Sujet Mines-Ponts 2018 : correction

1. Oui. Par exemple, dans la chaîne $t = aaa$, la chaîne $s = aa$ a comme occurrences $y = 2$ et $y' = 3$. Cependant, $2 \geq 3 - 2 + 1$.
2. Il ne peut y avoir d'occurrence $i \leq k - 1$. Une occurrence est donc comprise entre k et n. Il y en a au plus $n - k + 1$. Supposons que s et t sont constituées de k et n lettres identiques. Alors, il y a bien $n - k + 1$ occurrences de s dans t.
3.

```
let rec longueur l = match l with
    | [] -> 0
    | x::q -> 1+longueur q;;
```

4.

```
let rec prefixe s t = match s with
    | [] -> true
    | x::q -> if t=[] then false
              else let t=y::r in x=y && prefixe q r;;
```

La relation de récurrence pour la complexité C_k pour une chaîne de longueur k est $C_k = O(1) + C_{k-1}$. On a donc $C_k = O(k)$.
5.

```
let recherche_naive s t =
    let rec aux l k = match l with
    | [] -> []
    | x::q -> if prefixe s l then k:: aux q (k+1)
              else aux q (k+1)
     in aux t (longueur s);;
```

6. Cette fonction appelle $n - k + 1$ fois la fonction préfixe. Sa complexité est donc $(n - k + 1) \times O(k) = O(nk)$.
7. Soit $\alpha \in \Sigma$ et $q \in Q$ fixés. Si la suite $(\rho^j(q))_j$ n'atteint pas 0, elle est strictement décroissante et minorée par 0, ce qui est absurde pour une suite d'entiers. Ainsi, il existe j tel que $\rho^j(q) = 0$. Dans ce cas, $\delta(\rho^j(q), \alpha)$ est défini. Ainsi, il existe $j \geq 0$ tel que $\delta(\rho^j(q), \alpha)$ est défini.
8. On reprend l'automate $\mathcal{A}_1$ auquel on ajoute une transition de 1 vers 1 étiquetée par a, une transition de 2 vers 0 étiquetée par b, une transition de 3 vers 2 étiquetée par b, une transition de 3 vers 1 étiquetée par a.
9. $\mathcal{A}_1$ reconnaît le langage des mots qui se terminent par aba.
10.

```
let copie_afdr a= let n=Array.length a.final in
    let f=Array.copy a.final and t=Array.make_matrix n lambda -1 and r=Array.copy a.repli in
    for i=0 to n-1 do
        for j=0 to lambda-1 do
            t.(i).(j)<-a.transition.(i).(j)
            done;
        done;
    {final=f;transition=t;repli=r};;
```

11.

```
let enleve_repli A = let A1=copie_afdr A in
    let T=A1.transition in
    let k=Array.length T in
    for i=0 to k-1 do
        for j=0 to lambda-1 do
            if T.(i).(j)=-1 then
                T.(i).(j)<-T.(A1.repli.(i)).(j)
        done
    done;
    A1;;
```

Il y a deux boucles "for" imbriquées, l'une de taille k, l'autre de taille λ. En outre, pour chaque passage de boucle, il y a au plus $O(1)$ opérations effectuées. La complexité est donc $O(k\lambda)$.
12. On calcule $\delta(0, u_1 \cdots u_i)$ pour tout $1 \leq i \leq n$. Si $\delta(0, u_1 \cdots u_i) \in F$, alors on ajoute i à la liste des occurrences.
13.

```
let occurrences A u =
    let rec aux q u = match u with
        | [] -> if A.final.(q) then [q] else []
        | t::v -> let qprime=A.transition.(q).(t) in
           if A.final.(q) then q::(aux qprime v) else aux qprime v in
    aux 0 u;;
```

La complexité est en $O(n)$.
14. Il y a les états $0, 1, 2, 3, 4, 5$. Les transitions vont de 0 vers 1 étiquetée par a, 1 vers 2 étiquetée par b, 2 vers 3 étiquetée par a, 3 vers 4 étiquetée par b, 4 vers 5 étiquetée c.
Enfin, $\rho(1) = 0$, $\rho(2) = 0$, $\rho(3) = 1$, $\rho(4) = 2$, $\rho(5) = 0$.
15. Ce langage est celui des mots qui se termine par s.
16. On démontre par récurrence forte que les suffixes stricts du type $u_1 \cdots u_j$ de $u_1 \cdots u_i$ sont les suffixes de $u_1 \cdots u_{\rho^k(i)}$ pour $k \geq 1$. C'est vrai pour l'entier 1.
Par définition, $u_1 \cdots u_{\rho(i)}$ est le plus long préfixe st de $u_1 \cdots u_i$ de ce type. Les suffixes stricts du type $u_1 \cdots u_k$ de $u_1 \cdots u_i$ sont donc $u_1 \cdots u_{\rho(i)}$ et les suffixes stricts de $u_1 \cdots u_{\rho(i)}$. Par récurrence, ce sont donc les $u_1 \cdots u_{\rho^j(i)}$ pour $j \geq 1$.
En particulier, $u_1 \cdots u_{\rho(i)-1}$ est un préfixe de $u_1 \cdots u_{i-1}$. Ainsi, $u_1 \cdots u_{j_i-1}$ est un suffixe de $u_1 \cdots u_{j_{i-1}}$.
En particulier, le suffixe le plus long de $u_1 \cdots u_i$ du type $u_1 \cdots u_j$ est le suffixe le plus long de $u_1 \cdots u_{i-1}$ du type $u_1 \cdots u_k$ tel que $u_{k+1} = u_i$. Ainsi, ce suffixe est du type $u_1 \cdots u_{\rho^j(i-1}$ tel que $k = \rho^j(i-1)$ et $u_{k+1} = u_i$. Cet entier k est donc le plus petit tel que $u_1 \cdots u_k$ est un suffixe de u_k est suivi de u_i. C'est donc le plus petit j tel que $\delta(\rho^j(\rho(i-1)), u_i)$ est défini.
17.

```
let automate_kmp u =
    let k=longueur u in
    let f=Array.make (k+1) false and t=Array.make_matrix (k+1) lambda -1
    and r=Array.make (k+1) 0 in
    f.(k)=true;
    let rec aux i m = match m with
       | [] -> ()
       | a::q -> t.(i).(a) <- i+1; aux (i+1) q in
    aux 0 u;
    let rec aux2 i m = match m with
       | [] -> ()
       | a::q -> let j=ref i-1 in
          while t.(!j).(a)=-1 do
             j:=r.(!j)
          done;
          r.(i) <- t.(!j).(a); aux (i+1) q
    in aux2 1 m;
    {final=f;transition=t;repli=r};;
```

18. Pour tout k, $\delta(k, u) = k + 1$ si $\delta(k, u)$ est défini. On a donc $\rho(i) = \delta(\rho^{j_i}(\rho(i-1)), u_i) = \rho^{j_i}(\rho(i-1)) + 1$. De plus, $\rho(k) \leq k - 1$ pour tout k donc, par récurrence, $\rho^j(k) \leq k - j$ donc $\rho(i) \leq \rho^{j_i}(\rho(i-1)) + 1 \leq \rho(i-1) - j_i + 1$.
D'après ce qui précède, $\sum\limits_{i=1}^{k} j_i \leq \sum\limits_{i=1}^{k} (1 + \rho(i-1) - \rho(i)) = k + \rho(0) - \rho(k) = k$. Ainsi $\sum\limits_{i=1}^{k} j_i = O(k)$.
19. Les créations des tableaux sont en $O(k)$, la modification de f et t en $O(k)$. Enfin, la modification de r est en $O(\sum\limits_{i=1}^{k} j_i) = O(k)$.
La complexité globale est en $O(k)$.
20.

```
let recherche_kmp s t = occurences (enleve_repli (automate_kmp s)) t;;
```

La complexité est, d'après les calculs de complexité précédents, en $O(k) + O(k\lambda) + O(n) = O(k\lambda) + O(n)$.
21. Il s'agit du langage des mots qui se terminent par aa, ab ou ba.
22. On considère l'automate à 6 états : $0, 1, 2, 3, 4, 5$. Il y a une transition de 0 vers 1 étiquetée par b et des transitions de 0 vers 0 étiquetées par a et c ; une transition de 1 vers 2 étiquetée par a, une transition de 1 vers 3 étiquetée par c, une transition de 2 vers 4 étiquetée par a, une transition de 2 vers 5 étiquetée par b. Les états finals sont 3, 4 et 5. Enfin, le repli de $1, 2, 3$ et 4 est 0, celui de 5 est 1.
23.

```
let rec recherche_dictionnaire_kmp S t = match S with
    | [] -> []
    | s::S0 -> (recherche_kmp s t)@recherche_dictionnaire_kmp S0 t;;
```

La complexité est en $O(\lambda k) + O(n)$ pour chaque motif. Globalement, il faut additionner cette complexité pour chaque motif. Cela donne $O(|S|k\lambda) + O(n|S|)$.
24. On peut créer un AFDR qui reconnaissent les mots qui se terminent par l'un des motifs. Pour cela, il suffit de construire un automate KMP pour chaque motif. Ensuite, on fait en sorte que l'état initial de chacun d'entre eux soit le même et que les états suivants soient les premiers états de chaque automate, puis les seconds... Les replis doivent également être recalculés.