

□ **Exercice 1** : *complexité des algorithmes de recherche*

Dans tout cet exercice, on exprime la complexité en nombre de comparaisons effectués par l'algorithme. Et note t la longueur du texte et m la longueur du motif.

1. Montrer que la recherche naïve effectuée entre t et $t \times m$ comparaisons et donner des exemples dans laquelle ces deux bornes sont atteintes.

□ **Exercice 2** : *Algorithme naïf avec motif à caractère unique*

1. Montrer qu'il est possible d'améliorer l'algorithme de recherche naïf si on suppose que tous les caractères du motifs apparaissent une seule fois dans le motif.
 ⚙️ Faire par exemple la recherche de **abcd** dans le texte **abceababccabcbabb** et observer ce qu'il se passe lorsqu'on trouve un début de correspondance.
2. Donner une implémentation en C ou en OCaml de ce nouvel algorithme.

□ **Exercice 3** : *Dérouler l'algorithme de Boyer-Moore-Hoorspool*

1. Donner la table de décalage du motif **tele**
2. Donner les étapes du déroulement de l'algorithme de Boyer-Moore-Hoorspool pour recherche toutes les occurrences de ce motif dans le texte : **le telephone** ou **la television**

□ **Exercice 4** : *Rabin-Karp à deux motifs*

1. Ecrire une fonction `hash : string -> int` qui effectue la somme des carrés des code ASCII de la chaîne donnée en argument
2. Ecrire l'implémentation de l'algorithme de Rabin-Karp afin de recherche un motif dans un texte.
3. Adapter votre algorithme de façon à pouvoir rechercher les occurrences de deux motifs **m1** et **m2** qu'on suppose de même longueur.
4. Proposer une nouvelle modification si les motifs sont de taille différentes.

□ **Exercice 5** : *Quelques arbres de Huffman*

1. Donner l'arbre de Huffman obtenu pour un texte contenant 10 caractères ayant tous le même nombre d'occurrences n .
2. Donner l'arbre de Huffman obtenu pour un texte contenant 7 caractères dont les nombres d'occurrences sont 1, 2, 4, 8, 16, 32 et 64.
3. On donne l'arbre de Huffman suivant :

