Complexité:

- **En temps:** Pour chaque d et chaque i, retrouver la plus longue extension le long de la diagonale i. Correspond à trouver un préfixe commun entre un suffixe de P et un suffixe de T commençant à des positions connues (peut se faire en temps constant);
  - **Galil-Park** Prétraitement de P + utilisation de triplets de référence. $\mathcal{O}(m^2)$
  - **Landau-Vishkin** Arbres des suffixes de $t_1 t_2 \ldots t_n p_1 p_2 \ldots p_m$. $\mathcal{O}(m+n)$
  - **Ukkonen-Wood** Automate des suffixes $\mathcal{O}(m)$

$$\Longrightarrow \mathcal{O}(t(n+m)) + \text{prétraitement} = \mathcal{O}(tn)$$

On définit $L(e, d)$ comme étant la ligne d'indice le plus grand, sur la diagonale d qui a comme valeur e.

Avec cette définition, si $L(e, d) = m$ alors il y a une occurrence approximative de P, ayant e erreurs, se terminant en position $L(e, d) + d = m + d$ du texte.

Exemple:

Soit $T = AACGAGGAAC$, $P = AACGGG$ et $t = 2$, le nombre d'erreurs permis dans l'alignement:

	λ	A	A	C	G	A	G	G	A	A	C
λ	0	0	0	0	0	0	0	0	0	0	0
A	1		0								
A	2			1							
C	3										
G	4				0						
G	5				1						
G	6						1				

Calcul des $L(e, d)$:

$i \leftarrow \max(L(e-1, d-1)+1, L(e-1, d)+1, L(e-1, d+1))$
tant que $p_{i+1} = t_{d+i+1}$ **faire**
    $i \leftarrow i+1$
fin tant que
$L(e, d) \leftarrow i$

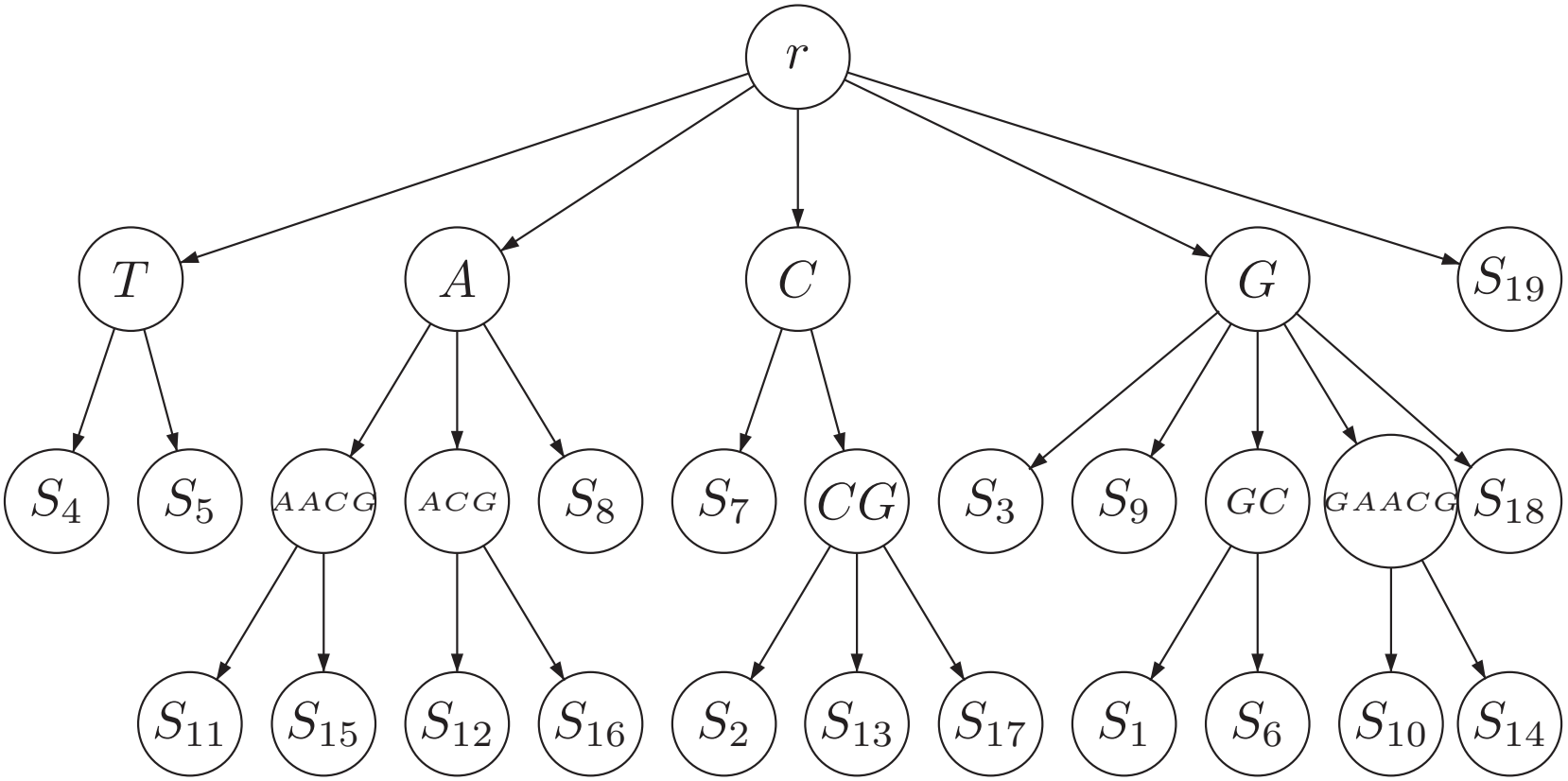
- Le but des 3 algorithmes est de calculer la boucle **tant que** en temps constant

Idée de Landau-Vishkin

1. On fait la concaténation du texte T et du mot P cherché et on obtient une séquence $S = t_1 \dots t_n p_1 \dots p_m$. On construit l'arbre des suffixes de $S\#$ en temps $\mathcal{O}(n+m)$.

2. On trouve les occurrences approximatives de P dans T, ayant au plus t erreurs, en calculant les L-diagonales correspondant à la table des distances D pour la recherche approximative de P dans T

Exemple: Si on a le mot $P = AACG$ et le texte $T = GCGTTGCAGGAACG$ alors, l'arbre des suffixes de $S = GCGTTGCAGGAACGAACG\#$ est:



Définition: On dénote $LCA(i, d)$ le *plus petit ancêtre commun* des feuilles représentant le suffixe $p_{i+1} \ldots p_m$ de P et le suffixe $t_{i+d+1} \ldots t_n$ de T.

Avec cette définition, l'algorithme de calcul des $L(e, d)$ devient:

$i \leftarrow \max(L(e-1, d-1)+1, L(e-1, d)+1, L(e-1, d+1))$

$q \leftarrow \textbf{longueur}(LCA(i, d))$

$L(e, d) \leftarrow i + q$

Idée de Ukkonen-Wood

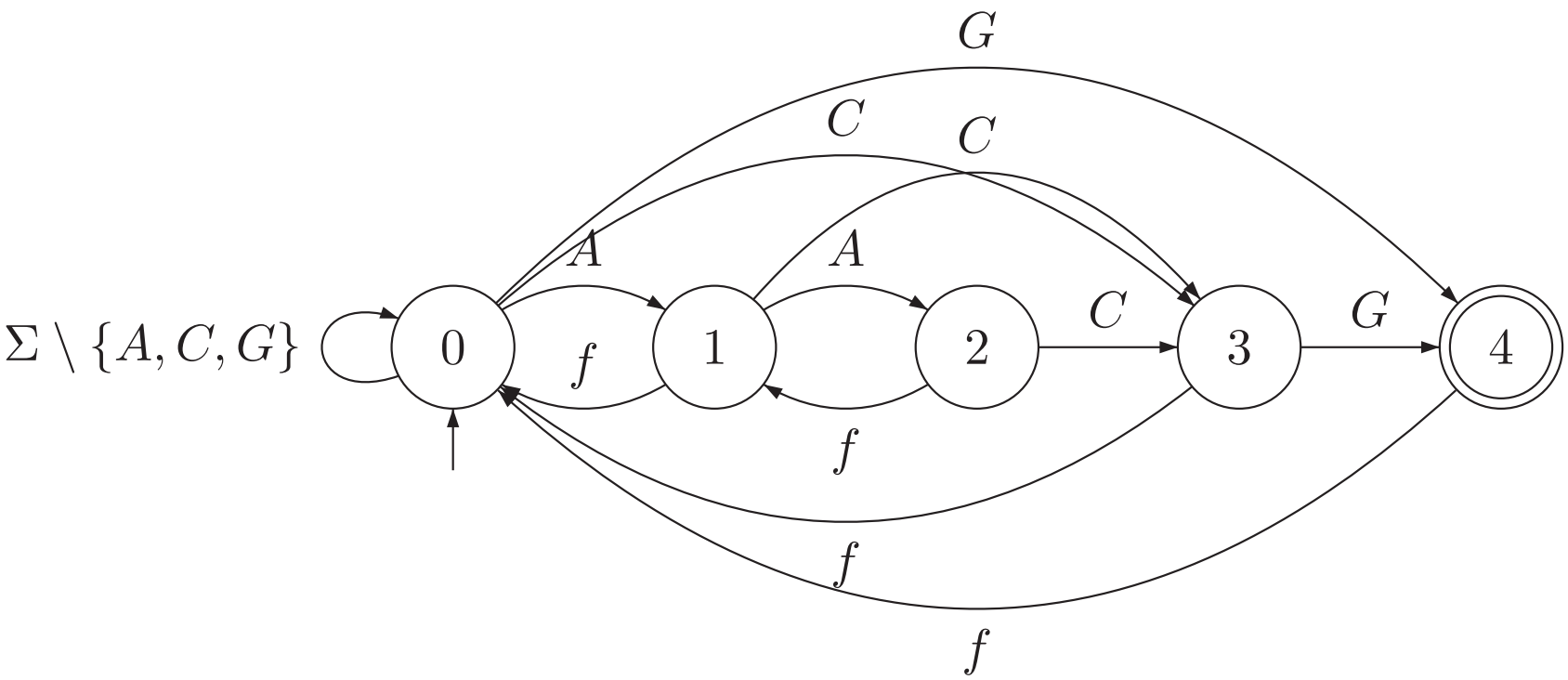
Ukkonen et Wood font un prétraitement de P consistant en la construction de l'*automate suffixe*, $S(P)$, de P.

L'automate $S(P)$ reconnaît le langage

$$\Sigma^* \cdot (P_1 + P_2 + \ldots + P_m),$$

où Σ est l'alphabet du texte T considéré et où $P_i = p_i \ldots p_m$. On appelle $S(P)$ l'automate suffixe de P étant donné qu'il reconnaît tous les mots se terminant par un suffixe de P.

Exemple: L'automate $S(P)$ du mot $P = AACG$ est le suivant:



Après la lecture de $t_1 \ldots t_j$ l'automate nous donne l'information suivante:

1. La longueur du plus long suffixe de $t_1 \ldots t_j$ qui est aussi un préfixe de P_1 ou P_2 ou $\ldots$ ou P_m.
2. La position dans P où se termine ce suffixe.

Cette information permet à Ukkonen et Wood de calculer les plus longs préfixes communs à tous les suffixes de P et T en temps constant et on a donc, encore une fois un algorithme ayant une complexité en temps de $\mathcal{O}(tn)$.

La méthode des 4 Russes[a] :

- En 1970, quatre mathématiciens russes, Arlazarov, Dinic, Kronrod et Faradzev, publient un algorithme permettant de calculer les entrées d'une matrice de dimension $n \times n$, reliée au calcul de la fermeture transitive d'un graphe orienté sur n sommets, en temps $\mathcal{O}(n^2/\log n)$

- On choisi un nombre l est choisi assez petit et qui divise parfaitement n. Ensuite, on subdivise la grande matrice $n \times n$ en sous-matrices de dimension $l \times l$ et on se sert des résultats précalculés pour trouver le résultat final.

- Cette méthode est communément appelée la *méthode des 4 Russes*.

[a] Arlazarov, V.L., Dinic, E.A., Kronrod, M.A. et Faradzev, I.A., *On economic construction of the transitive closure of a directed graph*, Dokl. Akad. Nauk. SSSR, 194, pp. 487–488 1970 (en Russe). Tranduction anglaise dans *Soviet Math. Dokl.*, 11, pp. 1209–1210, 1975.

Algo. de Masek-Paterson (Alig. global)[a] :

- En 1980, Masek et Paterson ont l'idée de se servir de la méthode des 4 Russes pour résoudre le problème de l'alignement global de deux mots en temps $\mathcal{O}(n^2/\log n)$.
- Étant donné deux mots de longueur n, ils commencent par choisir un petit nombre k, tel que k divise n. Ils précalculent ensuite toutes les sous-matrices possibles de dimension $(k+1) \times (k+1)$.
- Le calcul d'une de ces sous-matrices est entièrement déterminé par sa première ligne L, sa première colonne C et deux mots, m_1 et m_2, de longueur k sur l'alphabet Σ considéré.

[a] Masek, W.J. et Paterson, M.S., *A Faster Algorithm Computing String Edit Distance*, Journal of Computer and System Sciences, 20, pp. 18–31, 1980.

Algo. de Masek-Paterson (suite):

- Étant donné L, C, m_1 et m_2 fixés, on garde en mémoire la dernière ligne et la dernière colonne de la sous-matrice des distances correspondante.
- L'algorithme se termine par le calcul de la matrice des distances D, sous-matrices par sous-matrices, au lieu de cellule par cellule.
- Mais combien de sous-matrices doit-on précalculer avec cette méthode?

Combien de sous-matrices?

- Les lignes et les colonnes d'une table de distances D, entre deux mots de longueur n, comprennent des valeurs entre 0 et n. De plus, deux entrées successives, sur une même ligne ou une même colonne, diffèrent toujours de 0, 1 ou -1. On a donc environ $n \cdot 3^k$ premières lignes ou colonnes possibles dans des matrices de dimension $(k+1) \times (k+1)$.
- De plus, le nombre de mots de longueur k sur un alphabet Σ est $|\Sigma|^k$.
- On doit donc précalculer $n^2 \cdot 3^{2k} \cdot |\Sigma|^{2k}$ matrices de dimension $(k+1) \times (k+1)$.
- Chacune de ces matrice est calculée en temps $\mathcal{O}(k^2)$.

Exemple:

- Supposons que l'on veuille calculer la distance d'édition entre les deux séquences $GTCAGG$ et $CATAGT$.
- Ici la longueur des mots est 6 et on peut prendre, par exemple, $k = 3$.
- La première étape, que je ne ferai pas ici, consiste à précalculer toutes les sous-matrices de dimension 4×4 possibles sur l'alphabet $\{A, C, G, T\}$ (Il y a $6^2 \cdot 3^6 \cdot 4^6 = 107\ 495\ 424$)
- Garder en mémoire la dernière ligne et la dernière colonne de chacune de ces sous-matrices.

Exemple(suite): Initialisons la table des distances pour ce problème et subdivisons-la en matrices de dimension 4×4:

	λ	G	T	C	A	G	G
λ	0	1	2	3	4	5	6
C	1						
A	2						
T	3						
A	4						
G	5						
T	6						

Exemple(suite): On commence par aller chercher la dernière ligne et la dernière colonne de la sous-matrice en haut à gauche, c'est-à-dire la sous-matrice ayant comme première ligne 0123, comme première colonne 0123 et dont les sous-mots à aligner sont GTC et CAT. On obtient ce qui suit:

	λ	G	T	C	A	G	G
λ	0	1	2	3	4	5	6
C	1			2			
A	2			3			
T	3	3	2	3			
A	4						
G	5						
T	6						

Exemple(suite): On a maintenant l'information nécessaire au calcul des dernières lignes et colonnes de deux nouvelles sous-matrices. La première correspond aux vecteurs 3456, 3233 et aux sous-mots AGG et CAT et la deuxième aux vecteurs 3323 et 3456 et aux sous-mots GTC et AGT:

	λ	G	T	C	A	G	G
λ	0	1	2	3	4	5	6
C	1			2			5
A	2			3			4
T	3	3	2	3	3	3	4
A	4			3			
G	5			4			
T	6	5	4	5			

Exemple(suite): Finalement, on va chercher l'information correspondant à la dernière sous-matrice, en bas à droite. On obtient alors que la distance d'édition entre $GTCAGG$ et $CATAGT$ est 4::

	λ	G	T	C	A	G	G
λ	0	1	2	3	4	5	6
C	1			2			5
A	2			3			4
T	3	3	2	3	3	3	4
A	4			3			4
G	5			4			4
T	6	5	4	5	5	4	4

Idée pour réduire le nombre de sous-matrices:

- L'idée, pour diminuer le nombre de sous-matrices de dimension $(k+1) \times (k+1)$ à calculer, est de travailler avec des vecteurs de différences horizontales $(D(i,j) - D(i,j-1))$ et de différences verticales $(D(i,j) - D(i-1,j))$ au lieu de travailler avec des lignes et des colonnes pouvant apparaître dans la table D.

Exemple:

La sous-matrice suivante de la table des distances de l'exemple précédent était représentée par sa première ligne 3456, sa première colonne 3233 ainsi que les sous-mots AGG et CAT:

	C	A	G	G
λ	3	4	5	6
C	2			
A	3			
T	3			

Nous la représenterons maintenant par les mêmes sous-mots AGG et CAT et par le vecteur de différences horizontales 111, associé à la ligne 3456, et le vecteur de différences verticales -110, associé à la colonne 3233.

Idée (suite):

- Étant donné un vecteur de différences horizontales, correspondant à la première ligne d'une sous-matrice de distances de dimension $(k+1) \times (k+1)$, un vecteur de différences verticales, correspondant à la première colonne de la même sous-matrice, et deux mots m_1 et m_2 sur un alphabet Σ, Masek et Paterson donne un algorithme pour le calcul, en temps $\mathcal{O}(k^2)$, du vecteur de différences horizontales, correspondant à la dernière ligne de la sous-matrice, et du vecteur de différences verticales, correspondant à la dernière colonne de la sous-matrice
- Comme il y a 3^k vecteur de différences possibles, le nombre de sous-matrices à précalculer est réduit à $3^{2k} \cdot |\Sigma|^{2k}$.

4 Russes et recherche approximative

Dans un article publié en 1996, Wu, Manber et Myers [a] présentent deux algorithmes pour résoudre le problème de la recherche des occurrences approximatives, ayant au plus t erreurs, d'un mot P dans un texte T, employant la méthode des 4 Russes et ayant une complexité en temps de $\mathcal{O}(nm/\log n)$ et $\mathcal{O}(nt/\log n)$, respectivement.

[a] Wu, S., Manber, U. et Myers, G., *A Subquadratic Algorithm for Approximate Limited Expression Matching*, Algorithmica, 15, pp. 50–67, 1996.