Algorithme de Knuth-Morris-Pratt (KMP)

TRAN-THUONG Tien-Thinh

16 mars 2022

Présentation

Utilité

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

```python
def Naif(P, S): # P : Pattern, S: String
    p, s = len(P), len(S) # Longueur des chaines
    resultat = []
    for i in range(s-p): # Parcours de S
        j = 0
        while j < p and P[j] == S[j+i]:
            j += 1 # caractere similaire
        if j == p: # p caracteres similaires
            resultat.append(i)
    return resultat
"""
>>> Naif("ABCDABD", "ABC ABCDAB ABCDABCDABDE")
[15]
>>> Naif("AAAB", "AAAAAAAAAAAAA")
[]
"""
```

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

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
	O O O X
P	A B C D A B D
	O O O O O O X
>	A B C D A B D
	O O X
P	A B C D A B D
	O O O O O O O

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'indice
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

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

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

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

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

```
>>> Tableau("ABBAB")
[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

```
>>> P = "ABCDABD"
>>> S = "ABC ABCDAB ABCDABCDABDE"
>>> Recherche(P, S, Tableau(P))
[15]
>>> Recherche("AAAB", "AAAAAAAAAAAAA", Tableau("AAAB"))
[]
>>> Recherche("ABBAB", "ABBABBABCD", Tableau("ABBAB"))
[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

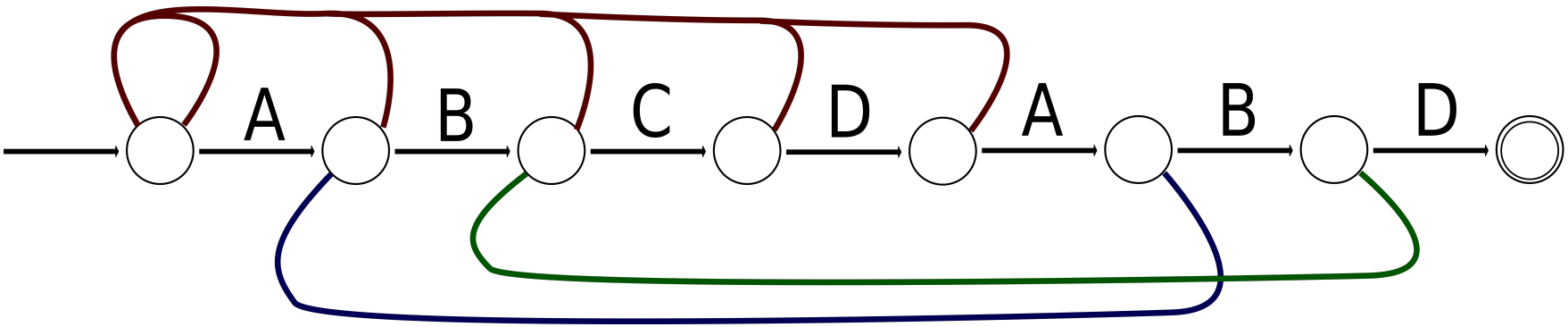

Figure – Schéma automate