Module HMIN121M

Recherche exacte d'un ensemble de motifs

Algorithme de Aho-Corasick

Sèverine Bérard

décembre 2018

Institut des Sciences de l'Evolution-Montpellier ISE-M – FDS, Université de Montpellier

Sommaire

Plan du cours

Définition du problème

- Généralisation du problème de recherche exacte d'un motif dans un texte

Définition du problème

- Généralisation du problème de recherche exacte d'un motif dans un texte
- Soient $\mathcal{P} = \{P_1, P_2, \ldots, P_k\}$ un ensemble de k motifs avec $\Sigma|P_i| = m$ et T un texte de longueur n

Définition du problème

- Généralisation du problème de recherche exacte d'un motif dans un texte
- Soient $\mathcal{P} = \{P_1, P_2, \ldots, P_k\}$ un ensemble de k motifs avec $\Sigma|P_i| = m$ et T un texte de longueur n

Définition du problème

- Généralisation du problème de recherche exacte d'un motif dans un texte
- Soient $\mathcal{P} = \{P_1, P_2, \ldots, P_k\}$ un ensemble de k motifs avec $\Sigma|P_i| = m$ et T un texte de longueur n

Recherche exacte d'un ensemble de motifs

Trouver toutes les occurrences des motifs de $\mathcal{P}$ dans T

Définition du problème

- Généralisation du problème de recherche exacte d'un motif dans un texte
- Soient $\mathcal{P} = \{P_1, P_2, \ldots, P_k\}$ un ensemble de k motifs avec $\Sigma|P_i| = m$ et T un texte de longueur n

Recherche exacte d'un ensemble de motifs

Trouver toutes les occurrences des motifs de $\mathcal{P}$ dans T

- De manière naïve, on pourrait utiliser pour chaque motif, une méthode de recherche en temps linéaire en la taille du texte (+ pré-traitement des motifs), ce qui donnerait une complexité en $O(kn + m)$

Définition du problème

- Généralisation du problème de recherche exacte d'un motif dans un texte
- Soient $\mathcal{P} = \{P_1, P_2, \ldots, P_k\}$ un ensemble de k motifs avec $\Sigma|P_i| = m$ et T un texte de longueur n

Recherche exacte d'un ensemble de motifs

Trouver toutes les occurrences des motifs de $\mathcal{P}$ dans T

- De manière naïve, on pourrait utiliser pour chaque motif, une méthode de recherche en temps linéaire en la taille du texte (+ pré-traitement des motifs), ce qui donnerait une complexité en $O(kn + m)$
- Mais on peut mieux faire ! But : $O(m + n + q)$ où q est le nombre d'occurrences des motifs dans le texte

Plan du cours

Arbre des mots-clés ou dictionnaire

Définition

Le *dictionnaire* pour un ensemble de motifs $\mathcal{P}$ est une arborescence $\mathcal{K}$ (c.-à-d. un arbre orienté enraciné) satisfaisant 3 conditions :

Arbre des mots-clés ou dictionnaire

Définition

Le *dictionnaire* pour un ensemble de motifs $\mathcal{P}$ est une arborescence $\mathcal{K}$ (c.-à-d. un arbre orienté enraciné) satisfaisant 3 conditions :

1. chaque arc est étiqueté avec exactement un caractère

Arbre des mots-clés ou dictionnaire

Définition

Le *dictionnaire* pour un ensemble de motifs $\mathcal{P}$ est une arborescence $\mathcal{K}$ (c.-à-d. un arbre orienté enraciné) satisfaisant 3 conditions :

1. chaque arc est étiqueté avec exactement un caractère
2. 2 arcs sortants d'un même nœud ont des étiquettes différentes

Arbre des mots-clés ou dictionnaire

Définition

Le *dictionnaire* pour un ensemble de motifs $\mathcal{P}$ est une arborescence $\mathcal{K}$ (c.-à-d. un arbre orienté enraciné) satisfaisant 3 conditions :

1. chaque arc est étiqueté avec exactement un caractère
2. 2 arcs sortants d'un même nœud ont des étiquettes différentes
3. chaque motif P_i de $\mathcal{P}$ correspond à un nœud v de $\mathcal{K}$, tel que l'étiquette-chemin de v soit exactement P_i, et chaque feuille de $\mathcal{K}$ correspond à un motif de $\mathcal{P}$

Arbre des mots-clés ou dictionnaire

Définition

Le *dictionnaire* pour un ensemble de motifs $\mathcal{P}$ est une arborescence $\mathcal{K}$ (c.-à-d. un arbre orienté enraciné) satisfaisant 3 conditions :

1. chaque arc est étiqueté avec exactement un caractère
2. 2 arcs sortants d'un même nœud ont des étiquettes différentes
3. chaque motif P_i de $\mathcal{P}$ correspond à un nœud v de $\mathcal{K}$, tel que l'étiquette-chemin de v soit exactement P_i, et chaque feuille de $\mathcal{K}$ correspond à un motif de $\mathcal{P}$

Remarque : certains nœuds de $\mathcal{K}$ sont donc numérotés pour les mettre en correspondance avec les motifs de $\mathcal{P}$ qu'ils représentent

Arbre des mots-clés ou dictionnaire

Définition

Le *dictionnaire* pour un ensemble de motifs $\mathcal{P}$ est une arborescence $\mathcal{K}$ (c.-à-d. un arbre orienté enraciné) satisfaisant 3 conditions :

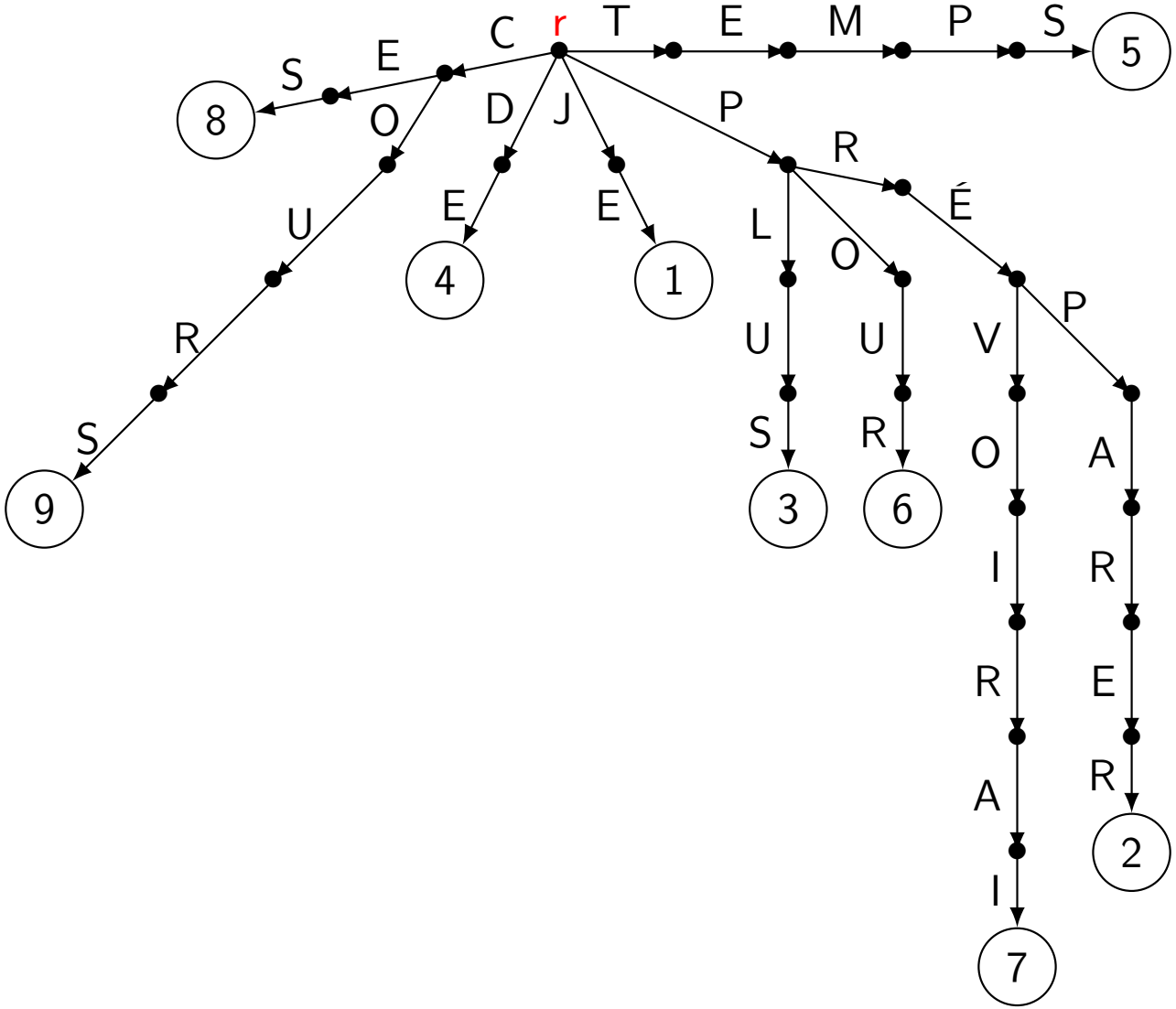
1. chaque arc est étiqueté avec exactement un caractère
2. 2 arcs sortants d'un même nœud ont des étiquettes différentes
3. chaque motif P_i de $\mathcal{P}$ correspond à un nœud v de $\mathcal{K}$, tel que l'étiquette-chemin de v soit exactement P_i, et chaque feuille de $\mathcal{K}$ correspond à un motif de $\mathcal{P}$

Remarque : certains nœuds de $\mathcal{K}$ sont donc numérotés pour les mettre en correspondance avec les motifs de $\mathcal{P}$ qu'ils représentent

Toutes les feuilles sont numérotées, ainsi que des nœuds internes si des motifs de $\mathcal{P}$ sont préfixes d'autres motifs de $\mathcal{P}$

Exemple

$\mathcal{P} = \{$JE, PRÉVOIRAI, PLUS, DE, TEMPS, POUR, PRÉPARER, CES, COURS$\}$



Plan du cours

Recherche de motifs en $O(nm)$

- Supposons que l'on dispose du dictionnaire $\mathcal{K}$ de l'ensemble $\mathcal{P}$, notons r sa racine et *num* l'application qui à tout nœud associe son éventuel numéro

Recherche de motifs en $O(nm)$

- Supposons que l'on dispose du dictionnaire $\mathcal{K}$ de l'ensemble $\mathcal{P}$, notons r sa racine et *num* l'application qui à tout nœud associe son éventuel numéro

---

Algorithme : Recherche naïve

Données : Le texte T de longueur n et le dictionnaire $\mathcal{K}$ de l'ensemble des mots $\mathcal{P}$ de longueur totale m

pour *(pos de* 1 *à* n*)* **faire**
  $v := r$; $j := pos$;
  tant que *(*$\exists$ *un arc* (v, v') *étiqueté par* $T[j]$*)* **faire**
    si $(num(v) = i)$ **alors**
      Écrire("P_i apparaît à la position ", pos) ;
    $v := v'$;
    $j := j + 1$;

Accélération : généralisation de KMP

- Dans l'algorithme précédent, on cherche tous les mots de $\mathcal{P}$ à partir de chaque position de début possible dans T

Accélération : généralisation de KMP

- Dans l'algorithme précédent, on cherche tous les mots de $\mathcal{P}$ à partir de chaque position de début possible dans T

- Quand on ne peut plus avancer (situation d'échec ou feuille), on reprend la recherche à partir de la position suivante dans T

Accélération : généralisation de KMP

- Dans l'algorithme précédent, on cherche tous les mots de $\mathcal{P}$ à partir de chaque position de début possible dans T
- Quand on ne peut plus avancer (situation d'échec ou feuille), on reprend la recherche à partir de la position suivante dans T
- L'idée ici est de reprendre la comparaison à partir de ce qu'on a déjà reconnu, de la même manière que dans les algorithmes MP et KMP

Accélération : généralisation de KMP

- Dans l'algorithme précédent, on cherche tous les mots de $\mathcal{P}$ à partir de chaque position de début possible dans T
- Quand on ne peut plus avancer (situation d'échec ou feuille), on reprend la recherche à partir de la position suivante dans T
- L'idée ici est de reprendre la comparaison à partir de ce qu'on a déjà reconnu, de la même manière que dans les algorithmes MP et KMP

$\rightarrow$ Liens échecs

Accélération : généralisation de KMP

- Dans l'algorithme précédent, on cherche tous les mots de $\mathcal{P}$ à partir de chaque position de début possible dans T
- Quand on ne peut plus avancer (situation d'échec ou feuille), on reprend la recherche à partir de la position suivante dans T
- L'idée ici est de reprendre la comparaison à partir de ce qu'on a déjà reconnu, de la même manière que dans les algorithmes MP et KMP

$\rightarrow$ Liens échecs

Hypothèse simplificatrice (que l'on contournera à la fin)

Pas de motif de $\mathcal{P}$ sous-chaîne d'un autre motif de $\mathcal{P}$.
Donc pas de nœud interne de $\mathcal{K}$ numéroté

Mise en place des liens échec

- Chaque nœud v de $\mathcal{K}$ est étiqueté avec la chaîne obtenue en concaténant dans l'ordre tous les caractères sur le chemin de la racine jusqu'à v, on note cette chaîne $\mathcal{L}(v)$

Mise en place des liens échec

- Chaque nœud v de $\mathcal{K}$ est étiqueté avec la chaîne obtenue en concaténant dans l'ordre tous les caractères sur le chemin de la racine jusqu'à v, on note cette chaîne $\mathcal{L}(v)$
- Pour chaque nœud v de $\mathcal{K}$, on définit $lp(v)$ la longueur du plus long suffixe propre de $\mathcal{L}(v)$ qui est préfixe d'un mot de $\mathcal{P}$

Mise en place des liens échec

- Chaque nœud v de $\mathcal{K}$ est étiqueté avec la chaîne obtenue en concaténant dans l'ordre tous les caractères sur le chemin de la racine jusqu'à v, on note cette chaîne $\mathcal{L}(v)$
- Pour chaque nœud v de $\mathcal{K}$, on définit $lp(v)$ la longueur du plus long suffixe propre de $\mathcal{L}(v)$ qui est préfixe d'un mot de $\mathcal{P}$

Mise en place des liens échec

- Chaque nœud v de $\mathcal{K}$ est étiqueté avec la chaîne obtenue en concaténant dans l'ordre tous les caractères sur le chemin de la racine jusqu'à v, on note cette chaîne $\mathcal{L}(v)$
- Pour chaque nœud v de $\mathcal{K}$, on définit $lp(v)$ la longueur du plus long suffixe propre de $\mathcal{L}(v)$ qui est préfixe d'un mot de $\mathcal{P}$

Lemme

Soit α le suffixe de $\mathcal{L}(v)$ de longueur $lp(v)$. Alors, il y a un unique nœud dans $\mathcal{K}$ d'étiquette α

Mise en place des liens échec

- Chaque nœud v de $\mathcal{K}$ est étiqueté avec la chaîne obtenue en concaténant dans l'ordre tous les caractères sur le chemin de la racine jusqu'à v, on note cette chaîne $\mathcal{L}(v)$
- Pour chaque nœud v de $\mathcal{K}$, on définit $lp(v)$ la longueur du plus long suffixe propre de $\mathcal{L}(v)$ qui est préfixe d'un mot de $\mathcal{P}$

Lemme

Soit α le suffixe de $\mathcal{L}(v)$ de longueur $lp(v)$. Alors, il y a un unique nœud dans $\mathcal{K}$ d'étiquette α

- Pour un nœud v de $\mathcal{K}$, soit n_v l'unique nœud de $\mathcal{K}$ étiqueté avec α (le suffixe de $\mathcal{L}(v)$ de longueur $lp(v)$). Quand $lp(v) = 0$, alors n_v est la racine de $\mathcal{K}$

Mise en place des liens échec

- Chaque nœud v de $\mathcal{K}$ est étiqueté avec la chaîne obtenue en concaténant dans l'ordre tous les caractères sur le chemin de la racine jusqu'à v, on note cette chaîne $\mathcal{L}(v)$
- Pour chaque nœud v de $\mathcal{K}$, on définit $lp(v)$ la longueur du plus long suffixe propre de $\mathcal{L}(v)$ qui est préfixe d'un mot de $\mathcal{P}$

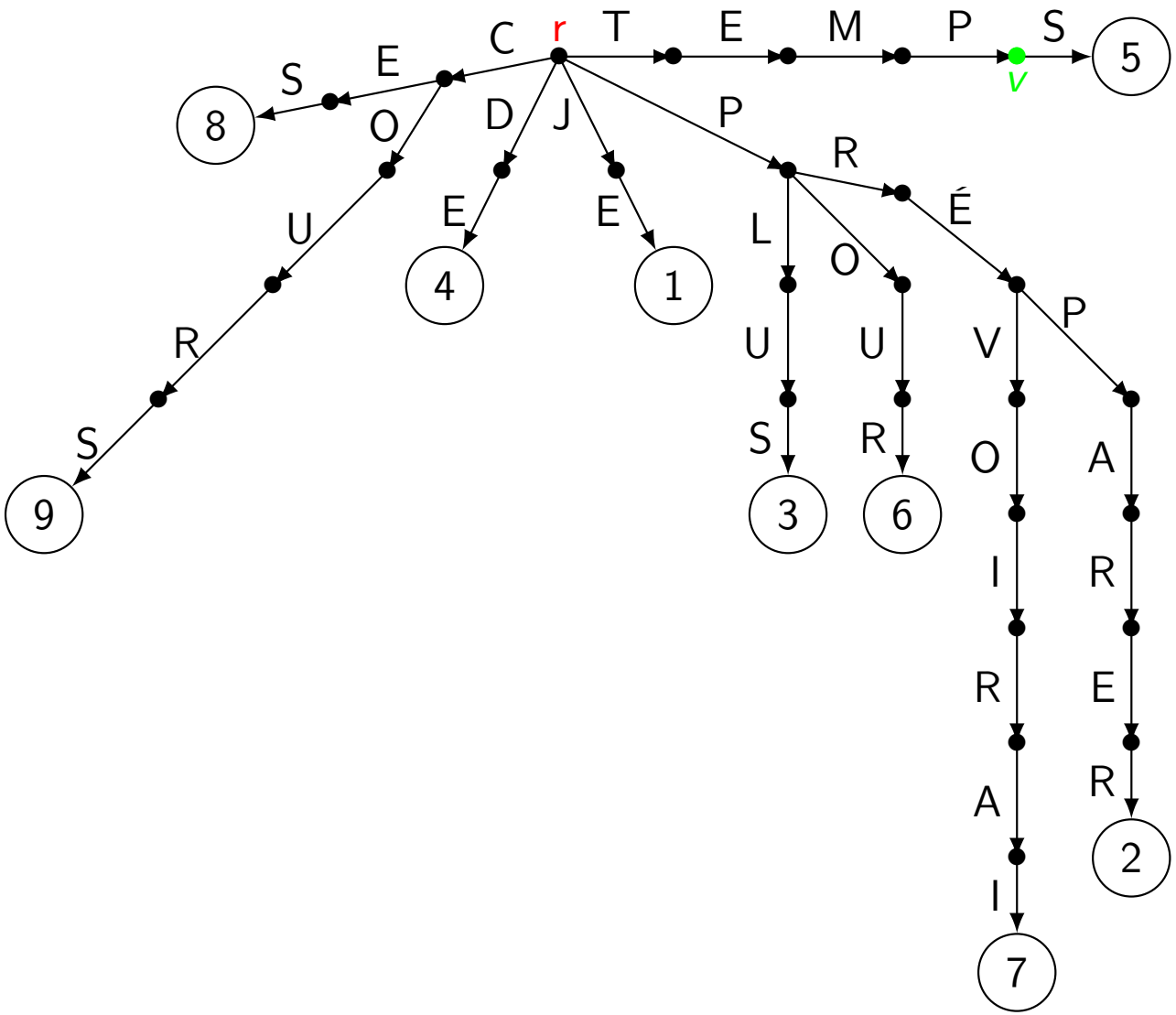
Lemme

Soit α le suffixe de $\mathcal{L}(v)$ de longueur $lp(v)$. Alors, il y a un unique nœud dans $\mathcal{K}$ d'étiquette α

- Pour un nœud v de $\mathcal{K}$, soit n_v l'unique nœud de $\mathcal{K}$ étiqueté avec α (le suffixe de $\mathcal{L}(v)$ de longueur $lp(v)$). Quand $lp(v) = 0$, alors n_v est la racine de $\mathcal{K}$
- On appelle lien échec l'arc (n, n_v)

Exemple

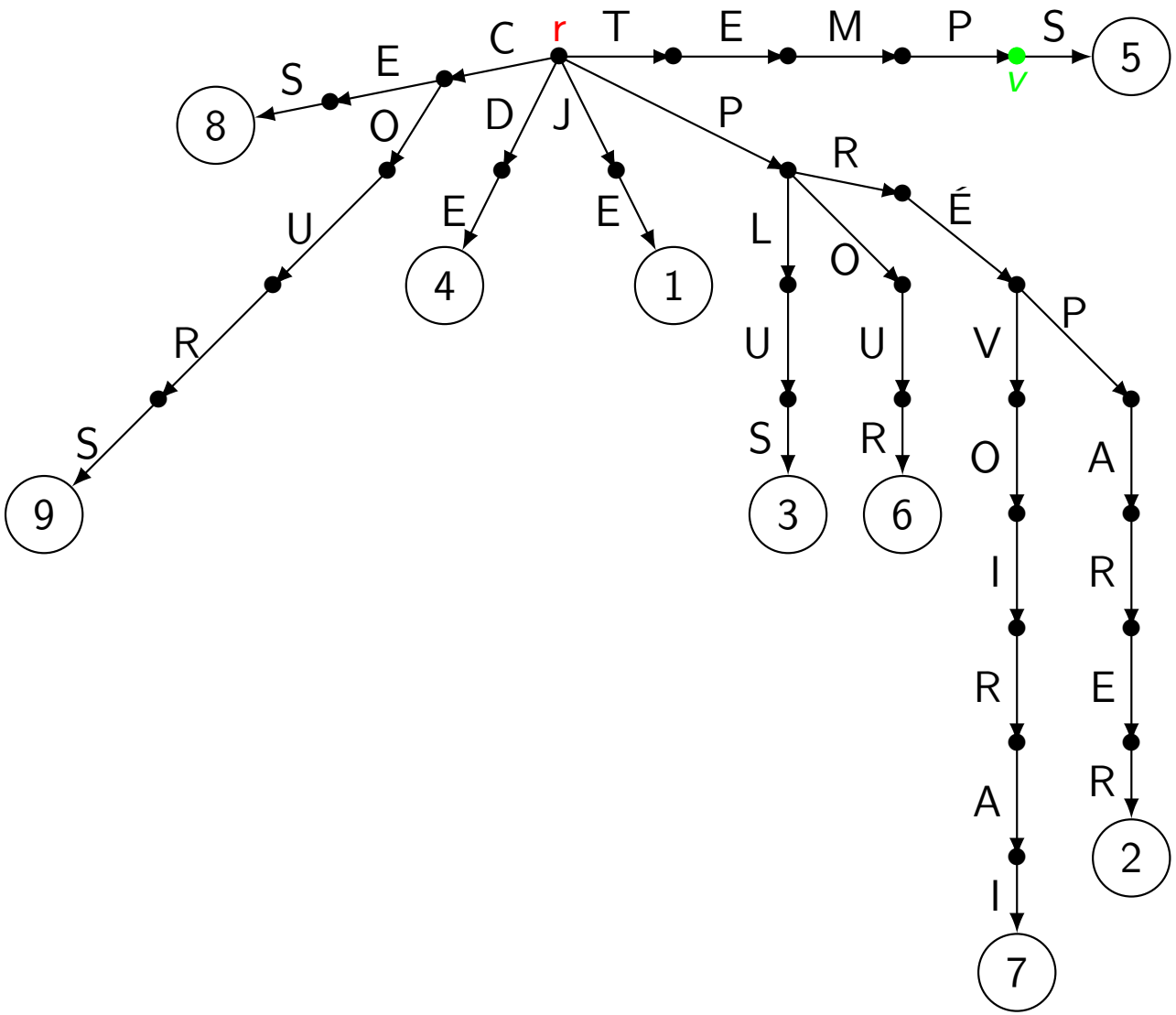
$\mathcal{L}(v)$=TEMP



Exemple

$\mathcal{L}(v)$=TEMP

$lp(v)$=1

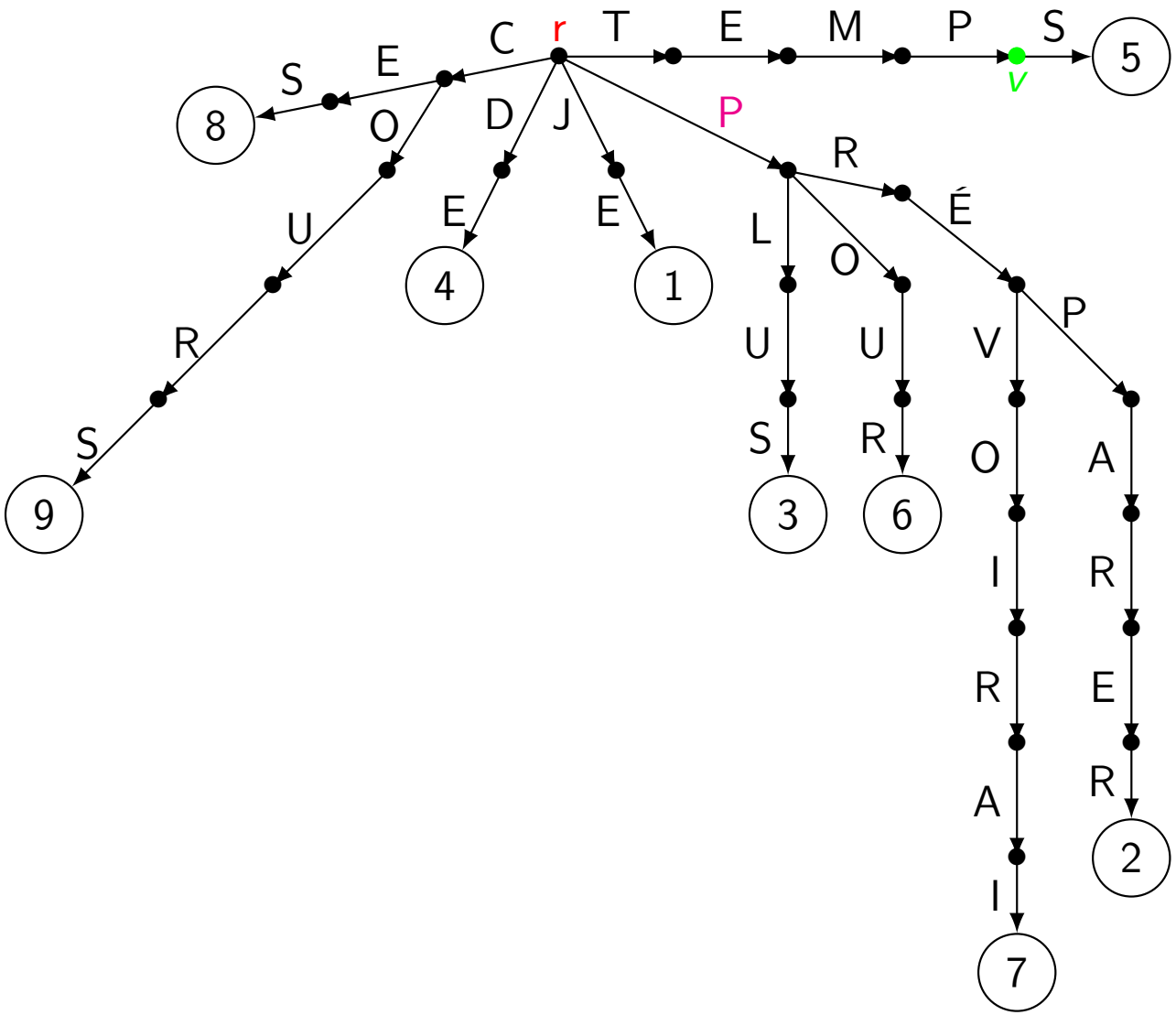


Exemple

$\mathcal{L}(v)$=TEMP

$lp(v)$=1

α=P



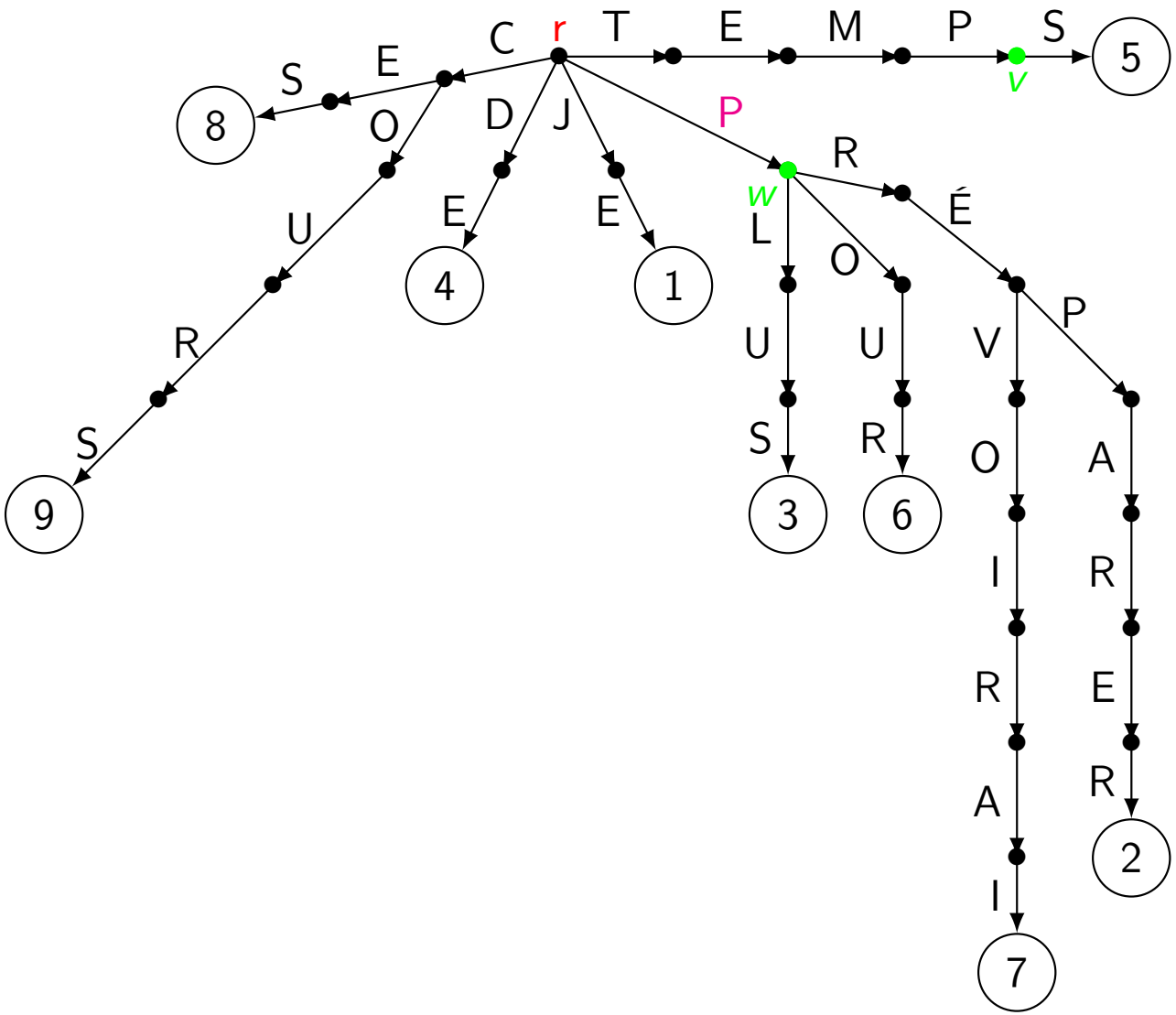
Exemple

$\mathcal{L}(v)$=TEMP

$lp(v)$=1

α=P

$n_v = w$



Exemple

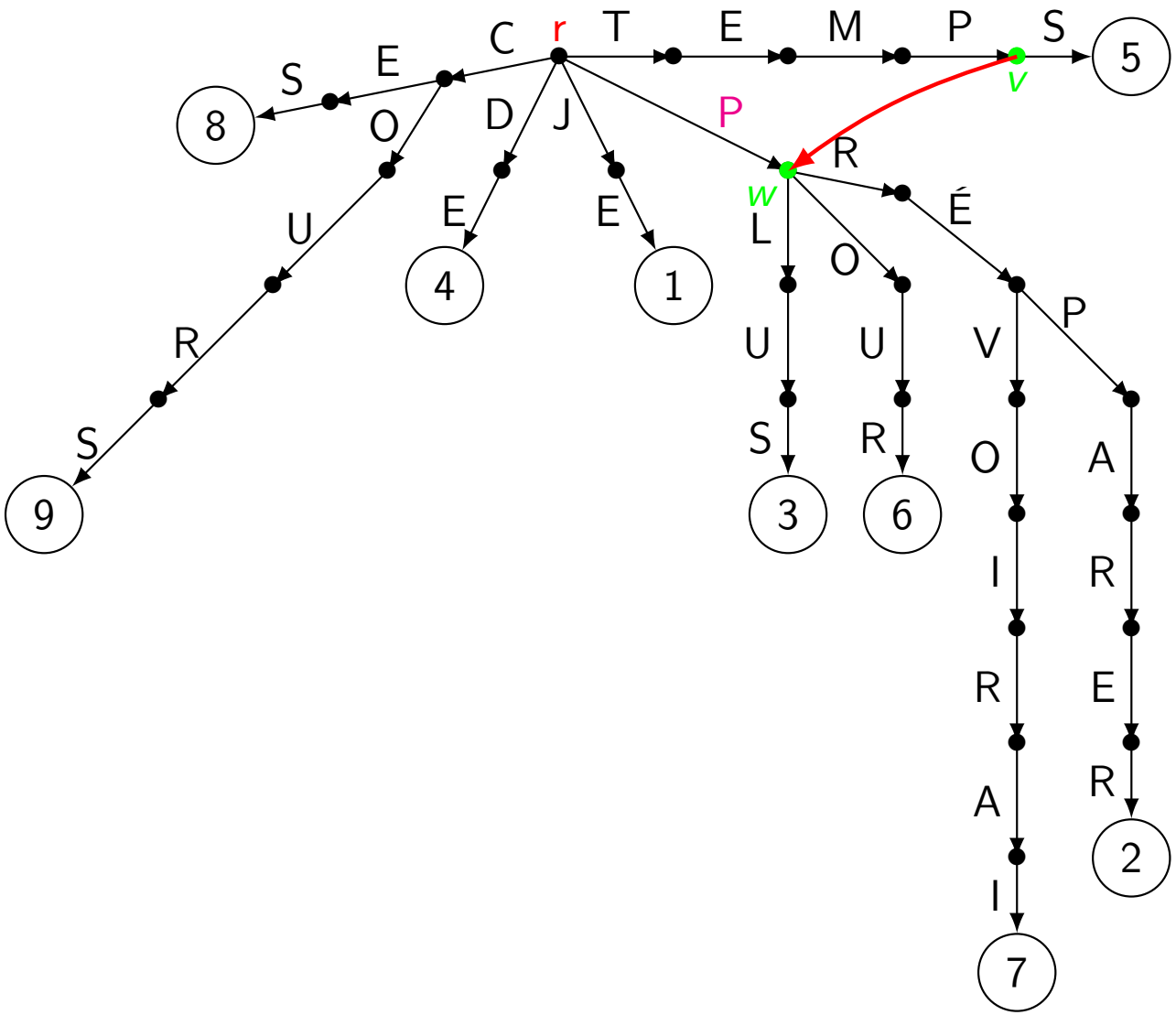
$\mathcal{L}(v)$=TEMP

$lp(v)$=1

α=P

n_v =w

Lien échec (v, n_v)



Exemple

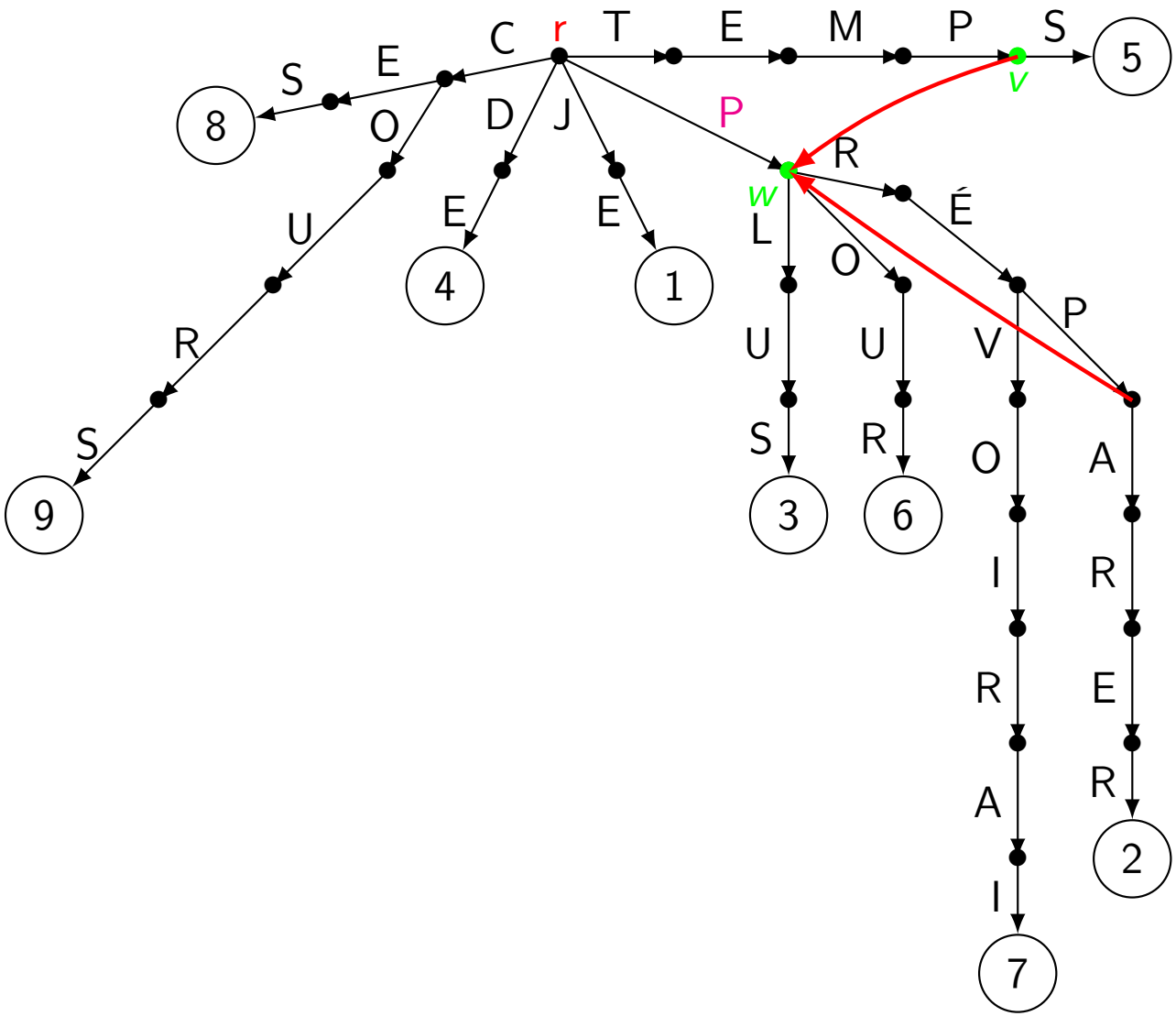
$\mathcal{L}(v)$=TEMP

$lp(v)$=1

α=P

n_v =w

Lien échec (v, n_v)



Seulement 2 liens échecs qui ne remontent pas vers la racine

Recherche de motifs en $O(n)$ (version Gusfield)

Algorithme : AC search

Données : Le texte T de longueur n et le dictionnaire $\mathcal{K}$ avec ses liens échec de l'ensemble des mots $\mathcal{P}$ de longueur totale m

$pos := 1$; $j := 1$; $v :=$ racine de $\mathcal{K}$;

répéter

  tant que *($\exists$ un arc (v, v') étiqueté par $T[j]$)* **faire**

    si *($num(v') = i$)* **alors**

      Écrire("P_i apparaît à la position ", pos) ;

    $v := v'$;

    $j := j + 1$;

  $v := n_v$;

  $pos := j - lp(v)$;

jusqu'à *($j > n$)*;

Recherche de motifs en $O(n)$ (version modifiée)

On avait remarqué en cours que dans certains cas, l'algorithme précédent ne pouvait plus avancer. Voici une version qui n'a pas ce problème :

Algorithme : AC search

Données : Le texte T de longueur n et le dictionnaire $\mathcal{K}$ avec ses liens échec de l'ensemble des mots $\mathcal{P}$ de longueur totale m

$pos := 1$; $j := 1$; $v :=$ racine de $\mathcal{K}$;

répéter

  tant que *($\exists$ un arc (v, v') étiqueté par $T[j]$)* **faire**

    si *($num(v') = i$)* **alors**

      Écrire("P_i apparaît à la position ", pos) ;

    $v := v'$;

    $j := j + 1$;

  si *($v =$ racine de $\mathcal{K}$)* **alors** $j := j + 1$; $pos := j$;

  sinon $v := n_v$; $pos := j - lp(v)$;

jusqu'à *($j > n$)*;

Calcul des liens échec en $O(m)$

L'algorithme suivant calcule pour un nœud v son lien échec

Algorithme : n_v

Données : Le dictionnaire $\mathcal{K}$ de racine r

$v' :=$ le parent de v dans $\mathcal{K}$; $w := n_{v'}$;

$x :=$ le caractère de l'arête (v', v) ;

tant que *(∄ un arc sortant de w étiqueté par x) ET ($w \neq r$)* **faire**

  $w := n_w$;

si *(∃ un arc (w, w') sortant de w étiqueté par x)* **alors** $n_v := w'$;

sinon $n_v := r$;

Il faut utiliser cet algorithme pour tous les nœuds de $\mathcal{K}$ dans un parcours en largeur, après avoir initialisé le lien échec de la racine : $n_r = r$

$\Rightarrow$ Complexité totale : $O(m)$

Plan du cours

Aho-Corasick complet : ôter l'hypothèse simplificatrice

- Jusqu'à présent, on supposait que l'ensemble $\mathcal{P}$ ne contenait pas de motif sous-chaîne d'un autre motif de $\mathcal{P}$

Aho-Corasick complet : ôter l'hypothèse simplificatrice

- Jusqu'à présent, on supposait que l'ensemble $\mathcal{P}$ ne contenait pas de motif sous-chaîne d'un autre motif de $\mathcal{P}$
- Nœud interne de $\mathcal{K}$ étiqueté avec un numéro non pris en compte dans l'algorithme de recherche AC-search

Aho-Corasick complet : ôter l'hypothèse simplificatrice

- Jusqu'à présent, on supposait que l'ensemble $\mathcal{P}$ ne contenait pas de motif sous-chaîne d'un autre motif de $\mathcal{P}$
- Nœud interne de $\mathcal{K}$ étiqueté avec un numéro non pris en compte dans l'algorithme de recherche AC-search

Aho-Corasick complet : ôter l'hypothèse simplificatrice

- Jusqu'à présent, on supposait que l'ensemble $\mathcal{P}$ ne contenait pas de motif sous-chaîne d'un autre motif de $\mathcal{P}$
- Nœud interne de $\mathcal{K}$ étiqueté avec un numéro non pris en compte dans l'algorithme de recherche AC-search

Lemme

Supposons que dans un dictionnaire $\mathcal{K}$ il existe une suite (possiblement vide) de liens échec depuis un nœud v vers un nœud numéroté i. Alors le motif P_i apparaît dans T et se termine à la position courante j à chaque fois que le nœud v est atteint durant la phase de recherche

Aho-Corasick complet : ôter l'hypothèse simplificatrice

- Jusqu'à présent, on supposait que l'ensemble $\mathcal{P}$ ne contenait pas de motif sous-chaîne d'un autre motif de $\mathcal{P}$
- Nœud interne de $\mathcal{K}$ étiqueté avec un numéro non pris en compte dans l'algorithme de recherche AC-search

Lemme

Supposons que dans un dictionnaire $\mathcal{K}$ il existe une suite (possiblement vide) de liens échec depuis un nœud v vers un nœud numéroté i. Alors le motif P_i apparaît dans T et se termine à la position courante j à chaque fois que le nœud v est atteint durant la phase de recherche

Corollaire

Supposons qu'un nœud v a été atteint pendant la phase de recherche. Alors le motif P_i apparaît dans T (terminant en j) :

1. si v est numéroté par i
2. ou s'il existe une suite de liens échec de v vers un nœud numéroté i

Algorithme de Aho-Corasick complet

Algorithme : AC search complet

Données : Le texte T de longueur n et le dictionnaire $\mathcal{K}$ avec ses liens échec de l'ensemble des mots $\mathcal{P}$ de longueur totale m

$pos := 1$; $j := 1$; $v :=$ racine de $\mathcal{K}$;

répéter

  tant que *($\exists$ un arc (v, v') étiqueté par $T[j]$)* **faire**

    si *($num(v') = i$) OU ($\exists$ une suite de liens échec de v vers un nœud numéroté i)* **alors**

      Écrire("P_i apparaît à la position ", pos) ;

    $v := v'$;

    $j := j + 1$;

  si *($v =$ racine de $\mathcal{K}$)* **alors** $j := j + 1$; $pos := j$;

  sinon $v := n_v$; $pos := j - lp(v)$;

jusqu'à $(j > n)$;

Détails d'implémentation

- Les énoncés précédents sont "haut niveau", il faut préciser quelques détails d'implémentation pour atteindre la complexité annoncée : $O(m + n + q)$

Pas au programme de l'examen 2018-19

Plan du cours

Références

Toute cette présentation est basée sur la section 3.4 du livre suivant :

[Gusfield, 97] Dan Gusfield, **Algorithms on Strings, Trees and Sequences** - Computer Science and Computational Biology, University of California, Davis. ISBN :9780521585194. Août 1997. *En anglais*