TD1.9 – Algorithme de Huffman / Diviser pour Régner

Objectifs

À la fin de cette séance, vous devriez être capable de :
- appliquer l'algorithme de Huffman sur un exemple simple ;
- évaluer la qualité d'un code compresseur ;
- utiliser la méthode *Diviser pour Régner* pour résoudre un problème.

Exercice 1 : Huffman (30 min)

On dispose d'un texte écrit avec un alphabet $\mathcal{A} = \{A, B, C, D, E, F, G, H\}$. Les proportions de ces symboles dans le texte sont données en pourcentage dans le tableau suivant :

Symbole	A	B	C	D	E	F	G	H
Proportion (%)	2	10	4	18	12	16	32	6

1. **Codage de longueur fixe**
   Donnez la taille du codage de longueur fixe nécessaire pour coder cet alphabet $\mathcal{A}$.
   Calculez l'entropie $\mathcal{H}(p) = -\sum_{i=1}^{k} p_i \log_2 p_i$ de cette distribution.
   Pourquoi le code de longueur fixe n'est-il (probablement) pas le codage le plus court en moyenne pour cet alphabet ?
2. **Codage de longueur variable**
   Construire avec l'algorithme de Huffman un codage de longueur variable pour $\mathcal{A}$. Donner l'arbre de codage correspondant et le tableau des codes. Calculer la longueur moyenne du codage et commenter le résultat en une phrase.
3. **Transmission de message**
   À faire par deux : simulez une transmission de message à l'aide de votre code de Huffman.
   - L'un d'entre vous joue le rôle d'émetteur : il choisit un mot de 5 à 10 lettres sur l'alphabet $\mathcal{A}$ et le code.
   - L'autre joue le rôle du récepteur : il décode le message binaire reçu.

   Vérifiez que tout s'est passé comme prévu. Sinon, expliquez pourquoi et résolvez le problème.
   Échangez les rôles et recommencez.

Exercice 2 : Majoritaire

Définition : un élément x est dit *majoritaire* dans un multi-ensemble E de n éléments, si et seulement si le nombre d'occurrences de x dans E est strictement supérieur à $n/2$.
Dans la suite E est un tableau indexé de 0 à $n-1$.

1. **Algorithme naïf**

   ```
   existe_majoritaire <- faux
   pour i de 0 à n-1
      c := 0
      j := 0
      tant que j < n et non(existe_majoritaire) faire
           si E[j]=E[i] alors c := c+1
           si (c > n/2) alors
                 existe_majoritaire := vrai
                 majo := E[i]
           j := j+1
   ```

   Décrivez (en deux lignes maximum) son fonctionnement.
   Quel est le coût au mieux et au pire de cet algorithme ?

2. **Diviser pour régner**
   On remarque les propriétés suivantes (que l'on admet pour l'instant).
   Si $E = E_1 \cup E_2$ avec E_1 et E_2 de mêmes tailles à un élément près (dans le cas où E comporte un nombre impair d'éléments), alors :
   — si x est majoritaire dans E, alors x est majoritaire dans E_1 ou dans E_2.
   — si x est majoritaire dans E_1 **et** dans E_2, alors il est majoritaire dans E.
   En utilisant ces propriétés, écrivez une procédure `majoritaire(i,j)` qui explore récursivement les deux moitiés de tableau, et cherche s'il existe un élément majoritaire dans `E[i..j]`. Elle renvoie :
   — `(-, 0)` s'il n'y a pas de majoritaire dans `E[i..j]`.
   — `(x, c)` si `x` est majoritaire avec un nombre d'occurrences égal à `c`.

3. **Complexité**
   Déterminez la complexité de cette procédure récursive, en vous inspirant de l'algorithme vu en cours.

4. **Vers un algorithme récursif**
   Démontrer les propriétés suivantes pour justifier la correction de l'algorithme récursif :
   — il y a **au plus** un élément majoritaire.
   Si $E = E_1 \cup E_2$ avec E_1 et E_2 de taille n' (si $n = 2 \times n'$) ou de tailles n' et $n'+1$ si $n = 2 \times n' + 1$, alors :
   — si x est majoritaire dans E, alors x est majoritaire dans E_1 ou dans E_2.
   — si x est majoritaire dans E_1 et dans E_2, alors il est majoritaire dans E.

Membres du groupe et responsabilités

	Appliquer des méthodes algorithmiques connues (diviser pour régner)	Gérer le temps
	Évaluer l'efficacité des algorithmes	Réguler les prises de parole
	Tester les algorithmes écrits Rechercher des cas problématiques	S'assurer que tous comprennent et sont en accord avec le rendu
	Représenter schématiquement les structures de données et leur traitement	Rédiger le rendu

Exercice 2 : Majoritaire

Diviser pour régner

En explorant récursivement les deux moitiés de tableau, écrivez une procédure `majoritaire(i,j)` qui cherche s'il existe un élément majoritaire dans `E[i..j]`. Elle renvoie :

— `(-, 0)` s'il n'y a pas de majoritaire dans `E[i..j]`.

— `(x, c)` si `x` est majoritaire avec un nombre d'occurrences égal à `c`.