Roadmap du projet C-Wire

Introduction

Le projet C-Wire a été réalisé par un groupe de trois membres : Romain, Ève, et Victor. Ce document retrace les étapes du projet, les contributions individuelles, les tests effectués, les résultats obtenus, ainsi que les outils utilisés.

Chronologie du projet

Début du projet : 03/11/2024

Prise en main du sujet et lecture détaillée des consignes. Division initiale des tâches et première planification.

Semaine 1-2 : Familiarisation avec les outils

Analyse du fichier CSV et compréhension de la topologie des données. Développement initial du script Shell (c-wire.sh) pour gérer le filtrage des données. Difficultés rencontrées : gestion des paramètres du script et adaptation aux grandes tailles de fichiers (~255 Mo).

Semaine 2 : Implémentation du programme en C

Romain : Création de l'arbre AVL pour optimiser les calculs. Gestion des erreurs et des cas de données incorrectes. Victor : Aide ponctuelle pour corriger des bugs mineurs dans le programme C (par exemple, erreurs de segmentation). Difficultés : manipulation des données volumineuses avec les structures AVL et optimisation mémoire.

Semaine 5 : Finalisation du script Shell

Intégration entre le script Shell et le programme C. Tests des options possibles du script et génération des fichiers CSV de sortie.

Semaine 5 : Documentation et rendu final

Ève : Rédaction du README et organisation du dépôt GitHub. Victor : Rédaction de la roadmap et synthèse des étapes clés.

Répartition des tâches

Romain :
- Développement majeur du programme en C (arbre AVL).
- Intégration entre le programme C et le script Shell.
- Réalisation des tests principaux et optimisation des performances.

Ève :
- Rédaction du README et documentation des instructions.
- Organisation et gestion des versions sur GitHub.
- Tests fonctionnels sur les options Shell et validation des sorties.

Victor :
- Assistance sur le développement en C (correction de bugs).
- Rédaction de la roadmap.

- Collaboration sur la conception initiale du script Shell.
- Validation des graphiques et analyse des résultats.

Tests et validation

Objectifs des tests :
- Valider la cohérence des résultats entre les différents cas d'utilisation.
- Tester la robustesse face à des entrées incorrectes ou incomplètes.
- Évaluer les performances sur des fichiers volumineux (>5 millions de lignes).

Cas de test :
1. Tests fonctionnels :
- `./c-wire.sh input/data.csv hvb comp` : Analyse des stations HV-B avec consommateurs entreprises.
- `./c-wire.sh input/data.csv lv all` : Analyse des postes LV pour tous les consommateurs.

2. Tests non valides :
- Paramètres manquants : `./c-wire.sh input/data.csv hvb`.
- Fichier inexistant : `./c-wire.sh missing_file.csv hvb comp`.

3. Tests de performance :
- Temps d'exécution mesuré pour des fichiers de différentes tailles.

Limitations et améliorations

Limitations :
- Le script n'inclut pas de gestion avancée des logs pour le débogage.
- Le programme est sensible aux structures non conformes du fichier CSV.

Améliorations potentielles :
- Ajouter une interface graphique pour simplifier l'utilisation.
- Implémenter un système de logs pour suivre l'état des étapes d'exécution.
- Optimiser davantage les performances pour les fichiers supérieurs à 500 Mo.

Outils utilisés

1. GitHub :
- Collaboration et gestion des versions.
- Stockage des fichiers et suivi des modifications.

2. Environnement Linux :
- Tests et exécutions simulant les conditions réelles.

3. Gnuplot :
- Génération des graphiques pour les postes LV.

4. ChatGPT :
- Assistance pour comprendre les concepts complexes (AVL, optimisation).
- Aide pour résoudre des erreurs de segmentation et des problèmes Shell.

Conclusion

Ce projet a été une expérience enrichissante, avec des apprentissages sur la gestion des données massives, les structures AVL et l'automatisation avec des scripts. Grâce à une collaboration équilibrée et des outils appropriés, nous avons livré un projet fonctionnel respectant les attentes du cahier des charges.