Algorithme de Knuth-Morris-Pratt (1977)

1 Principe

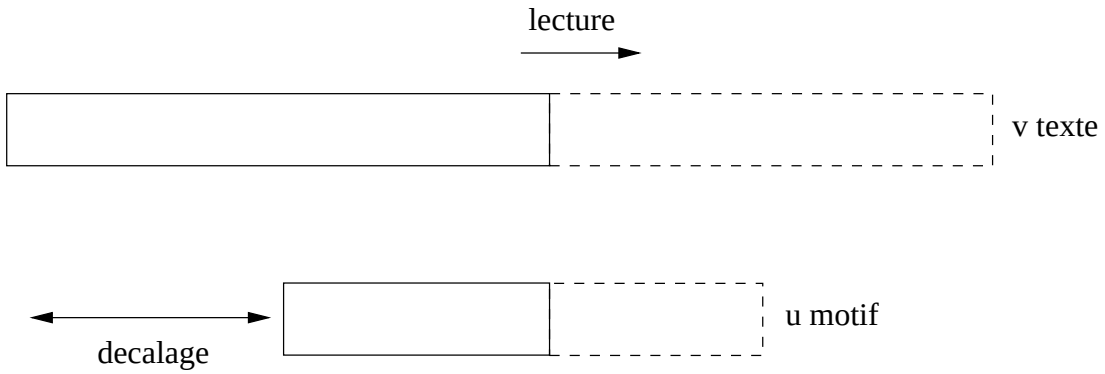

FIG. 1 – Principe de l'algorithme de Knuth-Morris-Pratt

Depuis un décalage de u, on avance simultanément la lecture sur u et v tant qu'on a égalité des lettres lues. Autrement dit, on calcule le plus grand suffixe de la partie de v lue qui est aussi préfixe de u.

Options de décalage

- Naïve : Décalage de u de 1. Mais obligation de recommencer le test de comparaison au départ.
- Plus économique : inutile de revenir en arrière sur v si on a bien calculé au fur et à mesure les plus grands suffixes des parties lues de v qui sont aussi préfixes de u. Il faut donc mieux penser les décalages. w' est un suffixe propre de w et aussi préfixe de u. Or, w est un préfixe de u et $|w'| < |w|$ donc w' est un préfixe propre de w. On va donc étudier les préfixes de w qui sont aussi des suffixes de w (notion de "bord" ou "bordure" de w). (Voir Fig.2)

Définition 1 (fonction π) *$\pi(w)$ est le plus grand préfixe propre de w qui est aussi un suffixe de w.*

L'idée est de calculer $\pi(w)$ pour tous les préfixes w de u. La bonne nouvelle est que ce calcul est possible en $\Theta(m)$ ($m = |u|$).

Lemme 1 (Formule de calcul incrémental)

$$\pi(wa) = \begin{cases} \pi(w)a & \text{si } w_{|\pi(w)|+1} = a \\ \pi(\pi(w)a) & \text{sinon} \end{cases}$$

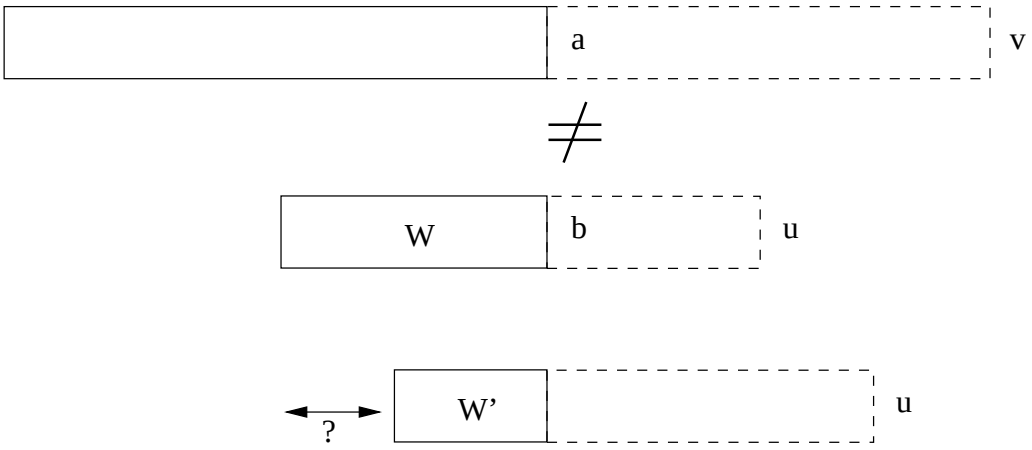

FIG. 2 – Mieux penser les décalages

Preuve : $\pi(wa)$ est un suffixe de wa donc $\pi(wa)$ s'écrit xa où x est un suffixe de w. De même, $\pi(wa)$ est un préfixe de wa donc x est un préfixe de w. Et donc, par définition de π, x est un préfixe et un suffixe de $\pi(w)$. On en déduit que $|\pi(wa)| = |x| + 1 \leq |\pi(w)| + 1$.

- Cas 1 : égalité, c'est à dire $|x| = |\pi(w)|$ soit $x = \pi(w)$ et $\pi(wa) = \pi(w)a$.
- Cas 2 : inégalité stricte : $|\pi(wa)| = |x| + 1 < |\pi(w)| + 1$

$$\left.\begin{array}{l} \pi(wa) \text{ suffixe de } wa \\ \pi(w)a \text{ suffixe de } wa \\ |\pi(wa)| < |\pi(w)| + 1 = |\pi(w)a| \end{array}\right\} \Rightarrow \pi(wa) \text{ suffixe propre de } \pi(w)a$$

$$\left.\begin{array}{l} \pi(wa) \text{ préfixe de } wa \\ \pi(w) \text{ préfixe de } wa \\ |\pi(wa)| \leq |\pi(w)| \end{array}\right\} \Rightarrow \begin{array}{l} \pi(wa) \text{ préfixe de } \pi(w) \\ \pi(wa) \text{ préfixe propre de} \pi(w)a \end{array}$$

Ainsi, par définition de π, $\pi(wa)$ est un préfixe et un suffixe de $\pi(\pi(w)a)$.

Inversement : si y est un préfixe et un suffixe propre de $\pi(w)a$, alors y est un préfixe de $\pi(w)$ donc de w donc de wa. Il est de plus un suffixe de wa. Et par définition de π, y est un préfixe et un suffixe de $\pi(wa)$. En appliquant à $y = \pi(\pi(w)a)$, on obtient que $\pi(\pi(w)a)$ est un préfixe et un suffixe de $\pi(wa)$.

Finalement, $\pi(wa) = \pi(\pi(w)a)$.

Définition 2 (fonction γ) *$\gamma(u, v)$ est le plus grand préfixe de u qui est aussi suffixe de v. (v est la partie lue du texte)*

Lemme 2 (Formule de calcul incrémental)

$$\gamma(u, va) = \begin{cases} \gamma(u, v)a & \text{si } u_{|\gamma(u,v)|+1} = a \\ \gamma(\pi(u), va) & \text{sinon} \\ \pi(\gamma(u, v)a) & \text{à priori OK} \end{cases}$$

L'avantage de la deuxième ligne sur la troisième, est que $\pi(u)$ est précalculé.

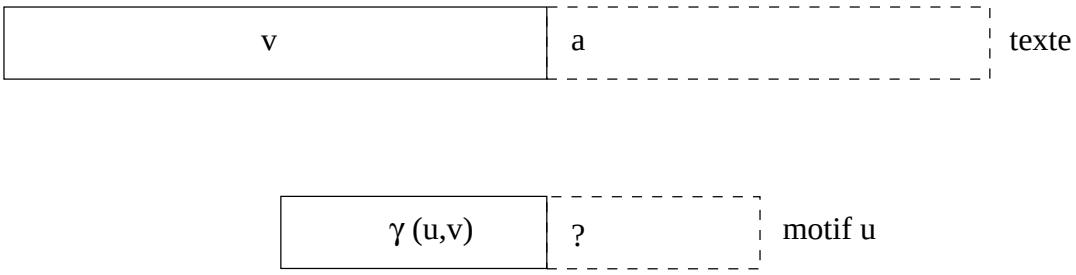

FIG. 3 – Calcul incrémental de γ

2 Implémentation

2.1 Préliminaires :

On a un texte $T[1...n]$ et un motif $P[1...m]$. On commence par calculer les bords pour P :

$$\Pi[i] = |\pi(P[1...i])| \ \forall \ 1 \leq i \leq m$$

Exemple, si $P = abaababa$.

Π	−1	0	0	1	1	2	3	2	3
	0	1	2	3	4	5	6	7	8

FIG. 4 – Exemple $P = abaababa$

Algorithm 1 Bords(P)

```
1: i ← 1, Π[0] ← −1
2: for j from 1 to m − 1 do
3:    Π[j] ← i
4:    while i ≥ 0 and P[j + 1] ≠ P[i + 1] do
5:       i ← Π[i]
6:    end while
7: end for
8: Π[m] ← i
9: return Π
```

Complexité : $\mathcal{O}(m)$, car la complexité est proportionnelle au nombre de test d'égalité de lettres. Or, pour chacun de ces tests, $2j - i = \alpha + \beta$ est strictement croissant, part de 0 et est majoré par $2m$. D'où une complexité en $\mathcal{O}(n)$.

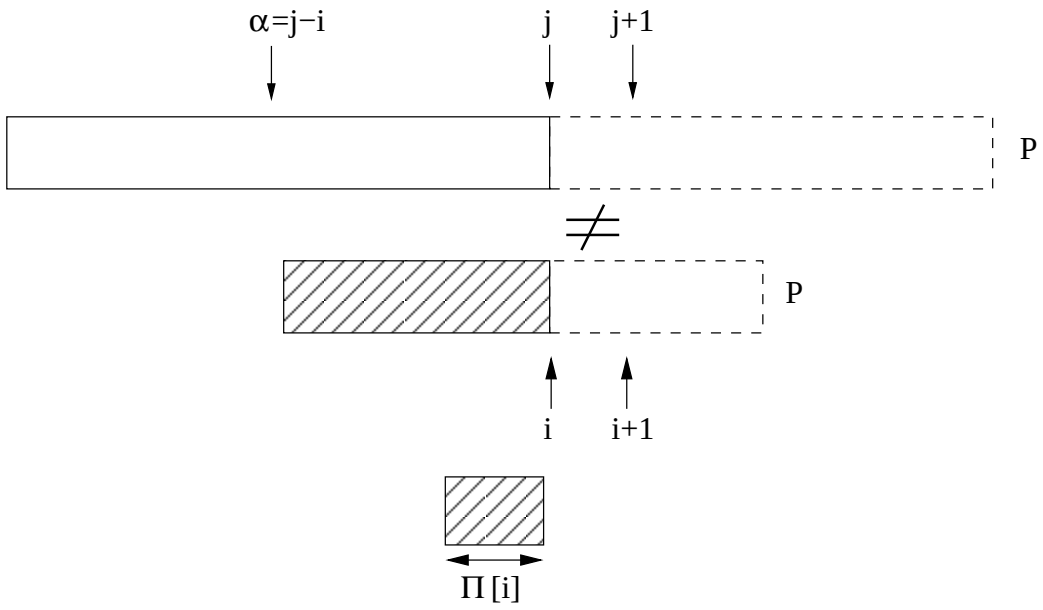

FIG. 5 – Calcul de Bord

2.2 Recherche :

Algorithm 2 Recherche(P, T)

```
1: Π ← Bords(P)
2: (i, j) ← (0, 0
3: while j < n do
4:    while i ≥ 0 and P[i + 1] ≠ T[j + 1] do
5:       i ← Π[i]
6:    end while
7:    (i, j) ← (i + 1, j + 1)
8:    if i = m then
9:       "Occurence !"
10:      i ← Π[i]
11:   end if
12: end while
```

Complexité : $\mathcal{O}(m+n)$.

- Calcul de Π : $\mathcal{O}(m)$
- Ensuite $\mathcal{O}(n)$, car $2j - i = \alpha + \beta$ est strictement croissant pour chaque test d'égalité, part de 0 et est majoré par $2n$. D'où une comlexité en $\mathcal{O}(n)$.

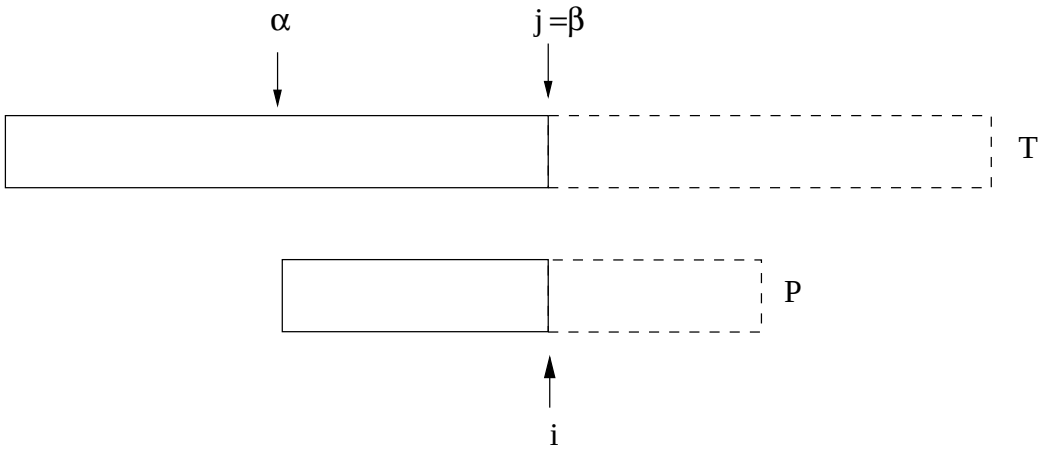

FIG. 6 – Recherche