Quelques éléments d'algorithmique du texte

felipe@iro.umontreal.ca

RALI
Dept. Informatique et Recherche Opérationnelle
Université de **Montréal**

V0.7

Last compiled: 8 octobre 2019

- Pour une très bonne exposition aux algorithmes du texte, lire : ***[Crochemore et al., 2001]***.
- Pour la partie Search, lire ***[Bowe, 2010]***

Plan

rali

Regroupement de noms propres *[Cohen et al., 2003]*

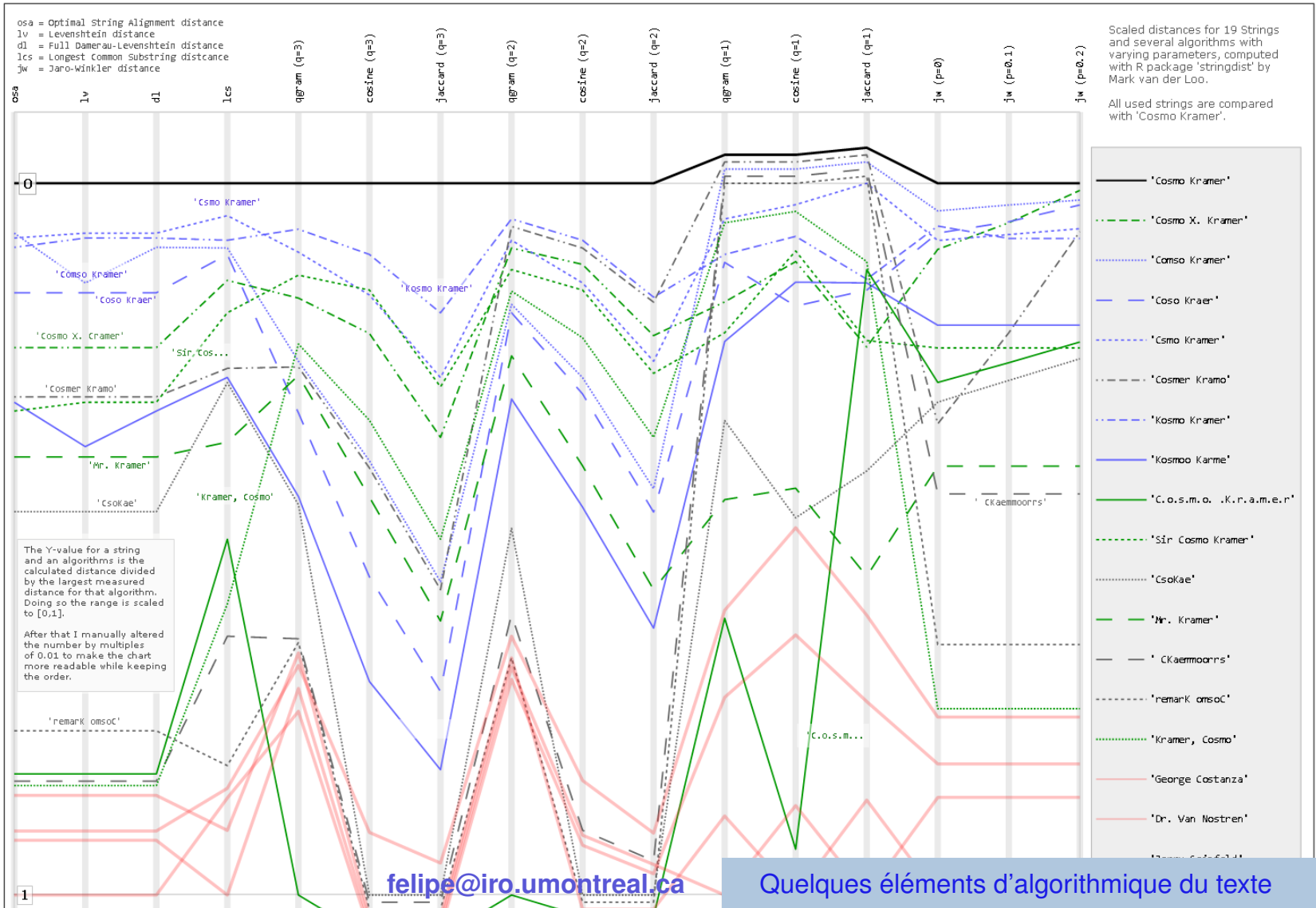
- identifier des doublons dans des noms propres (auteurs, restaurants, etc.)

  philippe's the original, "1001 n. alameda st.", "los angeles", "213/628-3
  philippe the original, "1001 n. alameda st.", "chinatown", "213-628-378

- datasets : `https://www.cs.utexas.edu/users/ml/riddle/data.html`

Regroupement de noms propres

https://www.joyofdata.de/blog/comparison-of-string-distance-algorithms/



Une phrases découle t-elle d'une autre ?

a) A smiling costumed woman is holding an umbrella

b) A happy woman in a fairy costume holds an umbrella

- selon SNLI *[Bowman et al., 2015]*, l'étiquette est **N**eutral (a**N**b)

- Un **baseline** consiste à utiliser la distance d'édition entre la prémisse (a) et la conclusion (b) comme trait
- Une variante de cela est disponible dans EOP (Excitement Open Plateform) *[Padó et al., 2015]*

rali

Application à la correction orthographique

- De nombreux types d'erreurs (qui peuvent être cumulés)
  - fautes d'orthographe : éphémaire, opignon, etc.
  - fautes de frappe : qviron, errueurs
  - impropriété (malaproprism) : il est âpre / Il est râpe
  - mots d'emprunts : tu n'as qu'à piper la sortie de ta commande avant de faire ton debugging
  - fautes de grammaire : la femme que j'ai vu.
  - erreurs de césure : il est con fondant/confondant
  - formes extra-lexicales : U2, OQP, etc.

- Les transparents qui suivent sont des idées simples qui illustrent comment on peut utiliser un modèle de langue pour faire un mini-correcteur orthographique.

rali

Application à la correction orthographique

Soit $\mathcal{L}$ un lexique fini contenant de nombreux mots d'une langue.

- Identifier dans un texte T_1^n les mots inconnus de $\mathcal{L}$ (on écarte ici les formes extra-lexicales chiffrées)
- Pour chaque mot inconnu u, réunir les mots de $\mathcal{L}$ qui lui sont proches au sens d'une distance à définir.

  soit v_u l'ensemble de ces mots proches de la forme u

- Si $h(u)$ désigne l'historique de u dans le texte, alors corriger u en $\hat{u}$:

$$\hat{u} = \underset{v \in v_u}{\operatorname{argmax}}\, p(v|h(u))$$

j'aime la apriture $\Rightarrow$ j'aime la aperture / friture / aptitude

Distance d'édition

- les mots les plus proches de asfalte :

asphalte	2	spalter	3	state	3
svelte	3	falce	3	fate	3
faute	3	faîte	3	halte	3

- Les mots les plus proches de ammateur :

armateur	1	amateur	1	aviateur	2
amateurs	2	animateur	2	armateurs	2
orateur	3	radiateur	3	sénateur	3

- Les mots les plus proches de courier :

courier	0	courrier	1	courtier	1
courir	1	courber	1	sourir	2
tourner	2	usurier	2	écourter	2

rali

Distance entre deux mots

- Une fonction $d : \Sigma^\star \times \Sigma^\star \rightarrow \mathcal{R}$ est une distance sur $\Sigma^\star$ si pour tout $u, v \in \Sigma^\star$ on a :
  - $d(u, v) \geq 0$
  - $d(u, v) = 0 \iff u = v$
  - $d(u, v) = d(v, u)$
  - $d(u, v) \leq d(u, w) + d(w, v) \quad \forall w \in \Sigma^\star$

- Si l'une de ces propriété n'est pas vérifiée, on parle plutôt de mesure de dissimilarité

Distance de Hamming

- Définie pour deux mots u et v de même longueur par le nombre de positions p où $u_p \neq v_p$

- Exemples :
  - d_h(ahuri,aluni) = 2
  - d_h(aluni,ahuri) = 2
  - d_h(salade,salace) = 1
  - d_h(salace,sagace) = 1
  - d_h(salade,sagace) = 2

Distance de Jaro-Winkler

- Adaptation par William E. Winkler (1999) d'une distance proposée par Matthew A. Jaro (1989)
- Adaptée pour les chaînes courtes (noms propres, mots de passe, etc.)
- Pas une distance (pas symétrique)

↪ Mesure de similarité entre 0 (absence de similarité) et 1

- Exemples :
  - d_{jw}(intimité,inimitié) = 0.96
  - d_{jw}(opinion,opignon) = 0.95
  - d_{jw}(bernard,bertrand) = 0.92
  - d_{jw}(lagacé,leveillé) = 0.57

rali

Distance de Jaro

$$d_j(x,y) = \frac{1}{3}\left(\frac{m}{|x|} + \frac{m}{|y|} + \frac{m-t}{m}\right)$$

- m est le nombre de caractères se correspondant dans x et y
  - **Def :** x_i et y_j se correspondent **ssi** :
    - $x_i = y_j$
    - $|i-j| \leq \rho \equiv \left\lfloor \frac{max(|x|,|y|)}{2} \right\rfloor$ -1
- t est le nombre de transpositions, cad le nombre de caractères qui diffèrent lorsqu'on dresse la séquence des caractères de chaque mot qui ont un correspondant dans l'autre chaîne, divisé par 2

Distance de Jaro

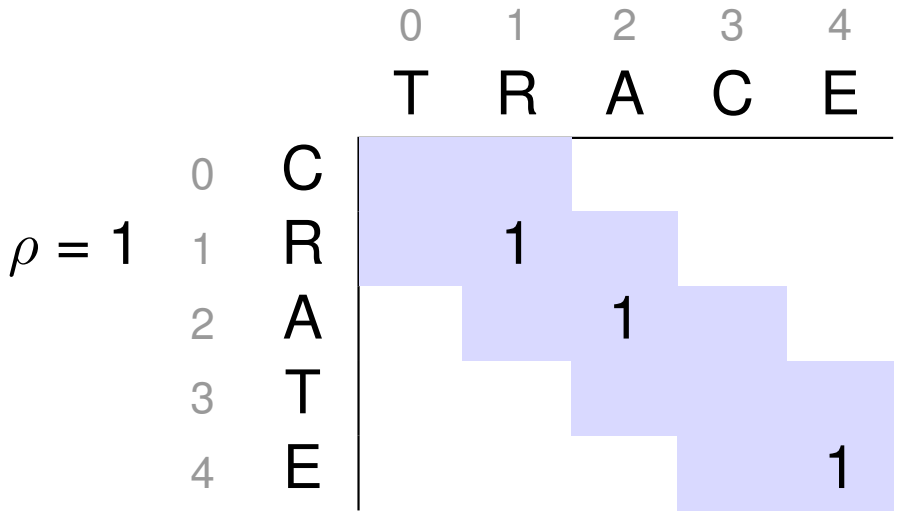
$\rho = 3$

		0	1	2	3	4	5	6
		B	E	R	N	A	R	D
0	B	1						
1	E		1					
2	R			1			1	
3	T							
4	R			1			1	
5	A					1		
6	N				1			
7	D							1

BERTRAND $\rightarrow$ B E R R A N D
BERNARD $\rightarrow$ B E R N A R D

- matchs = 7 ($m = 7$), transpositions = 1 ($t = 2/2 = 1$)
- $d_j(BERTRAND, BERNARD) = \frac{1}{3}\left(\frac{7}{8} + \frac{7}{7} + \frac{6}{7}\right) = 0.91$

Distance de Jaro



CRATE $\rightarrow$ R A E

TRACE $\rightarrow$ R A E

- $m = 3, t = 0$
- $d_j(CRATE, TRACE) = \frac{1}{3}\left(\frac{3}{5} + \frac{3}{5} + \frac{3}{5}\right) = 0.6$

Distance de Jaro-Winkler

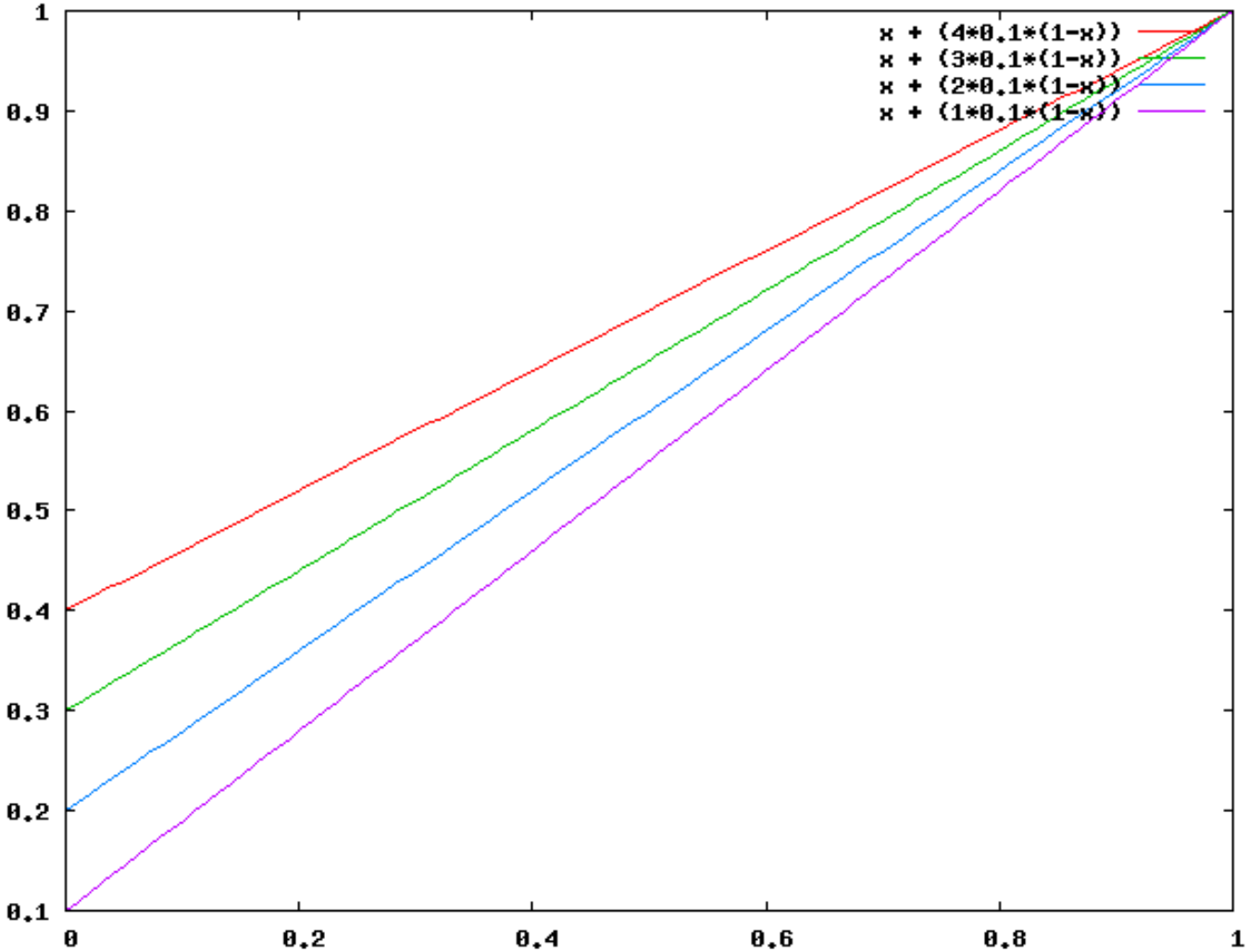
- Soit l la taille du plus grand préfixe commun entre les deux chaînes (max : $l = 4$)
- Soit p un coefficient ($p = 0.1$) :

$$d_{jw}(x, y) = d_j(x, y) + (l \times p \times (1 - d_j(x, y)))$$

intuition : favoriser les mots qui partagent un ”grand” préfixe

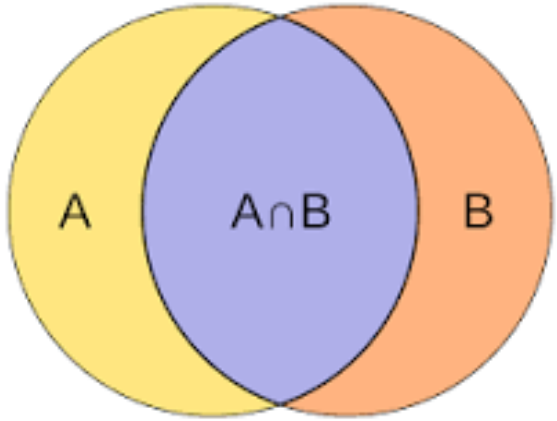
- $l = 3$ dans notre exemple (BERNARD, BERTRAND)
- d_{jw}(BERTRAND,BERNARD) = 0.937
- **note :** d_{jw}(CRATE,TRACE) = d_j(CRATE,TRACE)

Distance de Jaro-Winkler



- importance de la taille du préfixe entre les deux chaînes
- en abscisse : d_j, en ordonnée : d_{jw}

Dissimilarité de Jaccard



- indice de Jaccard : $J(A,B) = \frac{|A \cap B|}{|A \cup B|}$
- dissimilarité de Jaccard : $1 - J(A,b)$
- appliqué aux chaînes en considérant A et B comme des multisets (sacs de symboles)

	ch	hi	ie	en	ni	ic	he
chien	1	1	1	1			
niche	1				1	1	1
	J=1/7						

	c	h	i	e	n
chien	1	1	1	1	1
niche	1	1	1	1	1
	J=5/5				

Distance cosinus

- $cos(A, B) = \frac{A.B}{||A||_2.||B||_2}$

- on peut utiliser une représentation binaire
  $a_i = 1$ *si le symbole associé dans l'alphabet est présent dans A*
- ou utiliser une mesure comme tf-idf :
  - $tf_{\alpha,A}$: fréquence du symbole α dans la forme A
    $\log(tf_{\alpha,A} + 1)$
  - idf_α : inverse du nombre de formes ayant le symbole α
    $\log(idf_\alpha)$
  - tf.idf = produit des deux

Distance d'édition entre x et y

substitution substitution d'une lettre (l_1) de x par une lettre de (l_2) y ; soit sub(l_1,l_2) son coût
Ex : x =narine $\Rightarrow$ y =marine en substituant n à m dans x en position 1 dans x

insertion insertion d'une lettre (l) de y dans x ; soit ins(l) son coût
Ex : x =marine $\Rightarrow$ y =martine en insérant t en position 4 dans x

suppression opération inverse à la précédente, del(l) son coût

- Le nombre minimum d'opérations à appliquer pour transformer une chaîne en une autre est appelée la distance de Levenshtein [Levenshtein, 1966], ou encore la distance d'édition (edit distance).

rali

$d_L(voiture, moteur)$?

- Séquence de 4 opérations qui transforme la chaîne voiture en la chaîne moteur.

x	opération	coût
VOITURE	del(E)	1
VOITUR	sub(T,E)	1
VOIEUR	sub(I,T)	1
VOTEUR	sub(V,M)	1
MOTEUR		

Chaque opération, autre que *sub(a,a)* a un coût unitaire.

- Le coût de cette séquence de transformations est 4.
- Peut-on faire mieux ?

$d_L(voiture, moteur)$?

- Séquence de 4 opérations qui transforme la chaîne voiture en la chaîne moteur.

x	opération	coût
VOITURE	del(E)	1
VOITUR	sub(T,E)	1
VOIEUR	sub(I,T)	1
VOTEUR	sub(V,M)	1
MOTEUR		

Chaque opération, autre que *sub(a,a)* a un coût unitaire.

- Le coût de cette séquence de transformations est 4.
- Peut-on faire mieux ?
- Réponse en calculant une table d'édition

$d_L(voiture, moteur)$?

- $T[i,j] = d_L(x[1 \ldots i], y[1 \ldots j])$

(7)	(6)	M (1,d)	O (2,d)	T (3,d)	E (4,d)	U (5,d)	R (6,d)
V	(1,i)	(1,s)	(2,d)	(3,d)	(4,d)	(5,d)	(6,d)
O	(2,i)	(2,i)	(1,=)	(2,d)	(3,d)	(4,d)	(5,d)
I	(3,i)	(3,i)	(2,i)	(2,s)	(3,d)	(4,d)	(5,d)
T	(4,i)	(4,i)	(3,i)	(2,=)	(3,d)	(4,d)	(5,d)
U	(5,i)	(5,i)	(4,i)	(3,i)	(3,s)	(3,=)	(4,d)
R	(6,i)	(6,i)	(5,i)	(4,i)	(4,i)	(4,i)	(3,=)
E	(7,i)	(7,i)	(6,i)	(5,i)	(4,=)	(5,d)	(4,i)

- d(élétion) [1], i(nsertion), s(ubstitution), = (égalité)

1. Je sais c'est une horreur...

$d_L(voiture, moteur)$?

- On peut démontrer que :

$$T[i,j] = min \begin{cases} T[i-1,j-1] + sub(x[i], y[j]) \\ T[i-1,j] + ins(x[i]) \\ T[i,j-1] + del(y[j]) \end{cases}$$

- Cas particuliers aux bornes :

$$\begin{aligned} T[0,0] &= 0 \\ T[i,0] &= T[i-1,0] + ins(x[i]) \\ T[0,j] &= T[0,j-1] + del(y[j]) \end{aligned}$$

$d_L(voiture, moteur)$?

Require: $X = x_1, \ldots, x_N$ et $Y = y_1, \ldots, y_M$

function $Solve(i, j)$

if $i = 0$ **then**

if $j = 0$ **then**

return retourne 0

else

return $Solve(0, j-1) + del(Y[j])$

else

if $j = 0$ **then**

return $Solve(i-1, 0) + ins(X[i])$

else

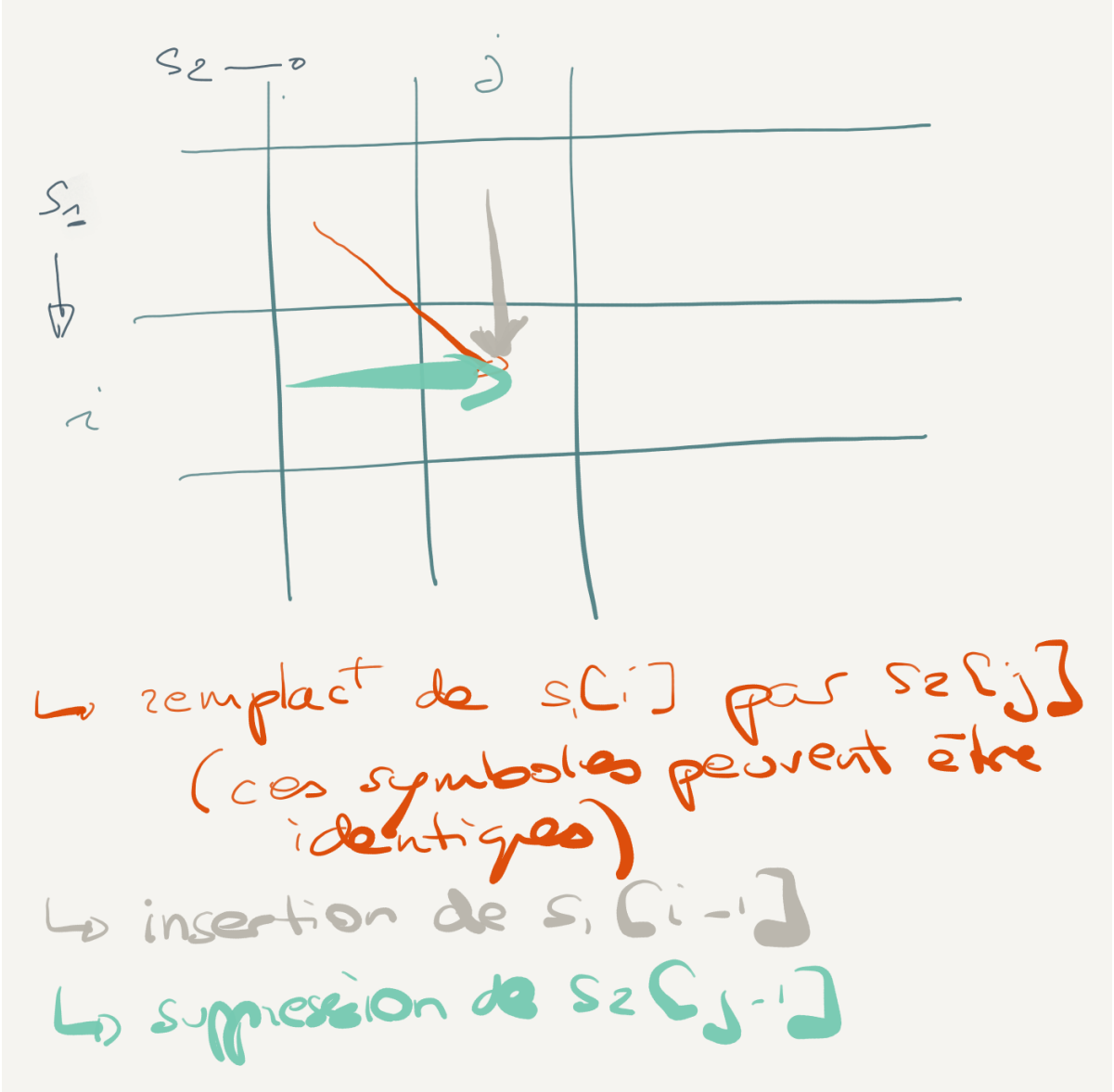
$i_1 \leftarrow Solve(i-1, j-1) + sub(X[i], Y[j])$

$i_2 \leftarrow Solve(i-1, j) + ins(X[i])$

$i_3 \leftarrow Solve(i, j-1) + del(Y[j])$

return $min(i_1, i_2, i_3)$

$d_L(voiture, moteur)$?



$$d_L(voiture, moteur)\,?$$

// init au bornes

for i $\leftarrow 0$ à I **do**

  T[i][0] $\leftarrow i$

for j $\leftarrow 1$ à J **do**

  T[0][j] $\leftarrow j$

// boucle principale

for i $\leftarrow 1$ à I **do**

  for j $\leftarrow 1$ à J **do**

$$\text{T[i][j]} = \min \begin{cases} T[i-1, j-1] & + & sub(x[i], y[j]) \\ T[i-1, j] & + & ins(x[i]) \\ T[i, j-1] & + & del(y[j]) \end{cases}$$

//Le résultat est dans T[I][J]

$d_L(voiture, moteur)$?

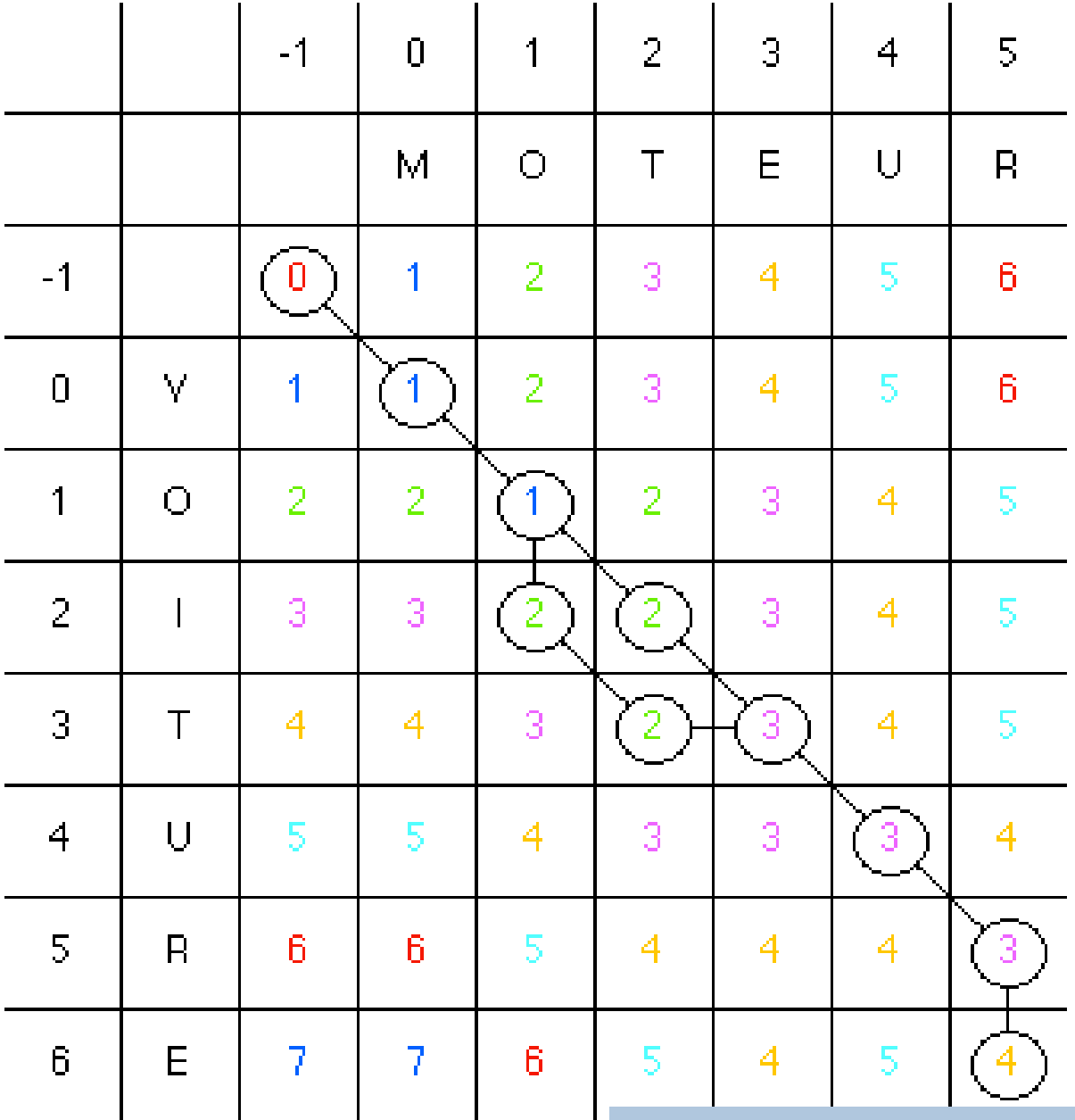
$i \leftarrow I$, $j \leftarrow J$, $a \leftarrow \{\}$

while $(i > 0)$ || $(j > 0)$ **do**
  if T[i][j-1]+del(y[j]) == T[i][j] **then**
    a $\leftarrow$ a $\cup$ del(y[j])
    j $\leftarrow$ j-1
  else if T[i-1][j]+ins(x[i]) == T[i][j] **then**
    a $\leftarrow$ a $\cup$ ins(x[i])
    i $\leftarrow$ i-1
  else
    a $\leftarrow$ a $\cup$ sub(x[i],y[j])
    i $\leftarrow$ i-1 , j $\leftarrow$ j-1

- L'ordre des tests influe sur le meilleur alignement retourné lorsqu'il y en a plusieurs
- On peut éviter cet algorithme en mémorisant dans chaque cellule de la table l'opération (2 bits sont nécessaires).

rali

$d_L(voiture, moteur)$?



		-1	0	1	2	3	4	5
			M	O	T	E	U	R
-1		0	1	2	3	4	5	6
0	V	1	1	2	3	4	5	6
1	O	2	2	1	2	3	4	5
2	I	3	3	2	2	3	4	5
3	T	4	4	3	2	3	4	5
4	U	5	5	4	3	3	3	4
5	R	6	6	5	4	4	4	3
6	E	7	7	6	5	4	5	4

$d_L(voiture, moteur)$?

- Les deux alignements de la figure précédente (chacun de coût minimal 4) :

V	O	I	T	U	R	E
M	O	T	E	U	R	
S	=	S	S	=	=	I

V	O	I	T		U	R	E
M	O		T	E	U	R	
S	=	D	=	I	=	=	I

  - Le second alignement correspond à la table de l'acétate 22
  - On peut représenter l'ensemble des alignements par un automate

Raffinements

[Ukkonen, 1985]

- Il existe un algo qui étant donné x_1^m, et $y = y_1^n$ et t un seuil, teste si $D(x, y) \leq t$ en un temps $O(t.\min(m, n))$ et en mémoire : $O(\min(t, n, m))$.
- en énumérant en ordre croissant les valeurs de t, on a un algo qui calcule la distance d'édition e en un temps $O(e.\min(m, n))$ et une mémoire $O(\min(e, m, n))$ (si l'alignement n'est pas requis)
- si les coûts sont unitaires, il existe des raffinements supplémentaires

rali

Propriétés

$$||x| - |y|| \leq d(x, y) \leq max(|x|, |y|)$$

- au pire on substitue les caractères de la plus courte chaine puis on ajoute ceux manquants de la plus longue (insertions) $\rightarrow$ inégalités de droite
- si les chaînes ne sont pas égales, elles diffèrent au moins de $||x| - |y||$, d'où l'inégalité de gauche
- **note :** distance nulle *ssi* les deux chaines sont identiques

rali

Language d'édition

On appelle language d'édition d'une chaîne x par rapport à un language L le language défini par l'ensemble des mots de L qui sont à une distance d'édition minimale de x :

$$d(L, x) = \min_{y \in L} d(x, y)$$

Ce language formel est utilisé par exemple en correction orthographique

Damerau-Levenshtein

// init au bornes

for i $\leftarrow 0$ à I **do**

  T[i][0] $\leftarrow i$

for j $\leftarrow 1$ à J **do**

  T[0][j] $\leftarrow j$

// boucle principale

for i $\leftarrow 1$ à I **do**

  for j $\leftarrow 1$ à J **do**

$$\text{T[i][j]} = \min \begin{cases} T[i-1,j-1] & + & sub(x[i],y[j]) \\ T[i-1,j] & + & ins(x[i]) \\ T[i,j-1] & + & del(y[j]) \end{cases}$$

if i ≥ 2 **and** j ≥ 2 **and** x[i] == y[j-1] **and** x[i-1] == y[j] **then**

$$\text{T[i][j]} = \min \begin{cases} \text{T[i][j]} \\ \text{T[i-2][j-2]} + cost\_transposition \end{cases}$$

//Le résultat est dans T[I][J]

Distance de Needleman-Wunsch

- plus de 6000 citations à ***[Needleman and Wunsch, 1970]*** !
- matrice de similarité sim de taille $\Sigma \times \Sigma$
- recherche les alignements de score maximal de manière analogue à la distance d'édition, en donnant une pénalité de trou identique d à chaque insertion ou suppression.

- la récurrence :

$$\begin{aligned} F_{ij} &= max \begin{cases} F_{i-1,j-1} + sim(x_i, y_j) \\ F_{i,j-1} + d \\ F_{i-1,j} + d \end{cases} \\ F_{0j} &= F_{i0}, \qquad \forall i, j \end{aligned}$$

Distance de Needleman-Wunsch

- avec $d = -5$ et sim :

	A	G	C	T
A	10	-1	-3	-4
G	-1	7	-5	-3
C	-3	-5	9	0
T	-4	-3	0	8

- l'alignement suivant de `AGACTAGTTAC` avec `CGAGACGT` a un coût de 1 :

A	G	A	C	T	A	G	T	T	A	C
C	G	A	–	–	–	G	A	C	G	T
-3	7	10	-5	-5	-5	7	-4	0	-1	0

- Est-ce un alignement optimal (cad de similarité maximale) ?

Plus longue sous-chaine commune à X et Y

- Soit X = $x_1 \dots x_N$ et $Y = y_1 \dots y_M$
  - $Z = z_1 \dots z_K$ est une sous-chaine de X, notée $Z = SC(X)$, *ssi* il existe $\rho : [1, K] \rightarrow [1, N]$ tel que :
    $Z = x_{\rho(1)} \dots x_{\rho(K)}$ avec $\rho(i) > \rho(j), \forall i > j$
  - Z est une sous-chaine commune à X et Y, notée $Z = SCC(X, Y)$ *ssi* Z=SC(Y) et Z=SC(X)

- **Exemple :** $X = ACGATCCACGT, Y = AGCTACGT$
  - AAA, CT,$ACGT$ sont des sous-chaines de X,
  - AA, $AGACT$ sont des sous-chaines communes à X et Y

Idée : X et Y sont d'autant plus proches qu'elle possèdent une sous-chaine commune qui est longue.

Plus longue sous-chaine commune à X et Y

- Soit $X = x_1 \dots x_N$ $Y = y_1 \dots y_M$ et $Z = z_1 \dots z_K$
- **si** Z est une plus longue sous-chaine de X et Y,
  ce que l'on note $Z = PLSCC(X, Y)$
- **alors**
  - **si** $x_N = y_M$ **alors** $z_K = x_N$ et $Z_1^{K-1} = PLSCC(X_1^{N-1}, Y_1^{M-1})$
  - **sinon**
    - **si** $z_K \neq x_N$ **alors** $Z_1^K = PLSCC(X_1^{N-1}, Y)$
    - **si** $z_K \neq y_M$ **alors** $Z_1^K = PLSCC(X, Y_1^{M-1})$

Récurrence

- $T[i,j]$ la longueur d'une PLSCC à $X[1...i]$ et $Y[1...j]$:

$$T[i,j] = \begin{cases} 0 & \text{si} \quad i*j=0 \\ T[i-1,j-1]+1 & \text{si} \quad i,j>0 \text{ et } x_i = y_j \\ max \begin{cases} T[i-1,j] \\ T[i,j-1] \end{cases} & \text{si} \quad i,j>0 \text{ et } x_i \neq y_j \end{cases}$$

- **Exemple :**

$$\overbrace{\overbrace{\overbrace{1+\max(1+\overbrace{1+\underbrace{PLSCC(\epsilon,\epsilon)}_{0}}^{PLSCC(A,A)=1}}^{PLSCC(AA,AA)=2}, \max(\overbrace{1+\underbrace{PLSCC(A,\epsilon)}_{0}}^{PLSCC(AA,A)=1}}^{PLSCC(AAB,A)=1}, \underbrace{PLSCC(AAB,\epsilon)}_{0}))}^{PLSCC(AAB,AA)=2}}^{PLSCC(AABB,AAB)=3}$$

- la version récursive amène à revisiter des sous-problèmes
  ex : $PLSCC(aaab, aaac) \Rightarrow$ 2 calculs de $PLSC(aaa, aaaa)$

Plus longue sous-chaine commune à X et Y

Require: T une table $N \times M$ dont un élément est T[i,j] = <score,back pointeur>

Ensure: T[N,M] contient la longueur d'une PLSSC de X et Y

for $i : 0 \rightarrow N$ **do** T[i,0] = $< 0, \downarrow >$

for $j : 1 \rightarrow M$ **do** T[0,j] = $< 0, \leftarrow >$

for $i : 1 \rightarrow N$ **do**

  for $j : 1 \rightarrow M$ **do**

    if $x_i = y_j$ **then**

      T[i,j] = $< T[i-1, j-1].score + 1, \swarrow >$

    else

      T[i,j] = $< T[i-1, j].score, \downarrow >$

      if T[i,j-1].score $>$ T[i,j].score **then**

        T[i,j] = $< T[i, j-1].score, \leftarrow >$

Plus longue sous-chaine commune à X et Y

T	0	↓	1	↓	2	↓	3	↓	4	↓	5	↓	6	↓	7	↓	8	↙
G	0	↓	1	↓	2	↓	3	↓	4	↓	5	↓	6	↓	7	↙	7	←
C	0	↓	1	↓	2	↙	3	↓	4	↙	5	↙	6	↓	6	←	6	←
A	0	↓	1	↙	2	↓	3	↙	4	↓	5	↓	6	↙	6	←	6	←
G	0	↓	1	↓	2	↓	3	↓	4	↓	5	↓	5	←	6	↙	6	←
C	0	↓	1	↓	2	↙	3	↓	4	↙	5	↙	5	←	5	←	5	←
A	0	↓	1	↙	2	↓	3	↙	4	↓	4	←	5	↙	5	←	5	←
C	0	↓	1	↓	2	↙	3	↓	4	↙	4	↙	4	←	4	←	4	←
G	0	↓	1	↓	2	↓	3	↓	3	←	3	←	3	←	4	↙	4	←
G	0	↓	1	↓	2	↓	3	↓	3	←	3	←	3	←	4	↙	4	←
A	0	↓	1	↙	2	↓	3	↙	3	←	3	←	3	↙	3	←	3	←
G	0	↓	1	↓	2	↓	2	←	2	←	2	←	2	←	3	↙	3	←
C	0	↓	1	↓	2	↙	2	←	2	↙	2	↙	2	←	2	←	2	←
A	0	↓	1	↙	1	←	1	↙	1	←	1	←	1	↙	1	←	1	←
X	0	↓	0	←	0	←	0	←	0	←	0	←	0	←	0	←	0	←
	Y		A		C		A		C		C		A		G		T	

- Y est une PLSCC(X,Y)

L'algorithme du Soundex

- Algorithme simple capable d'associer à chaque mot (suite de caractères) un *"code phonétique"*
- Il existe des soundex pour de nombreuses langues

- **Idée :** les mots proches phonétiquement ont le même code soundex

  soundex(computer) = soundex(camputter) = C513

- brevet déposé par Robert, R. Russel en 1918
- nombreuses variantes (ex : metaphone, NYYSIIS)

rali

Soundex Anglais

1. mettre le mot en majuscule, éliminer les ponctuations
2. garder la première lettre du mot
3. supprimer les occurences de : A E I O U H W Y
4. faire les changements suivants :
   - B F P V $\rightarrow$ 1 — bilabiales
   - C G J K Q S X Z $\rightarrow$ 2 — labiodentales
   - D T $\rightarrow$ 3 — dentales
   - L $\rightarrow$ 4 — alvéolaires
   - M N $\rightarrow$ 5 — vélaires
   - R $\rightarrow$ 6 — laryngales
5. si deux lettres adjacentes dans la chaîne de départ ont le même code, ne garder que la première de ces lettres (élimination des doublons)
6. retourner les 4 premiers caractères (compléter par des zéros le cas échéant)

Soundex Français

- Règles de conversion pour le français :

B P	$\rightarrow$	1	L	$\rightarrow$	4	G J	$\rightarrow$	7
C K Q	$\rightarrow$	2	M N	$\rightarrow$	5	X Z S	$\rightarrow$	8
D T	$\rightarrow$	3	R	$\rightarrow$	6	F V	$\rightarrow$	9

- On peut calculer le soundex du dictionnaire et encoder cela dans une table de hachage multiple de manière à retrouver en temps constant l'ensemble des mots proches d'un mot inconnu.

Plan

rali

Tableaux de suffixes (*Suffix arrays*)

Accès rapide à n'importe quelle séquence de mots

- **Idée :** indexer **toutes** les séquences possibles rencontrées dans un texte.
- **But :** pouvoir calculer des statistiques sur ces séquences pour découvrir par exemple des séquences pertinentes (au sens de la statistique utilisée).
- **Note :** Les transparents qui suivent reprennent les algorithmes décrits dans [Russell, 1998].

S = Some$_0$ of$_1$ the$_2$ words$_3$ of$_4$ the$_5$ sentence$_6$ are$_7$ the$_8$ same$_9$

- S contient $n = 10$ mots et 7 types.
- Soit un ordre sur les symboles ; par exemple l'ordre lexicographique :

  Some $<$ *are* $<$ *of* $<$ *same* $<$ *sentence* $<$ *the* $<$ *words*.
- Chaque position dans S débute un suffixe de S. Un suffix array (SFX) contient tous les suffixes triés :

0	7	4	1	9	6	8	5	2	3

- $SFX[i]$ indique le suffixe $S[SFX[i] \ldots n]$. Par exemple $S[6]$ vaut 8 qui est la position de départ du suffixe the same

Plus longs prefixes communs LPC

	0	1	2	3	4	5	6	7	8	9
S	Some	of	the	words	of	the	sentence	are	the	same

SFX	0	7	4	1	9	6	8	5	2	3

- $LPC[i]$ indique la taille du préfixe commun des séquences désignées par $SFX[i]$ et $SFX[i+1]$.

	0	1	2	3	4	5	6	7	8
LPC	0	0	2	0	0	0	1	1	0

- ex : $LPC[2]$ indique que les séquences of the words of the sequence are the same et of the sentence are the same partagent un préfixe de deux mots (of the)
- le calcul de LPC requiert un passage sur SFX et un accès au texte initial.

Trouver les occurrences d'une séquence

- **Input :** une occurrence d'une séquence key a été trouvée en position *found* dans *SFX*.
- **Output :** nombre d'occurrences de cette séquence

$left \leftarrow right \leftarrow found$
while $left > 0 \ \wedge \ lcps[left-1] \geq |key|$ **do**
  $left \leftarrow left - 1$
while $right < |lcps| \ \wedge lcps[right] \geq |key|$ **do**
  $right \leftarrow right + 1$
return $right - left + 1$

- Pas besoin de la table SFX.
- D'où vient $found$?

Extraire les séquences depuis la table LPC

But : extraire toutes les séquences d'au moins min_length mots et de fréquence minimale min_freq.

Structure : une séquence s est caractérisée par un triplet $\langle l, f, p \rangle$, où l est la longueur d'une séquence, f sa fréquence, et p la première position dans *SFX* (qui pointe sur une séquence dans le texte dont le préfixe est s).

- *of the* est caractérisée par le triplet $\langle 2, 2, 2 \rangle$.
- *the same* est caractérisée par le triplet $\langle 2, 1, 6 \rangle$

Idée : on maintient deux ensembles de triplets $active$ et $result$. Le premier est un ensemble temporaire, qui contient les séquences potentiellement intéressantes en cours d'analyse ; le second contient la réponse.

Extraction des séquences

$results \leftarrow active \leftarrow \emptyset$
$new\_length \leftarrow prev\_length \leftarrow min\_length$
$this\_pos \leftarrow 0$
while $this\_pos < |lcps|$ **do**
  $this\_length \leftarrow lcps[this\_pos]$
  if $this\_length < min\_length$ **then**
    for all $t = \langle len, freq, pos \rangle$ in $active$ **do**
      flush(t)
    $new\_length \leftarrow min\_length$
  else if $this\_length \geq prev\_length$ **then**
    for all $\langle len, freq, pos \rangle$ in $active$ **do**
      $freq \leftarrow freq + 1$
    while $new\_length \leq this\_length$ **do**
      $active \leftarrow active \cup \{\langle new\_length, 2, this\_pos \rangle\}$
      $new\_length \leftarrow new\_length + 1$
    $new\_length \leftarrow this\_length + 1$
  else
    for all $t = \langle len, freq, pos \rangle$ in $active$ **do**
      if $len \leq this\_length$ **then**
        $freq \leftarrow freq + 1$
      else
        flush(t)
    $new\_length \leftarrow this\_length + 1$
  $prev\_length \leftarrow this\_length$
  $this\_pos \leftarrow this\_pos + 1$
for all $t = \langle len, freq, pos \rangle$ in $active$ **do**
  flush(t)
return results

flush(t)
$active \leftarrow active - \{t\}$
if $freq \geq min\_freq$ **then**
  $results \leftarrow results \cup \{t\}$

Extraction des séquences

Trois cas de figure (exclusifs) dans cet algorithme :

$this\_length < min\_length$

On vide *active* en gardant (dans *result*) les séquences qui vérifient le critère de fréquence.

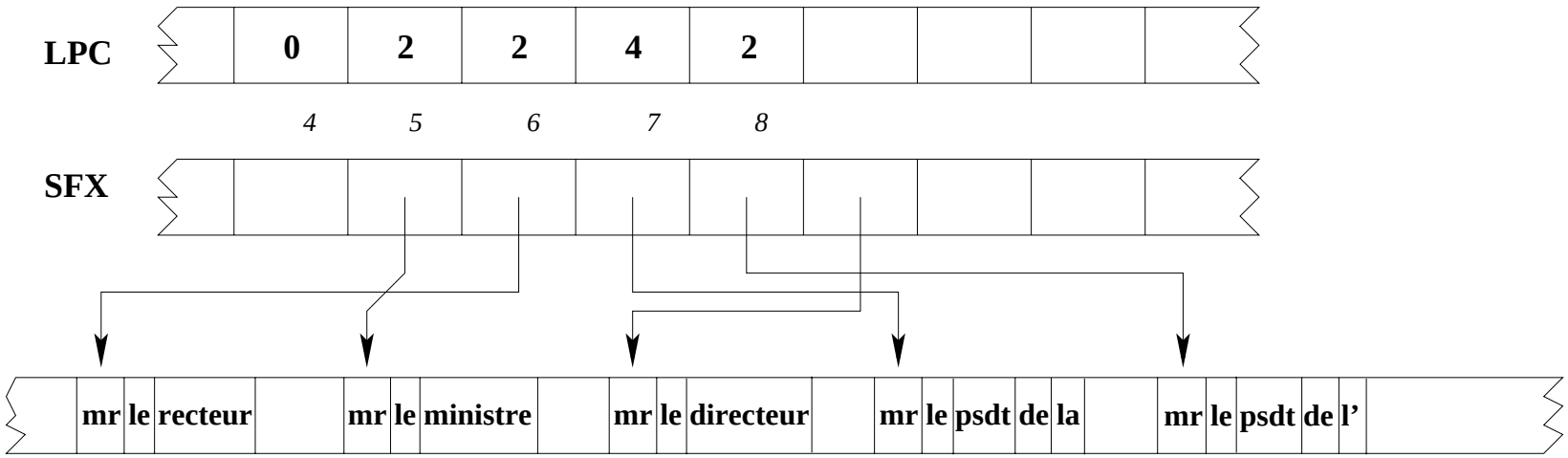
$this\_length \geq prev\_length$

- On augmente de 1 toutes les séquences déjà dans *active*.
- idée de la boucle sur new_length : “lorsqu’on voit une séquence de taille n, on voit également une séquence de taille $n - 1$”

$this\_length < prev\_length$ et $this\_length \geq min\_length$

- On augmente de 1 toutes les séquences d’au plus $this\_length$ mots dans *active*.
- On retire de *active* les autres séquences en gardant dans *result* celles qui vérifient le critère de fréquence

Évolution de $active$ ($min\_length = 2$)



$this\_pos$	$this\_length$	new_length	active	
4	0	2	$\{\}$	cas 1
5	2	3	$\{< 2, 2, 5 >\}$	cas 2
6	2	3	$\{< 2, 3, 5 >\}$	cas 2
7	4	5	$\{< 2, 4, 5 >, < 3, 2, 7 >, < 4, 2, 7 >\}$	cas 2
8	2	3	$\{< 2, 5, 5 >\}$	cas 3

Fréq. des séquences d'au moins deux mots

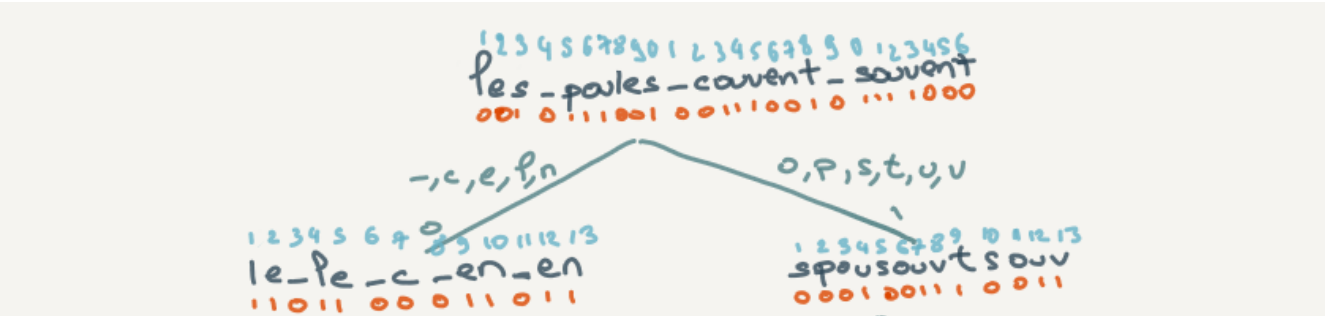
long.	fréq.	séquence	long.	fréq.	séquence
2	1760	de la	2	893	) :
2	1594	de l'	2	867	qu' il
2	1391	le président	2	778	, le
2	1083	monsieur le	2	724	: monsieur
3	1076	monsieur le président	3	724	: monsieur le
2	1069	le gouvernement	4	723	: monsieur le président
2	1040	à la	5	720	: monsieur le président ,
2	988	c' est	2	701	à l'
2	985	président ,	2	643	le député
3	983	le président ,	2	640	, je
4	971	monsieur le président ,	3	608	) : monsieur
2	913	que le	5	608	) : monsieur le président

rali

Wavelet Tree (binaire)

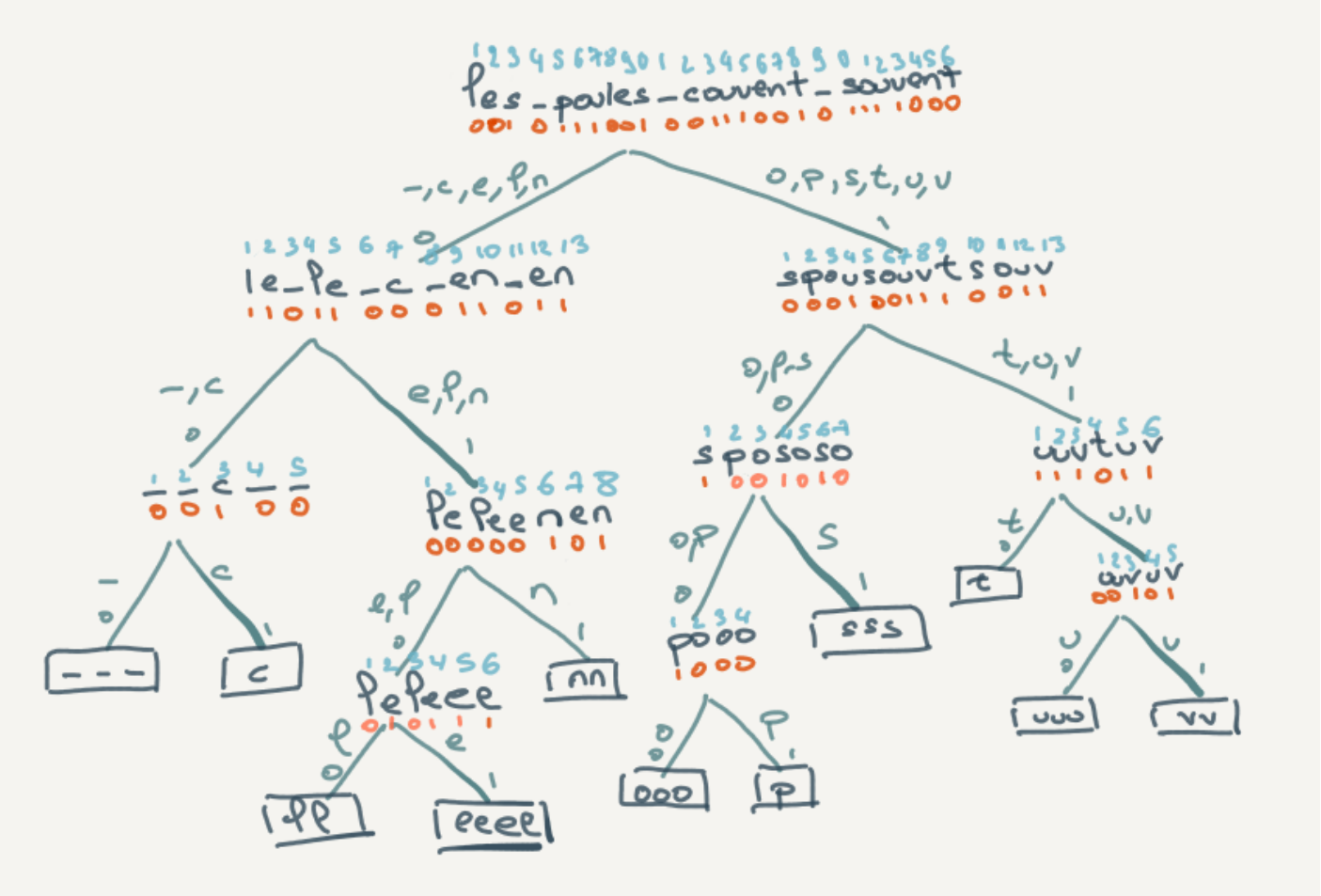
Soit Σ l'alphabet et σ sa taille et soit S la chaîne sur Σ^*. Le wavetree de S est défini récursivement par :

1. associer la première moitié des symboles de Σ à 0, l'autre à 1
2. grouper les symboles de S encodés par des 0 en un fils gauche
3. grouper les symboles de S encodés par des 1 en un fils droit
4. appliquer récursivement sur les fils jusqu'à épuisement des symboles



note : les chaînes ne sont pas stockées mais simplifient (je l'espère) la compréhension. Les indices en bleu clair seront utilisés plus tard et ne font pas non plus partie de l'arbre.

Wavelet Tree de les poules couvent souvent



- # feuilles : σ, # nœuds internes : $\sigma - 1$, profondeur : $\lfloor \log_2(\sigma) \rfloor$
- $\Sigma = \{\_, c, e, l, n, o, p, s, t, u, v\}$, $\sigma = 11$

rank et select sont dans un bateau

Soit s une chaîne de n symboles, on définit les opérations :

$rank_s(c, i)$ le nombre d'occurrences du symbole c dans s sur l'intervalle $[1 \dots i]$

$select_s(c, i)$ la position dans s de la ie occurrence de c

ex : soit s =acgtacggatga ($s[1]$ et $s[12]$ valent a)

- $rank_s(a, 8)$ vaut 2, $rank_s(g, 8)$ vaut 3
- $rank_s(a, 1)$ vaut 1, $select_s(g, 4)$ vaut 11

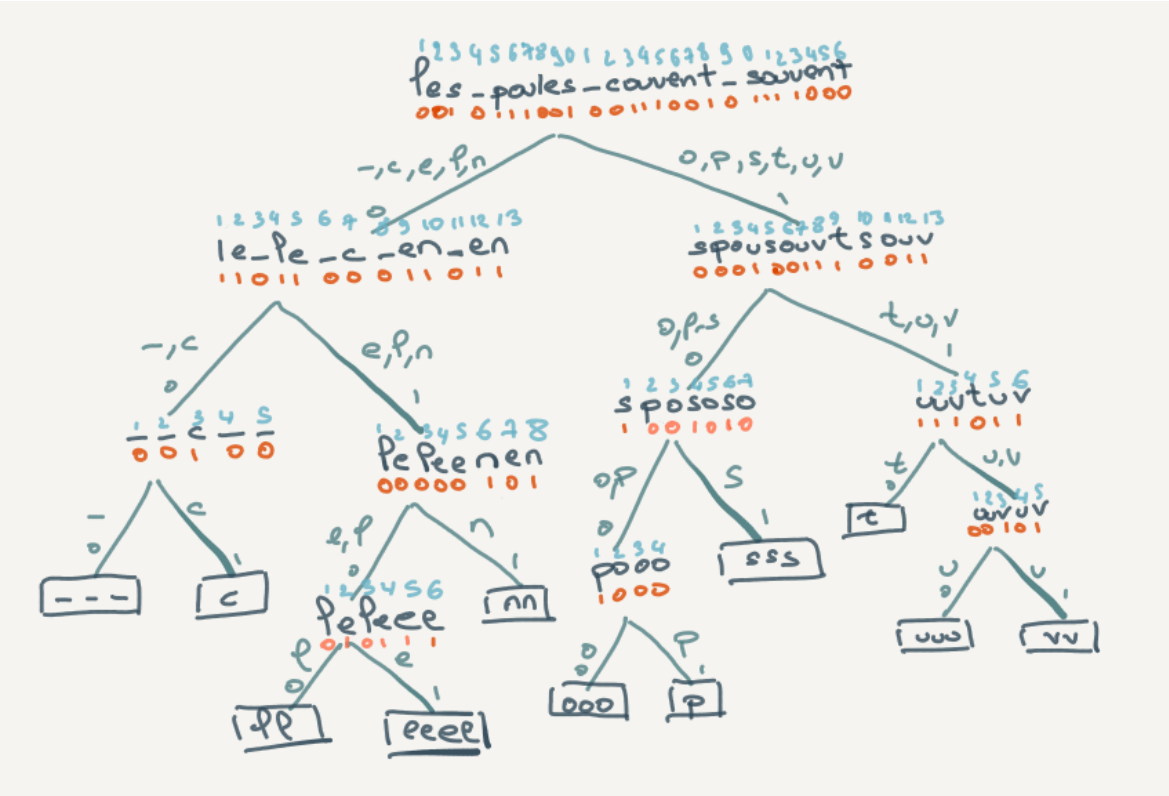
- $rank$ et $select$ à valeur dans $[0, n]$
- $rank_s(c, select_s(c, i)) = i$
- Un wavetree offre les opérations $rank$ et $select$ en $O(\log(\sigma))$ plutôt qu'en $o(\log n)$

Wavelet tree & $rank_s(c, i)$

$n \leftarrow root$
$c \leftarrow c_1, c_2, \ldots, c_\sigma$, l'encodage binaire de c induit par l'arbre
$k \leftarrow 0$
repeat
  $i \leftarrow rank_n(c_k, i)$
  $n \leftarrow (c_k == 1)?$ right_child(n) : left_child(n)
  $k \leftarrow k + 1$
until is_leaf(n)
return i

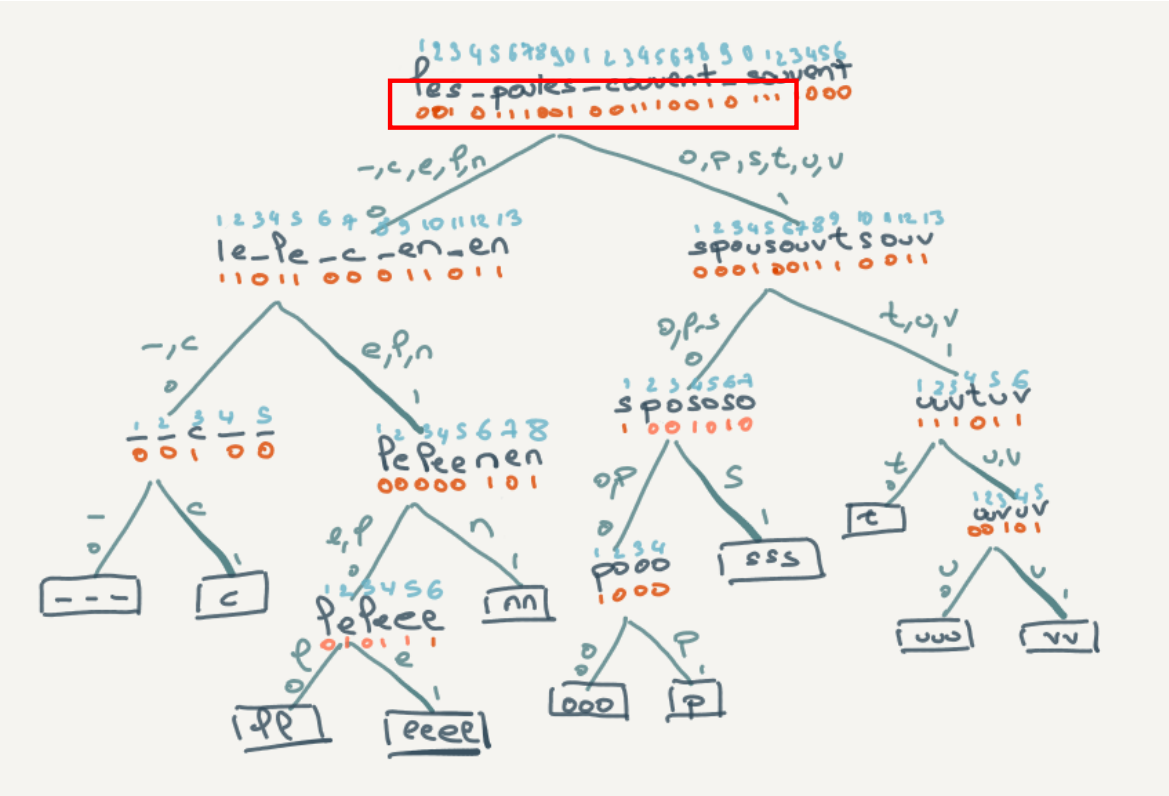
- on fait une descente récursive dans l'arbre $\Rightarrow log_2(\sigma)$ appels à $rank$ binaire (possible en $O(1)$)

Wavelet tree & $rank_s(o, 22)$



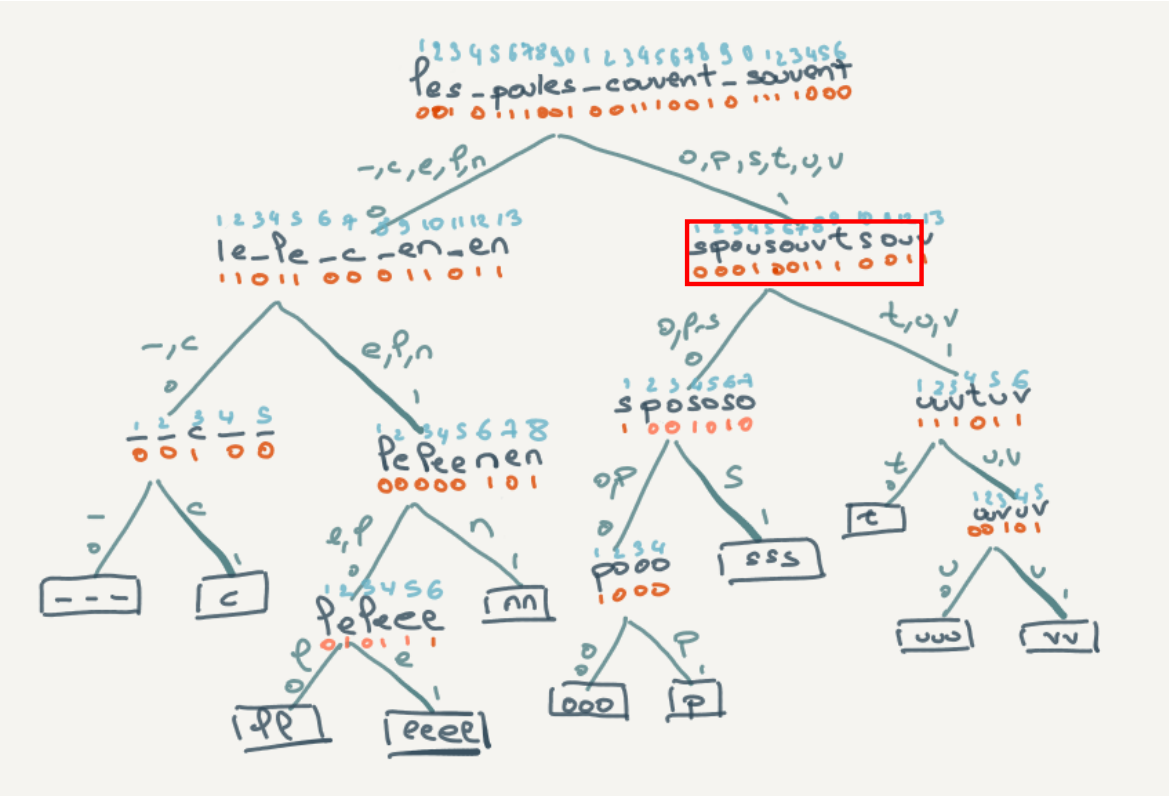
- o = 1, 0, 0, 0

Wavelet tree & $rank_s(o, 22)$



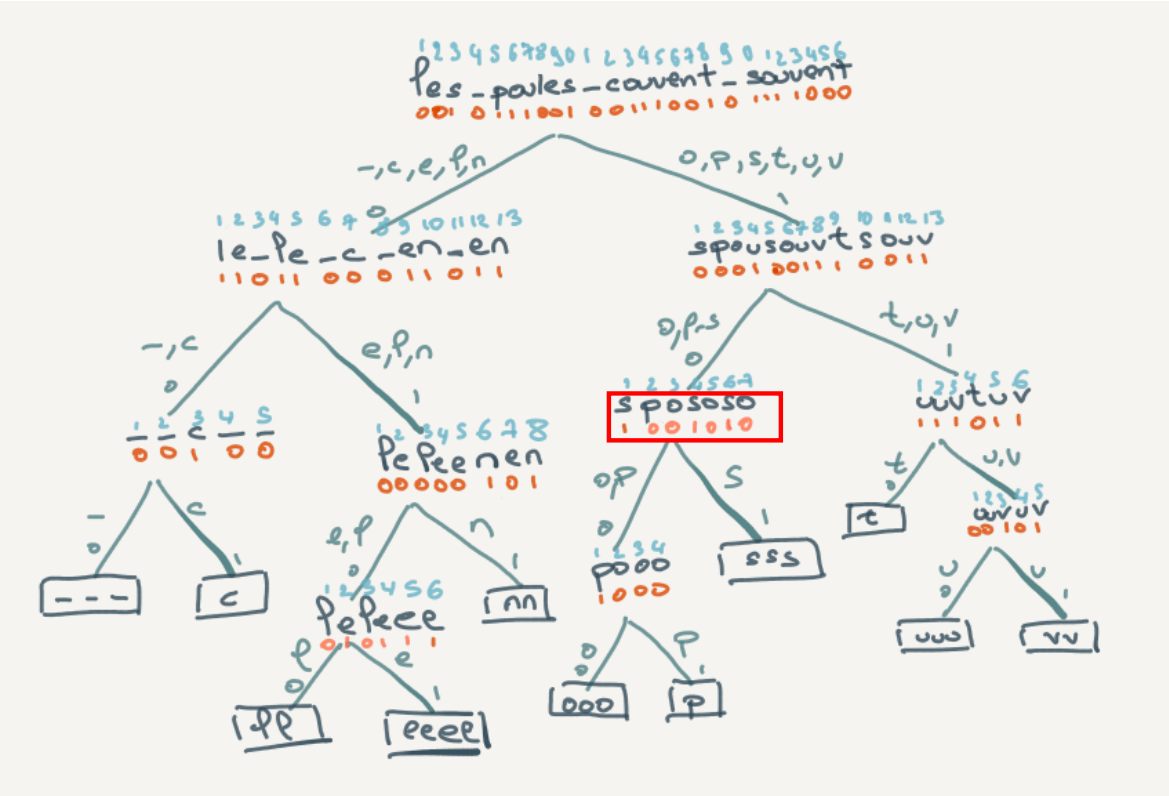
- o = 1, 0, 0, 0
- $rank(1, 22) = 12$

Wavelet tree & $rank_s(o, 22)$



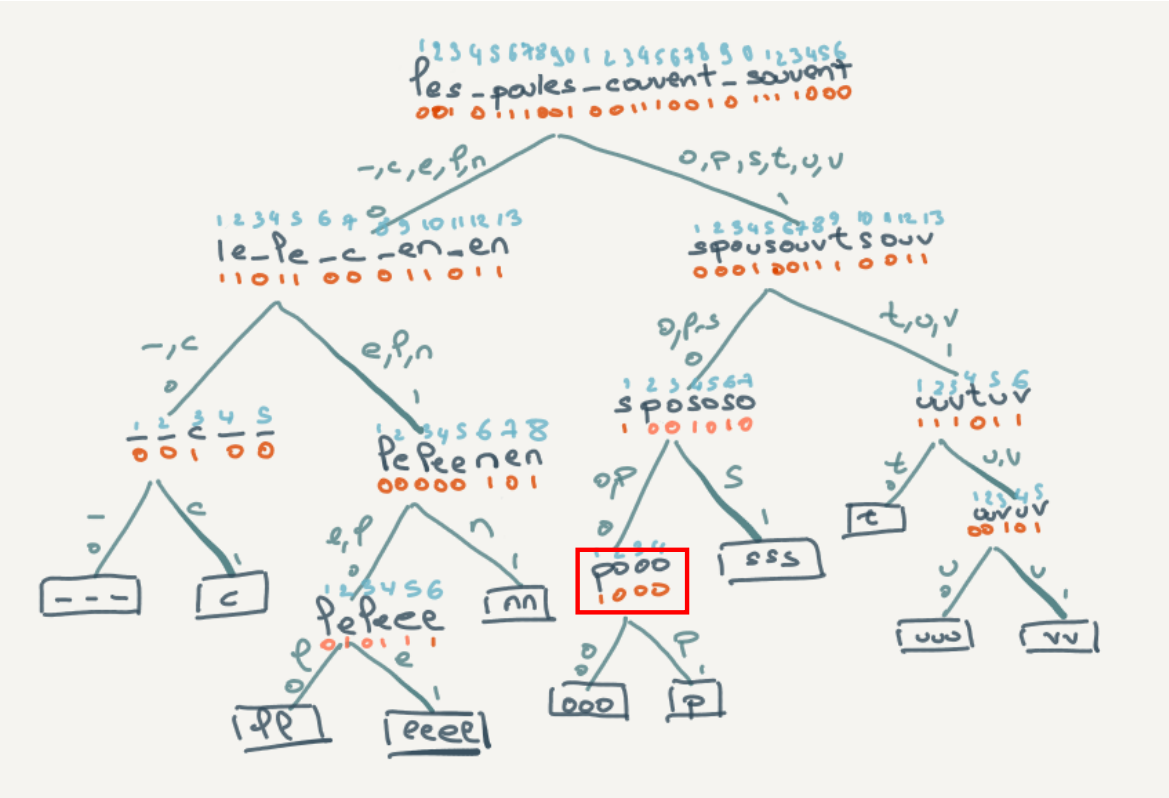
- o = 1, 0, 0, 0
- $rank(1, 22) = 12$
- $rank(0, 12) = 7$

Wavelet tree & $rank_s(o, 22)$



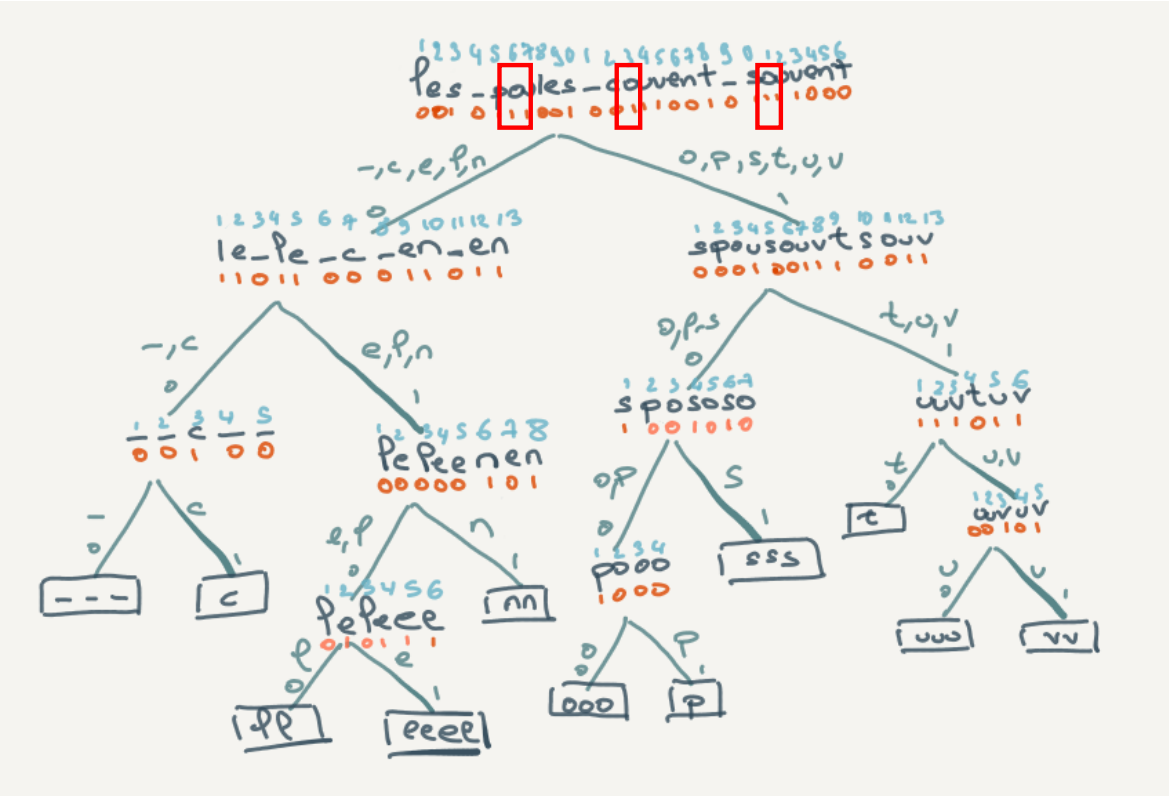
- o = 1, 0, 0, 0
- $rank(1, 22) = 12$
- $rank(0, 12) = 7$
- $rank(0, 7) = 4$

Wavelet tree & $rank_s(o, 22)$



- o = 1, 0, 0, 0
- $rank(1, 22) = 12$
- $rank(0, 12) = 7$
- $rank(0, 7) = 4$
- $rank(0, 4) = 3$

Wavelet tree & $rank_s(o, 22)$



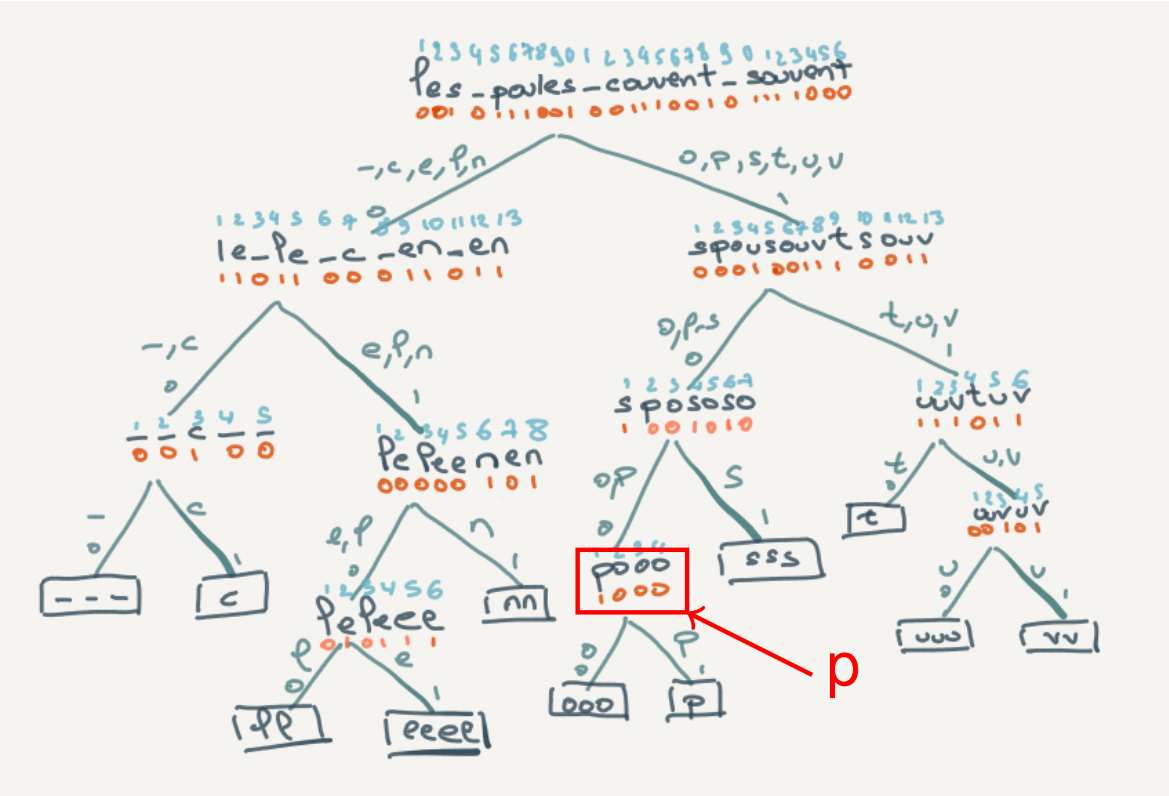
- o = 1, 0, 0, 0
- $rank(1, 22) = 12$
- $rank(0, 12) = 7$
- $rank(0, 7) = 4$
- $rank(0, 4) = 3$
- 3

Wavelet tree & $select_s(c, i)$

$n \leftarrow$ leaf which contains c
repeat
  $p \leftarrow$ parent(n)
  $c \leftarrow (p.fg == n)?\ 0 : 1$
  $i \leftarrow$ select$_p(c, i)$
  $n \leftarrow p$
until is_root(n)
return i

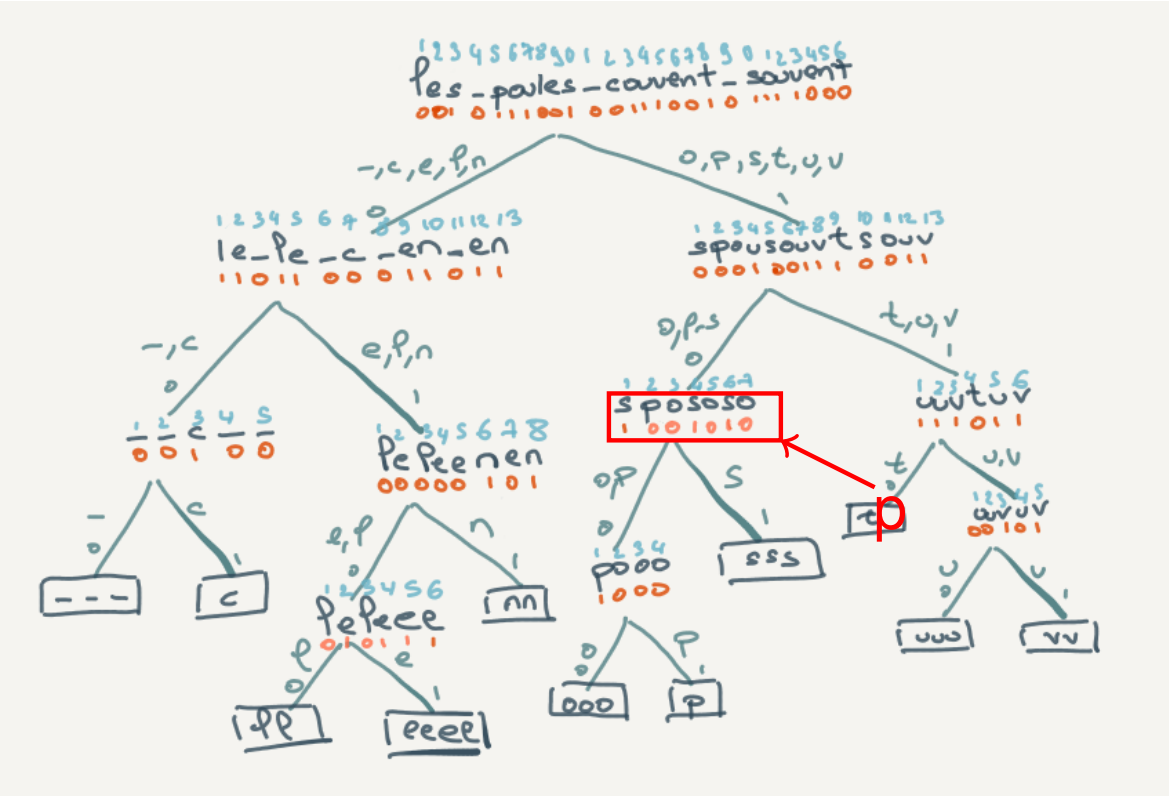
- on remonte depuis la feuille contenant $c \Rightarrow log_2(\sigma)$ appels à $select$ binaire (possible en $O(1)$)

Wavelet tree & $select_s(o, 3)$



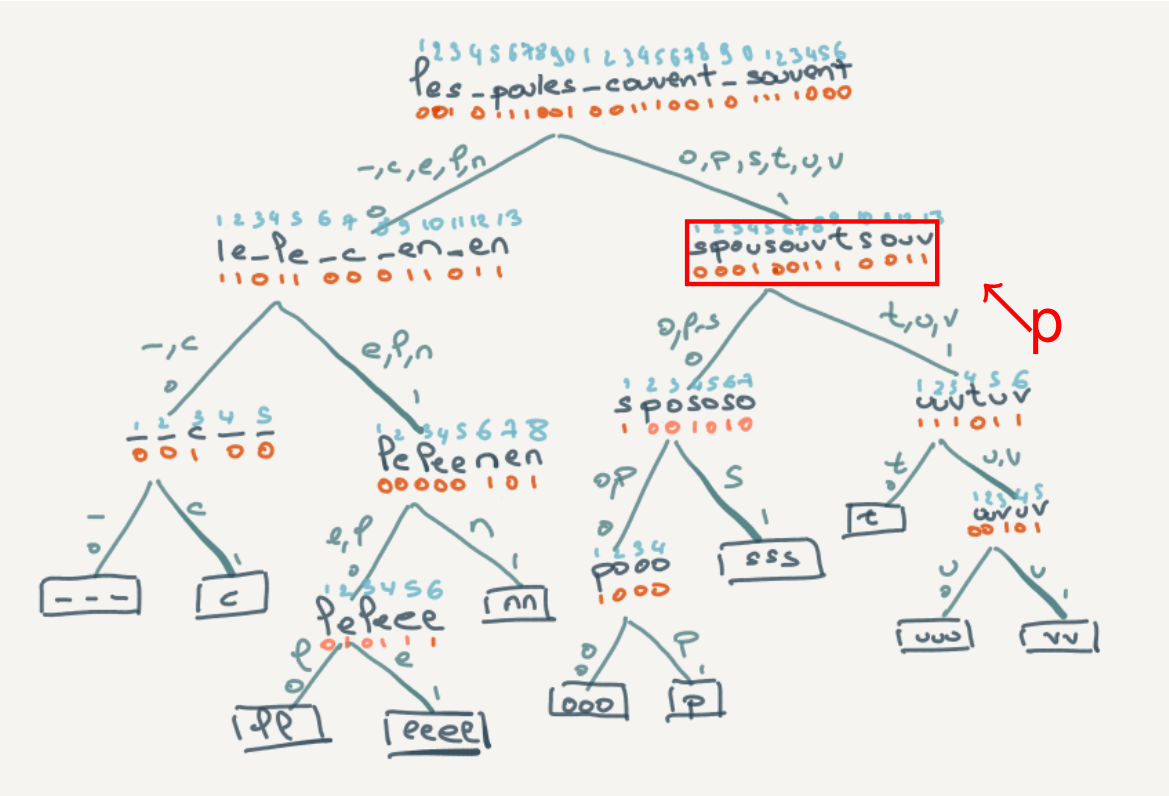
- $c = 0$,
  $i \leftarrow select_p(0, 3) = 4$

Wavelet tree & $select_s(o, 3)$



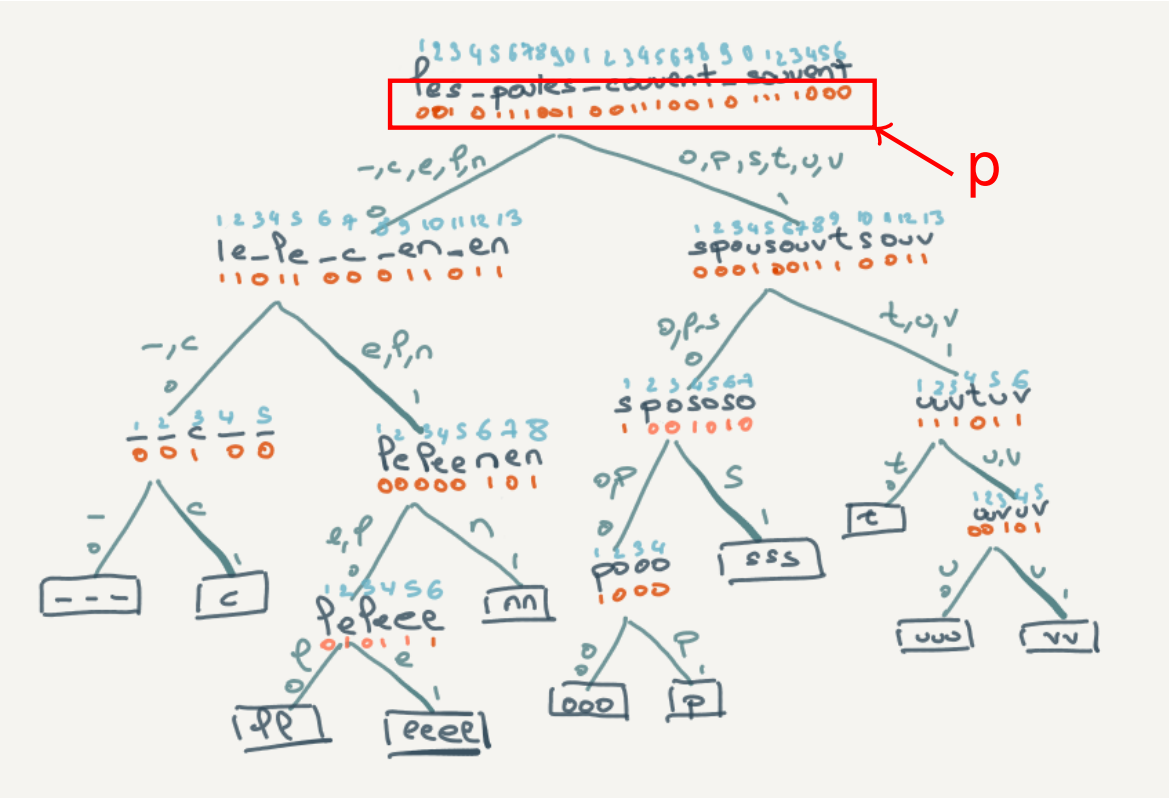
- $c = 0$,
  $i \leftarrow select_p(0, 3) = 4$
- $c = 0$,
  $i \leftarrow select_p(0, 4) = 7$

Wavelet tree & $select_s(o, 3)$



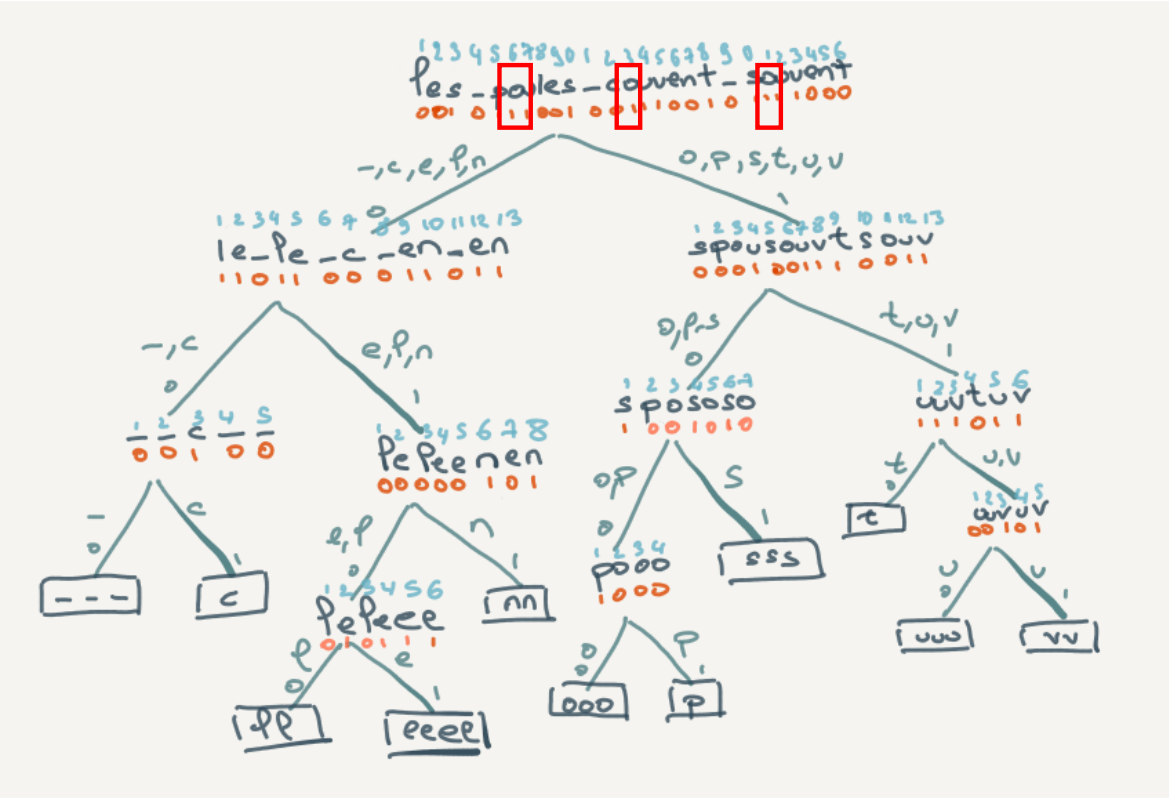
- $c = 0$,
  $i \leftarrow select_p(0, 3) = 4$
- $c = 0$,
  $i \leftarrow select_p(0, 4) = 7$
- $c = 0$,
  $i \leftarrow select_p(0, 7) = 11$

Wavelet tree & $select_s(o, 3)$



- $c = 0$, $i \leftarrow select_p(0, 3) = 4$
- $c = 0$, $i \leftarrow select_p(0, 4) = 7$
- $c = 0$, $i \leftarrow select_p(0, 7) = 11$
- $c = 1$, $i \leftarrow select_p(1, 11) = 21$

Wavelet tree & $select_s(o, 3)$



- $c = 0$,
  $i \leftarrow select_p(0, 3) = 4$
- $c = 0$,
  $i \leftarrow select_p(0, 4) = 7$
- $c = 0$,
  $i \leftarrow select_p(0, 7) = 11$
- $c = 1$,
  $i \leftarrow select_p(1, 11) = 21$
- 21

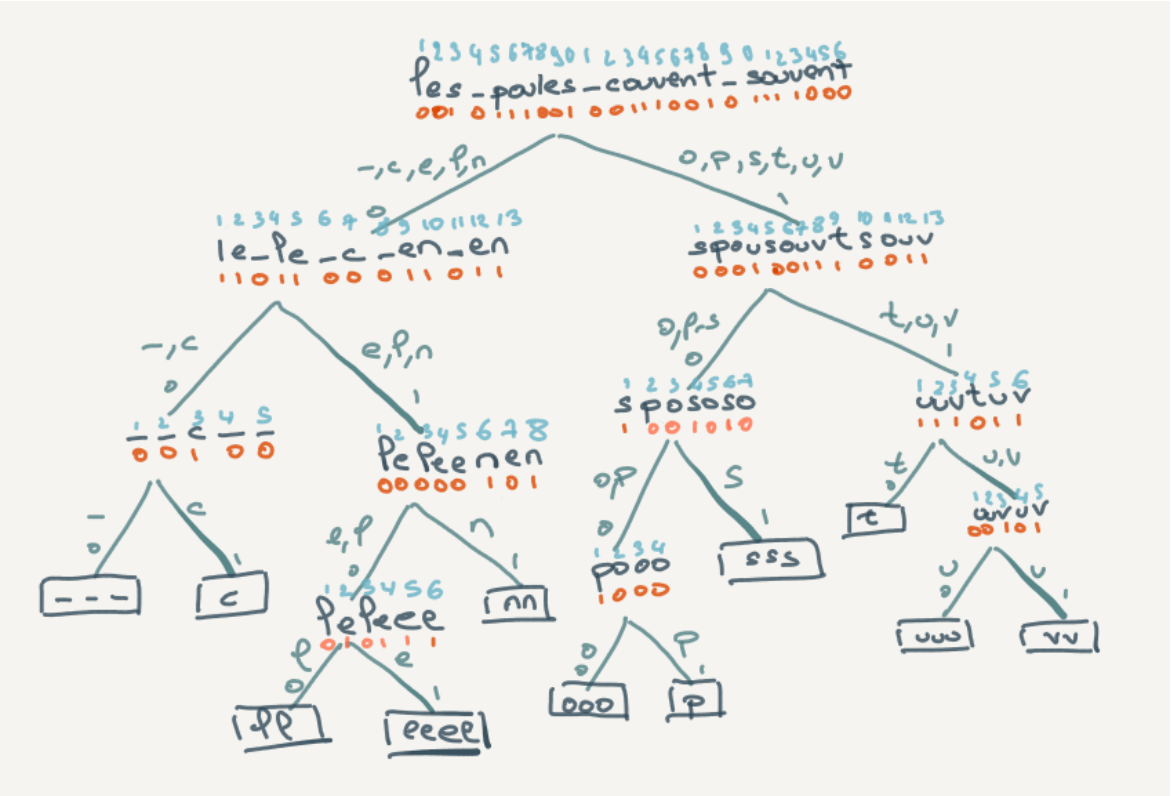
Un wavelet tree est une structure succinte

- Faible consommation de mémoire
  - chaque niveau = n bits, plus $o(\log_2(\sigma))$ nœuds à représenter
- On peut accéder à la chaîne de départ s sans la stocker.
  - accès à $s[i]$ (sans s) en $\log_2(\sigma)$ requêtes $rank$
  - même descente récursive que $rank_s(c, i)$, mais on se laisse guider par le bit à l'indice i

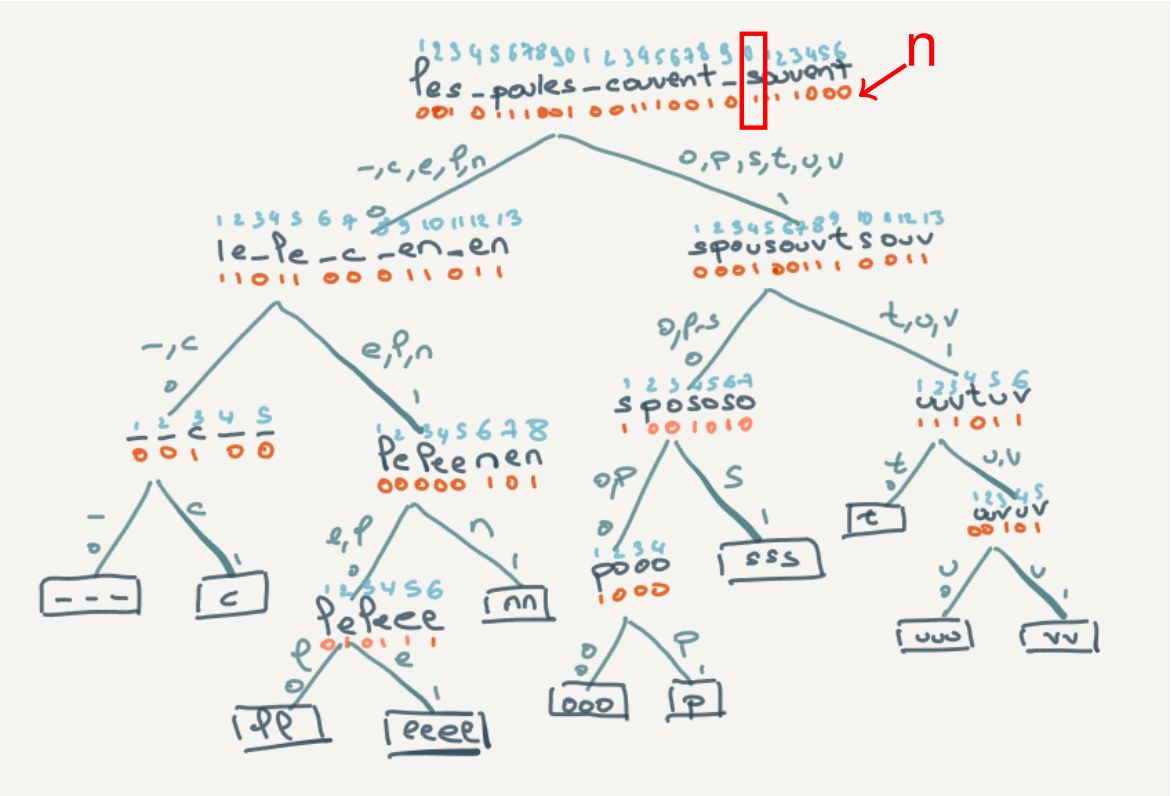
rali

Wavelet tree & $s[20]$

- $n \leftarrow root$

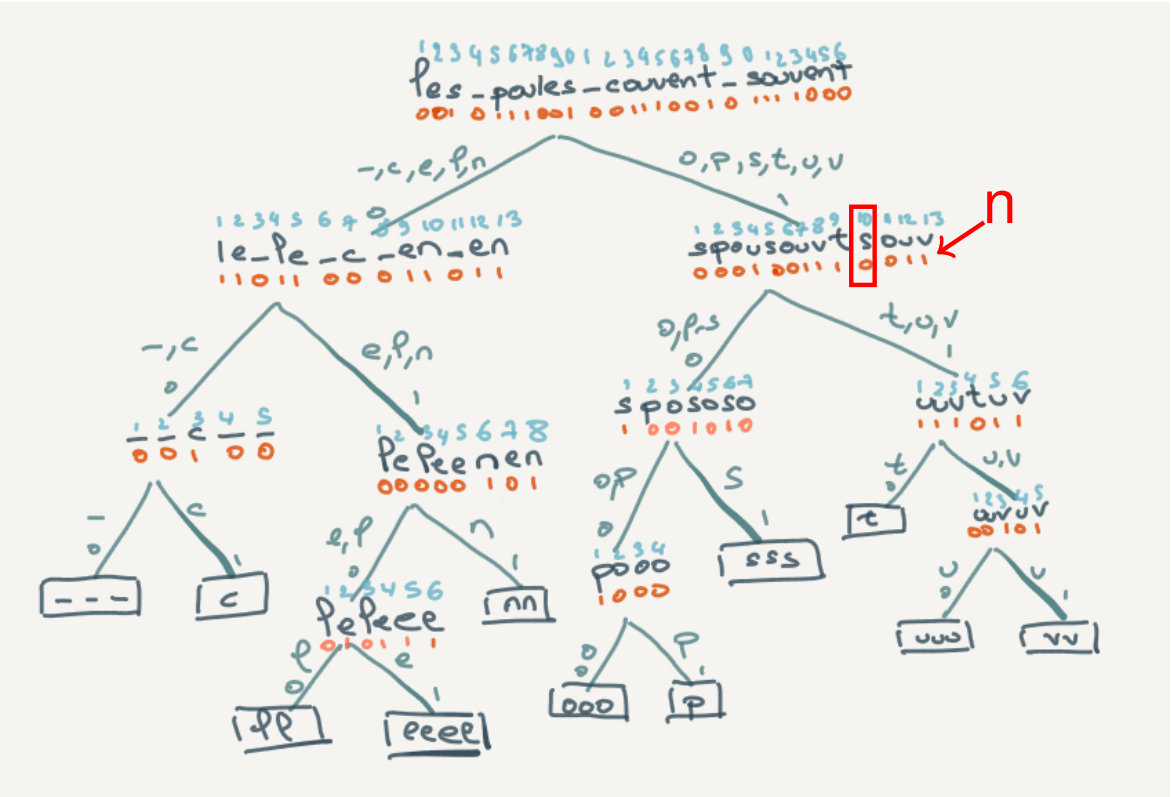


Wavelet tree & $s[20]$



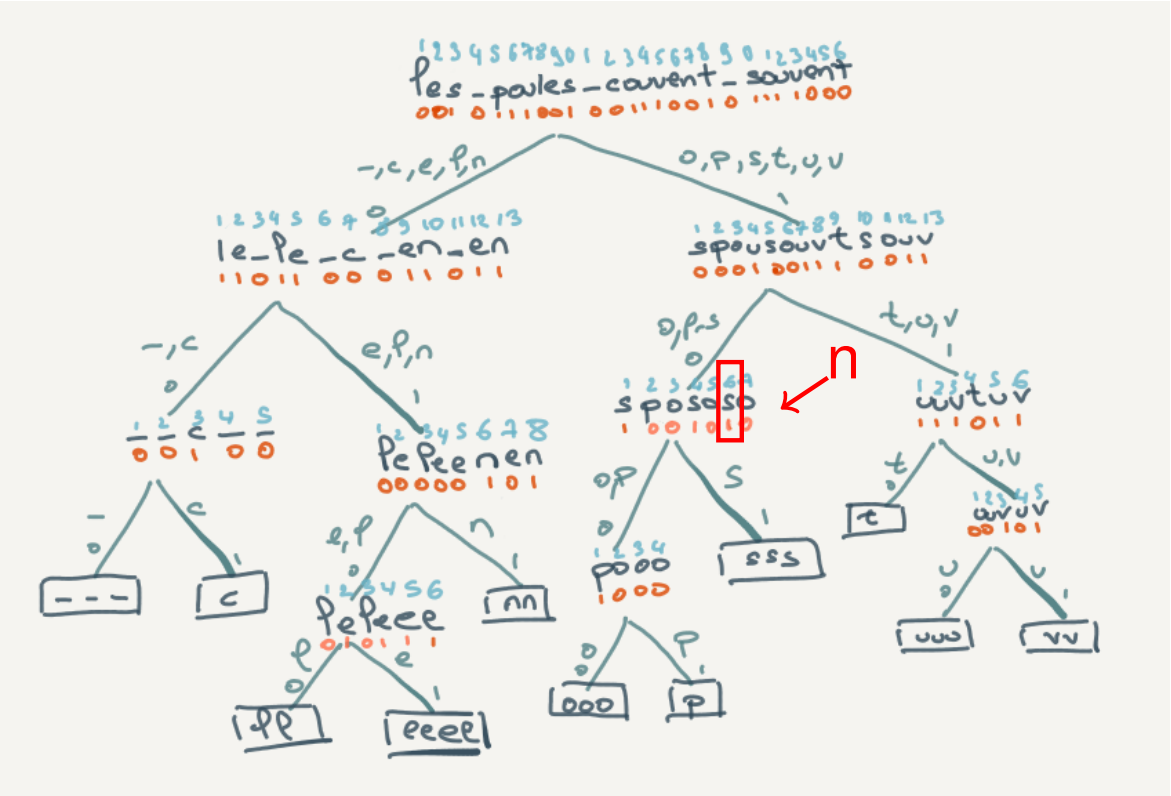
- $n \leftarrow root$
- n.get$(20) = 1$
  $rank(1, 20) = 10$
  $n \leftarrow n$.fils_droite

Wavelet tree & $s[20]$



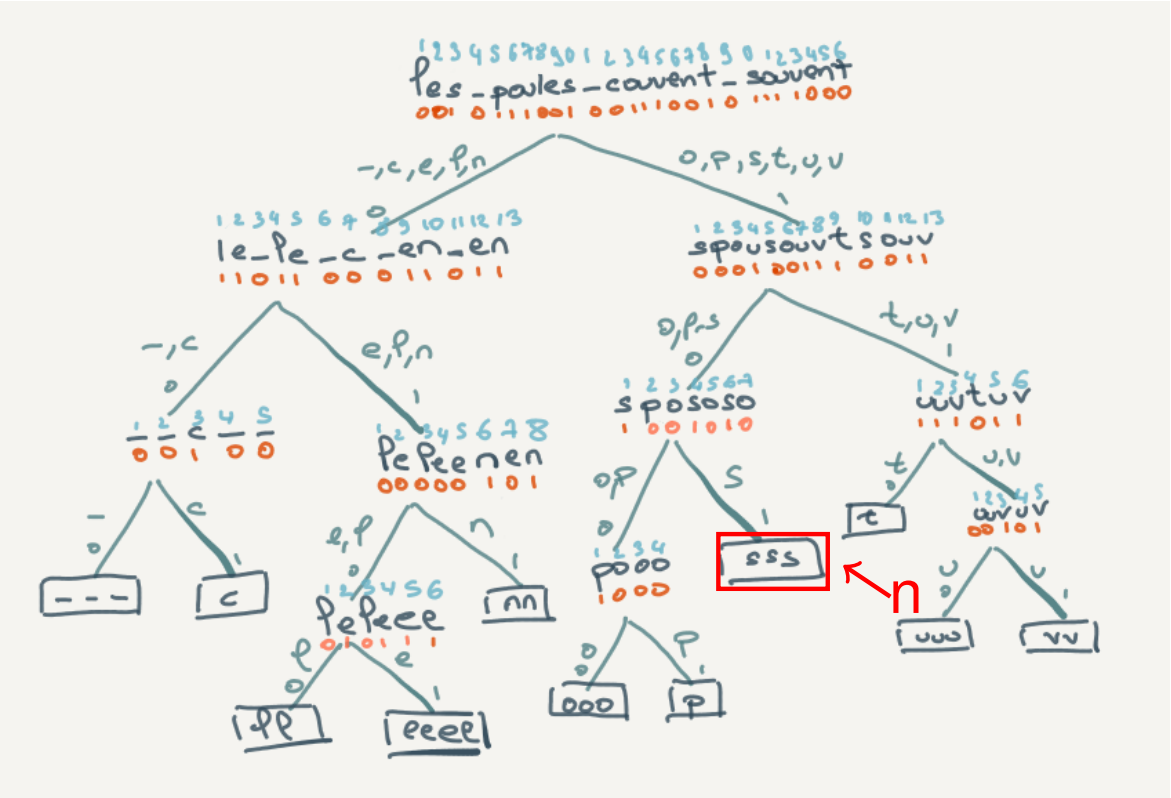
- $n \leftarrow root$
- $n.\text{get}(20) = 1$
  $rank(1, 20) = 10$
  $n \leftarrow n.\text{fils\_droite}$
- $n.\text{get}(10) = 0$
  $rank(0, 10) = 6$
  $n \leftarrow n.\text{fils\_gauche}$

Wavelet tree & $s[20]$



- $n \leftarrow root$
- $n.\text{get}(20) = 1$
  $rank(1, 20) = 10$
  $n \leftarrow n.\text{fils\_droite}$
- $n.\text{get}(10) = 0$
  $rank(0, 10) = 6$
  $n \leftarrow n.\text{fils\_gauche}$
- $n.\text{get}(6) = 1$
  $rank(0, 6) = 3$
  $n \leftarrow n.\text{fils\_droite}$

Wavelet tree & $s[20]$



- $n \leftarrow root$
- n.get$(20) = 1$
  $rank(1, 20) = 10$
  $n \leftarrow n$.fils_droite
- n.get$(10) = 0$
  $rank(0, 10) = 6$
  $n \leftarrow n$.fils_gauche
- n.get$(6) = 1$
  $rank(0, 6) = 3$
  $n \leftarrow n$.fils_droite
- n est la feuille associée à s
- s

Burrows-Wheeler Transform

- Une transformation réversible qui facilite la compression : Wikipedia.
- La chaîne BWT est liée à la chaîne originale par la table des suffixes :

$$BWT[i] = \begin{cases} S[SA[i]-1] & \text{si} \quad i \geq 1 \\ \$ & \text{si} \quad SA[i] == 1 \end{cases}$$

- $BWT[i]$ est donc le symbole qui précède le suffixe $SA[i]$ dans S

Burrows-Wheeler Transform et les poules

les_poules_couvent_souvent$	$les_poules_couvent_souvent
es_poules_couvent_souvent$l	_couvent_souvent$les_poules
s_poules_couvent_souvent$le	_poules_couvent_souvent$les
_poules_couvent_souvent$les	_souvent$les_poules_couvent
poules_couvent_souvent$les_	couvent_souvent$les_poules_
oules_couvent_souvent$les_p	ent$les_poules_couvent_souv
ules_couvent_souvent$les_po	ent_souvent$les_poules_couv
les_couvent_souvent$les_pou	es_couvent_souvent$les_poul
es_couvent_souvent$les_poul	es_poules_couvent_souvent$l
s_couvent_souvent$les_poule	les_couvent_souvent$les_pou
_couvent_souvent$les_poules	les_poules_couvent_souvent$
couvent_souvent$les_poules_	nt$les_poules_couvent_souve
ouvent_souvent$les_poules_c	nt_souvent$les_poules_couve
uvent_souvent$les_poules_co	oules_couvent_souvent$les_p
vent_souvent$les_poules_cou	ouvent$les_poules_couvent_s
ent_souvent$les_poules_couv	ouvent_souvent$les_poules_c
nt_souvent$les_poules_couve	poules_couvent_souvent$les_
t_souvent$les_poules_couven	s_couvent_souvent$les_poule
_souvent$les_poules_couvent	s_poules_couvent_souvent$le
souvent$les_poules_couvent_	souvent$les_poules_couvent_
ouvent$les_poules_couvent_s	t$les_poules_couvent_souven
uvent$les_poules_couvent_so	t_souvent$les_poules_couven
vent$les_poules_couvent_sou	ules_couvent_souvent$les_po
ent$les_poules_couvent_souv	uvent$les_poules_couvent_so
nt$les_poules_couvent_souve	uvent_souvent$les_poules_co
t$les_poules_couvent_souven	vent$les_poules_couvent_sou
$les_poules_couvent_souvent	vent_souvent$les_poules_cou
shift	+sort

Revenons aux poules . . .

```
les_poules_couvent_souvent$
es_poules_couvent_souvent$
s_poules_couvent_souvent$
_poules_couvent_souvent$
poules_couvent_souvent$
oules_couvent_souvent$
ules_couvent_souvent$
les_couvent_souvent$
es_couvent_souvent$
s_couvent_souvent$
_couvent_souvent$
couvent_souvent$
ouvent_souvent$
uvent_souvent$
vent_souvent$
ent_souvent$
nt_souvent$
t_souvent$
_souvent$
souvent$
ouvent$
uvent$
vent$
ent$
nt$
t$
```

pop

```
_couvent_souvent$
_poules_couvent_souvent$
_souvent$
couvent_souvent$
ent$
ent_souvent$
es_couvent_souvent$
es_poules_couvent_souvent$
les_couvent_souvent$
les_poules_couvent_souvent$
nt$
nt_souvent$
oules_couvent_souvent$
ouvent$
ouvent_souvent$
poules_couvent_souvent$
s_couvent_souvent$
s_poules_couvent_souvent$
souvent$
t$
t_souvent$
ules_couvent_souvent$
uvent$
uvent_souvent$
vent$
vent_souvent$
```

+sort

Revenons aux poules . . .

	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6
S	l	e	s	_	p	o	u	l	e	s	_	c	o	u	v	e	n	t	_	s	o	u	v	e	n	t

	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6
SA	11	4	19	12	24	16	9	2	8	1	25	17	6	21	14	5	10	3	20	26	18	7	22	14	23	15
		_		c		e				l		n		o		p		s		t		u				v

	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6
BWT	s	s	t	_	v	v	l	l	u	$	e	e	p	s	c	_	e	e	_	n	n	o	o	o	u	u
	1	1	1	0	1	1	0	0	1	0^2	0	0	1	1	0	0	0	0	0	0	0	1	1	1	1	1

- $C[i]$ premier indice dans SA d'un suffixe commençant par $\Sigma[i]$ (calculé en même temps que SA) :

	_	c	e	l	n	o	p	s	t	u	v
C	0	3	4	8	10	12	15	16	19	21	24

2. Par cohérence, il faut ajouter $ dans l'alphabet, ce que je n'ai pas fait...

FM-Index [Ferragina et al., 2007]

- SA + BWT = FM-Index (yeh)
- autorise le backsearch : recherche efficace de toutes les occurrences d'un patron P (chaîne) dans un texte
- en réduisant itérativement l'intervalle dans SA des suffixes qui commencent par P (ces suffixes sont adjacents dans SA)

  $s \leftarrow 1, e \leftarrow n, i \leftarrow |P|$
  repeat
  $s' \leftarrow C[P[i]] + rank_{BWT}(P[i], s-1) + 1$
  $e' \leftarrow C[P[i]] + rank_{BWT}(P[i], e)$
  $i \leftarrow i - 1$
  until i == 0
  return $[s', e']$

- $o(|P| \times \log_2(\sigma))$ [3]. Sans cela, la recherche d'un patron requiert $o(\log_2(n))$ comparaisons de chaînes

3. Attendez encore un peu . . .

Backsearch et les poules

Recherchons **toutes** les occurrences de P=ouv

1. $i = 3, s = 1, e = 26, C[P[i]] = C[v] = 24$

$$\begin{aligned} s' &= 24 + rank_{bwt}(v, 0) + 1 = 25 \\ e' &= 24 + rank_{bwt}(v, 26) = 26 \end{aligned}$$

$[25, 26]$ identifie (dans SFX) les suffixes commençant par v.

Backsearch et les poules

Recherchons **toutes** les occurrences de P=ouv

1. $i = 3, s = 1, e = 26, C[P[i]] = C[v] = 24$

$$\begin{aligned} s' &= 24 + rank_{bwt}(v, 0) + 1 = 25 \\ e' &= 24 + rank_{bwt}(v, 26) = 26 \end{aligned}$$

$[25, 26]$ identifie (dans SFX) les suffixes commençant par v.

2. $i = 2, s = 25, e = 26, C[P[i]] = c[u] = 21$

$$\begin{aligned} s' &= 21 + rank_{bwt}(u, 24) + 1 = 23 \\ e' &= 21 + rank_{bwt}(u, 26) = 24 \end{aligned}$$

$[23, 24]$ identifie (dans SFX) les suffixes commençant par uv.

Backsearch et les poules

Recherchons **toutes** les occurrences de P=ouv

1. $i = 3, s = 1, e = 26, C[P[i]] = C[v] = 24$

$$\begin{aligned} s' &= 24 + rank_{bwt}(v, 0) + 1 = 25 \\ e' &= 24 + rank_{bwt}(v, 26) = 26 \end{aligned}$$

$[25, 26]$ identifie (dans SFX) les suffixes commençant par v.

2. $i = 2, s = 25, e = 26, C[P[i]] = c[u] = 21$

$$\begin{aligned} s' &= 21 + rank_{bwt}(u, 24) + 1 = 23 \\ e' &= 21 + rank_{bwt}(u, 26) = 24 \end{aligned}$$

$[23, 24]$ identifie (dans SFX) les suffixes commençant par uv.

3. $i = 1, s = 23, e = 24, C[P[i]] = c[o] = 12$

$$\begin{aligned} s' &= 12 + rank_{bwt}(o, 22) + 1 = 14 \\ e' &= 12 + rank_{bwt}(o, 24) = 15 \end{aligned}$$

$[23, 24]$ est la réponse !

rali

Structure RRR [Raman et al., 2007]

- structure qui offre la compression d'une séquence de bits
- et qui, selon le codage, peut s'accompagner d'une requête rank en temps constant ($o(1)$)

- Application :
  - utilisé par exemple pour encoder BWT, ce qui permet (via *backsearch*) une recherche d'un patron P en $o(|P|)$
    - pensez au temps que mettrait à la recherche dans un énorme texte de toutes les occurrences d'une chaîne particulière !
  - utilisé pour accéder efficacement (en temps et en mémoire) aux comptes d'un modèle de langue sans hypothèse markovienne [Shareghi et al., 2016]

rali

Structure RRR *[Raman et al., 2007]*

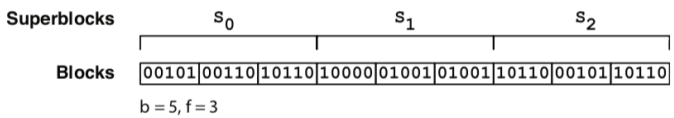


Figure 15: Block division scheme for 'Peter Piper...' Wavelet Tree's root node bit vector.

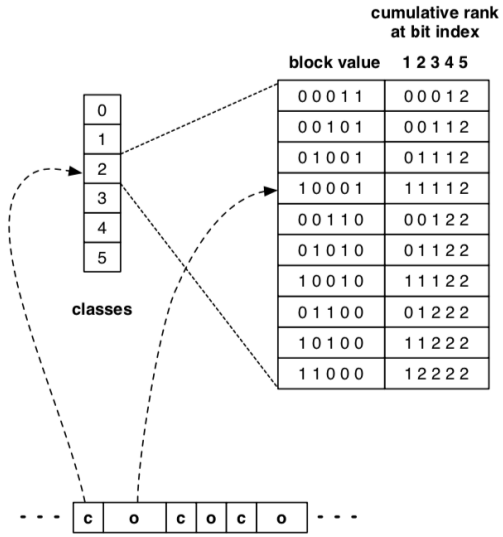


Figure 16: Binary RRR Count Table, with example lookup for class $c = 2$ and offset $o = 3$ in a RRR sequence.

- la clé d'accès à un bloc est son popcount c (nombre de bits à 1) ainsi que son offset o dans la table
- à un bloc est associé la somme de ses bits de 1 à $i \in [1, b]$
- il y a au plus $\binom{b}{c}$ offsets dans $G[c]$, et b+1 popcounts différents

Structure RRR *[Raman et al., 2007]*

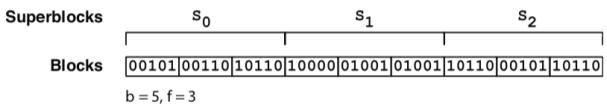


Figure 15: Block division scheme for 'Peter Piper...' Wavelet Tree's root node bit vector.

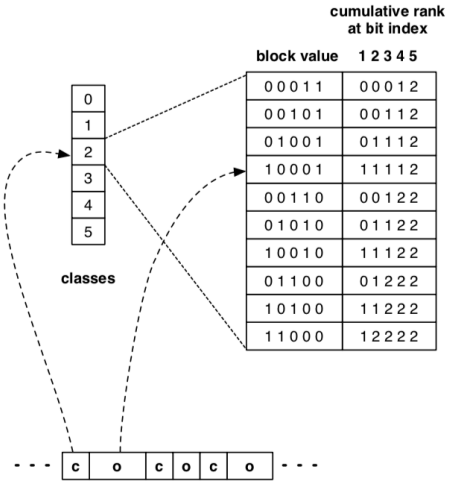


Figure 16: Binary RRR Count Table, with example lookup for class $c = 2$ and offset $o = 3$ in a RRR sequence.

- la compression (controlée par b) vient du fait qu'un bloc n'est codé qu'une seule fois.
- Si on stocke les comptes aux bornes des super-blocks (f en contrôle le nombre), calculer une requête $rank(b, i)$ revient à :
  - calculer block ($i_b = i/b$) et super-block ($i_s = i_b/f$)
  - prendre les comptes pré-calculés du super-bloc correspondant
  - ajouter les comptes des blocs après le superblock ($i_b \% f$)
  - ajouter le compte $G[c][o]$.cumulative$[i\%b]$

Récréation : Trouver le bon mot

If a fire breaks out, the alarm will ??

The boy doesn't know how to ?? his bicycle

The American congress can ?? a presidential veto

Before eating your bag of microwavable popcorn, you have to ?? it

Récréation : Trouver le bon mot

If a fire breaks out, the alarm will ring, sound

The boy doesn't know how to ride his bicycle

The American congress can overrule a presidential veto

Before eating your bag of microwavable popcorn, you have to nuke, cook it

---

Exemple de collocations pris de ***[Smadja, 93]***

Identifier des unités représentatives

- collocations (ex : *faim de loup*), formes figées (ex : *au fur et à mesure*), mots clés, termes, etc. [4]
- tâche non définie clairement (dépend de l'application visée), mais seriez-vous capable de classer ces séquences suivantes comme pertinentes/représentatives ou pas ?

▷ intérêts du canada	▷ intérêts des sociétés multi-nationales
▷ fin du printemps	▷ enfreint le règlement
▷ les lignes directrices sur les conflits d' intérêts	▷ n' a pas
▷ le projet de loi	▷ : monsieur le président , je
▷ le gouvernement ne	▷ et des

4. Lire les travaux de Igor Melcuk(UdeM/Linguistique)

Sélection par la fréquence

- pensez à la façon de coder cela . . . (hit : suffix arrays !)

long.	fréq.	séquence	long.	fréq.	séquence
2	1760	de la	2	893	) :
2	1594	de l’	2	867	qu’ il
2	1391	le président	2	778	, le
2	1083	monsieur le	2	724	: monsieur
3	1076	monsieur le président	3	724	: monsieur le
2	1069	le gouvernement	4	723	: monsieur le président
2	1040	à la	5	720	: monsieur le président ,
2	988	c’ est	2	701	à l’
2	985	président ,	2	643	le député
3	983	le président ,	2	640	, je
4	971	monsieur le président ,	3	608	) : monsieur
2	913	que le	5	608	) : monsieur le président

- pas très probant ...

PMI

- bcp de statistiques (lire [Manning and Schtze, 1999], chap 5),
- Pointwise Mutual Information en est une (information mutuelle ponctuelle)

$$I(x,y) = \log_2 \frac{P(x,y)}{P(x)P(y)} = \log_2 \frac{P(x|y)}{P(x)} = \log_2 \frac{P(y|x)}{P(y)}$$

- Ex. :[5]

	chambre	¬ chambre
house	31950	12004
¬ house	4793	848330

	communes	¬ communes
house	4974	38980
¬ house	441	852682

$$\log \frac{P(house|chambre)}{p(house)} = \log \frac{\frac{31950}{31950+4793}}{P(house)} \approx \log \frac{0.87}{P(house)}$$

$$\log \frac{P(house|communes)}{p(house)} = \log \frac{\frac{4974}{4974+441}}{P(house)} \approx \log \frac{0.92}{P(house)}$$

- très populaire, malgré une faiblesse connue (la voyez-vous ?)

5. Pris de [Manning and Schtze, 1999], page 179 (cas bilingue)

Likelihood ratio *[Dunning, 1993]*

- Deux événements A et B qui co-occurrent sont-ils reliés ?

$$\begin{aligned} -\log\lambda \quad = \quad & a\log a + b\log b + c\log c + d\log d + N\log N \\ & -(a+c)\log(a+c) - (a+b)\log(a+b) \\ & -(c+d)\log(c+d) - (d+b)\log(d+b) \end{aligned}$$

- Ex (co-occurrence = se suivent)

a : est le nombre de fois où A et B se suivent
b : est le nombre de fois où A apparait non suivi de B
c : est le nombre de fois où B est non précédé de A
d : est le nombre de fois où ni A ni B n'apparaissent, dans cet ordre, dans le corpus
N : a + b + c + d = taille du corpus

- Plus la quantité $-\log\lambda$ est grande, plus les événements sont statistiquement non indépendants

Likelihood ratio *[Dunning, 1993]*

- Le rapport de vraisemblance d'une hypothèse particulière (H_0) est donné par :

$$\lambda = \frac{max_{\omega \in \Omega_0} H(\omega; k)}{max_{\omega \in \Omega} H(\omega; k)}$$

où Ω est l'espace des paramètres, et Ω_0 est l'espace des paramètres correspondant à l'hypothèse ; k résume l'observation

- λ est positif et ≤ 1. Plus il est proche de 1, plus cela valide H_0
- Si on admet que événements observés sont la résultante d'une épreuve de Bernouilli (tirage au sort), la statistique suffisante est le compte de tirages "positifs" et la vraisemblance est régie par :

$$H(p; k, n) = \binom{n}{k} p^k (1-p)^{n-k}$$

Likelihood ratio *[Dunning, 1993]*

- $H_0 : p = p_1 = p_2$.
- le rapport de vraisemblance s'écrit :

$$\lambda = \frac{max_p H(p, p; k_1, n_1, k_2, n_2)}{max_{p_1,p_2} H(p_1, p_2; k_1, n_1, k_2, n_2)}$$

- Le maximum du dénominateur est obtenu pour $p_1 = \frac{k_1}{n_1}$ et $p_2 = \frac{k_2}{n_2}$

Likelihood ratio *[Dunning, 1993]*

- en posant $L(p, k, n) = p^k(1-p)^{n-k}$, on a :

$$\lambda = \frac{L(p, k_1, n_1) \times L(p, k_2, n_2)}{L(p_1, k_1, n_1) \times L(p2, k_2, n_2)}$$

avec $p_1 = \frac{k_1}{n_1}$, $p_2 = \frac{k_2}{n_2}$, et $p = \frac{k_1+k_2}{n_1+n2}$

- soit, en prenant le logarithme :

$$\begin{aligned} -\log \lambda = \quad & \log L(p_1, k_1, n_1) + \log L(p_2, k_2, n_2) \\ & - \log L(p, k_1, n_1) - \log L(p, k_2, n_2) \end{aligned}$$

- une grande valeur invalide H_0

Likelihood ratio *[Dunning, 1993]*

- Soit A et B deux mots (qui se suivent) dont nous voulons mesurer le degré de dépendance.
- H_0 (indépendance) : $p_1 = P(B|A) \equiv p_2 = P(B|\neg A) \equiv P(B)$
- Soit la table de contingence suivante (où par exemple, a (resp. b) indique le nombre de fois où B suit (resp. ne suit pas) A dans un corpus) :

	B	$\neg B$
A	a	b
$\neg A$	c	d

- $p_1 = \frac{a}{a+b}$ et $p_2 = \frac{c}{c+d}$.
- en développant, on retombe sur la formulation du rapport de vraisemblance de l'acétate 76
- une grande valeur de $-\log \lambda$ invalide l'hypothèse d'indépendance et milite donc en faveur de la dépendance

Test de vraisemblance

- 16 premières (et 4 dernières) séquences de deux mots selon le score $-\log\lambda$ sur 10 tranches du Hansard :

fréq.	$-\log\lambda$	séquence	fréq.	$-\log\lambda$	séquence
1391	4690.07	le président	985	2502.4	président ,
988	4400.53	c’ est	307	2387.39	p. 100
1083	3935.63	monsieur le	592	2117.01	la chambre
893	3504.88	) :	1594	1943.32	de l’
724	2930.47	: monsieur	1760	1896.21	de la
540	2899.87	j’ ai	643	1679.89	le député
1069	2734.26	le gouvernement	385	1670.77	nous avons
867	2650.22	qu’ il	293	1509.37	premier ministre
2	3.7e-6	maintenant un	10	2.2e-6	loi que
2	2.3e-6	là a	4	7.9e-7	jamais .

- **Question :** comment étendre cette métrique à des séquences de taille variable ?

Étendre une mesure d'association binaire

- Une séquence est non pertinente si on la retrouve comme préfixe ou suffixe de séquences pertinentes ***[Ries et al., 1995]***.
- Le score ρ d'une séquence w_1^n $(n \geq 2)$ peut être calculé à l'aide d'une mesure d'association binaire A par :

$$\rho(w_1^n) = \min_{i \in [1, n-1]} A(w_1^i, w_{i+1}^n)$$

- ce score mesure la "résistance d'une séquence à la division", et donc reflète d'une certaine manière son degré de cohésion

Étendre une mesure d'association binaire

- Les 12 séquences les mieux notées selon ρ sur 10 tranches du Hansard, et les 4 séquences les moins bien notées :

fréq.	$\rho(s)$	s=séquence	fréq.	$\rho(s)$	s=séquence
1076	6297.4	monsieur le président	608	3120.2	) : monsieur
1391	4690.0	le président	971	3075.9	monsieur le président ,
988	4400.5	c' est	723	2936.7	: monsieur le président
1083	3935.6	monsieur le	724	2930.4	: monsieur
893	3504.8	) :	540	2899.8	j' ai
608	3122.0	) : monsieur le président	459	2875.2	projet de loi
3	1.8e-6	est sur le	3	5.9e-7	des mesures .
4	7.9e-7	jamais .	2	3.9e-7	la souveraineté .

Score d'entropie *[Shimohata et al., 1997]*

$$
\begin{aligned}
e(w_1^n) &= (e_{left}(w_1^n) + e_{right}(w_1^n))/2 \\
e_{left}(s) &= \sum_{w/ws \in T} h(\frac{|ws|}{|s|}) \\
e_{right}(s) &= \sum_{w/sw \in T} h(\frac{|sw|}{|s|}) \\
h(x) &= x \log(x)
\end{aligned}
$$

- $e_{left}(s)$ (resp. $e_{right}(s)$) est nul quand seulement une forme suit (resp. précède) s dans toutes les occurrences de s
- il est maximal, lorsqu'il y a exactement $freq(s)$ types qui suivent (resp. précèdent) s.

- Intuitivement : une séquence cohérente devrait apparaître dans un nombre varié de contextes (entropie élevée)

rali

Score d'entropie

$$\left.\begin{array}{l} – \\ , \\ : \\ ce \end{array}\right\} \text{ monsieur } \left\{\begin{array}{l} a \\ le \end{array}\right.$$

faible entropie

$$(364) \left\{\begin{array}{l} abandon \\ accepté \\ \vdots \\ vraiment \end{array}\right\} \text{ le } \left\{\begin{array}{l} 10 \\ baril \\ \vdots \\ yukon \end{array}\right\} (440)$$

forte entropie

Score d'entropie

fréq.	$e(s)$	s=séquence	fréq.	$e(s)$	s=séquence
1760	2.5362	de la	393	2.39531	d' un
333	2.53098	a été	517	2.34532	que les
370	2.46355	, les	371	2.34049	d' une
256	2.44128	et les	156	2.2844	ont été
385	2.4374	nous avons	379	2.28096	il est

- 40% des séquences de deux mots ou plus ont un score nul avec cette métrique.

Séquences ($f \geq 2$) contenant aviation safety

	$\rho(s)$	$e(s)$	freq.	l	s
1	113.32	0.23	16	2	aviation safety
2	109.19	1.14	15	3	aviation safety board
3	92.02	0.24	15	3	canadian aviation safety
4	85.69	1.13	14	4	canadian aviation safety board
5	35.33	1.23	14	4	the canadian aviation safety
6	32.62	1.99	13	5	the canadian aviation safety board
7	12.91	0.69	2	4	aviation safety board recommendations
8	6.29	1.39	4	6	the canadian aviation safety board ,
9	5.97	0.69	4	5	canadian aviation safety board ,
10	5.67	0.69	4	4	aviation safety board ,
11	5.65	0.69	2	6	by the canadian aviation safety board
12	5.49	0.35	2	5	by the canadian aviation safety
13	5.28	0.35	2	7	of the canadian aviation safety board .
14	3.81	1.10	3	6	, the canadian aviation safety board
15	3.59	0.55	3	5	, the canadian aviation safety
16	2.71	0.32	3	6	the canadian aviation safety board .

- on peut (par ex.) éliminer une séquence si c'est le préfixe d'une séquence mieux notée par l'entropie
  ex : aviation safety (0.23) versus aviation safety board (1.14)

Bowe, A. (2010).
Multiary wavelet trees in practice.
Master's thesis, U. Melbourne.

Bowman, S. R., Angeli, G., Potts, C., and Manning, C. D. (2015).
A large annotated corpus for learning natural language inference.
In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642.

Cohen, W. W., Ravikumar, P., and Fienberg, S. E. (2003).
A comparison of string distance metrics for name-matching tasks.
In *Proceedings of the 2003 International Conference on Information Integration on the Web*, pages 73–78.

Crochemore, M., Hancart, C., and Lecroq, T. (2001).
Algorithmique du texte.

rali

Number ISBN-2-7117-8628-5. Vuibert, vuibert informatique edition.

Dunning, T. (1993).
Accurate methods for the statistics of surprise and coincidence.
Computational Linguistics, 19(1).

Ferragina, P., Manzini, G., Mäkinen, V., and Navarro, G. (2007).
Compressed representations of sequences and full-text indexes.

ACM Trans. Algorithms, 3(2).

Levenshtein, V. I. (1966).
Binary codes capable of correcting deletions, insertions and reversals.
Sov. Phys. Dokl., 6 :707–710.

Manning, C. D. and Schtze, H. (1999).
Foundations of Statistical Natural Language Processing.
MIT Press.

rali

Needleman, S. B. and Wunsch, C. D. (1970).
A general method applicable to the search for similarities in the amino acid sequence of two proteins.
Journal of molecular biology, 48(3) :443–453.

Padó, S., Noh, T.-G., Stern, A., Wang, R., and Zanoli, R. (2015).
Design and realization of a modular architecture for textual entailment.
Natural Language Engineering, 21(2) :167–200.

Raman, R., Raman, V., and Satti, S. R. (2007).
Succinct indexable dictionaries with applications to encoding k-ary trees, prefix sums and multisets.
CoRR, abs/0705.0552.

Ries, K., Buo, F. D., and Wang, Y.-Y. (1995).
Improved language modeling by unsupervised acquisition of structure.

In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP) 1995*, pages 193–196, Detroit, Michigan. IEEE.

Russell, G. (1998).
Identification of salient token sequences.
Internal Report, RALI.

Shareghi, E., Petri, M., Haffari, G., and Cohn, T. (2016).
Fast, small and exact : Infinite-order language modelling with compressed suffix trees.
Transactions of the Association for Computational Linguistics, 4 :477–490.

Shimohata, S., Sugio, T., and Nagata, J. (1997).
Retrieving collocations by co-occurrences and word order constraints.
In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 476–481, Madrid, Spain.

Smadja, F. (93).

rali

Retrieving collocations from text : Xtract.
Computational Linguistics, 19(1) :144–177.

Ukkonen, E. (1985).
Algorithms for approximate string matching.
Inf. Control, 64(1-3) :100–118.