Algorithmique et complexité : HLIN401

Une page recto verso manuscrite autorisée

1 Exercice

Soit un algorithme $\mathcal{A}_1$ de complexité dans $\Theta(n^4)$, programmé en un programme $\mathcal{P}_1$.

Sur une machine donnée, pour une donnée de taille n_0, $\mathcal{P}_1$ prend un temps t_0.
Quelle taille de problème $\mathcal{P}_1$ pourra-t-il traiter sur une machine dix mille fois plus rapide dans ce même laps de temps t_0 ?

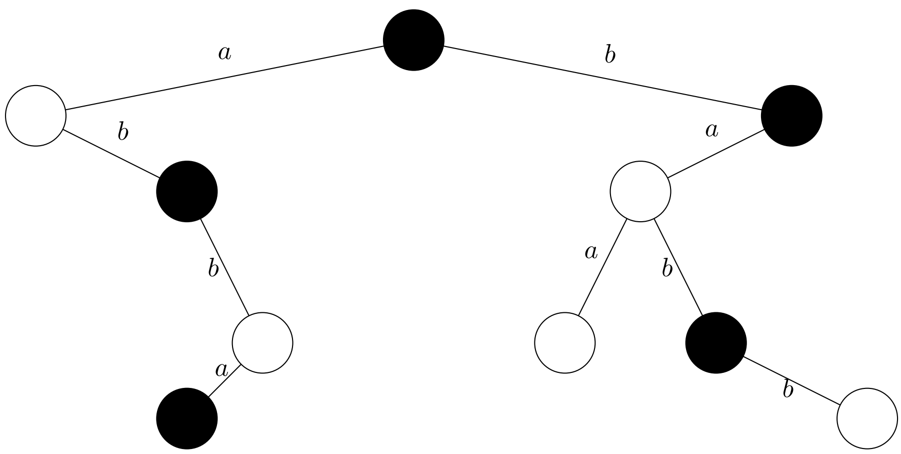
2 Problème : représentation d'un ensemble $\mathcal{S}$ de chaînes de caractères.

Question 1 Pourquoi la taille $\tau(\mathcal{S})$ d'un **ensemble** de **p** chaînes de caractères, avec au total N caractères est-elle dans $\Theta(N)$? (indiquez une représentation plausible simple).

Une structure de données pour représenter un ensemble de chaînes de caractères.

Les chaînes de caractères considérées contiennent seulement les caractères `a` et `b`.

La figure ci dessous représente l'ensemble de chaînes {"*babb*", "*ba*", "*abb*", "*baa*", "*a*"} .



Les noeuds sont noirs si et seulement si les chaînes correspondantes **ne sont pas** dans l'ensemble.

Question 2 Donner un exemple d'une arborescence de 6 sommets qui représente l'ensemble **vide** de chaînes de caractères.

Question 3 Donner une condition suffisante pour que la taille de la figure qui représente $\mathcal{S}$ soit dans $\mathcal{O}(\tau(\mathcal{S}))$. Cette condition ne devra pas empêcher de pouvoir représenter n'importe quel ensemble de chaînes de caractères.

Par la suite, on supposera cette condition vérifiée.

Les fonctions fournies sur les chaînes de caractères.

Vous disposez des constantes et fonctions suivantes :

— V_D est la chaîne (de caractères) vide.

— $AC(ch)$ (en $\Theta(|ch|)$) qui affiche la chaîne ch de caractères qui lui est passée en paramètre

— *a(ch)* et *b(ch)* : prennent en entrée et renvoient toutes deux une chaîne de caractères.
La chaîne renvoyée est la chaîne en entrée, augmentée à **droite** du caractère `a` (pour a) ou du caractère `b` (pour b). Ces deux fonctions sont de complexité dans $\Theta(1)$.
Par exemple $a(V_D)$ renvoie la chaîne de un caractère "a" et si ξ est la chaîne de caractères "aaa", $b(\xi)$ renvoie la chaîne de caractères "$aaab$".

Représentation de l'arborescence

Pour représenter l'arborescence de la page précédente, on utilisera la structure de données (dont les méthodes ne sont pas encore spécifiées)

```
struct sommet {bool present; sommet* ptA; sommet* ptB; ... };
```

Annexe : ordre lexicographique $<$ sur un alphabet à deux caractères `a` et `b`

Cet ordre est celui en usage dans les dictionnaires, et vous l'avez déjà vu.

Sa définition n'est redonnée ici que dans un but d'exhaustivité, et si vous comprenez les exemples ci dessous, cela suffit pour traiter le problème, sans avoir à se plonger dans la définition qui suit.

Exemples

— "$babaa$" $<$ "$babba$"

— "$babaa$" $<$ "$babaaa$"

Définition :
Étant données deux chaînes $A = \alpha_0\alpha_1...\alpha_p$ et $B = \beta_0\beta_1...\beta_q$, où chaque α_i et chaque β_j appartiennent à l'ensemble {`a`, `b`} (avec `a<b`), on dit que la chaîne A est lexicographiquement inférieure à la chaîne B (ce qu'on écrit $A < B$) si et seulement si

1. il existe un entier j, avec $0 \leq j \leq min(p, q)$, tel que $\alpha_i = \beta_i$ pour tout $i \in \{0,\ 1, \dots,\ j-1\}$ et $\alpha_j < \beta_j$, (c'est d'après cette règle, avec $j = 3$, que "$babaa$" $<$ "$babba$")
ou
2. $p < q$ et $\alpha_i = \beta_i$ pour tout $i \in \{0, 1, ..., p\}$.
cette règle s'applique pour avoir "$babaa$" $<$ "$babaaa$".

Question 4 Écrire une fonction d'affichage de chaînes, que l'on appellera AS

- qui prend en entrée `sommet*` $\mathcal{A}$ qui représente un ensemble $\mathcal{S}$ de chaînes de caractères
- et qui affiche les chaînes de $\mathcal{S}$ dans l'ordre lexicographique

Si on lui fournit la structure dessinée à la première page, AS doit afficher successivement "a", "abb" , "ba", "baa" et "$babb$".

Si vous utilisez (ce qui est conseillé) une fonction auxiliaire récursive, appelez la AR, spécifiez ses entrées et sa sortie et indiquez son appel initial en écrivant AS.

Question 5 Donner la complexité de votre fonction AS, la justifier et justifier qu'elle est optimum.

On dispose d'une **liste** $\mathcal{L}$ de chaînes (qui n'est **pas** fournie sous forme d'un arbre) et dont on veut imprimer les chaînes (toutes distinctes) dans l'ordre lexicographique avec une complexité optimum.

Question 6
Pourquoi l'algorithme qui consiste à trier d'abord les chaînes de la liste $\mathcal{L}$ dans l'ordre lexicographique, puis à les imprimer, n'est-il est pas de complexité linéaire ?

On dispose des fonctions élémentaires suivantes, toutes trivialement dans $\Theta(1)$:

- **sur une chaîne de caractères ch**
  - `bool CV?(ch)` renvoie un booléen indiquant si *ch* est vide.
  - `chaine CR(ch)` prend une chaîne **non vide** *ch* de caractères et renvoie la chaîne à laquelle on a enlevé son premier caractère (celui de gauche)
  - `bool a?(ch)` prend une chaîne **non vide** *ch* de caractères et renvoie un booléen indiquant si le premier caractère de cette chaîne est un `a`
    (si `a?(ch)` renvoie `faux`, c'est que le premier caractère de *ch* est le caractère `b`).
- **sur une liste de chaînes L**
  - `bool LN?(L)` renvoie un booléen indiquant si L **n**'est **pas** vide.
  - `liste LR(L)` prend une liste L **non vide** de chaînes et renvoie L privé de sa premiere chaîne.
  - `chaine LP(L)` prend une liste L **non vide** de chaînes et renvoie la premiere chaîne de L.

On veut écrire l'algorithme `AL(`$\mathcal{L}$`)` d'affichage de toutes les chaines de caractères de la liste $\mathcal{L}$. (bien sûr dans l'ordre lexicographique) en utilisant la structure de données arborescente.

Question 7 Écrivez la ou les méthodes qui vous seront nécessaires de manipulation de votre structure de données, et justifiez en la complexité en fonction de variables que vous choisirez vous même.

Question 8
Ecrivez maintenant l'algorithme `AL(`$\mathcal{L}$`)` d'affichage de toutes les chaines de caractères de la liste $\mathcal{L}$ (dans l'ordre lexicographique) et justifiez sa complexité optimum.