Knuth-Morris-Pratt (KMP)

Bach. Rodolfo Mercado Gonzales
Universidad Nacional de Ingeniería

Algoritmo KMP

Es un algoritmo determinístico que resuelve el problema de “string matching” de forma eficiente.

Problema de String Matching

Dado un patrón $\boldsymbol{P}$ y un texto $\boldsymbol{T}$, determinar si P aparece en T y enumerar las posiciones de todas las ocurrencias.

Definiciones

Sea $s = s_0 s_1 \dots s_{n-1}$ una cadena de longitud n.

- Un **prefijo** de s es una subcadena $pref = s_0 \dots s_{k-1}$, donde $k \in [1, n]$
- Un prefijo propio, es un prefijo distinto a la cadena original.
- Un **sufijo** de s es una subcadena $suf = s_{n-k} \dots s_{n-1}$, donde k $está\ en\ [1, n]$
- Un sufijo propio, es un sufijo distinto a la cadena original.

Definiciones

Un **borde** de una cadena s es una subcadena que es al mismo tiempo prefijo propio y sufijo propio.

Ejemplo:

Para $s = "abacab"$

Los prefijos propios: a, ab, $aba, abac, abaca$

Los sufijos propios: $b, ab, cab, acab, bacab$

Los bordes : ab

Prefix function

Sea $s = s_0 s_1 \dots s_{n-1}$ una cadena de longitud n.

La función de prefijos está definido como un arreglo $b[\,]$, tal que:

$b[\,i\,]$ es la longitud del borde más grande de la subcadena $s[0 \dots i]$

Prefix function

Ejemplo:

Para $s = "abcabcd"$

$b[\,0\,] = b[\,1\,] = b[\,2\,] = 0$

$b[\,3\,] = 1$, “a”

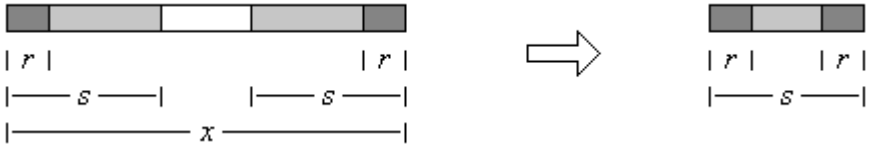
$b[\,4\,] = 2$, “ab”

$b[\,5\,] = 3$, “abc”

$b[\,6\,] = 0$

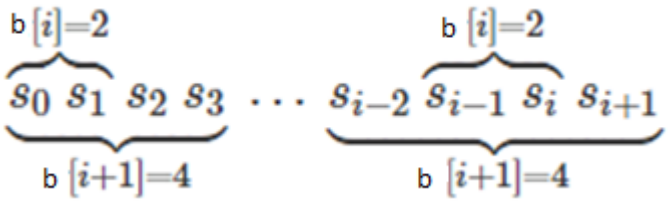
Construcción de Prefix function

Sean r y s bordes de una cadena x, además |r| < |s|, entonces r es borde de s



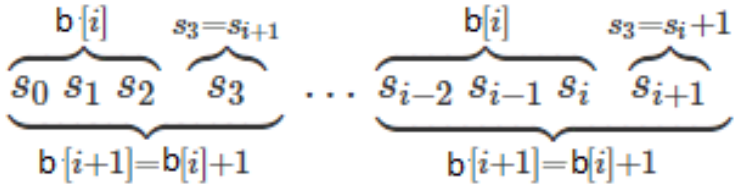
Construcción de Prefix function

Los valores de la función de prefijos pueden incrementar a lo más en 1 o disminuir en algún valor.



Construcción de Prefix function

Para saber si un borde puede ser extendido ($b[\,i+1\,] = b[\,i\,] + 1$), debemos comparar si $s[i+1] = s[\,b[\,i\,]\,]$



Prefix function

```
vector<int> prefixFunction(string s) {
    int n = s.size();
    vector<int> b(n);
    for (int i = 1; i < n; i++) {
        int j = b[ i - 1 ];
        while ( j > 0 && s[ i ] != s[ j ] ) j = b[ j - 1 ];
        if ( s[ i ] == s [ j ] ) j++;
        b[ i ] = j;
    }
    return b;
}
```

KMP

S = patrón + # + texto

Y Luego aplicamos prefix function

$$O(n+m)$$

```
vector<int> kmp(string p, string t) {
    int n = p.size(), m = t.size();
    vector<int> b(n + 1 + m), ans;
    string s = p + "#" + t;
    for (int i = 1; i < s.size(); i++) {
        int j = b[ i - 1 ];
        while ( j > 0 && s[ i ] != s[ j ] ) j = b[ j - 1 ];
        if ( s[ i ] == s [ j ] ) j++;
        b[ i ] = j;
        if( b[ i ] == n )ans.push_back( i - ( n + 1) - n + 1 );
    }
    return ans;
}
```

Problemas

SPOJ – A needle in the haystack

Referencias

- Prefix function, E-Maxx Algorithms in English.

¡ Good luck and have fun !