جامعة عبد المالك السعدي
Université Abdelmalek Essaâdi

IASD
INTELLIGENCE ARTIFICIELLE ET SCIENCES DE DONNEES

كلية العلوم و التقنيات بطنجة
Faculté des Sciences et Techniques de Tanger

UNIVERSITÈ ABDELMALEK ESSAADI

Master IA et Science de Données

Sujet : Projet Base de Données

Projet:

Mise en œuvre d'algorithmes de hachage et d'indexation dans une interface graphique

Réalisé Par :

MAHRI AYMANE
SABBAHI MOHAMED AMINE

Encadre Par :

Prof. Ezziyyani Mostafa

Année universitaire : 2023/2024

Table of contents:

Table of figures:

1. Introduction :

Dans le cadre de ce projet, nous avons entrepris la tâche passionnante d'implémenter des tables de hachage ainsi que des algorithmes d'indexation au sein d'une interface utilisateur graphique, en utilisant la bibliothèque Customtkinter avec le langage de programmation Python. Ces structures de données et algorithmes, incontournables dans le domaine de la gestion efficace des données, constituent la colonne vertébrale de nombreuses applications informatiques modernes.

Notre objectif principal était de créer une plateforme interactive permettant aux utilisateurs d'explorer et de comprendre les nuances des algorithmes de hachage tels que l'Essai Linéaire, le Hachage Quadratique, le Hachage Double, le Chaînage Interne, le Chaînage Séparé, ainsi que l'Arbre B+. Ces techniques jouent un rôle essentiel dans la recherche, la manipulation et l'organisation optimale des données, offrant des solutions efficaces pour diverses applications, allant des bases de données aux systèmes de gestion de fichiers.

Tout au long de ce rapport, nous détaillerons le processus de conception et d'implémentation de l'interface utilisateur graphique, mettant en lumière les choix techniques faits pour la représentation visuelle des algorithmes. Nous explorerons également en détail chacun des algorithmes implémentés, en fournissant des explications approfondies sur leur fonctionnement, leurs avantages et leurs limitations.

Par cette entreprise, nous cherchons à également à démontrer notre compréhension des concepts fondamentaux liés aux tables de hachage et aux structures d'indexation. À travers cette réalisation, nous espérons offrir une contribution significative à la compréhension et à l'application pratique de ces concepts dans le domaine de l'informatique.

Le rapport qui suit détaille donc notre parcours, de la conception à l'implémentation, offrant un aperçu complet du processus créatif et des défis rencontrés tout au long du développement de cette interface utilisateur graphique interactive.

2. Analyse des besoins et Objectifs :

Dans cette section, nous examinerons en détail les besoins et objectifs du projet, en identifiant les exigences fonctionnelles et non fonctionnelles de l'application de hachage.

Implémentation des Algorithmes de Hachage : L'application doit offrir une implémentation fonctionnelle des différents algorithmes de hachage mentionnés, y compris l'essai linéaire, le hachage quadratique, le double hachage, le chaînage interne, le chaînage séparé et l'arbre B+.

Interface Utilisateur Intuitive : Une interface utilisateur conviviale et intuitive doit être développée pour permettre aux utilisateurs d'interagir facilement avec les fonctionnalités de l'application, y compris l'insertion, la recherche, la suppression et la visualisation des données.

Gestion des Collisions : Une gestion efficace des collisions doit être intégrée dans chaque algorithme de hachage pour garantir la fiabilité et la cohérence des résultats.

Efficacité : Les algorithmes de hachage implémentés doivent être optimisés pour assurer des performances rapides et une utilisation efficace des ressources système.

Extensibilité : L'architecture de l'application devrait être conçue de manière à permettre une extension facile pour l'ajout de nouveaux algorithmes de hachage ou de fonctionnalités supplémentaires à l'avenir.

3. Interface proposée

Cette section présente de manière visuelle et commentée les solutions envisagées pour répondre aux besoins spécifiques du projet. À travers des captures d'écran de notre application développée avec le langage de programmation Python, nous mettons en lumière les différentes fonctionnalités clés et l'interface utilisateur intuitive que nous avons intégré.

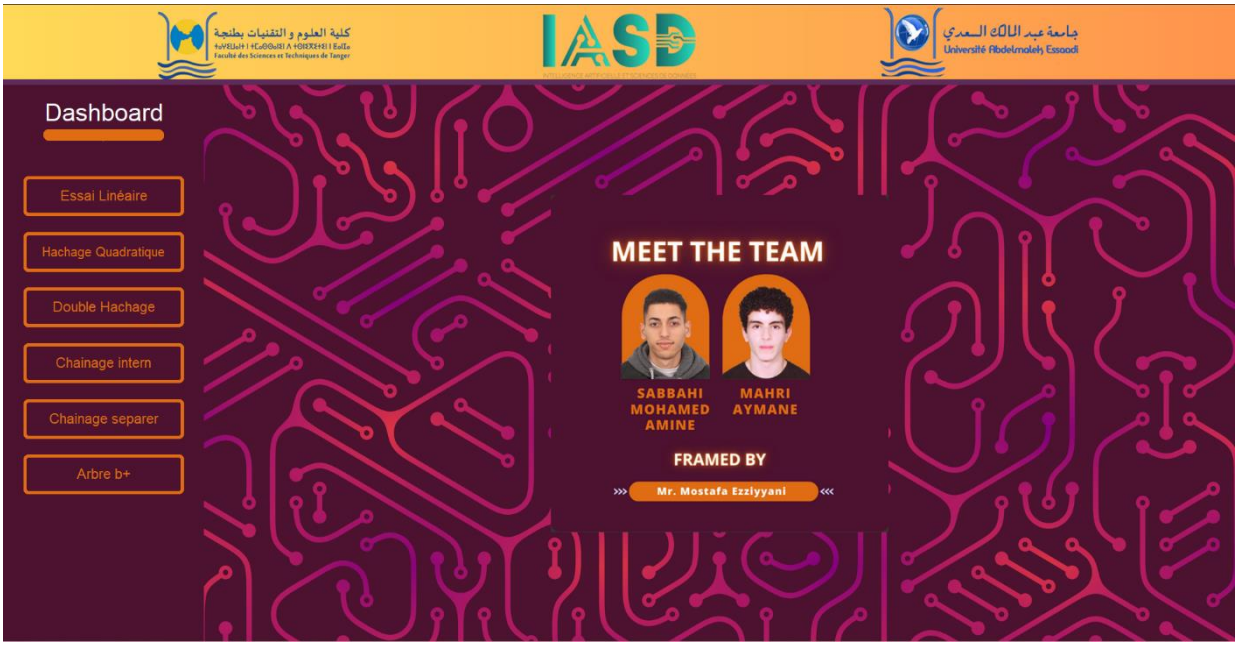


Figure 1: Page d'accueil d'interface graphique

L'interface d'accueil se présente comme un tableau de bord central, offrant une vue d'ensemble de notre application. Dans l'en-tête, elle affiche de manière élégante le nom et les logos de notre Master AISD ainsi que de notre université, tandis que sur la page de contenu, se trouve l'équipe de développement. À gauche de cette interface, les utilisateurs trouveront des boutons d'accès rapide, facilitant ainsi la navigation vers des sections spécifiques de l'application. Cette première capture d'écran propose une mise en page épurée avec des boutons clairement libellés, invitant les utilisateurs à explorer les fonctionnalités de l'application de manière intuitive.

Ces boutons, soigneusement agencés, dirigent vers des interfaces dédiées pour l'implémentation et le fonctionnement des différents algorithmes de hachage, tels que l'essai linéaire, le hachage quadratique, le hachage double, le chaînage interne et externe, ainsi que l'arbre B+. L'objectif de cette interface d'accueil est de simplifier la navigation et de fournir un point de départ convivial pour les utilisateurs, les guidant efficacement vers les différentes fonctionnalités de l'application.

4. Table de hachage

Une table de hachage est une structure de données qui permet de stocker et de récupérer des éléments de manière efficace en associant chaque élément à une clé unique. Lorsque plusieurs éléments ont la même clé de hachage (collision), différentes techniques sont utilisées pour gérer ces collisions. Voici une explication de ces techniques :

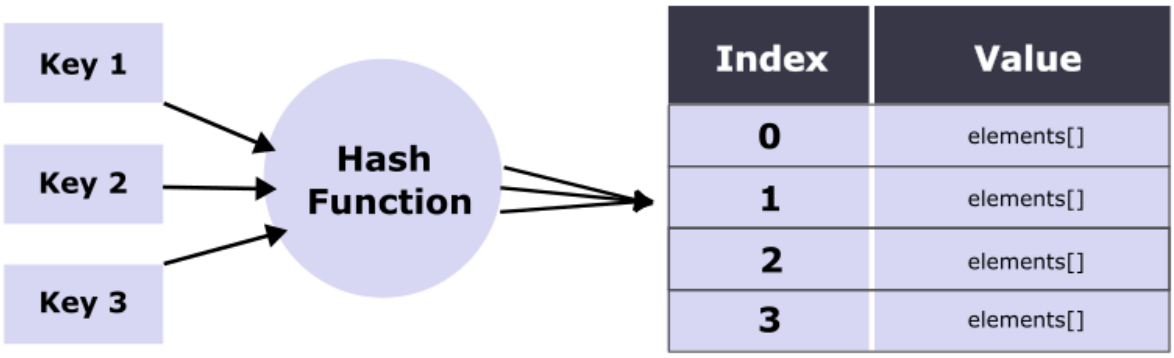


Figure 2: Tbale de hachage

- **Chaînage :** Dans cette approche, chaque case de la table de hachage contient une liste chaînée. Lorsqu'une collision se produit, l'élément en collision est ajouté à la liste chaînée correspondante à cette case. Ainsi, plusieurs éléments peuvent être associés à la même case sans problème.
- **Essai linéaire :** Aussi appelé "essai direct" ou "Linear Probing", cette technique consiste à rechercher linéairement dans la table de hachage pour trouver un emplacement vide lorsqu'une collision se produit. L'élément en collision est alors inséré à cet emplacement. Cette méthode peut souffrir du problème de clustering si plusieurs collisions successives se produisent.
- **Essai quadratique :** Similaire à l'essai linéaire, mais au lieu de parcourir linéairement les emplacements de la table de hachage, cette technique utilise une fonction quadratique pour déterminer les emplacements à vérifier en cas de collision. Cela aide à réduire le risque de clustering par rapport à l'essai linéaire.
- **Double hachage :** Cette méthode utilise une deuxième fonction de hachage pour calculer un décalage supplémentaire en cas de collision. En utilisant cette approche, les éléments en collision sont placés dans des emplacements différents de manière déterministe, réduisant ainsi le risque de clustering.

Chacune de ces techniques a ses avantages et inconvénients en termes d'efficacité et de gestion des collisions. Le choix de la méthode dépend souvent des besoins spécifiques de l'application et des caractéristiques des données à stocker.

5. Essai linéaire

5.1 Principe d'essai linéaire :

Le sondage linéaire est l'un des nombreux algorithmes conçus pour trouver la position correcte d'une clé dans une table de hachage. Lors de l'insertion des clés, nous atténuons les collisions en parcourant séquentiellement les cellules de la table. Une fois que nous trouvons la prochaine cellule disponible, nous y insérons la clé. De même, pour trouver un élément dans une table de hachage, nous parcourons linéairement les cellules jusqu'à ce que nous trouvions la clé ou que toutes les positions aient été parcourues.

Supposons maintenant que notre table de hachage ait une longueur arbitraire de 6 et que nous souhaitions insérer les éléments restants de [40, 310, 74, 9, 2]

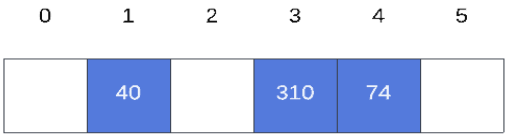


Figure 3: Table de hachage de taille 6

Selon notre fonction h(a), 9 sera inséré à l'index 5, tandis que le dernier élément de l'ensemble qui est 2 sera inséré à l'index 3 :

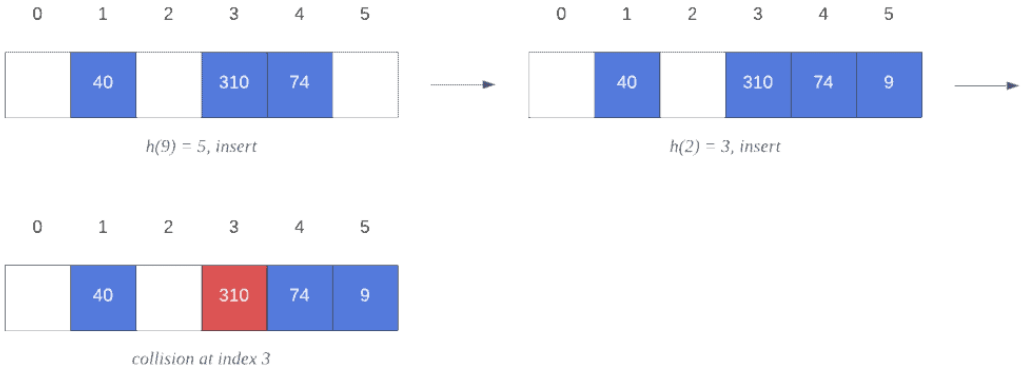


Figure 4: Insertion d'élément dans l'essai linéaire

Une fois que nous essayons d'insérer 2 dans notre table de hachage, nous rencontrons une collision avec la clé 310. Cependant, comme nous utilisons le sondage linéaire comme algorithme de résolution de collision, notre table de hachage aboutit à l'état suivant après avoir inséré tous les éléments dans [40, 310, 74, 9, 2]

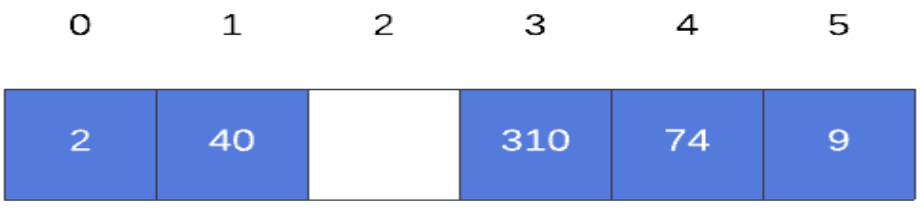


Figure 5: Cas de collision dans l'essai linéaire

5.2 Example d'implémentation dans l'interface graphique :

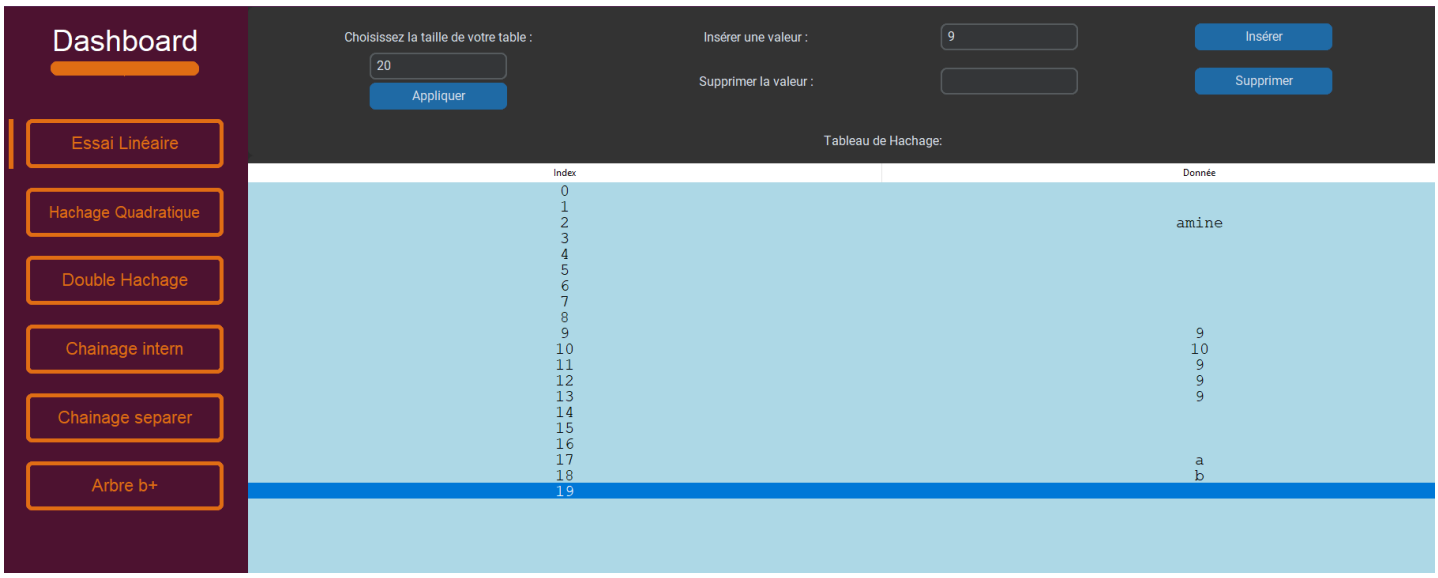


Figure 6: La page d'essai lineaire

Lorsque vous cliquez sur le bouton de sondage linéaire, l'application vous dirigera vers la page associée. En haut de la page, vous trouverez un tableau de contrôle où vous pourrez spécifier la taille de la table de hachage, ainsi que les éléments à insérer et à supprimer. Sous ce tableau de contrôle, vous trouverez la table de hachage présentée en deux colonnes : l'indice et sa valeur correspondante.

Notre table de hachage est capable de stocker non seulement des valeurs entières, mais aussi des chaînes de caractères. Dans le cas des chaînes de caractères, la clé est calculée en additionnant les codes ASCII de tous les caractères qui composent la chaîne. Ainsi, notre structure de données offre une flexibilité accrue pour stocker et manipuler différents types de données.

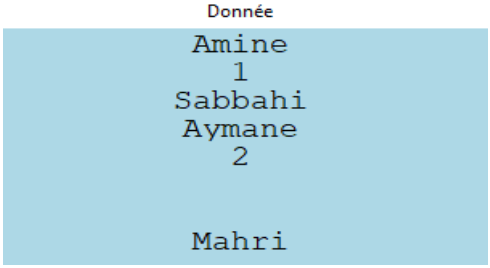


Figure 8: insertion des chaines de caracters

Figure 7: example essai lineaire

Pour illustrer le fonctionnement de notre table de hachage, prenons l'exemple suivant. Nous commençons avec une table de hachage d'une taille de 10, et tentons d'insérer les éléments 3-4-3-3-3-9.

Lors de l'insertion, le premier 3 sera placé à l'index 3 et le 4 sera placé au quatrième index. Cependant, lors de l'insertion du deuxième 3, une collision se produit, car l'index 3 est déjà occupé. Dans ce cas, l'algorithme de sondage linéaire cherchera le prochain espace libre dans la table pour insérer la valeur 3, et ainsi de suite pour tous les autres éléments.

6. Hachage quadratique :

6.1 Principe de hachage quadratique :

Le hachage par sondage quadratique est une technique de résolution de collision utilisée dans les tables de hachage. Contrairement au sondage linéaire où les éléments en collision sont recherchés de manière linéaire, le hachage par sondage quadratique utilise une fonction quadratique pour déterminer les emplacements à vérifier en cas de collision.

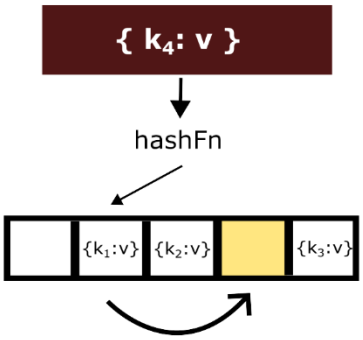


Figure 9: Hachage quadratique

Voici comment fonctionne le processus de hachage par sondage quadratique :

1- Lorsqu'une collision se produit lors de l'insertion d'un élément dans la table de hachage, au lieu de simplement parcourir les emplacements de manière linéaire, une fonction quadratique est utilisée pour calculer de nouveaux emplacements à vérifier.

2- La fonction quadratique peut être de la forme f(i)=(h(k) + c1 ∗ i + c2 ∗ i**2) modN, où h(k) est la fonction de hachage initiale, i est le numéro de l'itération de la recherche, c1 et c2 sont des constantes, et N est la taille de la table de hachage.

3- À chaque itération, le nouvel emplacement est calculé en ajoutant c1 ∗ i + c2 ∗ i**2 à l'emplacement initial h(k), puis en prenant le résultat modulo N.

4- Si l'emplacement calculé est déjà occupé, une nouvelle itération est effectuée en incrémentant i et en recalculant l'emplacement. Ce processus est répété jusqu'à ce qu'un emplacement vide soit trouvé.

Le hachage par sondage quadratique aide à réduire le phénomène de clustering observé dans le sondage linéaire en répartissant les éléments de manière plus uniforme dans la table de hachage. Cependant, il peut également souffrir de limitations similaires, notamment en cas de forte saturation de la table de hachage.

6.2 Example d'implémentation dans l'interface graphique :

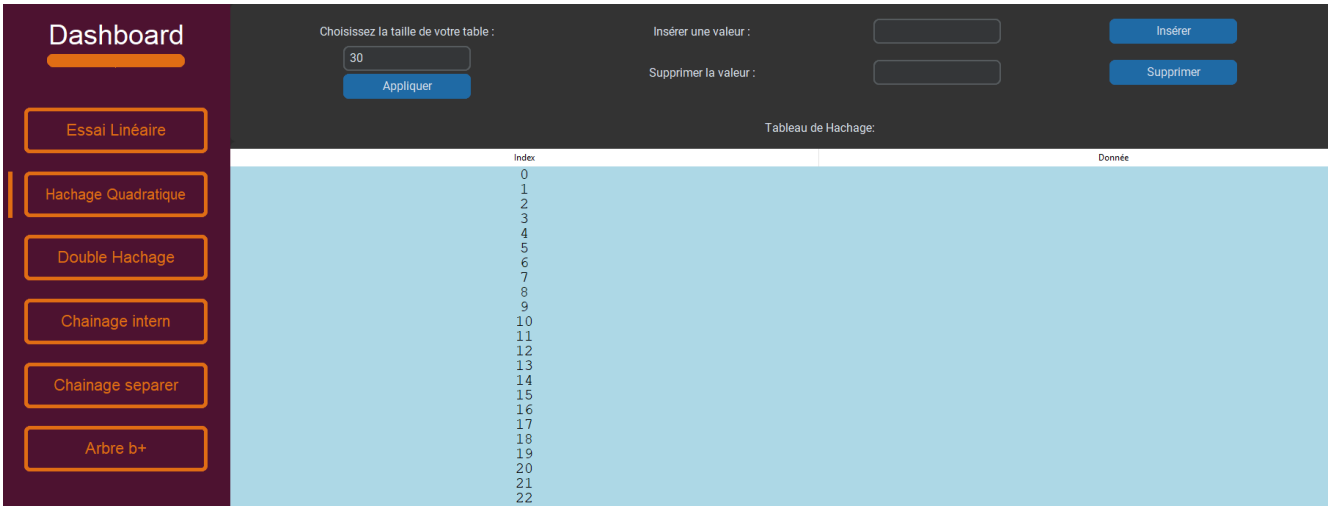


Figure 10: Page de hachage quadratique

Similaire à celui observé dans la page du sondage linéaire, lorsque vous cliquez sur le bouton de hachage quadratique, l'application vous redirigera vers la page correspondante. Il convient de noter que cette table de hachage supporte également les données de type chaînee de caracters.

Essayons maintenant de créer un exemple dans notre interface en utilisant l'algorithme de hachage quadratique avec une table de taille 30. Insérons plusieurs fois la même valeur, par exemple 33, afin de provoquer une collision.

Figure 11: Example d'implementation de hachage quadratique

Lors de l'insertion initiale de la valeur 33 dans la table de hachage, elle sera placée à un certain index calculé en utilisant la fonction de hachage quadratique. Cependant, lors des insertions ultérieures de la même valeur, une collision se produira, car la même fonction de hachage quadratique donnera le même index pour cette valeur.

L'algorithme de hachage quadratique tentera alors de trouver un nouvel emplacement pour insérer la valeur en utilisant une séquence quadratique d'indices. Il commencera par essayer l'indice suivant, puis l'indice suivant après le carré de l'itération, puis le suivant après le cube, et ainsi de suite. Ce processus se poursuivra jusqu'à ce qu'un emplacement vide soit trouvé.

7. Double hachage :

7.1 Principe de hachage quadratique :

Le hachage double, également connu sous le nom de double hachage, est une technique de résolution de collision utilisée dans les tables de hachage. Contrairement à d'autres méthodes telles que le chaînage ou le sondage linéaire, le hachage double utilise deux fonctions de hachage distinctes pour calculer les emplacements potentiels où insérer un élément en cas de collision.

Voici comment fonctionne le processus de hachage double :

1- Lorsqu'une collision se produit lors de l'insertion d'un élément dans la table de hachage, une deuxième fonction de hachage est utilisée pour calculer un décalage supplémentaire à partir de l'emplacement initial.

2- Cette deuxième fonction de hachage est généralement conçue pour produire des valeurs différentes de la première fonction de hachage, garantissant ainsi une distribution plus uniforme des éléments dans la table de hachage.

3- Si l'emplacement calculé à l'aide de la deuxième fonction de hachage est également occupé, un décalage supplémentaire est calculé en utilisant à nouveau la deuxième fonction de hachage, et ce processus est répété jusqu'à ce qu'un emplacement vide soit trouvé.

La formule de double hachage pour trouver un seau ouvert ou un élément particulier déjà placé dans la table de hachage est la suivante :

$$i = \big(H(k) + x * J(k)\big)\ mod\ N \quad \text{où:}$$

i represent l'indice du table.

N est la taille de la table.

H est la fonction de hachage.

k est la clé de l'élément à hacher.

J est la deuxième fonction de hachage.

Le double hachage aide à réduire les risques de clustering et à améliorer la distribution des éléments dans la table de hachage, ce qui peut conduire à de meilleures performances globales. Cependant, il est important de choisir judicieusement les deux fonctions de hachage pour éviter les collisions excessives et garantir une répartition efficace des éléments.

7.2 Example d'implémentation dans l'interface graphique :

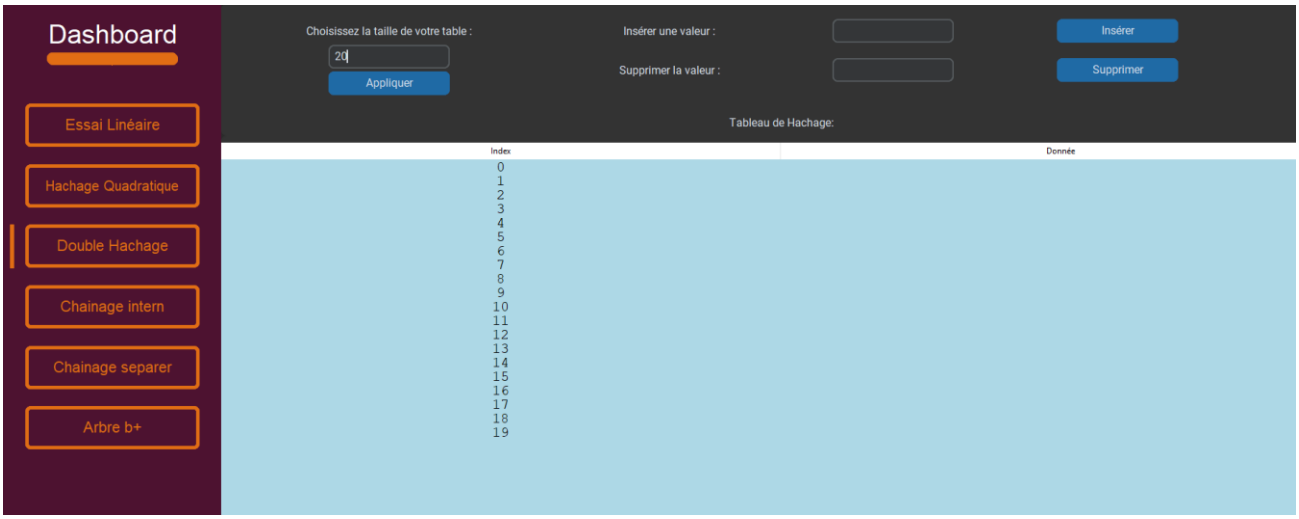


Figure 12: La page de double hachage

Lorsque vous cliquez sur le bouton de double hachage, l'application vous conduira vers la page dédiée à cette méthode. En haut de cette page, vous trouverez un tableau de configuration où vous pourrez spécifier la taille de la table de hachage ainsi que les éléments à ajouter ou à retirer. Sous ce tableau de configuration, la table de hachage sera affichée avec deux colonnes distinctes une colonne pour les indices et une autre pour les valeurs correspondantes.

Créons maintenant un exemple dans notre interface en utilisant l'algorithme de double hachage. Nous allons insérer la chaîne "AISD" plusieurs fois afin de provoquer un cas de collision.

Figure 13: Example double hachage

Avec le double hachage, lorsqu'une collision se produit, une seconde fonction de hachage est utilisée pour calculer un décalage supplémentaire à partir de la position initiale. La valeur est alors insérée dans la position obtenue après le second calcul de hachage.

Dans ce cas la première insertion de "AISD" est affectée à l'indice 14, et qu'une collision se produit lors de l'insertion suivante de "AISD", le double hachage recalculera l'indice en utilisant la seconde fonction de hachage, puis placera la valeur à cet emplacement. Ce processus se répète jusqu'à ce qu'une position vide soit trouvée.

8. Chainage interne :

8.1 Principe de chainage imterne :

Le chaînage interne est un principe fondamental dans la gestion des bases de données relationnelles, où les liens entre les enregistrements sont utilisés pour naviguer efficacement à travers les données. Ce principe repose sur l'utilisation de références ou de liens entre les enregistrements pour créer des relations logiques entre les données stockées dans différentes tables de la base de données.

Structure du Chaînage Interne : Dans le chaînage interne, chaque enregistrement ou tuple contient des références ou des liens vers d'autres enregistrements de la même table ou d'autres tables de la base de données. Ces liens peuvent être établis via des clés primaires et des clés étrangères, assurant ainsi des relations logiques entre les données.

Insertion de Données : Lors de l'insertion d'un nouvel enregistrement, ses références vers d'autres enregistrements sont définies en fonction de la logique métier et des contraintes de clés étrangères. Si un nouvel enregistrement doit être lié à un enregistrement existant, ses références sont mises à jour pour inclure les identifiants de ces enregistrements.

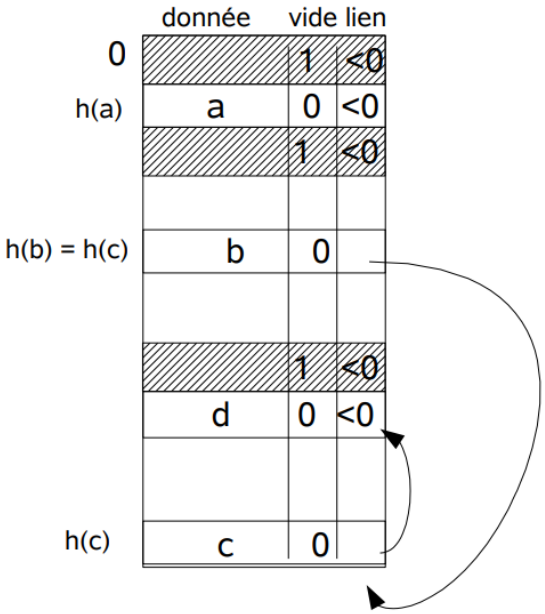


Figure 14: chainage interne

Suppression de Données : Lors de la suppression d'un enregistrement, les références vers cet enregistrement sont également supprimées ou mises à jour pour refléter la suppression. Les opérations de suppression peuvent nécessiter une gestion spéciale pour maintenir l'intégrité référentielle et éviter les incohérences dans les données.

Avantages du Chaînage Interne : Le chaînage interne permet une navigation efficace à travers les données, en minimisant le temps nécessaire pour accéder aux données connexes. Il facilite la gestion des relations entre les enregistrements et maintient l'intégrité référentielle de la base de données.

Limitations : Une mauvaise gestion des références peut entraîner des incohérences dans les données et des problèmes d'intégrité référentielle. Les opérations de mise à jour et de suppression peuvent être complexes et nécessiter une attention particulière pour éviter les erreurs.

8.2 Example d'implémentation dans l'interface graphique :

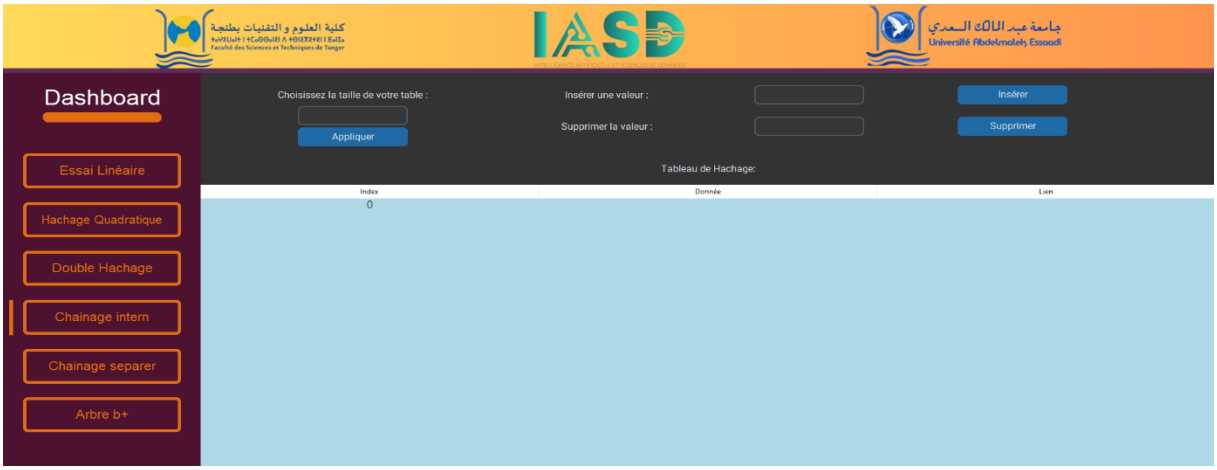


Figure 15: Page de chainage interne

On commence tout d'abord par initialiser la taille de notre table de hashage, dans ce cas on choisi la taille 10. Ensuite nous allons insérer les valeurs suivantes : 1 – 2 – 3 – 6

Index	Donnée	Lien
0		
1	1	
2	2	
3	3	
4		
5		
6	6	
7		
8		
9		

Figure 16: Insertion dans chainage interne

Chaque valeur entrée prend sa place dans ta table selon l'index donnée après le hashage de la valeur, nous allons maintenant insérer la valeur 11 qui possède le meme index que la valeur 1.

Index	Donnée	Lien
0		
1	1	4
2	2	
3	3	
4	11	1
5		
6	6	
7		
8		
9		

Figure 17: collision dans chainage interne

Nous remarquons que la valeur 11 prend la première place libre dans la table après la valeur 1 et on remaque aussi que le lien de la valeur 11 fait référence à sa place initiale (1) et meme chose pour la valeur 1 qui fait reference à la place de la nouvelle valeur insérée (4).

Suppimons maintenant la valeur 11 :

Index	Donnée	Lien
0		
1	1	4
2	2	
3	3	
4		
5		
6	6	
7		

Figure 18:Suppression dans chainage interne

La suppression se fait de manière rapide et sans problème !

9. Chaînage Séparé :

9.1 Principe de chaînage séparé :

Le chaînage séparé est une technique de gestion des collisions couramment utilisée dans les structures de données telles que les tables de hachage, notamment dans le contexte des bases de données. Contrairement au chaînage interne où les collisions sont gérées en ajoutant des éléments à la suite des autres dans une même liste, le chaînage séparé consiste à stocker les éléments ayant des clés de hachage identiques dans des structures de données distinctes, souvent des listes chaînées.

Structure du Chaînage Séparé : Dans le chaînage séparé, les enregistrements ou tuples qui partagent la même valeur de hachage sont stockés dans des listes chaînées distinctes. Chaque élément de la table de hachage contient une référence vers le début de la liste chaînée des enregistrements ayant la même valeur de hachage.

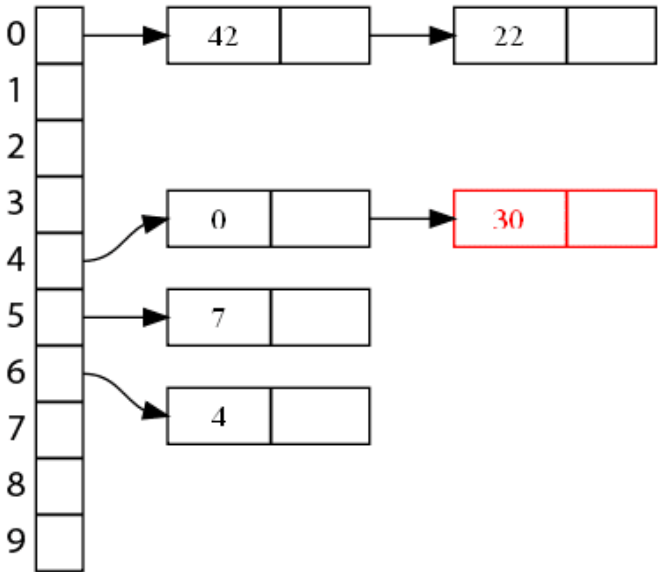


Figure 19: Chaînage séparé

Insertion de Données : Lors de l'insertion d'un nouvel enregistrement, son hachage est calculé et utilisé pour déterminer la position dans la table de hachage. Si la case correspondante de la table de hachage est vide, un nouveau nœud de liste chaînée est créé pour cet enregistrement. Si la case de la table de hachage est occupée, l'enregistrement est ajouté à la liste chaînée existante à cette position.

Suppression de Données : Lors de la suppression d'un enregistrement, le système calcule le hachage de la clé de l'enregistrement à supprimer et accède à la case correspondante de la table de hachage. Ensuite, il parcourt la liste chaînée associée pour localiser l'enregistrement à supprimer et le retire de la liste.

Avantages du Chaînage Séparé : Le chaînage séparé est efficace pour gérer les collisions dans les tables de hachage, car il permet de stocker un nombre arbitraire d'enregistrements ayant la même valeur de hachage. Il offre une recherche efficace lorsque les collisions sont rares, car la complexité de la recherche est généralement O(1) dans le meilleur des cas. Il est simple à mettre en œuvre et ne nécessite pas de redimensionnement fréquent de la table de hachage.

Limitations : La performance du chaînage séparé peut se dégrader si les collisions sont fréquentes, car cela peut entraîner des listes chaînées très longues, ce qui ralentit les opérations de recherche. La gestion des collisions peut être plus complexe, en particulier lorsque la distribution des clés de hachage n'est pas uniforme.

9.2 Example d'implémentation dans l'interface graphique :

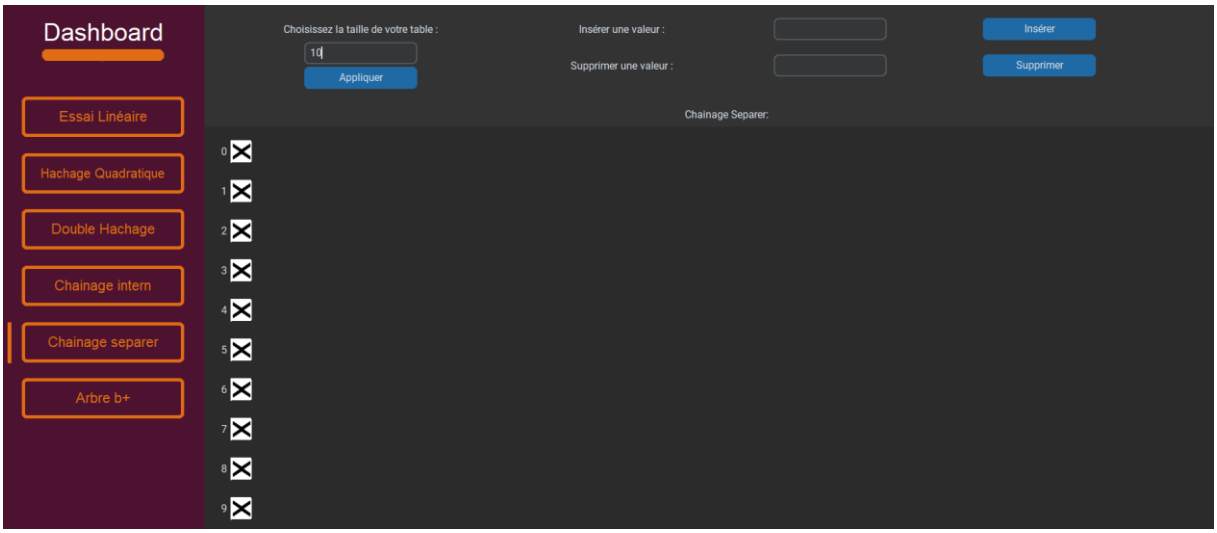


Figure 20: Page de chainage separe

Tous d'aborde on Initialison la taille de notre table, dans ce cas nous choissirons la taille 10, et on Insérons les valeurs 3-4-7 dans notre table :

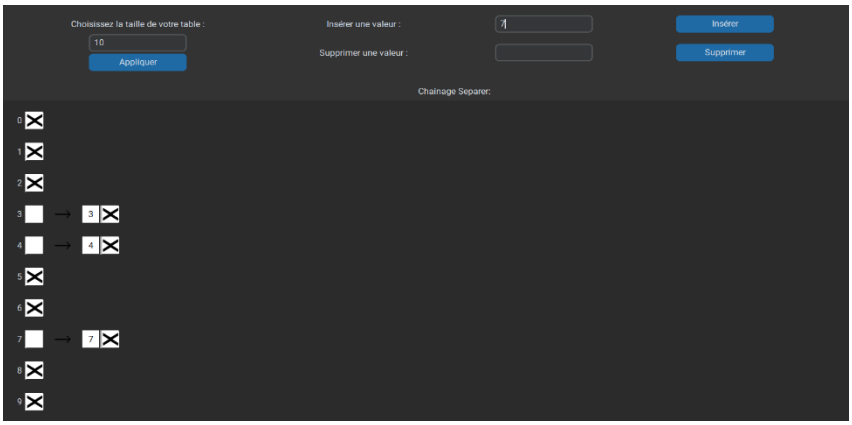


Figure 21: insertion dans chainage separe

Les valeurs sont bel et bien insérées dans leur case correcte selon leur valeur de hachage, on remarque aussi la création d'une liste chainée. Insérons maintenant les valeurs 13-23-24-17 :

De meme les valeurs sont bel et bien insérée correctement selon leur valeur de hachage, suivant une liste chainée.

Supprimons maintenant la valeur 7-23-4 :

Figure 22: Suppression dans chainage separe

La suppression se fait de manière efficace et convenable ! De meme pour le reste des valeurs 3 et 13 :

10. Arbre B+ :

10.1 Principe d'arbre B+ :

Les arbres B+ sont une structure de données hiérarchique utilisée principalement dans les systèmes de gestion de bases de données pour organiser et stocker des données de manière efficace.

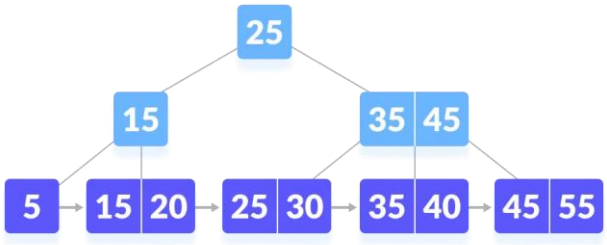


Figure 23: Arbre B+

Structure des Arbres B+ : Les arbres B+ sont des arbres équilibrés constitués de nœuds. Chaque nœud, à l'exception de la racine, contient au moins (m-1) clés, où m est le degré de l'arbre. Les clés dans chaque nœud sont ordonnées de manière ascendante. Les nœuds internes agissent comme des index pour les nœuds feuilles, qui contiennent les données proprement dites. Tous les nœuds feuilles sont à la même profondeur, ce qui facilite la recherche.

Insertion de Données : Lors de l'insertion d'une nouvelle donnée, l'arbre est parcouru depuis la racine jusqu'à atteindre le nœud feuille approprié. Si le nœud feuille n'est pas plein, la nouvelle donnée est simplement insérée à sa place dans le nœud feuille. Si le nœud feuille est plein après l'insertion, il est scindé en deux, et la clé médiane est promue au nœud parent. Cette opération peut se propager récursivement vers le haut de l'arbre si nécessaire.

Suppression de Données : Lors de la suppression d'une donnée, l'arbre est parcouru depuis la racine jusqu'au nœud feuille contenant la donnée à supprimer. Si la donnée à supprimer est trouvée dans un nœud feuille, elle est simplement retirée de ce nœud. Si cela entraîne une sous-capacité dans le nœud feuille, des ajustements sont effectués pour rétablir l'équilibre de l'arbre en fusionnant ou en redistribuant les clés entre les nœuds voisins.

Avantages des Arbres B+ : Les arbres B+ offrent une recherche efficace et des performances cohérentes pour une grande quantité de données. Ils sont adaptés pour les bases de données en raison de leur capacité à optimiser les opérations d'E/S et à faciliter les requêtes par intervalle. Leur structure équilibrée garantit une répartition uniforme des données, réduisant ainsi les risques de goulets d'étranglement.

Limitations : La complexité de l'implémentation et de la maintenance des arbres B+ peut être plus élevée par rapport à d'autres structures de données plus simples.

Les performances peuvent être affectées en cas de fragmentation ou de saturation de la mémoire principale.

10.2 Exemple d'implémentation :

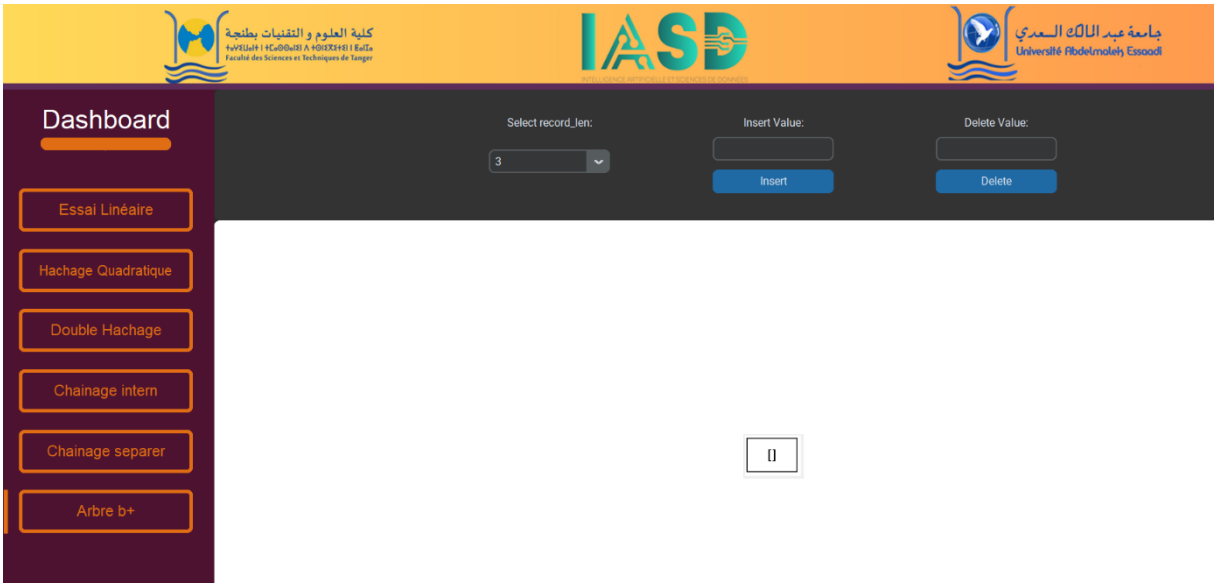
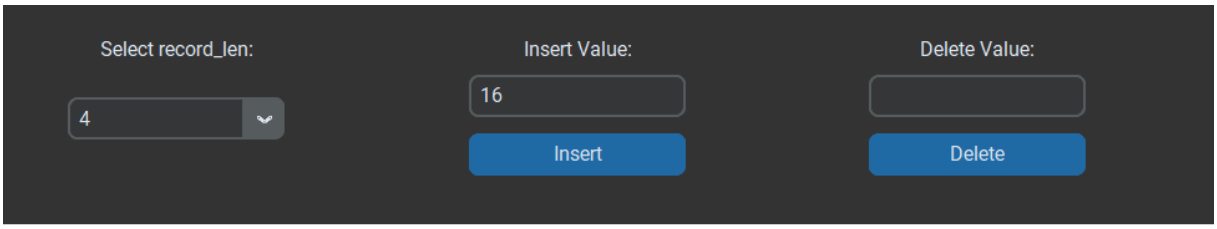


Figure 24: Page d'arbre B+

Commencons tout d'abord par initialiser le degrée de notre arbre b+, dans ce cas nous choisissons le degrée 4, apres on Insérons les valeurs suivantes dans l'arbre 1–30–8-11–15-2-16-16 :



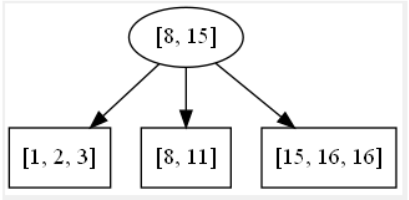
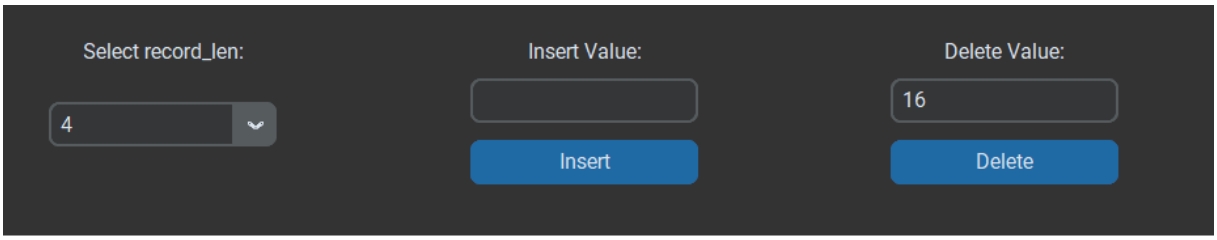


Figure 25: Insertion dans l'arbre B+.

L'insertion a été effectuée avec succès, suivant le principe de l'arbre B+, sans rencontrer de problèmes.

Essayons maintenant de supprimer les valeurs suivantes 3 - 8 - 16 -16 – 11 :



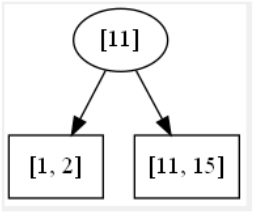


Figure 26: Suppression dans l'arbre B+

Et donc la suppression marche aussi de manière efficace et sans problème !

11. Technologies utilisées et *outils de développement :* :

11.1 Langage de programmation Python :

Figure 27: Logo de Python

Python est un langage de programmation interprété de haut niveau reconnu pour sa lisibilité et sa polyvalence. Python est largement utilisé dans des domaines tels que le développement web, la science des données et l'intelligence artificielle, grâce à sa simplicité et sa vaste bibliothèque.

11.2 Customtkinter :

Figure 28: Logo de Customtkinter

CustomTkinter est une extension de la bibliothèque Tkinter en Python, conçue pour améliorer les capacités esthétiques des interfaces graphiques utilisateur (GUIs) basées sur Tkinter. Elle offre des widgets supplémentaires, des options de style et des fonctions utilitaires, simplifiant ainsi la création d'interfaces modernes et conviviales pour les applications desktop.

11.3 Visual studio code :

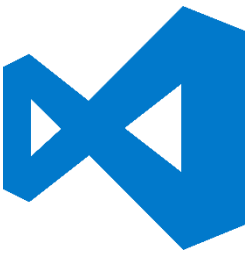


Figure 29: logo Visual studio code

Visual Studio Code est un éditeur de code source qui peut être utilisé avec une variété de langages de programmation, notamment Java, JavaScript, Python,Go, Node. js et C++. Il est basé sur le cadre Electron, qui est utilisé pour développer des applications Web Node. js qui s'exécutent sur le moteur de présentation Blink.

11.4 Git :

Figure 30: Logo de Git

Git est un système de contrôle de version décentralisé largement utilisé dans le développement de logiciels. Il permet de suivre les modifications du code source, de gérer les différentes versions du projet et de faciliter la collaboration entre les développeurs en permettant le partage et la fusion des modifications.

11.5 Git Hub :

Figure 31: Logo de Git Hub

GitHub est une plateforme d'hébergement de code basée sur Git. Elle offre des fonctionnalités supplémentaires telles que le suivi des problèmes, la gestion des projets et la possibilité de collaborer avec d'autres développeurs. GitHub facilite également le partage et la découverte de projets open source.

12. Conclusion Générale:

Le développement de cette application de hachage a été une entreprise passionnante et enrichissante, offrant une exploration approfondie des différents algorithmes et techniques utilisés pour organiser les données. À travers ce rapport, nous avons examiné les objectifs, les exigences et les réalisations de ce projet.

L'analyse des besoins et des objectifs a permis de définir clairement les fonctionnalités essentielles de l'application, notamment l'implémentation des algorithmes de hachage, une interface utilisateur conviviale . Cette analyse a été cruciale pour orienter le développement de l'application et assurer sa pertinence et son utilité pour les utilisateurs.

La mise en œuvre des différents algorithmes de hachage a été un processus complexe mais gratifiant, nécessitant une compréhension approfondie des principes sous-jacents ainsi qu'une expertise en programmation et en conception logicielle. Chaque algorithme a été soigneusement développé et optimisé pour garantir des performances optimales tout en répondant aux exigences de sécurité et de fiabilité.

Enfin, ce projet ne se termine pas avec la conclusion de ce rapport. Des opportunités d'amélioration subsistent, notamment l'exploration de nouveaux algorithmes de hachage, l'optimisation des performances et l'ajout de fonctionnalités supplémentaires pour enrichir l'expérience utilisateur. Avec un engagement continu envers l'innovation et l'excellence technique, cette application de hachage a le potentiel de devenir une ressource précieuse dans le domaine de la sécurité des données et de l'informatique.

En conclusion, ce projet représente une étape importante dans notre compréhension et notre application des concepts de hachage, tout en fournissant un outil précieux pour la communauté informatique. Nous espérons que ce rapport servira de guide utile pour ceux qui souhaitent explorer davantage ce domaine fascinant et en constante évolution.

13. Bibliographie :

[1] : https://www.baeldung.com/cs/hashing-linear-probing

[2] : https://www.geeksforgeeks.org/open-addressing-collision-handling-technique-in-hashing

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

[4] : https://www.cs.usfca.edu/~galles/visualization/Algorithms.html

[5] : https://carmencincotti.com/fr/2022-10-17/sondage-quadratique-adressage-ouvert-tables-de-hachage

[6] : https://www.iro.umontreal.ca/~csuros/IFT2015/H09/materiel/hashing-pp12_17.pdf

[7] : https://carmencincotti.com/fr/2022-10-24/double-hachage-adressage-ouvert-tables-de-hachage/

[8] : https://www.youtube.com/watch?v=g44q1E6qKnw&t=146s

[9] : https://www.youtube.com/watch?v=KyUTuwz_b7Q&t=269s

[10] : https://www.youtube.com/watch?v=n6QwxA8s8KI

[11] : https://www.programiz.com/dsa/b-plus-tree#google_vignette

[12] ; : https://www.youtube.com/watch?v=4T7QHDbIzxA

[13] : https://www.youtube.com/watch?v=Y01r643ckfI&list=PLfZw_tZWahjxJl81b1S-vYQwHs_9ZT77f

[14] : https://customtkinter.tomschimansky.com/documentation/

[15] : https://carmencincotti.com/fr/2022-10-03/le-chainage-separe-faq-tables-de-hachage/

[16] : https://www.techno-science.net/glossaire-definition/Table-de-hachage-page-2.html