

Algorithme de Knuth-Morris-Pratt KMP

TRAN-THUONG Tien-Thinh

24 mars 2022

Présentation

1 Utilité

2 Algorithme Naïf

3 KMP

- Exemple simple
- Tableau
- Recherche
- Automate

L'algorithme KMP

Un algorithme de recherche d'une chaîne de caractères P (Pattern) de taille p au sein d'une autre chaîne S (String) de taille s avec $p \leq s$.

- L'algorithme Naïf est en $O(s \times p)$
- KMP est en $O(s)$

Algorithme Naïf

```
1 def Naif(P, S): # P : Pattern, S: String
2     p, s = len(P), len(S) # Longueur des chaines
3     resultat = []
4     for i in range(s-p): # Parcours de S
5         j = 0
6         while j < p and P[j] == S[j+i]:
7             j += 1 # caractere similaire
8         if j == p: # p caracteres similaires
9             resultat.append(i)
10    return resultat
11 """
12 >>> Naif("ABCDABD", "ABC ABCDAB ABCDABCDABDE")
13 [15]
14 >>> Naif("AAAB", "AAAAAAAAAAAAA")
15 []
16 """
```

Le pire cas est en $O(s \times p)$

KMP - Exemple

S	A B C	A B C D A B	A B C D A B C D A B D E
P	A B C D A B D		
	0 0 0 X		
P		A B C D A B D	
		0 0 0 0 0 0 X	
>			A B C D A B D
			0 0 X
P			A B C D A B D
			0 0 0 0 0 0 0

L'idée est de mettre en place un tableau pour Pattern permettant de ne pas devoir traiter plusieurs fois un même caractère.

KMP - Tableau

Indice	0	1	2	3	4	5	6
Pattern P	A	B	C	D	A	B	D
Tableau T	0	0	0	0	1	2	0

```
1 def Tableau(P): # P : Pattern
2     T = [0] # Tableau d'indices
3     j, p = 0, len(P)
4     for i in range(1, p+1): # Recherche pour i
5         while P[i] != P[j] and j > 0:
6             j = T[j-1]
7         if P[i] == P[j]:
8             j += 1
9         else:
10            j = 0
11        T.append(j)
12    return T
```

KMP - Tableau exemple

Pattern P		A	B	C	D	A	B	D
Tableau T		0	0	0	0	1	2	0

```
1 >>> Tableau("ABCDABD")
2 [0, 0, 0, 0, 1, 2, 0]
```

Pattern P		A	A	A	B
Tableau T		0	1	2	0

```
1 >>> Tableau("AAAB")
2 [0, 1, 2, 0]
```

Pattern P		A	B	B	A	B
Tableau T		0	0	0	1	2

```
1 >>> Tableau("ABBAB")
2 [0, 0, 0, 1, 2]
```

KMP - Recherche

```
1 def Recherche(P, S, T): # Pattern String Tableau
2     p, s = len(P), len(S)
3     i, m = 0, 0 # i parcourt P, m parcourt S
4     resultat = []
5     while m + i < s: # Parcourt de S
6         if P[i] == S[m + i]: # Meme caractere
7             i += 1
8             if i == p: # On trouve P dans S
9                 resultat.append(m)
10                m += i - T[i-1]
11                i = T[i-1]
12            elif i > 0: # Modifie m en fonction de T
13                m += i - T[i-1]
14                i = T[i-1]
15            else:
16                m += i + 1
17    return resultat
```


KMP - Exemple de recherche

```
1 >>> P = "ABCDABD"
2 >>> S = "ABC ABCDAB ABCDABCDABDE"
3 >>> Recherche(P, S, Tableau(P))
4 [15]
5 >>> Recherche("AAAB", "AAAAAAAAAAAA", Tableau("AAAB"))
6 []
7 >>> Recherche("ABBAB", "ABBABBABCD", Tableau("ABBAB"))
8 [0, 3]
```

KMP - Automate

S : ABC ABCDAB ABCDABCDABDE

Pattern P		A	B	C	D	A	B	D
Tableau T		0	0	0	0	1	2	0

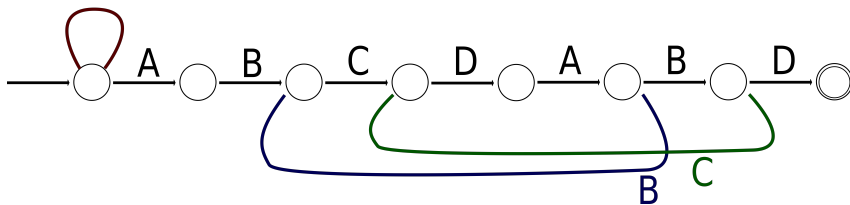


Figure – Schéma automate