

Corrigé de la Composition d'Informatique

Jean-Berstel Jean-Jacques Lévy

14 février 2001 – 4h

Question 1

	<i>b</i>	<i>a</i>
<i>a</i>		<i>b</i>
	<i>a</i>	
<i>b</i>	<i>b</i>	
	<i>c</i>	<i>a</i>

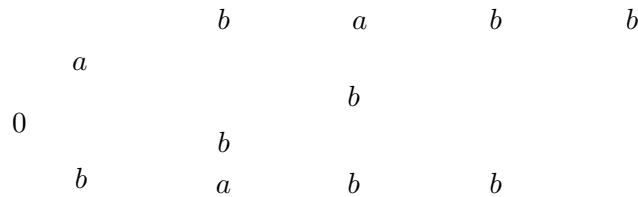
Question 2 `static int filsDe(int p, char c, Arbre a) {
 for (Liste x = a.succ[p]; x != null; x = x.suivant)
 if (x.etiq == c)
 return x.sommet;
 return -1;
}`

Question 3 `static int descendantDe(int p, String w, Arbre a) {
 for (int i = 0; i < w.length(); ++i) {
 p = filsDe (p, w.charAt(i), a);
 if (p == -1)
 return -1;
 }
 return p;
}`

Question 4 `static void inserer(String w, Arbre a) {
 int p = 0;
 for (int i = 0; i < w.length(); ++i) {
 int q = filsDe (p, w.charAt(i), a);
 if (q == -1) {
 a.succ[p] = new Liste (w.charAt(i), a.n, a.succ[p]);
 p = a.n++;
 } else
 p = q;
 }
}`

Question 5 `static Arbre nouvelArbre(String[] x) {
 Arbre a = new Arbre();
 for (int i = 0; i < x.length; ++i)
 inserer (x[i], a);
 return a;
}`

Question 6



Question 7

```

static void insererSuffixe(int i, String w, Arbre a) {
    int p = 0;
    for ( ; i < w.length(); ++i) {
        int q = filsDe (p, w.charAt(i), a);
        if (q == -1) {
            a.succ[p] = new Liste (w.charAt(i), a.n, a.succ[p]);
            p = a.n++;
        } else
            p = q;
    }
    a.terminal[p] = true;
}

static Arbre nouvelArbreSuffixe(String w) {
    Arbre a = new Arbre();
    for (int i = 0; i <= w.length(); ++i)
        insererSuffixe (i, w, a);
    return a;
}
  
```

Question 8

```

static int[] nombreDeFacteurs(int N, Arbre a) {
    int[] cw = new int[N + 1];
    visiter (0, 0, cw, a);
    return cw;
}

static void visiter (int p, int profondeur, int[] cw, Arbre a) {
    ++ cw[profondeur];
    for (Liste x = a.succ[p]; x != null; x = x.suivant)
        visiter (x.sommet, profondeur+1, cw, a);
}
  
```

Question 9 a) La borne supérieure est atteinte quand tous les facteurs sont différents (par exemple quand toutes les lettres de w sont différentes). Alors $F(w) = 1 + N + N - 1 + \dots + 1 = 1 + N(N + 1)/2$. Inversement, la borne inférieure est atteinte quand tous les facteurs de même longueur sont égaux (par exemple quand toutes les lettres de w sont égales). Alors $F(w) = N + 1$. En conclusion

$$N + 1 \leq F(w) \leq 1 + N(N + 1)/2$$

b) $F(w)$ est le nombre de sommets de l'arbre suffixe. En dessinant l'arbre suffixe dans le plan cartésien en suivant l'axe des abscisses pour un arc étiqueté a et l'axe des ordonnées pour les arcs étiquetés b , on obtient $F(w) = (n + 1)^2$ (tous les points du carré de côté n).

Question 10 a) Un tel facteur est suffixe de w .

b) Chaque fois que xa est facteur pour une lettre a , le mot ya est aussi facteur.

c) On note $H_w(k)$ l'ensemble des facteurs de longueur k de w . Alors $C_w(k+1) = C_w(k) + \delta_w(k)$ avec $\delta_w(k) = \sum_{x \in H_w(k)} d(x) - 1$. Pour que la somme $\delta_w(k)$ soit négative, il doit exister un facteur

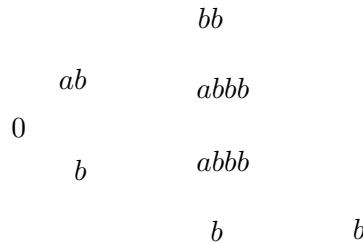
$x_0 \in H_w(k)$ de degré 0. D'après (a), il est le seul de son espèce. Donc tous les autres facteurs de longueur k sont de degré 1 et $\delta_w(k) = -1$.

d) Supposons l'assertion fautive, et soit w un mot de longueur minimale pour lequel l'implication est fautive, donc tel qu'il existe un entier k avec $C_w(k) = C_w(k+1)$ et $C_w(k+1) < C_w(k+2)$. Soit u le suffixe de longueur k de w , et posons $w = w'a$, où a est la dernière lettre de w .

(i) Si u n'est pas facteur de w' , alors $C_{w'}(k+i) = C_w(k+i) - 1$ pour $i \geq 0$, car w' compte pour chaque longueur un facteur de moins que w , à savoir le suffixe de w de cette longueur. En vertu de la minimalité de w , on a $C_{w'}(k+1) \geq C_{w'}(k+2)$, donc par l'équation aussi $C_w(k+1) \geq C_w(k+2)$, contradiction.

(ii) Si u est facteur de w' , alors tous les facteurs de w de longueur k sont de degré 1 exactement. Comme $C_w(k+1) < C_w(k+2)$, il existe au moins un facteur x de longueur $k+1$ de w de degré ≥ 2 . Mais alors, son suffixe de longueur k est aussi de degré ≥ 2 , contradiction.

Question 11 Pour $w = ababbb$



Question 12 Un arbre compacté a $N + 1$ sommets terminaux. Si le mot ne contient pas deux lettres distinctes, son nombre de sommets est $N + 1$. Dans le cas contraire, soit k le nombre de feuilles de l'arbre. On a $k \leq N$ parce que la racine est terminal sans être une feuille (mot non vide). Tous les sommets non terminaux sont de degré au moins 2. Il résulte des propriétés des arbres que le nombre de sommets de degré au moins 2 est au plus $k - 1$, et la racine est parmi ces sommets. Les sommets non terminaux sont donc en nombre au plus $k - 2$. Le nombre total de sommets est donc majoré par $k - 2 + N + 1 \leq 2N - 1$.

Question 13 `static int longueurMot (Arbre a) { return longueurMotR(a, 0); }`

```
static int longueurMotR (Arbre a, int p) {
    int r = 0;
    for (Liste x = a.succ[p]; x != null; x = x.suivant)
        r = Math.max (r, 1 + longueurMotR(a, x.sommet));
    return r;
}
```

Question 14 `static ArbreA transformer (Arbre a) {`

```
    ArbreA r = new ArbreA(); r.n = a.n;
    transformerR (r, 0, longueurMot(a), a);
    return r;
}
```

```
static void transformerR (ArbreA r, int p, int N, Arbre a) {
    for (Liste x = a.succ[p]; x != null; x = x.suivant) {
        int q = x.sommet;
        transformerR (r, q, N, a);
        ListeA qArc = r.succ[q];
        if (qArc == null)
            ajouterArcA(p, N - 1, N, q, r);
        else if (qArc.suivant == null && !r.terminal[q])

```

```

        ajouterArcA(p, qArc.debut - 1, qArc.afin, qArc.sommet, r);
    else
        ajouterArcA(p, qArc.debut - 1, qArc.debut, q, r);
    }
    r.terminal[p] = a.terminal[p];
}

static void ajouterArcA(int x, int deb, int afin, int y, ArbreA a) {
    a.succ[x] = new ListeA (deb, afin, y, a.succ[x]);
}

```

Question 15

```

static ArbreA epurer (ArbreA a) {
    ArbreA r = new ArbreA(); r.n = 0;
    epurerR (r, a, 0);
    return r;
}

static int epurerR (ArbreA r, ArbreA a, int p) {
    int p1 = r.n++; ListeA y = null;
    for (ListeA x = a.succ[p]; x != null; x = x.suivant) {
        int q = x.sommet; int q1 = epurerR (r, a, q);
        y = new ListeA(x.debut, x.afin, q1, y);
    }
    r.succ[p1] = y;
    r.terminal[p1] = a.terminal[p];
    return p1;
}

```

Question 16

```

static int lpc(int i, int j, int k, int l, String w) {
    int h = 0;
    while ( i < j && k < l && w.charAt(i)== w.charAt(k) ) {
        ++i; ++k; ++h;
    }
    return h;
}

```

Question 17

```

static void insererSuffixe(int i, String w, int p, ArbreA a) {
    ListeA arc;
    if ( i == w.length() || (arc = arcDe(p, w.charAt(i), w, a)) == null)
        ajouterSuffixeAuBout(p, i, w, a);
    else {
        int h = lpc(arc.debut, arc.afin, i, w.length(), w);
        int q = arc.sommet;
        if (arc.debut + h < arc.afin) {
            int r = briserArc(p, arc, arc.debut + h, a);
            q = r;
        }
        insererSuffixe(i + h, w, q, a);
    }
}

static ListeA arcDe(int p, char c, String w, ArbreA a) {
    for (ListeA x = a.succ[p]; x != null; x = x.suivant)
        if (w.charAt(x.debut) == c)
            return x;
    return null;
}

static void ajouterSuffixeAuBout(int p, int i, String w, ArbreA a) {
    if (i == w.length())

```

```

        a.terminal[p] = true;
    else {
        int t = a.n++; a.terminal[t] = true;
        ajouterArcA(p, i, w.length(), t, a);
    }
}

static int briserArc(int p, ListeA arc, int k, ArbreA a) {
    int r = a.n++;
    ajouterArcA(r, k, arc.afin, arc.sommet, a);
    arc.sommet = r; arc.afin = k;
    return r;
}

```

Question 18 `static ArbreA nouvelArbre(String w) {`
 `ArbreA a = new ArbreA();`
 `for (int i = 0; i <= w.length(); i++)`
 `insérerSuffixe(i, w, 0, a);`
 `return a;`
`}`

Question 19 En partant à partir de la racine, on avance dans le chemin vers p correspondant au facteur x en décidant sur la première lettre des étiquettes des arcs issus des sommets rencontrés. Le choix entre les arcs se fait séquentiellement et prend $O(k)$ opérations. Au total, le calcul de p se fait en $O(E \times k)$.

Question 20 Le facteur $x = ay$ correspond au sommet p . Alors soit p est un noeud terminal, soit p est un sommet de degré au moins 2. Dans le premier cas, x est un suffixe, et donc également y . Il lui correspond donc bien un noeud terminal q . Dans le deuxième cas, il existe deux facteurs xz et xz' ($zz' \neq \varepsilon$). Comme $x = ay$, il existe aussi deux facteurs yz et yz' , et donc le facteur y correspond à un noeud q de degré au moins supérieur à deux. Il correspond bien à un sommet dans l'arbre des préfixes.

Question 21 `static int[] lienSuffixe(String w, ArbreA a) {`
 `int[] ls = new int[a.n];`
 `for (int i = 0; i < ls.length; ++i) ls[i] = -1;`
 `faireLienSuffixe(0, ls, w, a);`
 `return ls;`
`}`

`static void faireLienSuffixe(int p, int[] ls, String w, ArbreA a) {`
 `if (p != 0 && ls[p] != -1)`
 `for (ListeA x = a.succ[p] ; x != null; x = x.suivant) {`
 `int q = x.sommet;`
 `int q1 = descendantDe(ls[p], x.debut, x.afin, w, a);`
 `faireLienSuffixe(q1, ls, w, a);`
 `ls[q] = q1;`
 `faireLienSuffixe(q, ls, w, a);`
 `}`
`}`