Knuth Morris Pratt (Complexité)

Leçons : 907 ; 921 ; 926

Références : CORMEN–LEISERSON–RIVEST–STEIN, *Introduction à l'algorithmique* (p.921)

But de l'algorithme : trouver toutes les occurences d'un motif P de taille m dans un texte T de taille n.

Par convention, les tableaux commencent à l'indice 1.

Dans l'algorithme, on a besoin d'un tableau auxiliaire π qui pour chaque indice i du tableau donne LA LONGUEUR DU PLUS GRAND SUFFIXE DU SOUS-MOT DE P QUI FINIT À L'INDICE i QUI EST AUSSI PRÉFIXE PROPRE DE P. On peut dire aussi que le tableau π est le tableau qui pour chaque indice i donne LA LONGUEUR DU PLUS GRAND BORD DU SOUS MOT DE P QUI FINIT À L'INDICE i. La définition du bord d'un mot est justement un préfixe propre qui est aussi suffixe du mot.

Exemple : Le motif *abcabd* a pour tableau auxiliaire $\pi = [|0; 0; 0; 1; 2; 0|]$ car pour *a*, *ab*, *abc* leur bord est de longueur nulle (par convention *a* n'est pas suffixe et préfixe propre, donc de bord de longueur nulle). Cependant pour *abca*, *a* est préfixe propre et suffixe et sa longueur est 1, puis *abcab* a pour bord *ab* de longueur 2. Pour finir, *abcabd* n'a pas de bord de longueur non nulle.

Algorithme 1 Calcul-π

```
Fonction Calcul-π(P) :
m = longueur(P)
π = tableau de taille m
π[1] = 0
k = 0  [indice de parcours du motif P]
Pour i de 2 à m faire
    Tant que (k > 0 et π[k+1] ≠ π[i]) faire
        k = π[k]
    Fait
    Si (π[k+1] = π[i]) alors
        k = k + 1
    Fin Si
    π[i] = k
Fin Pour
    π
Fin
```

On veut étudier la complexité de cet algorithme. On a une boucle 'for' de longueur $m-1$ avec des opérations à coup constant à l'intérieur sauf la boucle 'while'.

On va compter le nombre d'itérations maximum de la boucle 'while' quand on exécute le programme. On remarque trois propriétés :

— $k \geqslant 0$ car tout élément de π est positif (car chaque élément représente une longueur de mot)

— $\pi(k) < k$ car $k < i$ donc quand on définit $\pi(i)$, on met un élément strictement inférieur à i.

— k s'incrémente au plus de 1 à chaque fois dans la boucle 'for', donc $m-1$ fois au plus dans le programme

Ainsi on diminue strictement au plus $m-1$ fois la valeur de k dans le programme, donc on passe au plus $m-1$ fois dans la boucle 'while'. Ainsi la complexité est en $\mathcal{O}(m)$.

On peut maintenant donner l'algorithme de Knuth Morris Pratt

Algorithme 2 Knuth Morris Pratt

```
Procédure KMP((T, P))
    n = longueur(T)
    m = longueur(P)
    π = Calcul − π(P)
    k = 0  [indice de parcours du motif P]
    Pour i de 1 à n faire
        Tant que (k > 0 et π[k + 1] ≠ T[i]) faire
            k = π[k]
        Fait
        Si (π[k + 1] = T[i]) alors
            k = k + 1
        Fin Si
        Si (k = m + 1) alors
            Afficher "occurence de P à l'indice i − m + 1"
            k = π(k)
        Fin Si
    Fin Pour
Fin
```

Exemple : Pour le texte $T = abcabcabd$, on veut chercher le motif $P = abcabd$.

i	1	2	3	4	5	6	7	8	9	
T	a	b	c	a	b	c	a	b	d	
	↕	↕	↕	↕	↕	×				
P	a	b	c	a	b	d				
k	1	2	3	4	5					
					2					car $\pi(5) = 2$ et $c \neq d$
P				a	b	c	a	b	d	
k					2	3	4	5	6	

On a modifié 5 en 2 car on entrait une fois dans la boucle 'while'. Il faut faire tourner l'algorithme à la main pour bien se convaincre de sa correction (ce n'est pas une preuve de correction !).

On veut maintenant étudier la complexité de cet algorithme. On raisonne de la même manière que pour le programme **Calcul-π**. Le seul changement est la boucle 'for' de longueur n. Le nombre d'itérations de la boucle 'while' est majoré par n cette fois-ci. On a donc une complexité en $\mathcal{O}(n)$.

Ainsi l'utilisation de l'algorithme de Knuth Morris Pratt utilise une complexité temporelle et spatiale de $\mathcal{O}(m)$ pour le prétraitement, puis pour l'algorithme on a une complexité temporelle en $\mathcal{O}(n)$. L'algorithme de Knuth Morris Pratt est donc une amélioration de l'algorithme de Morris Pratt qui stockait l'automate des occurrences ($\mathcal{O}(m|\Sigma|)$ en espace où Σ est l'alphabet utilisé).

Remarques :

Pour utiliser des mots précis, on utilise de la complexité amortie, mais il faut savoir exactement comment prouver le résultat avec les techniques style agrégat, potentiel, comptable, ici on raisonne plutôt par agrégat.

Si on parle de Morris Pratt il faut savoir plus en détail cette histoire d'automates des occurrences. Cela tombe bien c'est un développement développé plus haut : Automates des occurrences

Dans ce développement, je développe beaucoup l'exemple pour bien faire comprendre l'algorithme.