ALGORITHMIQUE AVANCE AVEC PYTHON

Présentation Générale

Ce mini-projet a été conçus pour nous 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

Instructions : inscrivez le contexte du cahier des charges

Exigences

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.

Modéliser une architecture réseau

Date limite : 23/03/2025 à 23h59 sur MyGes

Optimiser le routage des paquets en utilisant des arbres binaires pour structurer une table de routage.

Instructions : inscrivez les exigences.

Points d'attentions

Compliquer d'afficher le chemin le plus cours en fonction du temps d'exécution de la méthode de recherche la plus rapide, sans la lib time

Utilisation de la méthode BFS pour l'affichage du chemin de la racine vers le nœud "réseau" rechercher

Ajout d'un fichier cache pour stocker les recherches et optimiser la recherche de l'utilisateur

Utilisation de deux paths pour la recherche avec les trois méthodes et pour le chemin de la racine jusqu'au nœud recherché

l'arbre binaire représente le réseau informatique d'une entreprise.

On considère qu'il n'y a aucune redondance, et que le ficher cache a été réaliser afin d'apprendre à extraire des données, les formater et de les réutiliser.

Instructions : inscrivez les points d'actions à prendre en compte.

Tâches

ALGORITHMIQUE AVANCE AVEC PYTHON

Présentation Générale

Ce mini-projet a été conçus pour nous 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

Instructions : inscrivez le contexte du cahier des charges

Exigences

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.	Modéliser une architecture réseau
Date limite : 23/03/2025 à 23h59 sur MyGes	Optimiser le routage des paquets en utilisant des arbres binaires pour structurer une table de routage.

Instructions : inscrivez les exigences.

Points d'attentions

Compliquer d'afficher le chemin le plus cours en fonction du temps d'exécution de la méthode de recherche la plus rapide, sans la lib time

Utilisation de la méthode BFS pour l'affichage du chemin de la racine vers le nœud "réseau" rechercher

Ajout d'un fichier cache pour stocker les recherches et optimiser la recherche de l'utilisateur

Utilisation de deux paths pour la recherche avec les trois méthodes et pour le chemin de la racine jusqu'au nœud recherché

l'arbre binaire représente le réseau informatique d'une entreprise.

On considère qu'il n'y a aucune redondance, et que le ficher cache a été réaliser afin d'apprendre à extraire des données, les formater et de les réutiliser.

Instructions : inscrivez les points d'actions à prendre en compte.

Tâches

Instructions

Étape 1 : Ajouter les modules à créer sur les autocollants jaunes.
Étape 2 : Ajouter le sujet de chaque module sur les autocollants gris.
Étape 3 : Ajouter les fonctionnalités de chaque module sur les autocollants verts.
Étape 4 : Ajouter les fonctionnalités supplémentaires sur les autocollants cyans.
Étape 5 : Ajouter les Ressources documentaires à créer (descriptions/documentations).

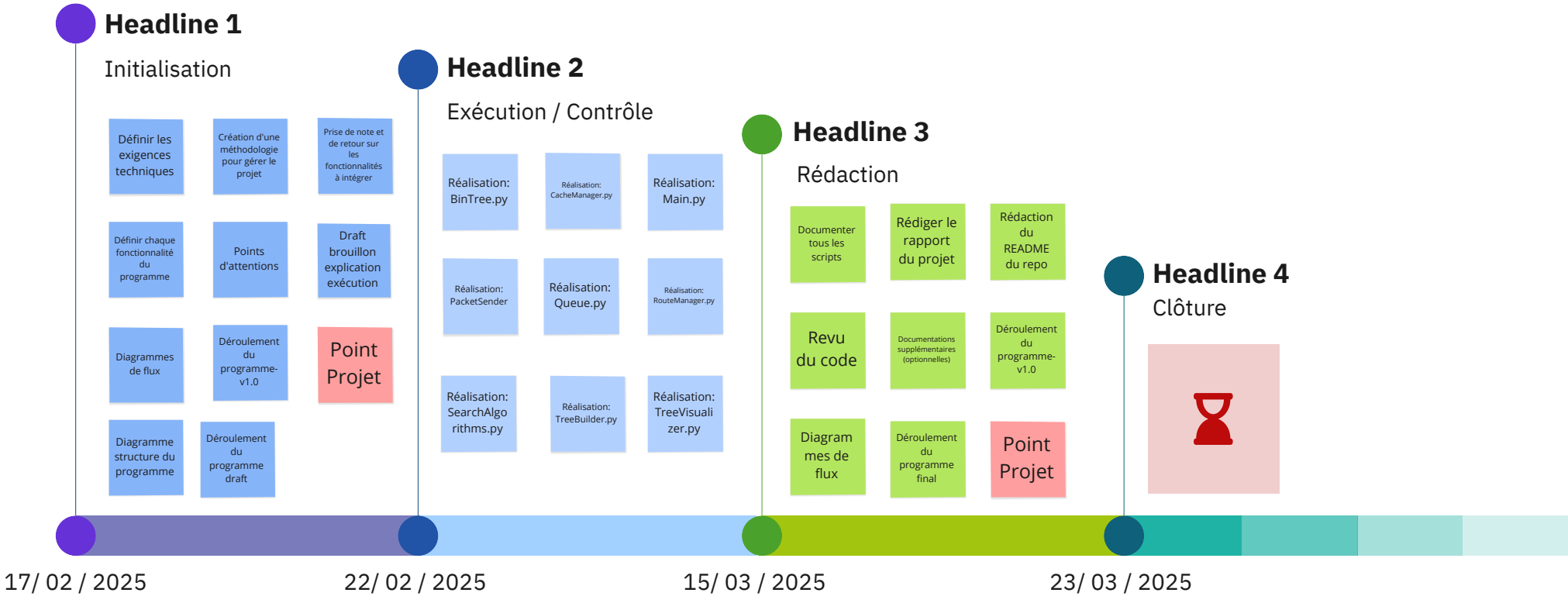
Définir une hiérarchisation des taches : les taches prioritaires en haut de la liste.
Tâche fini : ajouter le symbole pour valider la tâche.
Bloquante : changer la couleur des autocollants en rouge si vous rencontrez un point bloquant.

Release Legend

Modules | Fonctions | Fonctions supplémentaires | points bloquant

Sujets | Ressources documentaires

Off-boarding plan for Maxence



Explication du programme

Ce diagramme représente la structure du programme et les interactions entre les différents fichiers

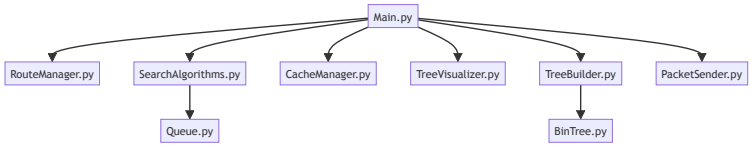
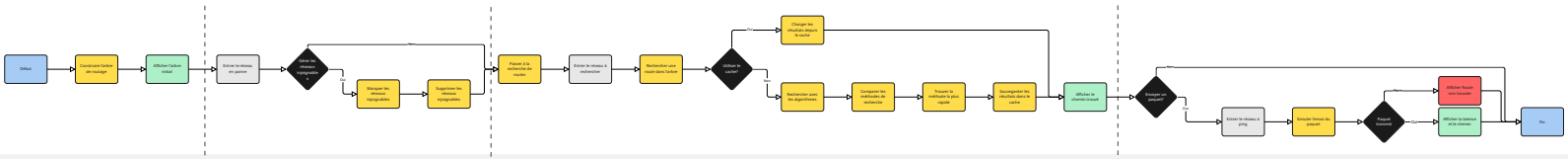
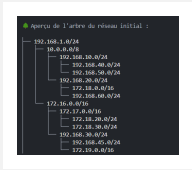
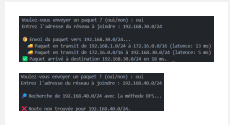


Diagramme de flux qui illustre le déroulement du programme, depuis la construction de l'arbre jusqu'à l'envoi de paquets.
Le diagramme montre les étapes principales et les interactions entre les différents modules.







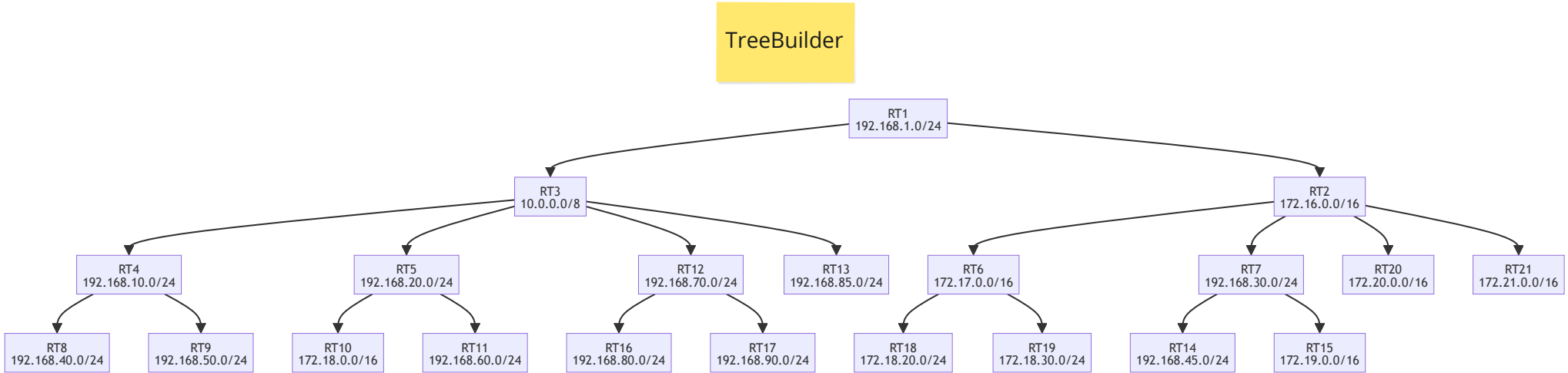
Explication du Diagramme de Flux

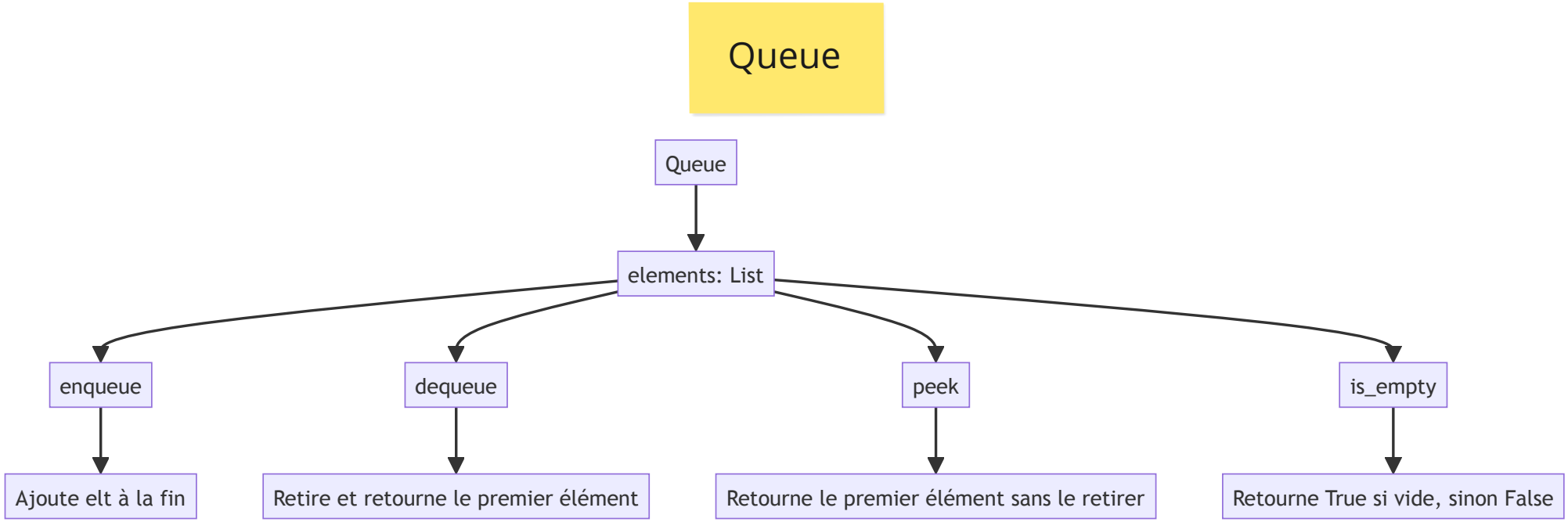
1. **Début** : Le programme commence par la construction de l'arbre de routage.
2. **Construire l'arbre de routage** : L'arbre est construit à partir des sous-réseaux et des routeurs définis dans `TreeBuilder.py`.
3. **Afficher l'arbre initial** : L'arbre est affiché dans le terminal grâce à `TreeVisualizer.py`.
4. **Gérer les réseaux injoignables** : L'utilisateur est invité à entrer un réseau injoignable. Si un réseau est marqué comme injoignable, il est supprimé de l'arbre.
5. **Rechercher une route dans l'arbre** : L'utilisateur entre une adresse IP à rechercher. Le programme vérifie si le résultat est déjà dans le cache.
6. **Utiliser le cache** : Si le résultat est dans le cache, il est chargé directement. Sinon, le programme utilise les algorithmes de recherche pour trouver la route.
7. **Comparer les méthodes de recherche** : Les algorithmes de recherche (préfixe, infixe, suffixe) sont comparés pour trouver la méthode la plus rapide. En parallèle, la méthode BFS est utilisée pour trouver le chemin le plus court en ignorant la cache vers le réseau recherché.
8. **Trouver la méthode la plus rapide** : La méthode la plus rapide est sélectionnée, et le résultat est sauvegardé dans le cache ainsi que le chemin depuis la racine.
9. **Afficher le chemin trouvé** : Si le chemin est trouvé, la route identifiée par la méthode la plus efficace sera affichée à l'utilisateur, ainsi que le chemin parcouru et le chemin depuis la racine.
10. Livre
11. **Envoyer un paquet** : L'utilisateur est invité à envoyer un paquet sur le réseau. Si oui, le paquet est calculé avec une latence aléatoire.
12. **Simuler l'envoi du paquet** : Le programme simule l'envoi du paquet en affichant chaque saut et la latence totale.
13. **Fin** : Le programme se termine après l'envoi du paquet ou si l'utilisateur choisit de ne pas envoyer de paquet.

Fonctionnalités Couvertes par le Diagramme

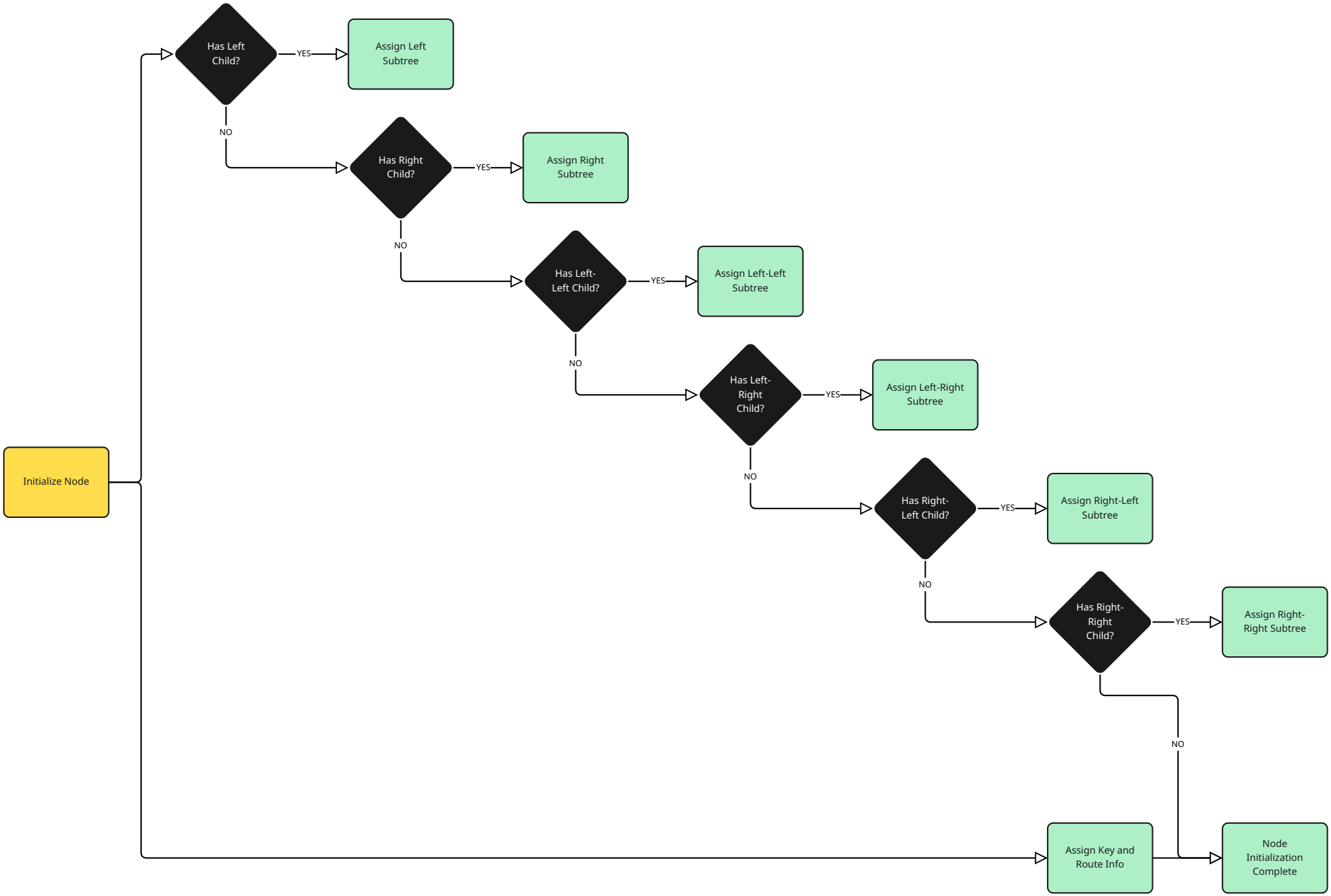
- **Construction de l'arbre** : Gérée par `TreeBuilder.py`
- **Gestion des réseaux injoignables** : Gérée par `RouteManager.py`
- **Recherche de routes** : Gérée par `SearchAlgorithms.py` et `CacheManager.py`
- **Envoi de paquets** : Géré par `PacketSender.py`
- **Visualisation de l'arbre** : Gérée par `TreeVisualizer.py`

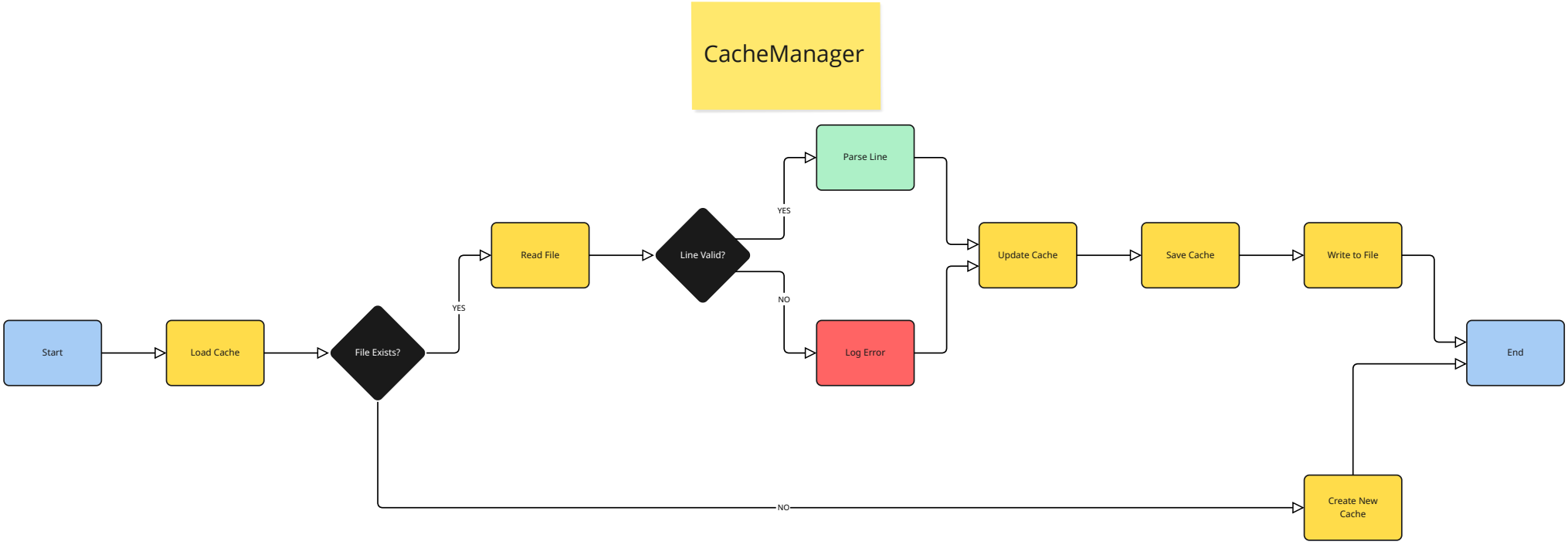
Ce diagramme de flux résume le déroulement logique du programme et montre comment les différents modules interagissent pour offrir les fonctionnalités principales.


TreeBuilder
RT1
192.168.1.0/24
RT3
10.0.0.0/8
RT2
172.16.0.0/16
RT4
192.168.10.0/24
RT5
192.168.20.0/24
RT12
192.168.70.0/24
RT13
192.168.85.0/24
RT6
172.17.0.0/16
RT7
192.168.30.0/24
RT20
172.20.0.0/16
RT21
172.21.0.0/16
RT8
192.168.40.0/24
RT9
192.168.50.0/24
RT10
172.18.0.0/16
RT11
192.168.60.0/24
RT16
192.168.80.0/24
RT17
192.168.90.0/24
RT18
172.18.20.0/24
RT19
172.18.30.0/24
RT14
192.168.45.0/24
RT15
172.19.0.0/16

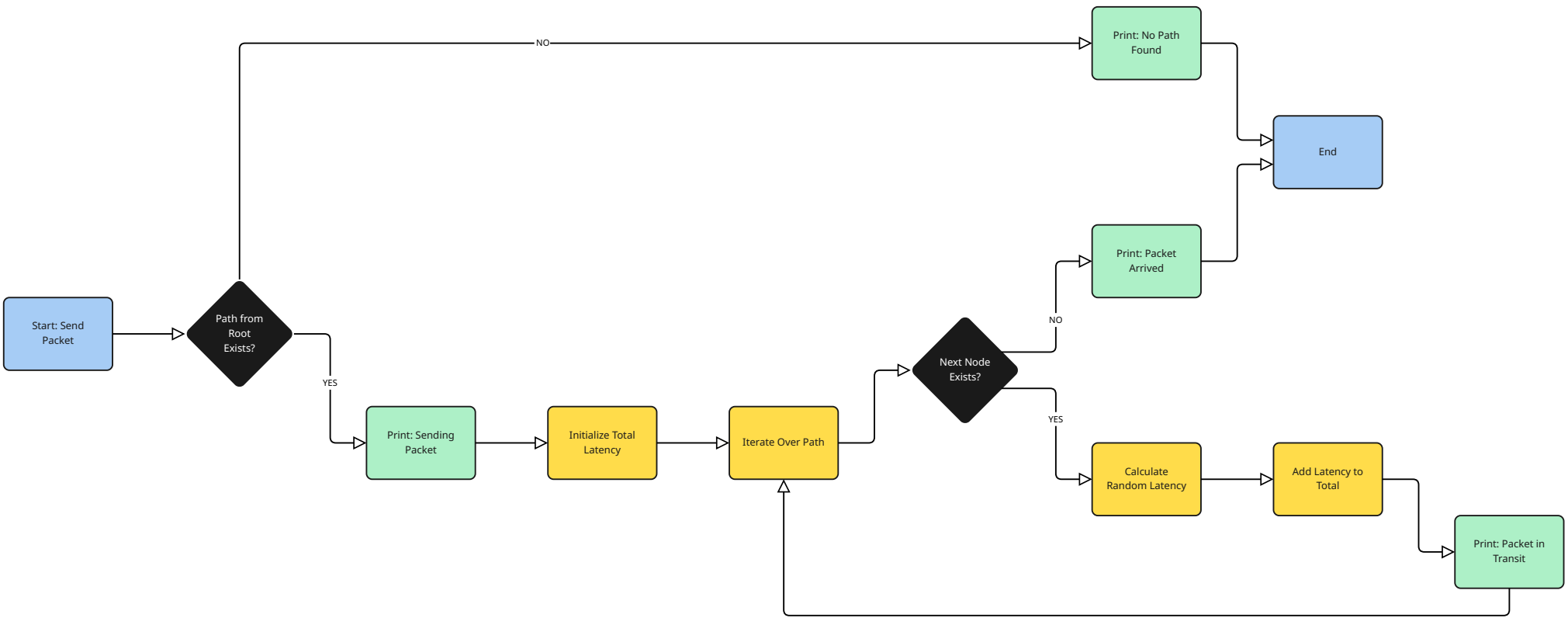

Queue
Queue
elements: List
enqueue
dequeue
peek
is_empty
Ajoute elt à la fin
Retire et retourne le premier élément
Retourne le premier élément sans le retirer
Retourne True si vide, sinon False

BinTree

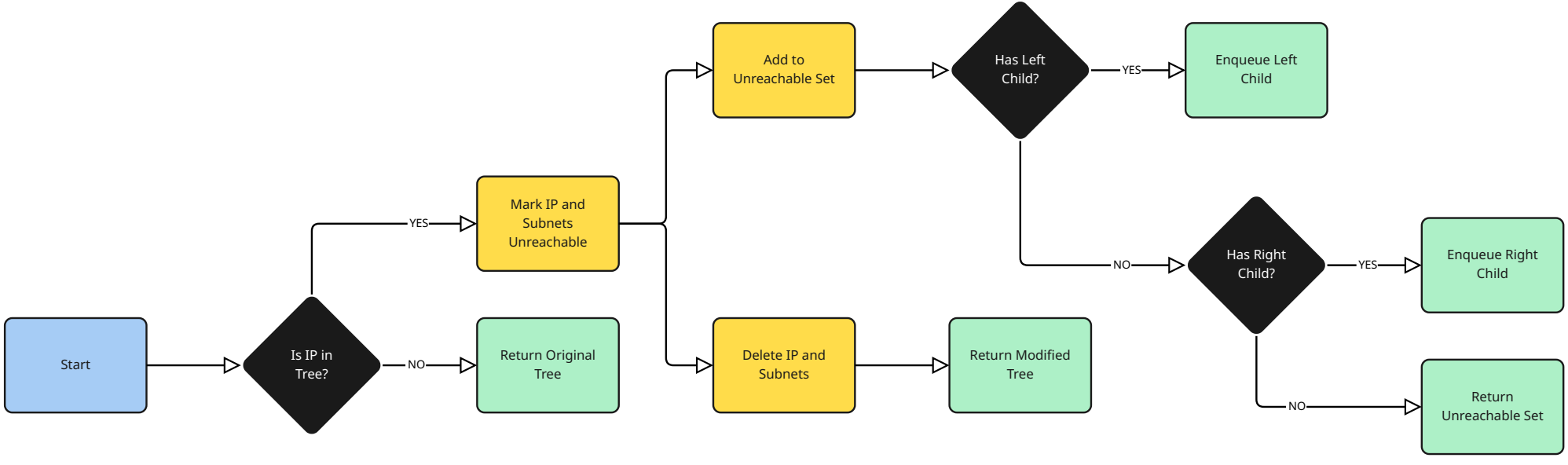

Initialize Node
Has Left Child?
YES
Assign Left Subtree
NO
Has Right Child?
YES
Assign Right Subtree
NO
Has Left-Left Child?
YES
Assign Left-Left Subtree
NO
Has Left-Right Child?
YES
Assign Left-Right Subtree
NO
Has Right-Left Child?
YES
Assign Right-Left Subtree
NO
Has Right-Right Child?
YES
Assign Right-Right Subtree
NO
Assign Key and Route Info
Node Initialization Complete

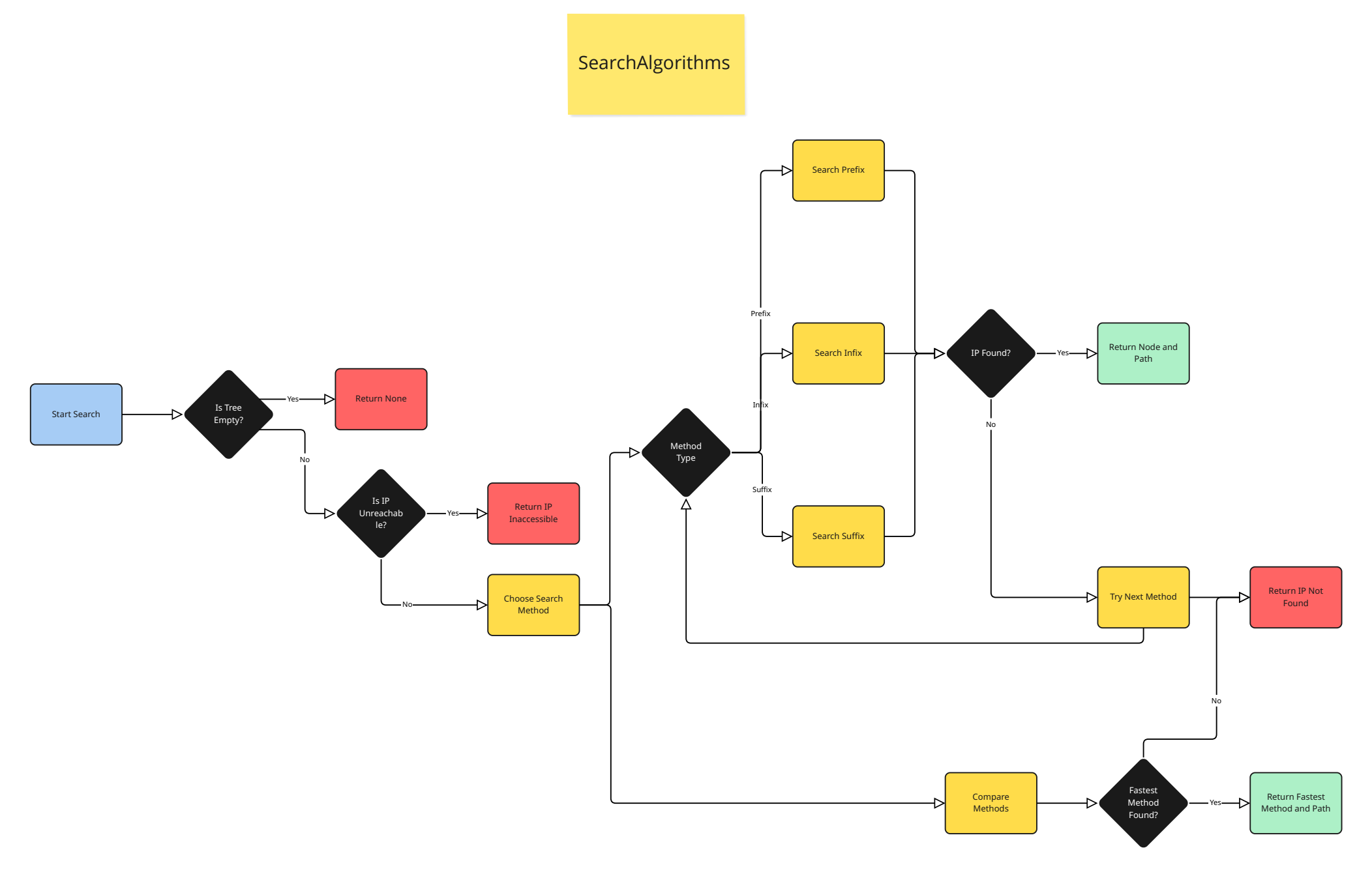

CacheManager
Start
Load Cache
File Exists?
YES
NO
Read File
Line Valid?
YES
NO
Parse Line
Log Error
Update Cache
Save Cache
Write to File
End
Create New Cache

PacketSender



RouteManager




SearchAlgorithms
Start Search
Is Tree Empty?
Yes
Return None
No
Is IP Unreachable?
Yes
Return IP Inaccessible
No
Choose Search Method
Method Type
Prefix
Search Prefix
Infix
Search Infix
Suffix
Search Suffix
IP Found?
Yes
Return Node and Path
No
Try Next Method
Return IP Not Found
No
Compare Methods
Fastest Method Found?
Yes
Return Fastest Method and Path

TreeVisualizer