Master M1/TC Algorithmique Avancée et Complexité – Examen Mini Projet

Exercice 1 : Recherche de la plus longue sous-séquence commune

Soient les deux séquences X = <A, L, G, O, R, I, T, H, M, E> et Y = <L, A, R, M, E>.

1. (5.5P) Dérouler l'algorithme LONGUEUR-PLSC de recherche de la plus longue sous-séquence commune.
2. (3p)Proposer un algorithme affichant toutes les plus longues sous-séquences communes ?
3. (1.5p)Donner la complexité en moyenne de LONGUEUR-PLSC ? Justifier !

Exercice 2 : Compression de données et codages de Huffman

Soit un fichier contenant les caractères suivants avec leurs fréquences :

Lettres	J	K	R	S	T	U
Fréquences	41	9	8	12	5	1

1. (5.5P) Dérouler toutes les étapes de l'algorithme HUFFMAN.
2. (2p) Ecrire l'algorithme INSERER(Q, *z*) qui consiste à insérer le nœud *z* ayant la fréquence somme des deux fréquences min préalablement extraites avec EXTRAIRE-MIN.
3. (1p) Proposer un invariant à l'algorithme HUFFMAN, puis démontrer qu'il est correct.
4. (1.5p)Donner la complexité de HUFFMAN dans le meilleur des cas ? Justifier !

Exercice 1

Solution (https://zanotti.univ-tln.fr/ALGO/I51/PLSC.html)

	j	0	1	2	3	4	5
i		EPS	L	A	R	M	E
0	EPS	0	0	0	0	0	0
1	A	0	0 ↑	1 ↖	1 ←	1 ←	1 ←
2	L	0	1 ↑	1 ↑	1 ↑	1 ↑	1 ↑
3	G	0	1 ↑	1 ↑	1 ↑	1 ↑	1 ↑
4	O	0	1 ↑	1 ↑	1 ↑	1 ↑	1 ↑
5	R	0	1 ↑	1 ↑	2 ↖	2 ←	2 ←
6	I	0	1 ↑	1 ↑	2 ↑	2 ↑	2 ↑
7	T	0	1 ↑	1 ↑	2 ↑	2 ↑	2 ↑
8	H	0	1 ↑	1 ↑	2 ↑	2 ↑	2 ↑
9	M	0	1 ↑	1 ↑	2 ↑	3 ↖	3 ←
10	E	0	1 ↑	1 ↑	2 ↑	3 ↑	4 ↖

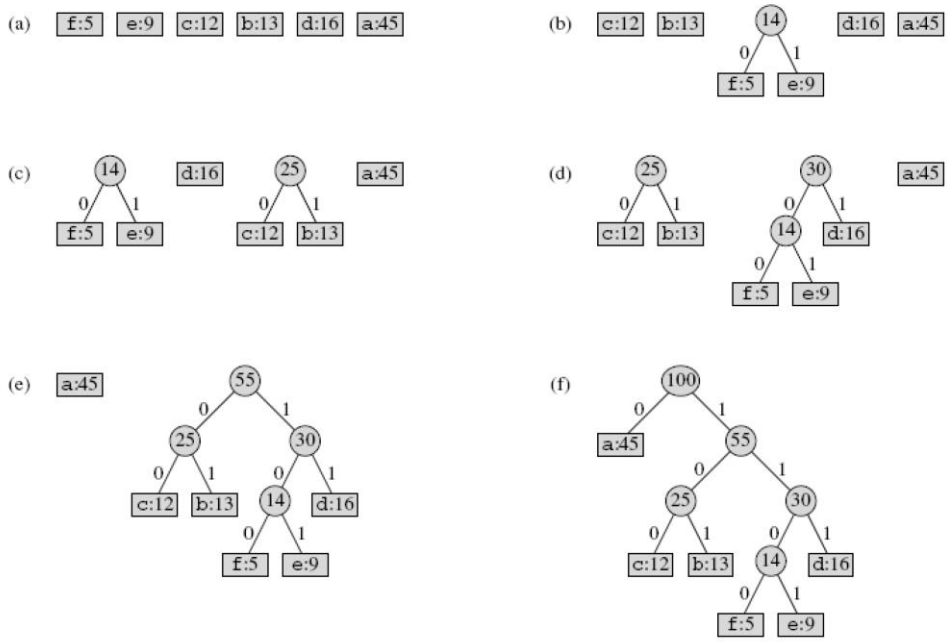
```
Algorithm LONGUEUR-PLSC2(b, X, i, j)
    if (j=0) or (j=0) then
        return
    endif
    if (b[i, j] = '↖') then
        LONGUEUR-PLSC2(b, X, i-1, j-1)
        Imprimer(x_i)
    else
        if (b[i, j] = ↑) or (c[i-1, j] = c[i, j-1]) then
            LONGUEUR-PLSC2(b, X, i-1, j)
        endif
        if (b[i, j] = ←) or (c[i-1, j] = c[i, j-1]) then
            LONGUEUR-PLSC2(b, X, i, j-1)
        endif
    endif
```

La complexité en moyenne de LONGUEUR-PLSC est $\theta(n*m)$ où n et m sont les longueurs des deux séquences, car l'algorithme affecte toutes les cases de la matrice c[1..n, 1..m], où pour chaque case il exécute un nombre fini d'instructions en $\theta(1)$.

Exercice 2

Il suffit de remplacer dans la trace suivante :

Remplacer	J	K	R	S	T	U
de fréquences	45	13	12	16	9	5
Par	**a**	**b**	**c**	**d**	**e**	**f**
de fréquences	41	9	8	12	5	1




```
Algorithm TasExtraireMin(T)
        R := T[1]
        if T.Nb == 0 then
                return erreur
        endif
        T[1] := T[T.Nb]
        T.Nb − −
        Entasser(T, 1)
        return R
```

```
Algorithm INSERER(Q, z)
        if Q.Nb == TAILLE_MAX then
                return erreur
        endif
        Q.Nb + +
        i := Q.Nb       //indice de la nouvelle feuille.
        while (i/2 ≥ 1) ∧ (T[i/2] < z) do
                Q[i] := Q[i/2]
                i := i/2
        endwhile
        T[i] := z
```

Invariant de l'algorithme de Huffman : Dans tous les arbres construits, les poids des nœuds de n'importe quel niveau h sont plus petits que les niveau supérieurs h-1, h-2, …1.
Initialisation : pas d'arbres. Propriété vérifiée.
Maintenance : supposons la propriété est vrai à une itération. A l'itération suivante, le nouveau nœud est la somme des deux nœuds extraits. Puisque les arbres de ces deux nœuds respectent la propriété, alors nécessairement la propriété reste vérifiée par le fait que le nouveau nœud est la somme des nœuds extraits, qui est donc supérieure aux nœuds extraits, et donc aussi supérieurs à tous les niveaux inférieurs.
Terminaisons : Dans L'arbre final unique, la valeur de tout nœud est supérieure à ses fils et donc à tous les descendants. Ainsi l'optimalité de l'algorithme est assurée, car les nœuds les moins fréquents ont les codes les plus longs, puis les nœuds suivants, ainsi de suite, jusqu'au nœud le plus fréquent qui aura le code le plus court.

Complexité de Huffman : l'algorithme boucle (n-1) sur l'algorithme ExtraireMin qui est en O(log(n)). D'où une complexité en O(n*log(n)).
Dans le meilleur des cas, si toutes les fréquences sont égales, alors ExtraireMin serait en $\theta(1)$, d'où $\theta(n)$.