ALGORITHMIQUE AVANCE AVEC PYTHON

PILE, FILE, ARBRE BINAIRE ET ARBRE BINAIRE DE RECHERCHE

(MINI-PROJET)

Présentation Générale

Ces mini-projets sont conçus pour vous permettre de comprendre et d'implémenter des structures d'arbres binaires en Python. Ils impliquent des concepts d'algorithmie avancée, notamment la manipulation des arbres, l'optimisation de parcours et l'application à des contextes pratiques comme la compilation et les réseaux. Une extension est ajoutée pour inclure la notion d'arbre binaire de recherche (ABR), permettant d'optimiser certaines opérations de recherche et d'organisation des données.

Consignes Générales

- Chaque projet doit être réalisé en **binôme ou individuellement**.
- L'utilisation de bibliothèques externes est interdite : **tout doit être codé à la main**.
- Une bonne structuration du code est attendue : **des fonctions claires et bien commentées**.
- Un rapport expliquant la logique adoptée doit accompagner le code source.

Modalités d'Évaluation

- Date limite : **23/03/2025** à **23h59** sur **MyGes**
- Compréhension et implémentation des concepts d'arbres binaires et d'arbres binaires de recherche (45%)
- Qualité et organisation du code (25%)
- Tests et démonstration du programme (20%)
- Clarté du rapport final (10%)

ALGORITHMIQUE AVANCE AVEC PYTHON

PILE, FILE, ARBRE BINAIRE ET ARBRE BINAIRE DE RECHERCHE

(MINI-PROJET)

Projet 1 : Mise en place d'un Compilateur avec les Arbres Binaires

1. Objectif

Ce projet vise à développer un mini-compilateur capable d'analyser et d'exécuter un langage simple en utilisant des arbres binaires pour représenter l'arbre syntaxique abstrait (AST).

2. Fonctionnalités Attenues

1. **Analyse Lexicale (Tokenisation)**
    - Transformer l'expression d'entrée en une liste de jetons.
    - Identifier les types de jetons : nombres, opérateurs (+, -, *, /), variables et affectations.
    - Gérer les espaces et ignorer les caractères inutiles.
    - Utilisation d'une **pile** pour gérer les jetons et prioriser les opérations.
2. **Analyse Syntaxique (Parsing)**
    - Construire un arbre binaire représentant l'expression.
    - Respecter la précédence des opérateurs (ex. multiplication avant addition).
    - Gérer correctement les parenthèses pour imbriquer les expressions.
    - Utiliser une **pile** pour stocker temporairement les opérateurs et construire l'arbre syntaxique.

3. **Interprétation et Exécution**
    - Parcourir l'arbre syntaxique en mode postfixe (post-order traversal).
    - Effectuer les calculs à partir des feuilles vers la racine.
    - Associer les variables à leurs valeurs dans une table de symboles.
    - Utiliser une **pile** pour évaluer l'expression en postfixe.
4. **Exemple de Mini-Langage**

```
x = 5 + 3
y = x * 2
print(y)  # Affiche 16
```

5. **Extension avec l'ABR**
    - Stocker les variables et leurs valeurs dans un arbre binaire de recherche.
    - Permettre une récupération et mise à jour efficace des valeurs des variables.

ALGORITHMIQUE AVANCE AVEC PYTHON

PILE, FILE, ARBRE BINAIRE ET ARBRE BINAIRE DE RECHERCHE

(MINI-PROJET)

Projet 2 : Optimisation du routage des paquets

1. Objectif

Ce projet vise à modéliser une architecture réseau et optimiser le routage des paquets en utilisant des arbres binaires pour structurer une table de routage.

2. Fonctionnalités Attenues

1. **Représentation d'un Réseau sous Forme d'Arbre Binaire**
    - Chaque nœud de l'arbre représente un routeur.
    - Les branches de l'arbre représentent les connexions entre les routeurs.
    - Chaque feuille correspond à une destination finale du réseau.
    - Utilisation d'une **file** pour stocker les paquets en attente d'envoi.
2. **Implémentation d'un Algorithme de Routage**
    - Construire l'arbre de routage avec des valeurs de latence ou de distance sur chaque connexion.
    - Utiliser un parcours de l'arbre (ex. parcours en profondeur ou en largeur) pour trouver le chemin optimal vers une destination.
    - Comparer plusieurs chemins et renvoyer celui avec le coût le plus faible.
    - Utilisation d'une **file** pour explorer les nœuds lors du parcours en largeur (BFS).

ALGORITHMIQUE AVANCE AVEC PYTHON

PILE, FILE, ARBRE BINAIRE ET ARBRE BINAIRE DE RECHERCHE

(MINI-PROJET)

3. **Simulation d'un Envoi de Paquets**
    - Prendre une destination définie et simuler l'envoi d'un paquet depuis la racine de l'arbre.
    - Suivre le chemin optimal calculé précédemment et afficher les nœuds traversés.
    - Gérer les cas où une route est obstruée et proposer une alternative.
    - Utilisation d'une **file** pour gérer l'envoi et le traitement des paquets en attente.
4. **Extension avec l'ABR**
    - Stocker les routes et leur coût dans un arbre binaire de recherche.
    - Permettre une recherche rapide du meilleur chemin.
    - Optimiser la gestion des mises à jour des routes dans le réseau.