivent tous être des mots existants (*i.e.* des mots du dictionnaire). Dans le jeu original, la règle est la suivante. Si un joueur pose ou construit un mot par une des actions ci-dessus et qu'un adversaire doute de sa validité, ce dernier à la possibilité de défier le joueur. Si après vérification, le mot n'est pas valide, le coup est annulé (le joueur reprend ces cartes), le joueur subit une pénalité et il passe son tour. Dans le cas contraire, c'est l'adversaire qui l'a défié qui subit une pénalité. Un défi ne peut concerner qu'un mot formé lors du denier coup. Si au moins un autre joueur a joué sont tour, un mot invalide est considéré comme correct et la partie continue avec. Dans votre projet, cette règle est adaptée de manière à ce que la vérification soit systématique.

Lorsqu'il n'y a plus de carte au talon, la pile des cartes exposées est reprise, battue et remise sur la table dans la position originale (la première carte au sommet du talon ainsi constitué est retournée face visible et forme la première carte exposée), puis le jeu continue comme avant.

2 Cahier des charges

L'application que vous devez développer doit permettre à des joueurs de réaliser une partie dans sa totalité. Les joueurs sont désignés par leur numéro d'ordre (1, 2, *etc*). Le joueur 1 débute le premier tour, le joueur 2 débutera le second tour, *etc*. Le nombre de joueurs est reçu en paramètre de l'application (`lexicon.exe 3` par exemple).

Initialement, votre programme doit gérer le mélange du paquet de cartes, la distribution des cartes aux joueurs et la sélection de la première carte exposée. À chaque étape du jeu, le programme doit afficher la situation courante du jeu :

- L'identité du joueur devant jouer le prochain coup.
- La carte exposée.
- Les cartes détenues par ce joueur.
- La liste numérotée des mots déjà posés sur la table.

La carte joker (qu'elle soit sur la pile des cartes exposées ou au sein d'un mot déjà posé) est représentée par un point d'interrogation. Les autres cartes sont représentée par la lettre qu'elles portent.

Un extrait d'une trace d'exécution est donné en annexe. Vous pourrez y trouver le format d'affichage de la situation de jeu.

Les coups joués par les joueurs peuvent prendre l'une des formes suivantes.

- `T <lettre>` (pour Talon)

  La lettre doit correspondre à une de celles détenues par le joueur. La carte correspondante est placée au dessus des cartes exposées et la première carte du talon est ramassée par le joueur.
- `E <lettre>` (pour Exposée)

  La lettre doit correspondre à une de celles détenues par le joueur. La carte correspondante remplace celle située au dessus des cartes exposées et le joueur ramasse cette dernière.
- `P <mot>` (pour Poser)

  Les lettres composant le mot doivent être détenues par le joueur. S'il lui manque une lettre mais qu'il détient la carte joker, cette carte est employée pour former le mot. Le mot est posé sur la table et les cartes correspondantes sont retirées de la main du joueur.
- `R <numéro> <mot>` (pour Remplacer)

  Le numéro doit désigner un mot présent sur la table et le nouveau mot doit pouvoir être construit à partir de celui-ci en y remplaçant des lettres par d'autres lettres détenues par le joueur. Ici aussi, le joker peut être employé pour remplacer une lettre manquante. Le nouveau mot remplace sur la table celui désigné par le numéro.
- `C <numéro> <mot>` (pour Compléter)

  Le numéro doit désigner un mot présent sur la table et le nouveau mot doit pouvoir être construit à partir de celui-ci en y insérant des lettres détenues par le joueur. Ici aussi, le joker peut être employé pour remplacer une lettre manquante. Le nouveau mot remplace sur la table celui désigné par le numéro.

Avant toute chose, votre programme doit afficher la liste des commandes qu'il est supposé supporter. Un programme complet devra afficher le message "`(Commandes valides : TEPRC)`" indiquant que les commandes `T`, `E`, *etc* sont prises en compte. Les commandes non présentes dans cette liste ne seront pas testées mais vous serez pénalisés pour leur absence.

Vous disposez d'un dictionnaire de la langue française. Il est composé de 369085 mots. Votre application devra faire l'hypothèse que le fichier texte correspondant se trouve dans le répertoire courant où est lancé l'application (surtout ne pas mettre de chemin absolu pour désigner le fichier).

Les coups sont saisis sur une seule ligne de texte. Les espaces servent de séparateur et peuvent être répétés. Toute commande de format incorrect doit être signalée par le message "`Coup invalide, recommencez`" et cela indépendamment de la cause de l'erreur. Aucune saisie ne doit pouvoir provoquer un arrêt brutal de l'application.

Pour les commandes `P`, `R` et `C`, le mot proposé doit appartenir au dictionnaire qui vous est fourni. Si ce n'est pas le cas, le joueur l'ayant proposé reçoit une pénalité de 3 points et son coup est annulé (*i.e.* c'est au prochain joueur de jouer). Votre programme doit afficher le message "`Mot invalide, vous passez votre tour`".

À la fin de chaque tour (*i.e.* dès qu'un joueur a posé toutes ses cartes), le score des joueurs est mis à jour et affiché. Les joueurs ayant dépassé 100 points sont exclus de la partie, les cartes sont distribuées aux joueurs restants et un nouveau tour débute.

À la fin de la partie (*i.e.* il ne reste pas au moins 2 joueurs en course), le programme se termine automatiquement. Vous pourrez trouver le format d'affichage des messages particuliers de fin de tour, d'affichage des scores et de fin de partie dans la trace d'exécution donnée en annexe.

3 Qui, quoi et quand?

Votre projet doit être fait en binôme. Les groupes de 3 ne seront pas acceptés. Évitez de faire votre projet tout seul (soit vous êtes très fort et des personnes ont besoin de votre aide, soit vous avez des difficultés et il faut vous faire aider).

Vous pouvez employer les structures de données vues en cours ou développer les vôtres. Par contre l'usage des chaînes de caractère du C++ (le type `string`) ainsi que les conteneurs de la bibliothèque standard (`vector`, `list`, `map`, `stack`, etc) est strictement interdit. En cas de doute, contactez moi.

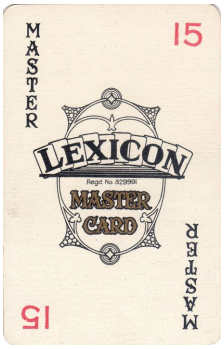
Vous devez porter une attention particulière à la rédaction de votre dossier. Sa qualité est déterminante pour l'évaluation de votre travail. Votre dossier doit être un unique document `pdf` dont la composition est la suivante :

- Une page de garde indiquant le nom et **le groupe** des membres du binôme, l'objet du dossier.
- Une table des matières de l'ensemble du dossier.
- Une brève introduction du projet.
- Le graphe de dépendance des fichiers sources de vos applications. Tous les composants (qu'ils soient réutilisés ou développés) de vos applications devront figurer sur le graphe (*cf.* Cours 4).
- Le code source des tests unitaires que vous aurez écrits (précisez quels tests passent et lesquels échouent).
- Un bilan du projet (les difficultés rencontrées, ce qui est réussi, ce qui peut être amélioré).
- En annexe, le code complet de vos sources (triez les fichiers selon un ordre logique).

Nous vous rappelons que le critère principal de notation est la structuration de votre code. Votre rapport doit mettre en avant la qualité de celle-ci.

Tous les éléments (constantes, types et fonctions) déclarés au sein d'un fichier `.h` doivent être documentés.

Avant le **vendredi 12 janvier 2024**, vous devez déposer une archive (`.zip`) contenant votre rapport (`.pdf`) ainsi que tous les fichiers sources (`.cpp` et `.h`) de votre application. Ces derniers doivent être réunis, au sein de l'archive, dans un dossier nommé `src`.



Annexe

qté	lettre	points	qté	lettre	points	qté	lettre	points
2	A	10	1	J	6	3	S	8
2	B	2	1	K	8	3	T	8
2	C	8	2	L	8	3	U	8
2	D	6	1	M	8	1	V	8
5	E	10	3	N	8	1	W	8
1	F	2	2	O	8	1	X	2
2	G	4	1	P	8	1	Y	4
2	H	8	1	Q	4	1	Z	2
4	I	10	3	R	8	1	Joker	15

Table 1: Nombre de cartes et points associés à chaque lettre

```
(Commandes valides : TEPRC)
* Joueur 1 (A) AEIOPRTUYZ
> P PORTEZ

* Joueur 2 (A) DFGHJKLMQS
1 - PORTEZ
> E K

* Joueur 1 (K) AIUY
1 - PORTEZ
> P AU

* Joueur 2 (K) ADFGHJLMQS
1 - PORTEZ
2 - AU
> R 2 MU

* Joueur 1 (K) IY
1 - PORTEZ
2 - MU
> T Y

* Joueur 2 (Y) AADFGHJLQS
1 - PORTEZ
2 - MU
> P GALAS

* Joueur 1 (Y) AI
1 - PORTEZ
2 - MU
3 - GALAS
> C 2 MUAI

Le tour est fini
* Scores
Joueur 1 : 0 point
Joueur 2 : 35 points
```

```
* Joueur 2 (?) ABCDEFGHIJ
> E H

* Joueur 1 (H) KLMNOPQRST
> P MORT

* Joueur 2 (H) ?ABCDEFGIJ
1 - MORT
> P ABCES

* Joueur 1 (H) KLNPQS
1 - MORTS
2 - ABCE?
> R 2 ABCES

* Joueur 2 (H) DFGIJ
1 - MORT
2 - ABCES
> P FIG
Mot invalide, vous passez votre tour

* Joueur 1 (H) ?KLNPQ
1 - MORTS
2 - ABCES
> ...

...

Le tour est fini
* Scores
Joueur 1 : 76 points
Joueur 2 : 105 points

La partie est finie
```

Figure 1: Exemple de session – en rouge, les données saisies par les joueurs, en bleu, les messages affichés par le programme.