Oraux blancs MPI
type CCMT

Florian Bourse

Exercice 1

1. Donner l'arbre de Huffman construit sur la chaîne de caractères `bacabada`.
2. Nommer la structure de donnée abstraite la plus adaptée pour implémenter l'algorithme de Hufmann ?
3. Donner la complexité de la construction de l'arbre de Huffman.
4. On suppose à présent que notre algorithme prend en entrée des couples (lettre, occurences) triés par ordre croissant du nombre d'occurence. Montrer que dans cette situation, on peut améliorer la complexité en utilisant deux files.

Exercice 2

On considère le code suivant, effectuant un parcours d'un graphe orienté $G = (S, A)$ représenté par listes d'adjacences :

```
type couleur = Blanc | Gris | Noir
let dfs (g : int list array) (s : int) =
  let n = Array.length g in
  let colors = Array.make n Blanc in
  let rec parcours i =
    if colors.(i) = Blanc then
      begin
        colors.(i) <- Gris;
        List.iter parcours g.(i);
        colors.(i) <- Noir
      end
  in
  parcours s;
  colors
```

1. Décrire le tableau `colors` renvoyé par l'appel `dfs g s`.
2. Calculer la complexité de la fonction `dfs`.
3. Décrire l'ensemble des sommets de couleur grise lors de l'exécution de l'appel récursif `parcours i`.
   On pourra considérer l'arborescence du parcours de graphe effectué.
4. Proposer un algorithme permettant de tester si un graphe orienté possède un cycle et évaluer sa complexité.
5. Proposer un algorithme permettant de compter le nombre de cycles d'un graphe orienté et évaluer sa complexité.