Université Abdelmalek Essaadi

Faculté des Sciences et Techniques

Tanger

Rapport Réalisation d'une application de Hachage et Indexation

Réalisé par : ARICHI Fatima zahra et EL MEZIANE Chaima.

Encadré par : Pr. EZZIYYANI Mostafa

Année Universitaire : 2023/2024

Table des matières

Introduction

Le développement d'une application de gestion de structures de données représente une réponse essentielle aux besoins croissants en matière de manipulation efficace de données dans un large éventail de domaines. Dans un monde où l'information est omniprésente et où sa gestion optimale est cruciale, une application robuste et conviviale pour gérer ces données est indispensable. Ce rapport vise à documenter en détail le processus de développement de notre application, qui répond à ces exigences en offrant à l'utilisateur une interface graphique conviviale pour gérer deux types majeurs de structures de données : les arbres B et les tables de hachage.

Contexte du Projet

Le projet est né d'une nécessité pressante de répondre aux exigences croissantes des utilisateurs en matière de manipulation de structures de données complexes. Les arbres B et les tables de hachage, étant des structures fondamentales dans le domaine du stockage et de la gestion des données, sont largement utilisés dans une multitude de contextes. Cependant, leur manipulation peut parfois s'avérer complexe et nécessiter des compétences techniques avancées. C'est dans ce contexte que notre application entre en jeu, avec pour objectif principal de simplifier l'utilisation de ces structures de données tout en offrant une expérience utilisateur agréable et intuitive.

Objectifs

Les objectifs principaux de notre application sont les suivants :

1. **Offrir une sélection de structures de données :** L'un des piliers de notre application est de fournir à l'utilisateur le choix entre deux types majeurs de structures de données : les arbres B et les tables de hachage. Cette flexibilité permet à l'utilisateur de sélectionner la structure qui correspond le mieux à ses besoins spécifiques en termes de stockage et de manipulation des données. En offrant cette diversité, notre application s'adapte aux différentes situations et exigences des utilisateurs.

2. **Faciliter les opérations de manipulation :** L'application vise à permettre à l'utilisateur d'effectuer un large éventail d'opérations sur les structures de données sélectionnées. Cela comprend des opérations telles que l'insertion de nouvelles données, la modification des données existantes, la suppression d'éléments ainsi que la recherche efficace dans les données stockées. En fournissant une interface intuitive et conviviale, notre objectif est de rendre ces opérations aussi simples et accessibles que possible, même pour les utilisateurs moins expérimentés.

3. **Créer une interface graphique intuitive :** Un autre aspect clé de notre application est la conception d'une interface utilisateur graphique (GUI) intuitive. Cette interface permet à l'utilisateur d'interagir de manière efficace avec les données, en fournissant des fonctionnalités de visualisation en temps réel pour refléter les modifications apportées aux structures de données. Des graphiques dynamiques, des représentations visuelles des données et des indicateurs en temps réel sont intégrés à l'interface pour offrir une expérience immersive et informative à l'utilisateur.

4. **Assurer la convivialité et la performance :** Enfin, un objectif majeur de notre application est d'assurer à la fois convivialité et performance. Cela implique non seulement de concevoir une interface facile à utiliser, mais aussi de garantir que l'application fonctionne de manière fluide et réactive.

Description du Projet

1. Le langage de programmation

Python est un langage de programmation interprété, de haut niveau, polyvalent et orienté objet. Créé par Guido van Rossum, Python met l'accent sur la lisibilité du code, la simplicité syntaxique et la facilité d'apprentissage. Son écosystème dynamique et sa communauté active en font un choix populaire pour une variété d'applications, allant du développement web à l'analyse de données et à l'apprentissage automatique.

Figure 1 : Logo du langage de programmation Python

Caractéristiques de l'utilisation de Python dans ce projet :

- **Simplicité et Lisibilité :** La syntaxe claire et lisible de Python facilite la compréhension du code, ce qui est particulièrement avantageux pour les étudiants novices en programmation.

- **Polyvalence :** Python est un langage polyvalent adapté à une large gamme de tâches, de la manipulation de données à l'implémentation d'algorithmes d'apprentissage automatique, répondant ainsi aux exigences variées du projet.

- **Écosystème d'Apprentissage Automatique :** L'abondance de bibliothèques dédiées à l'apprentissage automatique en Python, telles que scikit-learn, offre une base solide pour la mise en œuvre des différents algorithmes requis dans le projet.

- **Communauté Active :** La communauté Python active assure un support continu, des mises à jour régulières et un accès à une vaste gamme de ressources éducatives.

- **Facilité d'Intégration :** Python peut être intégré facilement avec d'autres technologies, facilitant l'incorporation d'interfaces utilisateur, de représentations graphiques des données et d'autres composants essentiels du projet.

- **Documentation Abondante :** La documentation complète de Python simplifie le

développement, tandis que les nombreuses ressources disponibles encouragent la rédaction d'une documentation exhaustive pour le projet.

- **Facilité de Visualisation :** Les bibliothèques de visualisation de données telles que Matplotlib et Seaborn en Python facilitent la création d'interfaces graphiques pour représenter les données de manière claire et explicite, répondant ainsi aux besoins de visualisation du projet.

2. Définition d'arbre B et de Hachage

Arbre B

L'arbre B est une structure de données en informatique, principalement utilisée pour organiser et stocker des données de manière efficace dans les bases de données et les systèmes de fichiers. Il se compose de nœuds interconnectés, comprenant plusieurs clés et des pointeurs vers d'autres nœuds. Les caractéristiques principales de l'arbre B incluent une hauteur équilibrée, des nœuds internes avec un nombre minimal de clés et des nœuds feuilles contenant les données finales. Sa conception permet des opérations de recherche, d'insertion et de suppression efficaces, en assurant des performances optimales même pour des ensembles de données volumineux.

Hachage

Le hachage est un processus utilisé en informatique pour mapper des données de taille arbitraire (telles que des chaînes de caractères, des fichiers, ou des valeurs numériques) vers des valeurs de taille fixe, généralement appelées « hachés » ou « hash codes ». Ce processus est réalisé à l'aide d'une fonction de hachage, qui prend en entrée les données originales et les transforme en une valeur de hachage unique correspondante. Voici les différents types de hachage utilisés dans notre application :

- Hachage Linéaire
  Le hachage linéaire est une technique de résolution de collisions dans une table de hachage. Lorsqu'une collision se produit (c'est-à-dire que deux clés de hachage se mappent sur la même position), cette méthode déplace linéairement vers l'avant dans la table jusqu'à ce qu'une position vide soit trouvée.

- Double Hachage
  Le double hachage est une technique de résolution de collisions dans une table de hachage qui utilise deux fonctions de hachage. Lorsqu'une collision se produit, la deuxième fonction de hachage est utilisée pour calculer un décalage supplémentaire, ce qui permet de trouver une nouvelle position pour stocker la clé. Cette méthode vise à réduire la formation de clusters et à améliorer les performances.

- Chaînage Séparé
  Le chaînage séparé est une méthode de résolution de collisions dans une table de hachage où chaque emplacement de la table pointe vers une liste chaînée contenant toutes les clés qui se sont hachées à cet emplacement. Lorsqu'une collision se produit, la nouvelle clé est simplement ajoutée à la liste chaînée correspondante. Cette méthode garantit un accès rapide aux éléments et peut être efficace pour gérer les collisions.

- Chaînage Interne
  Le chaînage interne est une technique de résolution de collisions dans une table de hachage où chaque emplacement de la table peut contenir plusieurs éléments en les chaînant ensemble à l'intérieur de la même case. Cela signifie que chaque emplacement de la table peut contenir plusieurs clés, réduisant ainsi le besoin de stocker des pointeurs supplémentaires et économisant de l'espace mémoire.

Fonctionnalités Clés

1. Page d'accueil
   Lorsque l'utilisateur lance notre application, il est immédiatement accueilli par une interface conviviale conçue pour simplifier au maximum le processus de manipulation des données. L'utilisateur il est accueilli par une interface conviviale lui permettant de choisir entre l'utilisation d'un arbre B ou d'une table de hachage.

Figure 2 : Interface d'accueil

2. Arbre B
   Dans la section dédiée à l'arbre B, notre application offre à l'utilisateur une gamme complète de fonctionnalités pour interagir avec cette structure de données complexe de manière efficace et intuitive. L'utilisateur peut non seulement effectuer des opérations basiques telles que l'insertion, la modification, la suppression et la recherche de nœuds dans l'arbre, mais il peut le faire de manière transparente grâce à une interface conviviale et réactive.

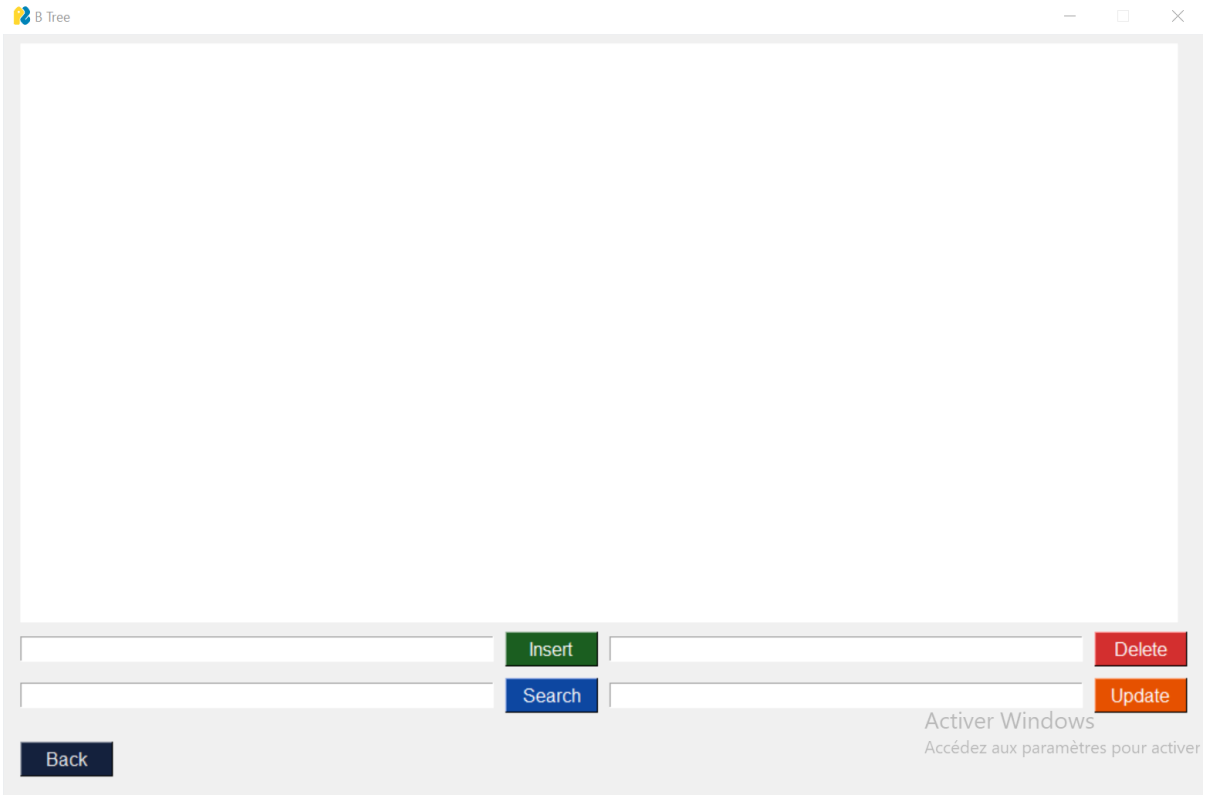

Figure 3 : Interface d'Arbre B

Les nœuds de l'arbre s'ajoute au fur et à mesure l'utilisateur ajoute un élément dans l'arbre B

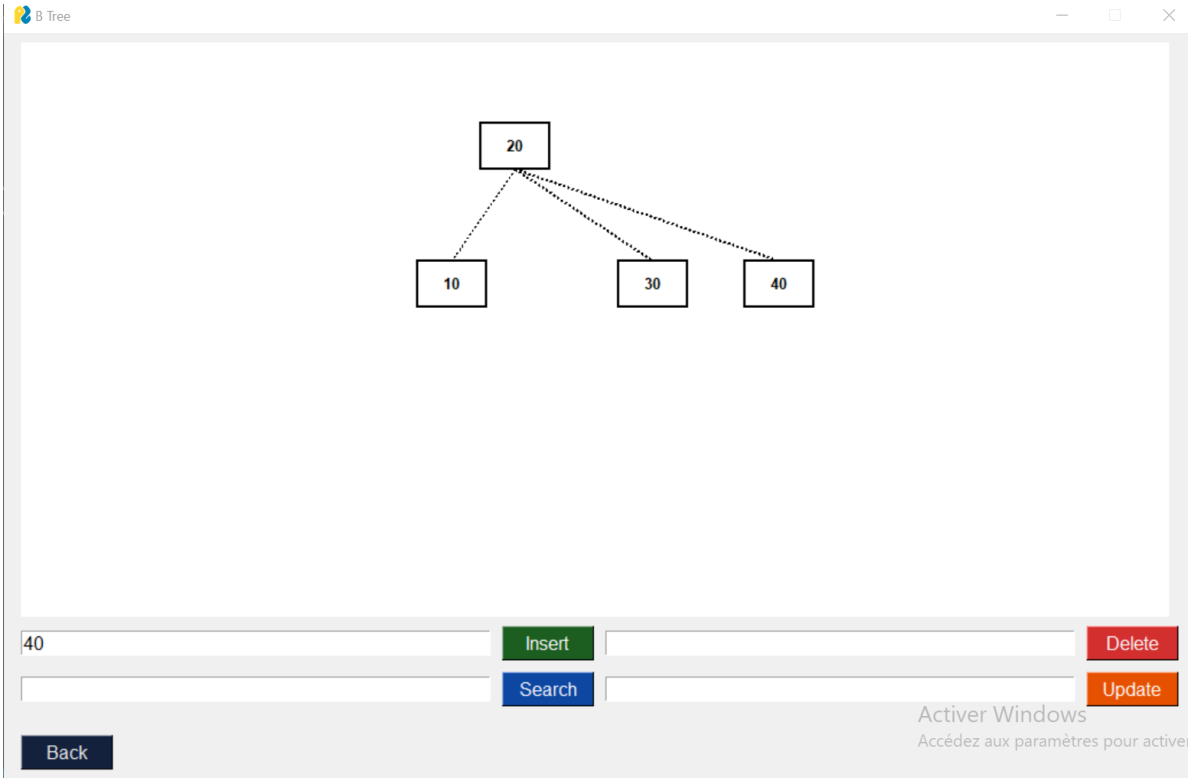

Figure 4 : Interface de visualisation d'une Arbre de 4 noeuds

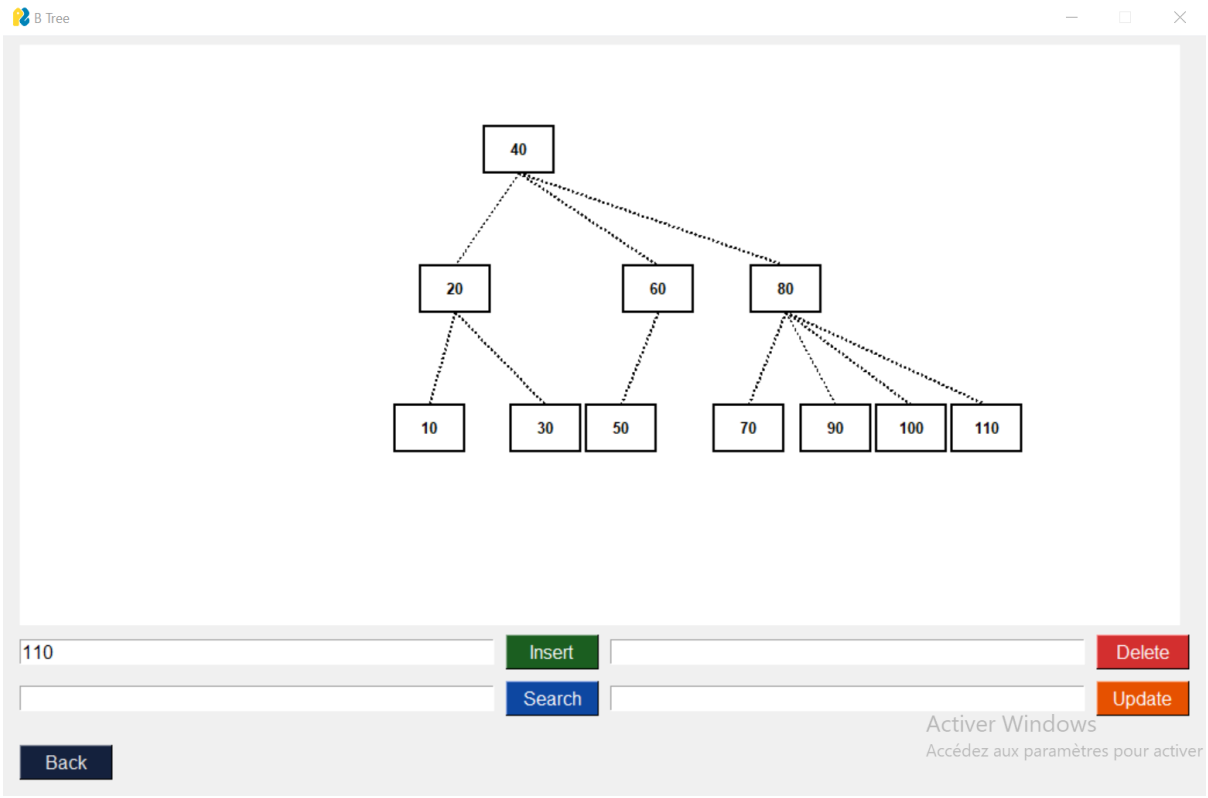

Figure 5 : Interface de visualisation d'une Arbre de 11 noeuds

On a essayé de supprimer l'élément : 100

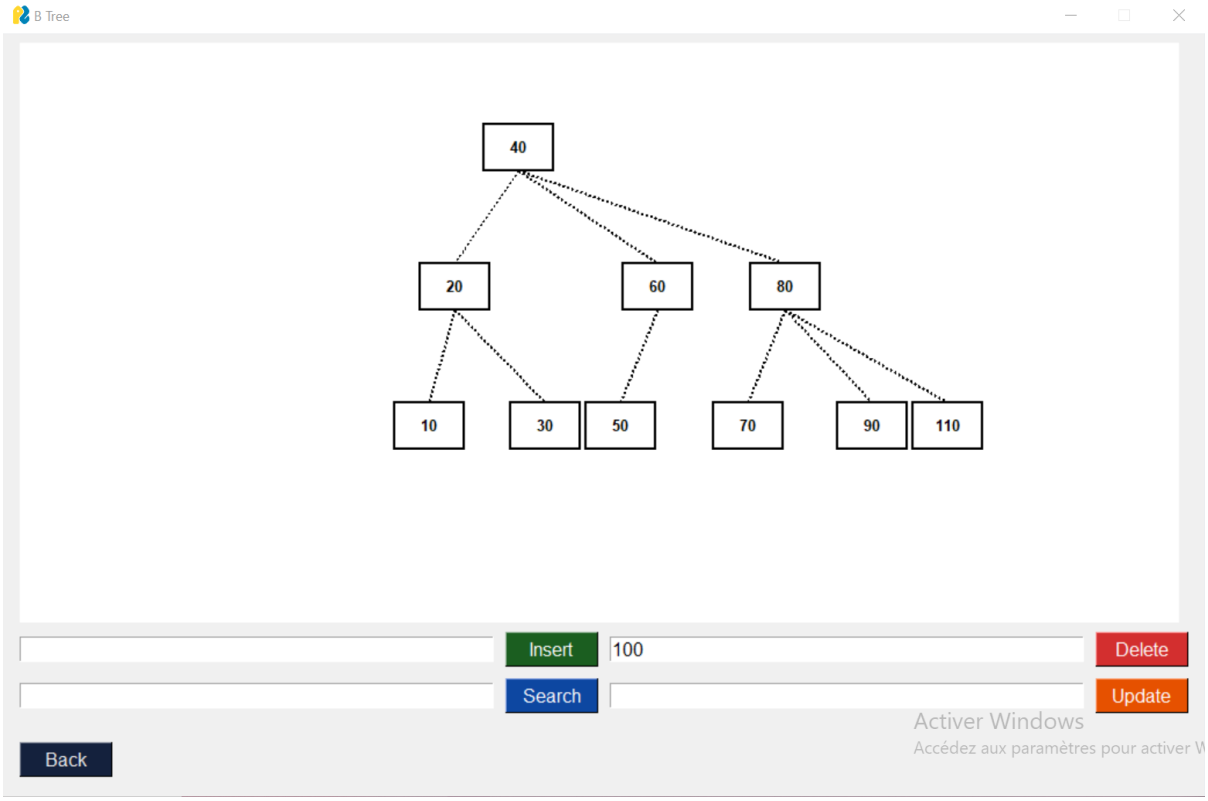

Figure 6 : Interface de visualisation de l'Arbre après suppression de l'element 100

Ici on va essayer de modifier : 40 avec 90

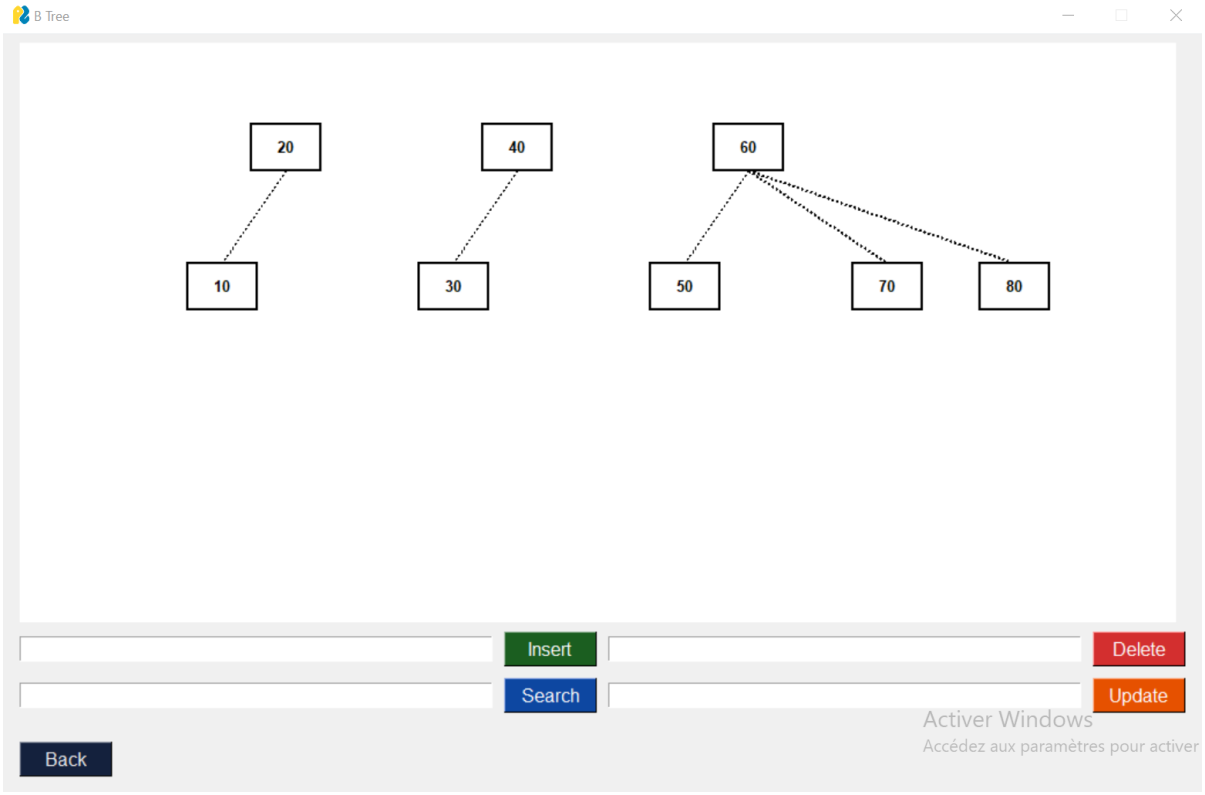

Figure 7 : Interface de visualisation de l'Arbre avant modification de 40

Voici le résultat :

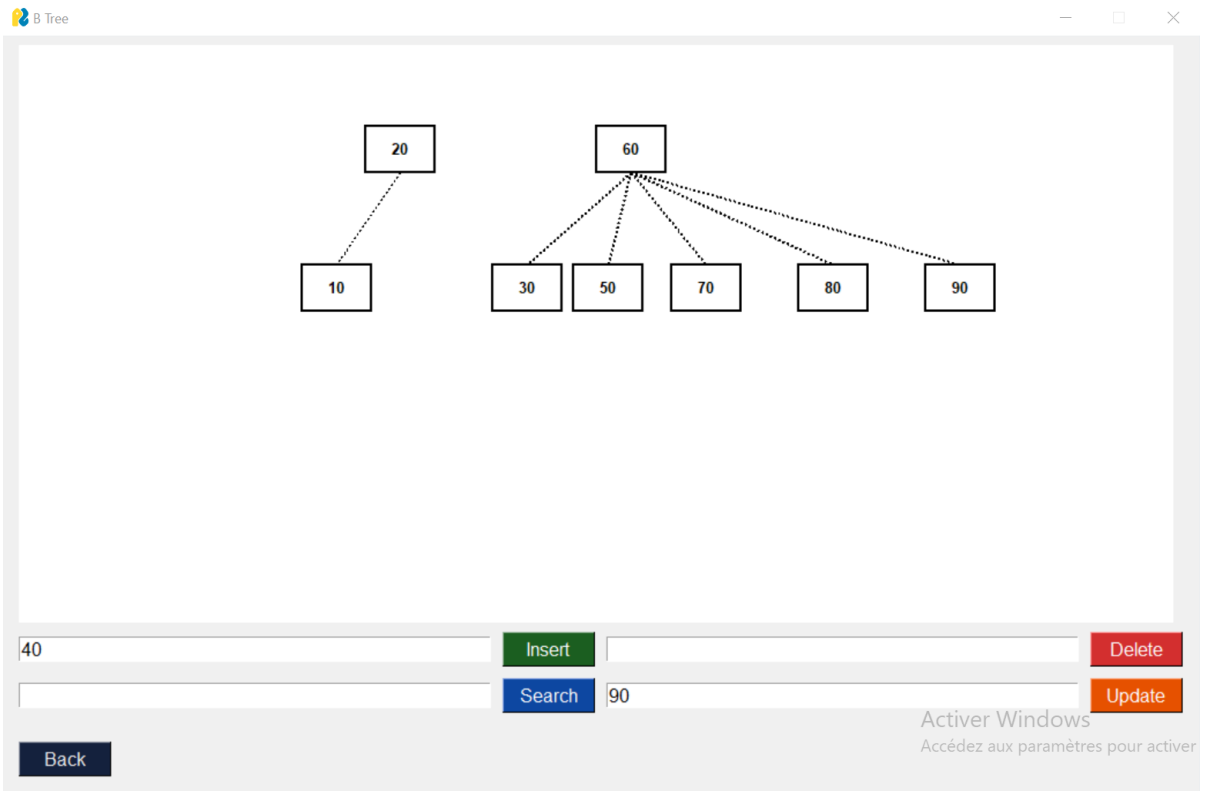

Figure 8 : Interface de visualisation de l'Arbre après modification de 40

On a essayé de chercher sur l'élément : 10

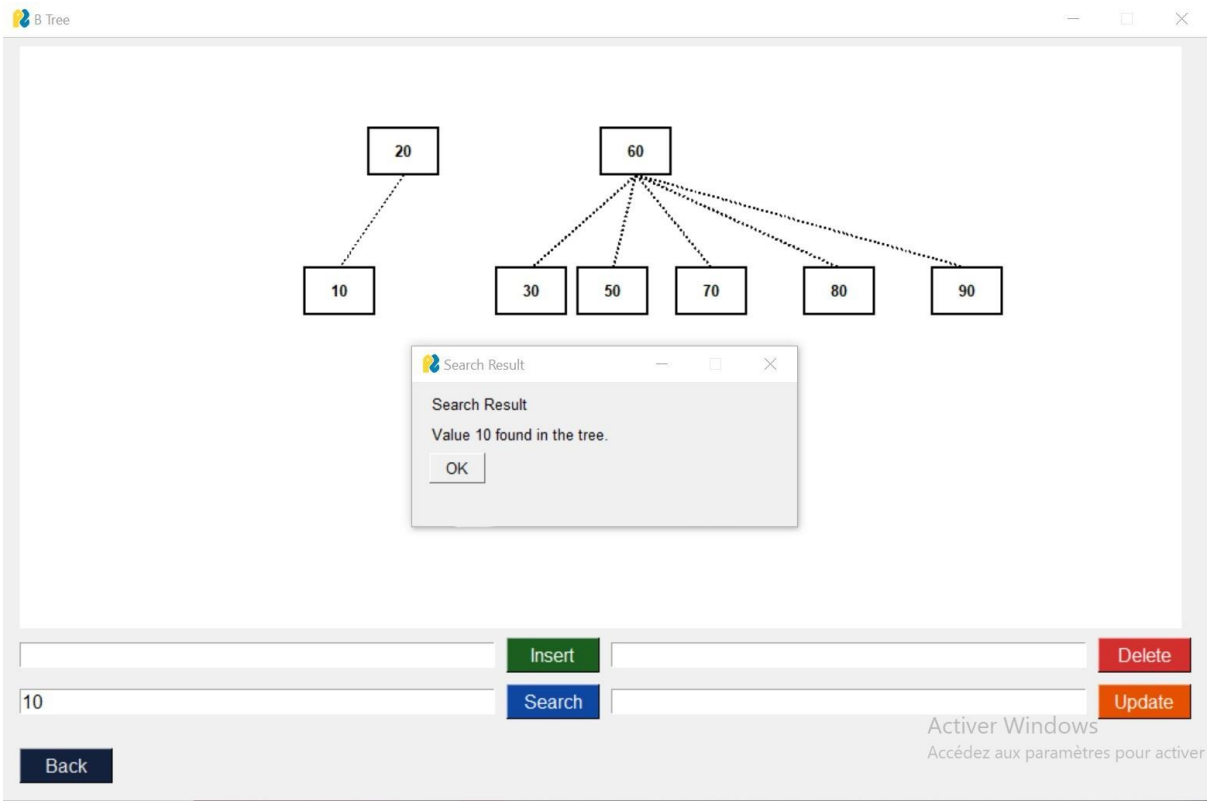

Figure 9 : Interface de visualisation de la fonction de recherche d'un element

3. Hachage

Dans la section de la table de hachage, l'utilisateur peut ajouter des éléments à la table. À travers une interface bien conçue, il peut saisir les données à insérer et choisir parmi différents types de hachage disponibles, tels que le hachage linéaire, le double hachage, le chaînage interne ou le chaînage séparé, l'utilisateur peut supprimer des éléments et vider entièrement la table.

On a choisi une taille fixe de la table : 28

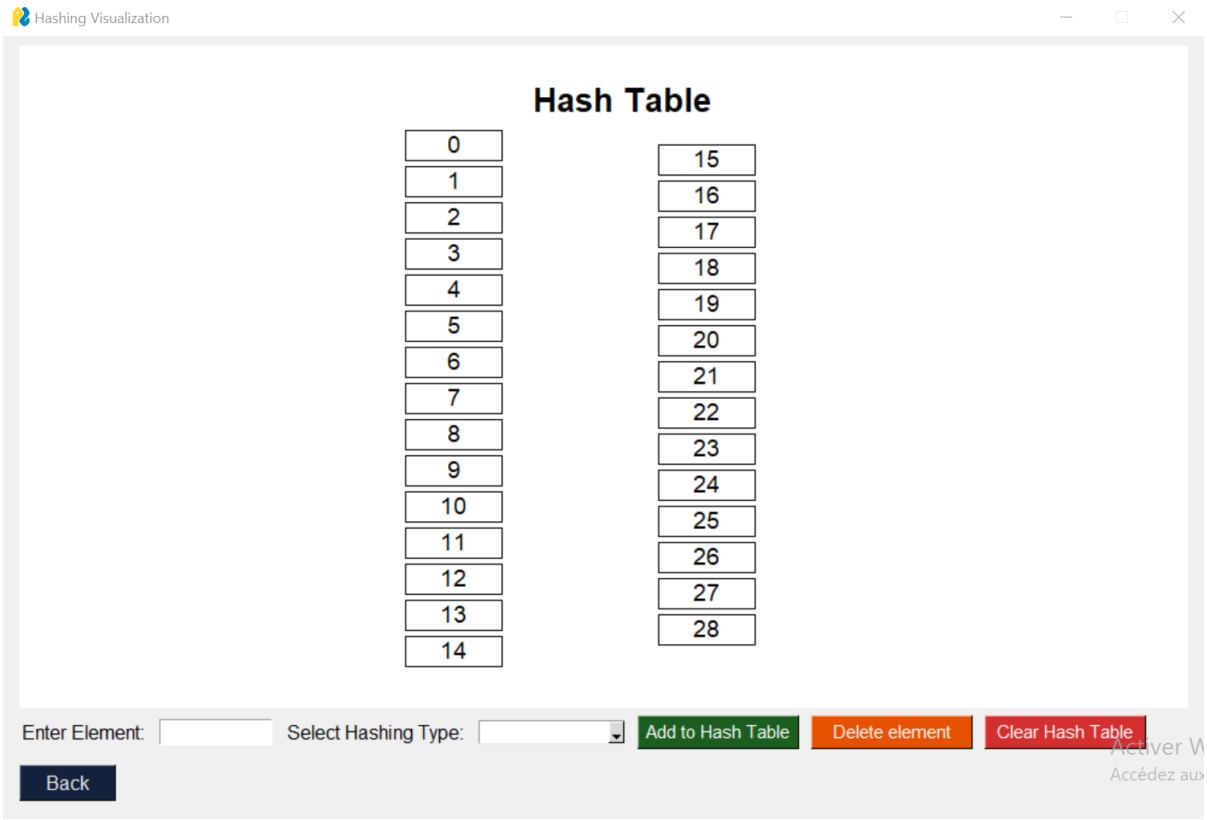

Figure 10 : Interface de Hachage

On a essayé de choisir d'insérer l'élément 1, 5 fois, en choisissant le type d'Hachage linéaire

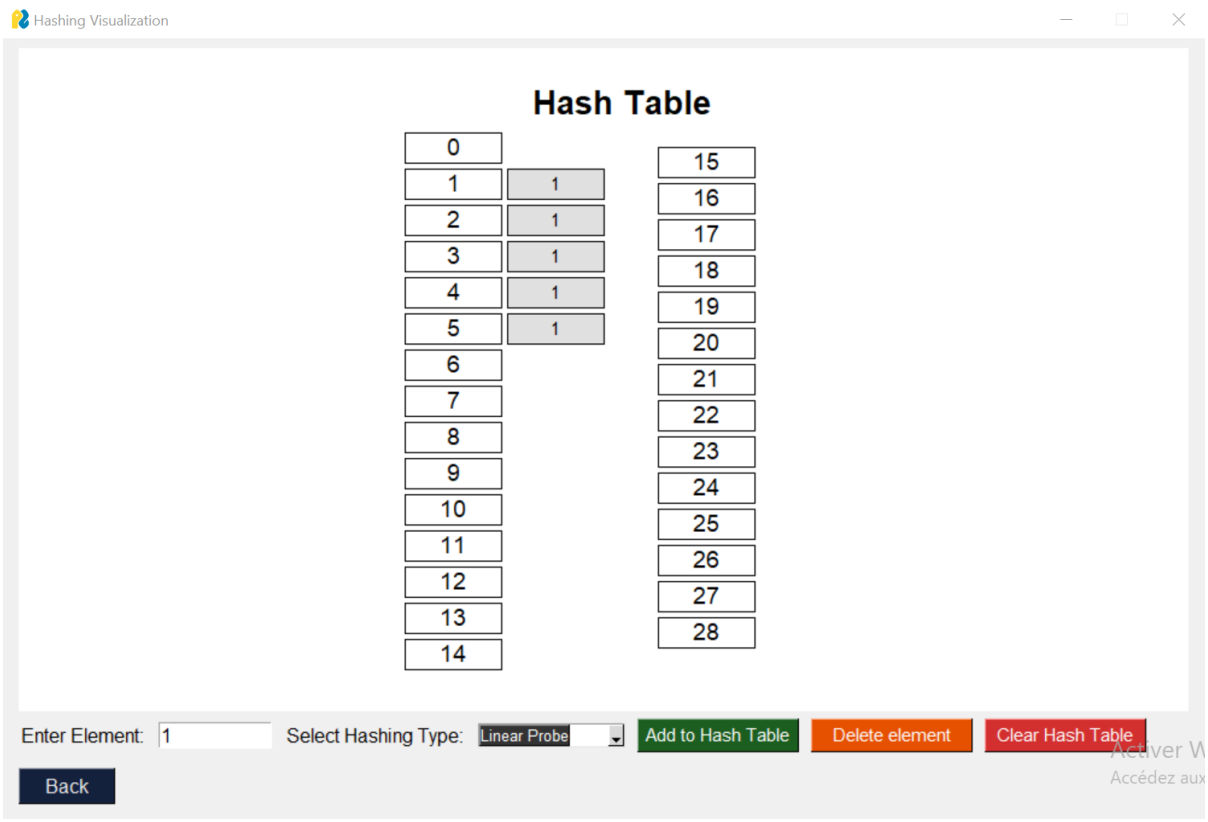

Figure 11 : Visualisation de Hachage lineaire

Ici on a essayé d'ajouter l'élément 1, 4 fois en choisissant le type Double Hachage

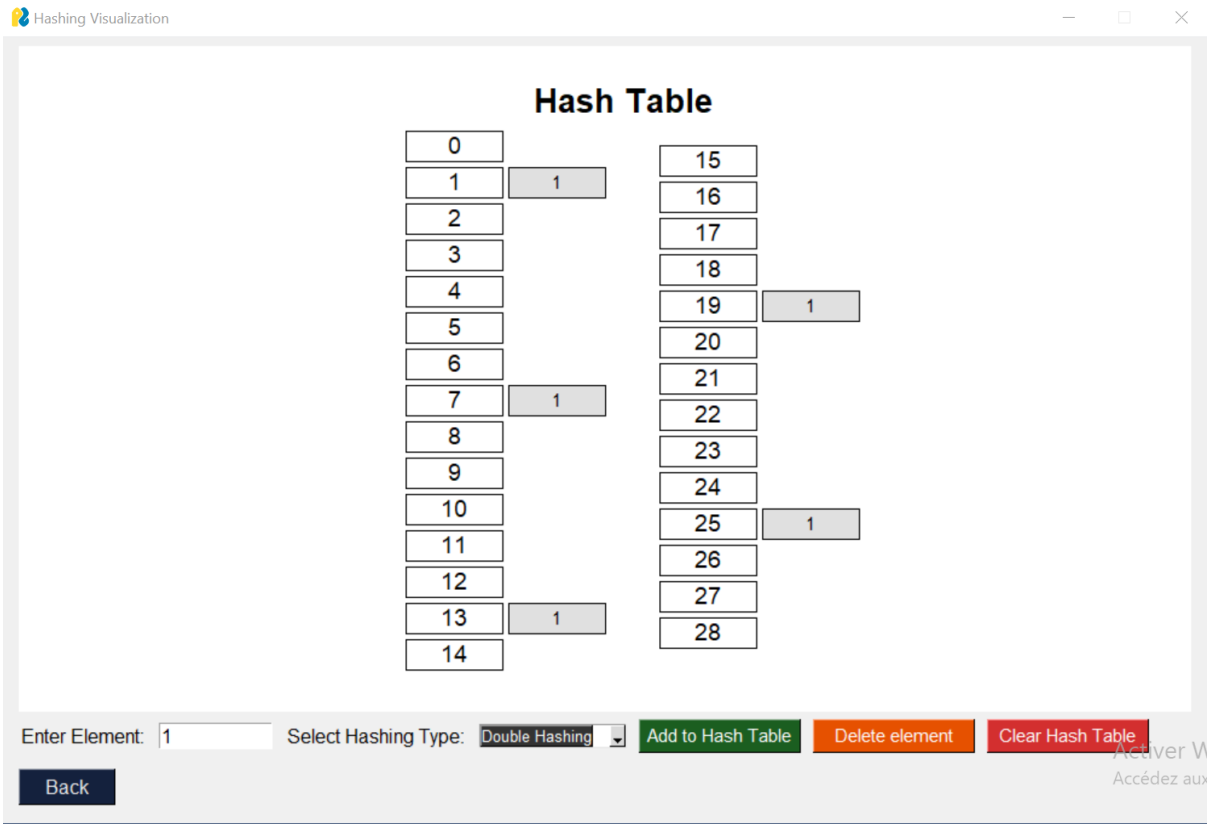

Figure 12 : Visualisation de Double Hachage

En choisissant le type Chainage séparée

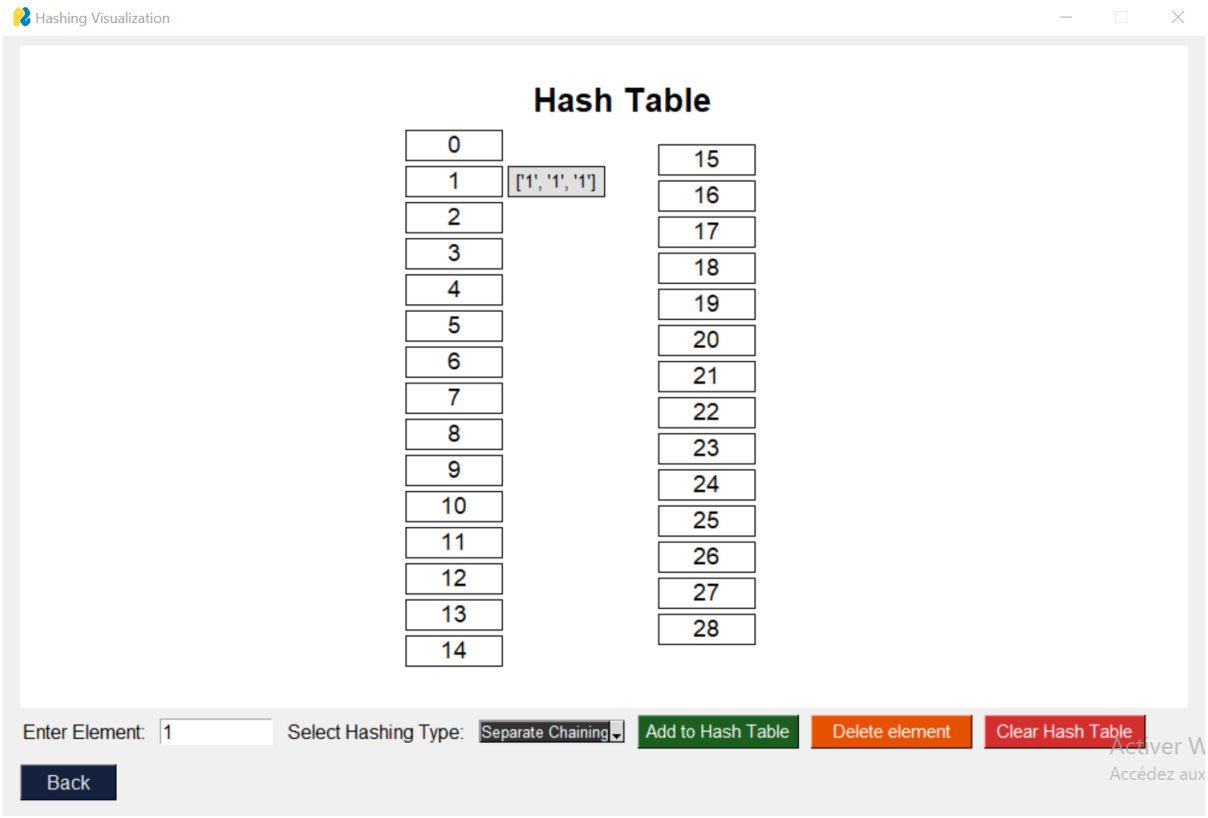

Figure 13 : Interface de Chainage séparée

Et en choisissant le type Chainage Interne

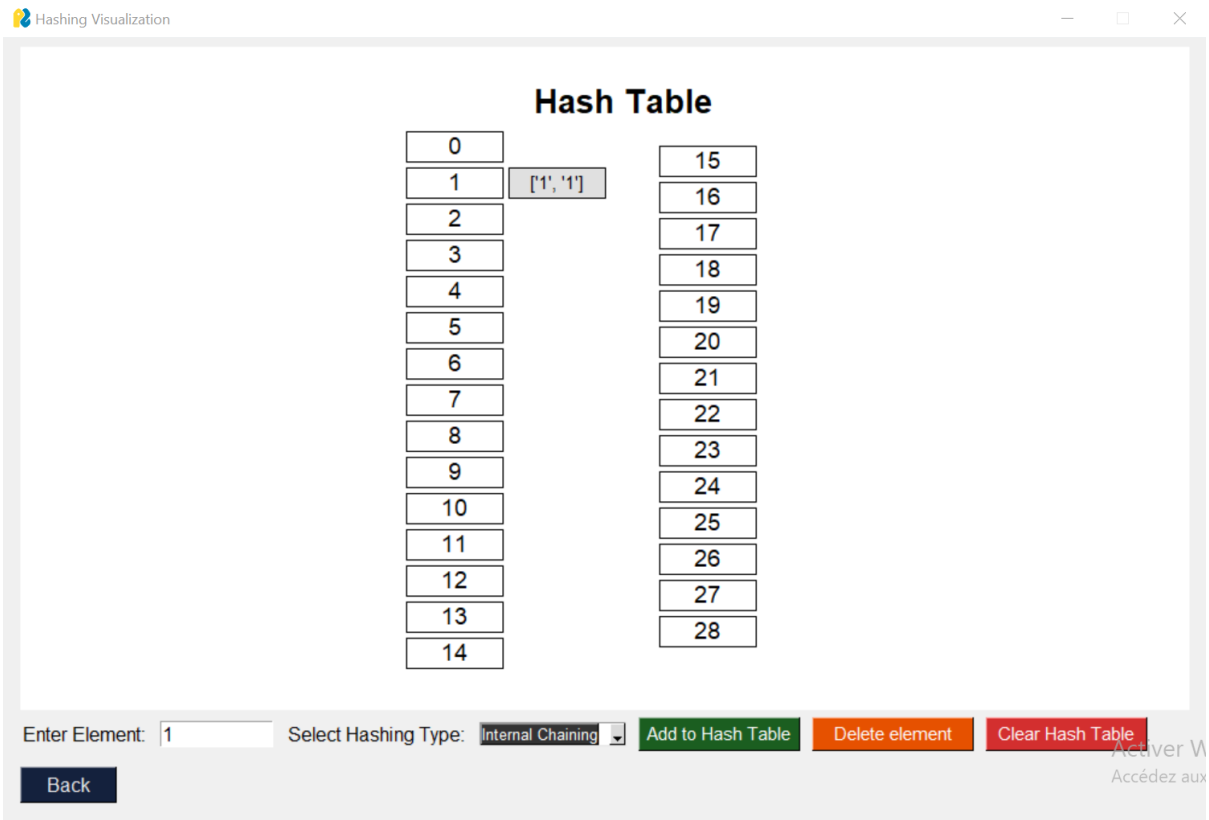

Figure 14 : Interface de Chainage Interne

On peut aussi utiliser les boutons de suppression d'un élément, et un bouton pour vider la table.

Méthodologie de Développement

1. Processus de Développement
   Le développement de l'application a suivi une approche itérative, avec une planification minutieuse, une conception détaillée et des tests rigoureux à chaque étape du processus.

   Arbre B :

   Pour l'implémentation de l'arbre B, voici les paramètres et les équations utilisés :

   Paramètres :

   t : Le degré minimum de l'arbre B. C'est un paramètre crucial qui détermine le nombre maximal de clés qu'un nœud peut contenir avant de devoir être fractionné.

   Équations :

   Lors de l'insertion d'une clé dans l'arbre B, nous vérifions si le nombre de clés dans un nœud dépasse la capacité maximale autorisée avant de le fractionner. Cette capacité maximale est déterminée par
   La formule :

   Nombre maximal de clés dans un nœud = (2 * self.t) - 1

   Si le nombre de clés dans un nœud atteint ou dépasse ce seuil lors de l'insertion, le nœud est fractionné en deux, maintenant ainsi la propriété d'équilibre de l'arbre B.

   Ces paramètres et équations sont utilisés pour garantir que chaque nœud dans l'arbre B reste dans les limites de capacité spécifiées, assurant ainsi une structure équilibrée et des performances efficaces lors des opérations d'insertion, de recherche et de suppression.

   Hachage :

   Dans le cadre de notre processus de développement, nous avons utilisé des éléments spécifiques dans la conception de la classe HashTable pour garantir la gestion efficace des données. Voici les équations et paramètres utilisés :

   Paramètres :

   prime: Nous avons utilisé un nombre premier fixe pour simplifier le processus de double hachage. Ce nombre est défini comme suit :

   Équations :

   - Dans la méthode custom_hash_2(self, element), nous utilisons une équation pour calculer le pas pour le double hachage. L'équation est la suivante :

     step = prime - (int(element) % prime)

     Cette équation permet de calculer le pas en utilisant le nombre premier fixe et l'élément donné.

     Ces éléments ont été soigneusement sélectionnés et intégrés dans notre implémentation de la classe HashTable pour garantir des opérations efficaces de hachage et de gestion des collisions dans notre application.

- Dans la méthode custom_hash(self, element), nous utilisons une équation simple pour calculer le hachage personnalisé d'un élément. L'équation est la suivante :

  return int(element) % self.size

  Cette équation applique l'opération de modulo à la valeur entière de l'élément par la taille de la table de hachage (self.size). Cette opération garantit que le résultat du hachage est un nombre entier compris entre 0 et self.size - 1, ce qui permet de déterminer l'indice où l'élément sera stocké dans la table de hachage.

2. Technologies Utilisées

   Pour l'interface utilisateur, nous avons utilisé la bibliothèque PySimpleGUI, qui offre une manière simple et efficace de créer des interfaces graphiques en Python. Pour le logique métier, nous avons utilisé les fonctionnalités natives de Python.

   Définition générale de la bibliothèque PySimpleGUI :

   PySimpleGUI est un wrapper d'API Python pour le module Tkinter qui permet au programmeur d'utiliser les mêmes éléments d'interface utilisateur qu'avec Tkinter mais avec une interface plus intuitive. PySimpleGUI a été créé dans le but de créer un processus de développement d'interface graphique Python plus convivial et ajoute la possibilité de définir des mises en page personnalisées.

   Avec PySimpleGUI, les développeurs peuvent créer rapidement et facilement des fenêtres, des mises en page et des widgets tels que des boutons, des champs de saisie, des curseurs, etc., en utilisant une syntaxe concise et simple. La bibliothèque abstrait en grande partie la complexité de la programmation GUI, permettant aux développeurs de se concentrer sur la fonctionnalité de leurs applications plutôt que sur les subtilités de l'interface graphique.

Figure 15 : Logo de la bibliothèque PySimpleGUI

Environnement de Développement

L'environnement de développement de ce projet a été soigneusement sélectionné pour assurer une productivité optimale et une collaboration efficace.

- **Visual Studio Code (VSCode) :** Un éditeur de code source développé par Microsoft, reconnu pour sa légèreté, sa rapidité, et sa polyvalence. Il propose une interface utilisateur intuitive, un support robuste pour la gestion de projet, et une compatibilité étendue avec plusieurs langages de programmation.

Figure 16 : Logo de l'éditeur Vscode

- **Git :** Système de contrôle de version distribué conçu pour suivre les modifications du code source pendant le développement logiciel. Git facilite la collaboration en permettant aux développeurs de travailler simultanément sur des projets, de gérer les versions et de fusionner les modifications en toute sécurité.

Figure 17 : Logo de Git

- **GitHub :** Plateforme collaborative de développement basée sur Git. GitHub offre un hébergement de projets, la gestion des problèmes, le suivi des modifications, et facilite la collaboration entre les développeurs. Elle encourage également la contribution communautaire et l'intégration transparente avec d'autres services.

Figure 18 : Logo de Github

Conclusion

En conclusion, l'application de gestion de structures de données développée avec Python et PySimpleGUI offre une solution efficace et conviviale pour manipuler les arbres B et les tables de hachage. Grâce à une conception soigneusement élaborée et une mise en œuvre rigoureuse des fonctionnalités, l'application répond aux besoins des utilisateurs tout en offrant une expérience utilisateur agréable et intuitive. Ce projet démontre la puissance et la polyvalence de Python dans le développement d'applications interactives, ainsi que le potentiel de la bibliothèque PySimpleGUI pour la création d'interfaces utilisateur sophistiquées.

Bibliographie

https://www.cs.usfca.edu/~galles/visualization/BTree.html

https://www.cs.usfca.edu/~galles/visualization/ClosedHash.html

https://docs.pysimplegui.com/en/latest/

https://www.programiz.com/dsa/b-tree

https://docs.python.org/3/library/tk.