tp09 :
Arbre de Merkle

Master 2 I2L, 2020/2021

Les arbres de Merkel sont des arbres binaires de hachage. Ils sont utilisés pour comparer rapidement deux ensembles et détecter les différences. Ils sont utilisés notamment dans les systèmes pair à pair de téléchargement, dans certaines bases de données nosql (BD distribuée de grand volume), ou encore en cryptographie dans les blockchains (bitcoin, etc.).

Le but est de construire des arbres de Merkel afin de comparer deux fichiers.

Exercice 1 : Définition

a- Trouver la définition des arbres de Merkel.
b- Donner un exemple d'un tel arbre.

Exercice 2 : Implémentation en Haskell

2.a. Nous allons utiliser comme fonction de hachage la fonction SHA256. Tester les lignes suivantes qui calcule le SHA256 d'une chaine de caractères :

```
import Data.Digest.SHA256 (hash)
import Data.Char          (ord)

stringToOctet s =
    map (\x -> fromIntegral (ord x)) s

main = do
    print $ hash [65, 66]
    print $ hash (stringToOctet "salut les nazes")
```

2.b. Définir une fonction `hashString` qui à partir d'une chaine de caractère et une taille de block (exprimé en nombre de caractères) construit une liste de couples. Le premier élément de chaque couple est la clé de hachage d'un block et le second élément est l'identifiant du block.
Par exemple, la chaine de caractère `"AAB.AAA.AAA.AAB.AAA."` peut être découpé en block de taille 4 pour obtenir une liste de la forme :
`[ ([34,...], 0), ([45,...], 1), ([129,...], 2), ([145,...], 3), ([54,...], 4) ]`.

Indication : voir la fonction `splitAt`.

2.c. Définir une fonction `consMerkelTree` qui construit l'arbre de Merkel d'une chaine de caractère. Evidement, vous pourrez utiliser `hashString`, puis définir une fonction `split2` qui séparer une liste en un couple de listes de même longueur (à l'unité près).

2.d Définir un `main` qui puisse lire un fichier puis le mettre sous forme d'arbre de Merkel.

Exercice 3 : Comparaison de 2 fichiers

3.a Définir une fonction qui compare et détecte toutes les différences entre la représentation sous forme d'arbre de Merkel de 2 chaines de caractères.

3.b Calculer la complexité de votre algorithme.