Projet Wordle
Partie 3 : Analyse, Justification et Documentation

Amine Bessaa Matricule: 232431652101

19 décembre 2025

Introduction

Ce document présente l'analyse complète du projet Wordle, incluant le **jeu Wordle** (Partie 1) et le **solveur automatique** (Partie 2). Nous détaillons la stratégie utilisée, les structures de données choisies, la complexité algorithmique ainsi que la documentation des principales fonctions.

1 Analyse et stratégie

1.1 Wordle Game (Partie 1)

1.1.1 Stratégie du jeu

Le Wordle Game choisit aléatoirement un mot secret à partir d'un dictionnaire. Le joueur dispose de 6 tentatives pour deviner ce mot. À chaque tentative, un feedback est fourni afin d'aider le joueur à ajuster sa prochaine proposition.

1.1.2 Utilisation du feedback

Le feedback est défini comme suit :
— **G (Green)** : lettre correcte et bien placée
— **Y (Yellow)** : lettre présente mais mal placée
— **X** : lettre absente du mot
Ce mécanisme respecte les règles officielles du jeu Wordle et gère correctement les lettres répétées.

1.1.3 Efficacité

Cette approche est efficace car :
— Le feedback est clair et intuitif pour le joueur
— Le mot secret est choisi aléatoirement
— Le dictionnaire garantit la validité des mots proposés

1.2 Wordle Solver (Partie 2)

1.2.1 Stratégie du solveur

Le solveur automatique commence avec l'ensemble complet du dictionnaire. À chaque tentative, il choisit un mot encore possible, puis utilise le feedback pour éliminer les mots incompatibles.

1.2.2 Élimination des possibilités

Un mot est conservé uniquement s'il produit exactement le même feedback que celui observé lors de la tentative précédente. Cela permet de réduire progressivement l'espace de recherche.

1.2.3 Pourquoi cette stratégie fonctionne

— Chaque feedback réduit strictement le nombre de mots possibles
— Le solveur ne teste jamais un mot déjà éliminé
— La convergence vers le mot secret est garantie

2 Justification des structures de données

2.1 Structures utilisées

— `char words[MAX_WORDS][WORDLEN+1]` : stockage du dictionnaire
— `int possible[MAX_WORDS]` : indicateur de mots encore valides
— `char guess[], target[], secret[]` : mots temporaires

2.2 Choix des tableaux

Les tableaux permettent :
— Un accès rapide aux mots
— Un parcours séquentiel simple
— Une implémentation claire et efficace

2.3 Alternatives envisagées

— Listes chaînées : suppression facile mais parcours plus lent , cest Vrai : La suppression y est facile.
Mais dans Word Solver :
Il faut vérifier chaque mot.
Il faut simuler un retour d'information pour chaque mot.
L'insertion dynamique n'est pas pertinente.

Les tableaux ont été retenus pour leur simplicité et leurs performances suffisantes.

3 Analyse de complexité

3.1 Wordle Game

— Chargement du dictionnaire : $O(n)$

— Vérification d'un mot : $O(n)$
— Feedback : $O(WORDLEN^2)$

3.2 Wordle Solver

— Filtrage des mots : $O(n \times WORDLEN)$
— Choix du prochain mot : $O(n)$
— Feedback : $O(WORDLEN^2)$

3.3 Complexité spatiale

— Dictionnaire : $O(n \times WORDLEN)$
— Tableaux auxiliaires : $O(n)$
La mémoire utilisée reste raisonnable même pour plusieurs milliers de mots.

4 Documentation du code

4.1 Fonction `load_dictionary`

Rôle : Charger les mots depuis un fichier texte et filtrer ceux de longueur 5.

4.2 Fonction `feedback`

Rôle : Comparer un mot proposé avec le mot secret et produire le feedback Wordle.

4.3 Fonction `compatible`

Rôle : Vérifier si un mot est compatible avec un feedback donné.

4.4 Fonction `filter_words`

Rôle : Éliminer les mots incompatibles du dictionnaire.

4.5 Exemple de code commenté

```
// Elimine les mots incompatibles avec le feedback
void filter_words(char dict[][WORDLEN + 1], int possible[],
                  int n, char guess[], char fb[]) {

    for (int i = 0; i < n; i++) {
        if (possible[i] && !compatible(dict[i], guess, fb)) {
            possible[i] = 0; // mot exclu
        }
    }
}
```

Conclusion

Le projet Wordle combine un jeu interactif et un solveur automatique. Le Wordle Game offre une expérience fidèle au jeu original, tandis que le solver exploite efficacement le feedback pour réduire l'espace de recherche. Les choix algorithmiques et les structures de données assurent un bon compromis entre simplicité, performance et lisibilité du code.