

□ **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.
2. Déterminer de même les bornes du nombre de comparaisons effectuées par l'algorithme de Boyer-Moore-Hoorspool et donner un exemple où elles 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 **abceababccabcbdbb** 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*

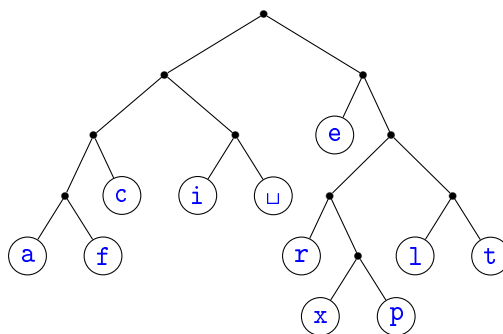
1. Donner la table de décalage du motif **tele**
2. Donner les étapes du déroulement de l'algorithme de Boyer-Moore-Hoorspool pour recherche toutes les occurrences de ce motif dans le texte : **le telephone** ou **la television**

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

1. Ecrire une fonction `hash : string -> int` qui effectue la moyenne des code ASCII de la chaîne donnée en argument en leur affectant pour coefficient leur position dans la chaîne (le coefficient du premier caractère est 1).
2. Ecrire l'implémentation de l'algorithme de Rabin-Karp avec cette fonction de hachage afin de rechercher 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 :



Donner les codes des caractères présents dans l'arbre.

4. Donner le texte dont la compression est :
 11101001111011101111010111101110110101011000010100011001100010000001010111010
5. Calculer la taille initiale du texte et la taille du texte compressé.

□ **Exercice 6** : *Algorithme LZW*

1. Décrire le fonctionnement de l'algorithme LZW sur une chaîne qui contient n caractères identiques.
2. On considère la compression d'un texte contenant 5 caractères différents. Donner une chaîne de caractères de longueur 20 pour laquelle l'algorithme LZW ne réduit pas la taille du résultat par rapport à l'entrée.