Knuth-Morris-Pratt(KMP)

André Luiz Tragancin Filho
João Henrique Faes Battisti

Introdução

- O algoritmo de Knuth-Morris-Pratt foi desenvolvido em 1977.
- É um algoritmo de busca de string.
- O objetivo do algoritmo é encontrar uma substring dentro de uma string.

Algoritmo

```
private int[] aux;

public KMP(String texto, String padrao){
    aux = new int[padrao.length()];
    tabelaFalha(padrao);
    int pos = posMatch(texto,padrao);
    if(pos == -1){
        System.out.println("No match found");
    }
    else{
        System.out.println("Match found at index "+pos);
    }
}
```

Fig.1 - Função KMP

Algoritmo

```
private void tabelaFalha(String padrao){
    int n = padrao.length();
    int pos = 2;
    int cnd = 0;
    aux[0] = -1;
    aux[1] =  0;

    while(pos < padrao.length()){
        if(padrao.charAt(pos-1) == padrao.charAt(cnd)){
            aux[pos] = cnd + 1;
            cnd = cnd + 1;
            pos = pos + 1;
        }
        else if(cnd > 0){
            cnd = aux[cnd];
        }
        else{
            aux[pos] = 0;
            pos = pos + 1;
        }
    }
}
```

Fig.2 - Função Tabela Falha

Algoritmo

```
private int posMatch(String texto, String padrao){
    int i =0 , j = 0;
    int lens = texto.length();
    int lemp = padrao.length();
    while ( i < lens && j < lemp){
        if(texto.charAt(i) == padrao.charAt(j)){
            i++;
            j++;
        }
        else{
            if(j==0){
                i++;
            }
            else{
                j = aux[j-1] + 1;
            }
        }
    }
    return ((j==lemp) ? (i - lemp) : -1);
}
```

Fig.3 - Índice do Match

Demonstração KMP

http://whocouldthat.be/visualizing-string-matching/

Complexidade KMP

De tempo:

Melhor caso = Ω(n)

Pior caso = O(m)

De espaço:

Complexidade = Θ(n)

O(n) Palavra

O(m) Texto

Comparação com Naive

Fig.4 - Pseudo código de Naive

http://whocouldthat.be/visualizing-string-matching/

```
NAIVE (texto,palavra)
1  m <- comprimento [texto]
2  n <- comprimento [palavra]
3  for i <- 0 to m do
4     cont = 0
5     for j <- 0 to n do
6        if( texto[i+j] != padrao[j] )
7           break
8        cont++
9     if(cont == n)
10     return Match Found na posição i
```

Complexidade Naive

De tempo:

Melhor caso = Ω(n)

Pior caso = O(m*n)

De espaço:

Complexidade = Θ(1)

O(n) Palavra

O(m) Texto

Referências

CORMEN, Thomas H. Algoritmos: teoria e prática. 2. ed. Rio de Janeiro: CAMPUS, 2002.

http://whocouldthat.be/visualizing-string-matching/