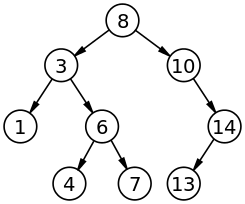
**Synthèse algorithmique – Séance 2**

**Une image contenant croquis, cercle, dessin, diagramme

Description générée automatiquementRappel**

* Différents types de parcours :
* Largeur
* Longueur
  + - * + Préfixe
        + Infixe
        + Postfixe



* C’est un arbre binaire dans lequel chaque nœud possède une clé (valeur) de tel sorte que chaque nœud du sous-arbre de gauche ait une clé plus petite ou égale au nœud considéré, et inversement pour le sous-arbre droit
* Opérations :
  + Recherches
  + Insertion
  + Suppression

*Tri par tas*

* *Une image contenant capture d’écran, Caractère coloré, Bleu Majorelle, léger

  Description générée automatiquement*Utilise les listes et les arbres
* Se base sur le fonctionnement de la structure de données du tas
* Associer un arbre binaire et une liste

Une image contenant horloge, capture d’écran, cercle, Police

Description générée automatiquement

* Algorithme :

1. S’assurer qu’on respecte la propriété « Max-Heap »

*Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement*2. Echanger la racine avec le dernier élément – Swap

3. Réduire la taille du tas de 1 – Remove

4. Repositionner l’élément le plus grand à la racine – Heapify

5. Recommencer au point 2

*Tas*

* Arbre binaire parfait
* Respecte la propriété du tas : la clé d'un nœud parent a une plus haute priorité que les clés de ses enfants

*Utilité du tri par tas*

* OS
* GUI
* Moteurs de recherche

**Les tables de hachage**

*Introduction*

* Structure de données reposant sur des tableaux
* Structure de données statique
* Algorithme de recherche performant
* Comment ?
  + La position de l’élément dans le tableau est fonction de l’élément lui-même

*Notion de clé*

* Partie d’un élément qui permet de désigner le contenu de cet élément sans ambiguïté
* Exemples :
  + Etudiant – Matricule
  + Abonné – Numéro de client
  + Etc

*Hachage ?*

* Comment stocke-t-on les mots de passe ?
* Très souvent, ils sont enregistrés mais illisibles.
* Comment ?

*Fonction de hachage*

* Utilisation d’une fonction de hachage qui va calculer une empreinte unique à partir de données fournies
* Attention donnée non récupérables !

*Propriétés*

* Longueur de la signature identique quel que soit la taille des données
* Unidirectionnel
* Impossible de prédire la signature
* 2 données différentes = 2 signatures différentes

*Exemple*

Une image contenant texte, capture d’écran, Police

Description générée automatiquementUne image contenant texte, capture d’écran, Police

Description générée automatiquement

*Collisions !*

=> Utilisation de fonctions de hachage à faible taux de collisions

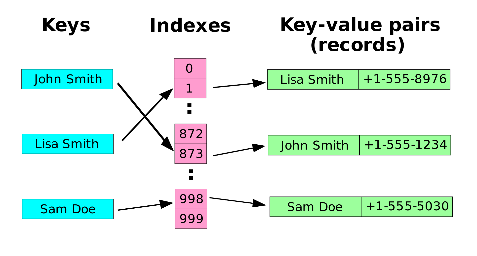
*Salage*

* Ajouter un élément à la donnée pour en modifier la signature
* Augmente la sécurité face à différentes attaques
* Eviter des doublons de mot de passe
* Exemple :
  + Python > 6e3604888c4b4ec08e2837913d012fe2834ffa83
  + Pythonsalé > 7d32954c9a36e2cdccea761bf462932b1cb12539

*Exemple de fonctions*

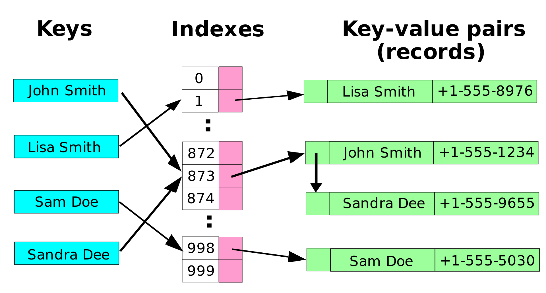
* MD5 : Bas niveau de sécurité car collisions possibles et trouvables – Signature de 128 bits
* SHA-1 : Taux de collisions moins élevés que MD5 mais trouvables – Signature 160 bits
* SHA-2 (SHA-256 – SHA-512) : Standards actuels - Signature de 256 et 512 bits

*Table de hachage*



* Recherche par clé très simple : Hash de la clé = index dans le tableau
* Sauf si collision !

*Solutions*

* Par chainage
* Par adressage ouvert

Une image contenant texte, Police, capture d’écran, reçu

Description générée automatiquement*Utilité ?*

* Complexité temporelle extrêmement intéressante pour chercher, ajouter ou supprimer un élément
* Base de données, caches, etc

*SHA-256*

* Etape 1 – Prétraitement des données
* Objectif : Compatibilité des données avec l’algorithme
  + Padding
  + Data Length
  + Permet de garantir l’intégrité des données
* Etape 2 – Initialisations des constantes
* Objectif : Constantes nécessaires à l’algorithme
  + Constantes (h0-h7) calculées à partir des 32 premiers bits et des 8 premiers nombres premiers
* Etape 3 – Division en blocs
  + Objectif : Taille fixe de bloc pour le traitement
  + Blocs de 512 bits
* Etape 4 – Traitement des blocs
  + Objectif : Produire le hachage
  + Traitement récursif – Traitement individuel
  + Transformation des blocs (Permutation, rotation, modulo, etc)