Algoritmo Knuth-Morris-Pratt

Los creadores del algoritmo fueron Donald Knuth, Vaughan Pratt y James H. Morris y se encarga de realizar busquedas de una subcadena dentro de una cadena. Para ello utiliza información basada en los fallos previos.

El algoritmo trata de localizar la posición de comienzo de una cadena, dentro de otra. Antes que nada con la cadena a localizar se precalcula una tabla de saltos (conocida como tabla de fallos) que después al examinar entre si las cadenas se utiliza para hacer saltos cuando se localiza un fallo.

Supongamos una tabla 'F' ya precalculada, y supongamos que la cadena a buscar esté contenida en el array 'P()', y la cadena donde buscamos esté contenida en un array 'T()'. Entonces ambas cadenas comienzan a compararse usando un puntero de avance para la cadena a buscar, si ocurre un fallo en vez de volver a la posición siguiente a la primera coincidencia, se salta hacia donde sobre la tabla, indica el puntero actual de avance de la tabla. El array 'T' utiliza un puntero de avance absoluto que considera donde se compara el primer carácter de ambas cadenas, y utiliza como un puntero relativo (sumado al absoluto) el que utiliza para su recorrido el array 'P'. Se dan 2 situaciones:

Mientras existan coincidencias el puntero de avance de 'P', se va incrementando y si alcanza el final se devuelve la posición actual del puntero del array 'T'
Si se da un fallo, el puntero de avance de 'T' se actualiza hasta, con la suma actual del puntero de 'P' + el valor de la tabla 'F' apuntado por el mismo que 'P'. A continuación se actualiza el puntero de 'P', bajo una de 2 circunstancias; Si el valor de 'F' es mayor que -1 el puntero de 'P', toma el valor que indica la tabla de salto 'F', en caso contrario vuelve a recomenzar su valor en 0.

Ejemplo del algoritmo en lenguaje C

```
/* Fuente: http://see-programming.blogspot.com/2013/07/c-program-to-implement-knuthmorrispratt.html */

#include <stdio.h>
#include <string.h>
#include <stdlib.h>

  /*
   * finds the position of the pattern in the given target string
   * target - str, patter - word
   */

  int  kmpSearch(char *str, char *word, int *ptr) {
        int i = 0, j = 0;

        while ((i + j) < strlen(str)) {
                /* match found on the target and pattern string char */
                if (word[j] == str[i + j]) {
                        if (j == (strlen(word) - 1)) {
                                printf("%s located at the index %d\n",
                                        word, i + 1);
                                return;
                        }
                        j = j + 1;
                } else {
                        /* manipulating next indices to compare */
```

```
                        i = i + j - ptr[j];
                        if (ptr[j] > -1) {
                                j = ptr[j];
                        } else {
                                j = 0;
                        }
                }
        }
}
/* find the overlap array for the given pattern */
void findOverlap(char *word, int *ptr) {
        int i = 2, j = 0, len = strlen(word);

        ptr[0] = -1;
        ptr[1] = 0;

        while (i < len) {
                if (word[i - 1] == word[j]) {
                        j = j + 1;
                        ptr[i] = j;
                        i = i + 1;
                } else if (j > 0) {
                        j = ptr[j];
                } else {
                        ptr[i] = 0;
                        i = i + 1;
                }
        }
        return;
}

int main() {
        char word[256], str[1024];;
        int *ptr, i;

        /* get the target string from the user */
        printf("Enter your target string:");
        fgets(str, 1024, stdin);
        str[strlen(str) - 1] = '\0';

        /* get the pattern string from the user */
        printf("Enter your pattern string:");
        fgets(word, 256, stdin);
        word[strlen(word) - 1] = '\0';

        /* dynamic memory allocation for overlap array */
        ptr = (int *)calloc(1, sizeof(int) * (strlen(word)));

        /* finds overlap array */
        findOverlap(word, ptr);
        /* find the index of the pattern in target string */
        kmpSearch(str, word, ptr);

        return 0;
}
```