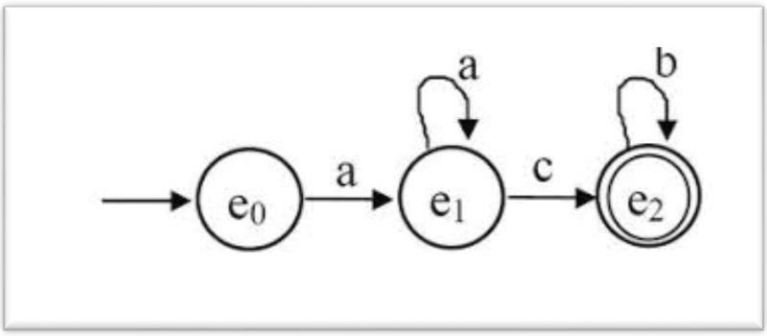
Análisis de Algoritmos

”Empate de cadenas”

Fuerza Bruta y Rabin Karp



***Docente**: Roberto Oswaldo Cruz Leija.*

***Alumna:** Paulina Medrano Hurtado.*

14/11/2019

Introducción

El objetivo de búsqueda de cadenas es halar la localización de un patrón de texto específico dentro de un texto la (eg, una oración, un párrafo), los requisitos son velocidad eficiencia, hay un gran número de algoritmos como lo es fuerza bruta, Rabin-Karp, Knuth- Morris-Pratt. En esta práctica se pretende implementar empate de cadenas atravez de el algoritmo fuerza bruta y Rabin-Karp, ambos en lenguaje c.

Descripción del problema

"Fuerza Bruta"

El algoritmo de fuerza bruta compara el pátron con el texto un carácter a la vez, hasta encontrar que no coinciden los caracteres.

Complejidad: Dado un patrón de M caracteres de longitud, y un texto de N caracteres de longitud. O(N).

- Peor caso; compara el patrón con cada subcadena de un texto de longitud M.
- Mejor caso; encuentra el patrón en las primeras M posiciones del texto.

"Rabin Karp"

El algoritmo de Rabin-Karp es un algoritmo de búsqueda de cadenas creado por Richard M. Karp y Michael O. Rabin que utiliza el hash para encontrar cualquiera de un conjunto de cadenas de patrones en un texto.

Una subcadena de una cadena es otra cadena que aparece en. Por ejemplo, ver es una subcadena de stackoverflow . No debe confundirse con la subsecuencia porque cover es una subsecuencia de la misma cadena. En otras palabras, cualquier subconjunto de letras consecutivas en una cadena es una subcadena de la cadena dada.

En el algoritmo de Rabin-Karp, generaremos un hash de nuestro patrón que estamos buscando y verificaremos si el hash de nuestro texto coincide con el patrón o no. Si no coincide, podemos garantizar que el patrón no existe en el texto . Sin embargo, si coincide, el patrón puede estar presente en el texto.

Este algoritmo se utiliza en la detección de plagio. Dado el material de origen, el algoritmo puede buscar rápidamente en un papel ejemplos de oraciones del material de origen, ignorando detalles como el caso y la puntuación. Debido a la abundancia de las cadenas buscadas, los algoritmos de búsqueda de una sola cadena no son prácticos aquí. De nuevo, el algoritmo Knuth-Morris-Pratt o el algoritmo de búsqueda de cadenas de

Boyer-Moore es un algoritmo de búsqueda de cadenas de un solo patrón más rápido que Rabin-Karp . Sin embargo, es un algoritmo de elección para la búsqueda de múltiples patrones. Si queremos encontrar alguno de los números grandes, digamos k, patrones de longitud fija en un texto, podemos crear una variante simple del algoritmo de Rabin-Karp.

¿Cómo funciona?

A. Hashing El hash es una forma de asociar valores utilizando una función hash para asignar una entrada a una salida. El propósito de un hash es tomar una gran cantidad de datos y poder ser representados por una forma más pequeña. El hash es increíblemente útil y tiene algunas desventajas, a saber, colisiones. El algoritmo Rabin Karp utiliza una función hash para la búsqueda de substrings en el texto y sigue el concepto de que si dos cadenas son iguales, sus hash values también serán iguales. Sin embargo, pueden existir casos donde los hash values son iguales y las cadenas son diferentes, entonces cuando sus hash values son iguales, cada carácter se compara individualmente.

B. Rolling Hash Consiste en precalcular todos los hashes incrementando de nuestro string, Comenzando en el h[i] que calculará el hash de la primera posición de nuestro string h[1]=t[0]+h[0]*b Seguidamente se calcula el segundo hash que sería para las primeras dos posiciones, sumando el anterior hash h[2]=t[1]+h[1]*b y así sucesivamente se sigue hasta tener un h[n] que tendrá por contenido el hash de todo el string Debido a que el número crece exponencialmente, se debe utilizar aritmética modular, pero esto conlleva un problema ya que al aplicar un módulo el número se va reduciendo, de modo que se da la posibilidad de que ese número ya esté utilizado, lo que nos puede generar colisiones(falsos positivos). Rabin y Karp llegaron a la conclusión de que utilizando un número primo como módulo los códigos hash se pueden repartir mejor en nuestra tabla.

Código

```
#include <stdio.h>
#define max 5000
void FuerzaBruta(char a[max],char b[max]){
      int n,m, i;
      int contador=0;
      //Obtenemos los tamaÃ±os de la cadena
      n=strlen(a);
      //Obtenemos el tamaÃ±o del patron
    m=strlen(b);
     //Recorremos la cadena desde el intevalo de [0,n-m] ya que no se
puede hacer comparaciones en la cadena en el intevalo [n-m,n] ya que
se saldria del arreglo
     for( i=0;i<=n-m;i++){
            //Empezamos con un auxioliar poniendolo en la posicion de
la cadena a verificar
        int l=i;
            //Empezamos con un auxioliar poniendolo en la posicion del
patron a verificar
            int j=0;
            //Se itera hasta que el auxiliar que mueve a la cadena sea
mayor que el tamaÃ±o del patron y que contenga el mismo caracter en
esas posiciones
        while(a[l]==b[j] &&  j<=m) {
                  //Se mueven los dos auxliares
            l++; j++;
            }
            //Si el auxiliar que mueve al patron es mayor al tamaÃ±o
de patron -1 quiere decir que tenemos un encuentro ya que se cimplio
que todos los carecatres de ese sub conjunto es igual al patron
        if(j > m-1){
                   //Se imprime la posicion donde se encuentra una
pocurrencia
                  printf("El patron se encontro en %d \n",i+1);
                  //Se aumenta el numero de ocurrencias
            contador++;
            }
       }
       //Si el contador de ocurrencias es mayor que 0 significa que se
encontro alemenos una vez de lo cntrario no se encontro
       if(contador > 0)printf("Se encontraron %d ocurrencias del
patron \n",contador);
       else printf("No se encontro el patron");
}

int main(){
      char cadena[max],patron[max];
      int opcion;
      do{
```

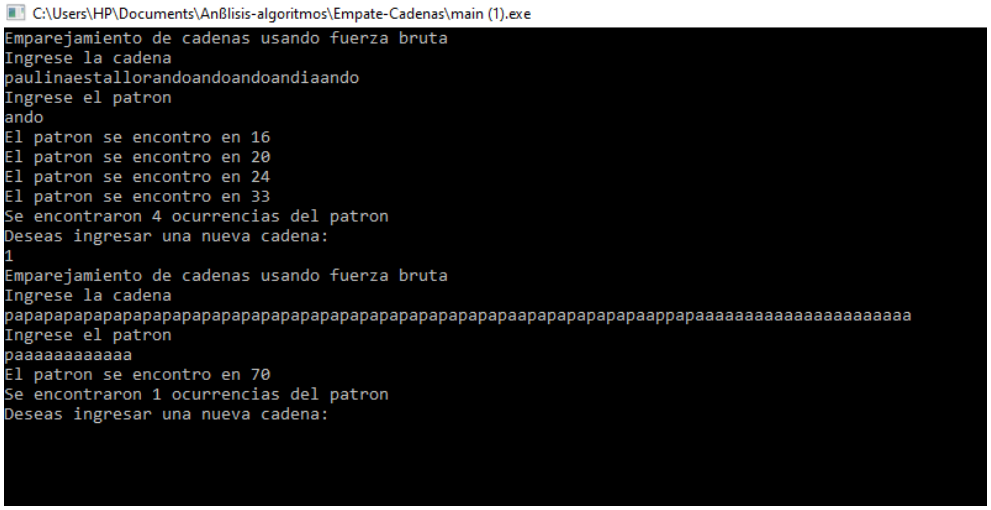
```
            printf("Emparejamiento de cadenas usando fuerza bruta
\n");
            printf("Ingrese la cadena \n");
            scanf("%s",cadena);
            printf("Ingrese el patron \n");
            scanf("%s",patron);
            FuerzaBruta(cadena,patron);
            printf("Deseas ingresar una nueva cadena: \n");
            scanf("%d",&opcion);
      }while(opcion!=2);

}
```

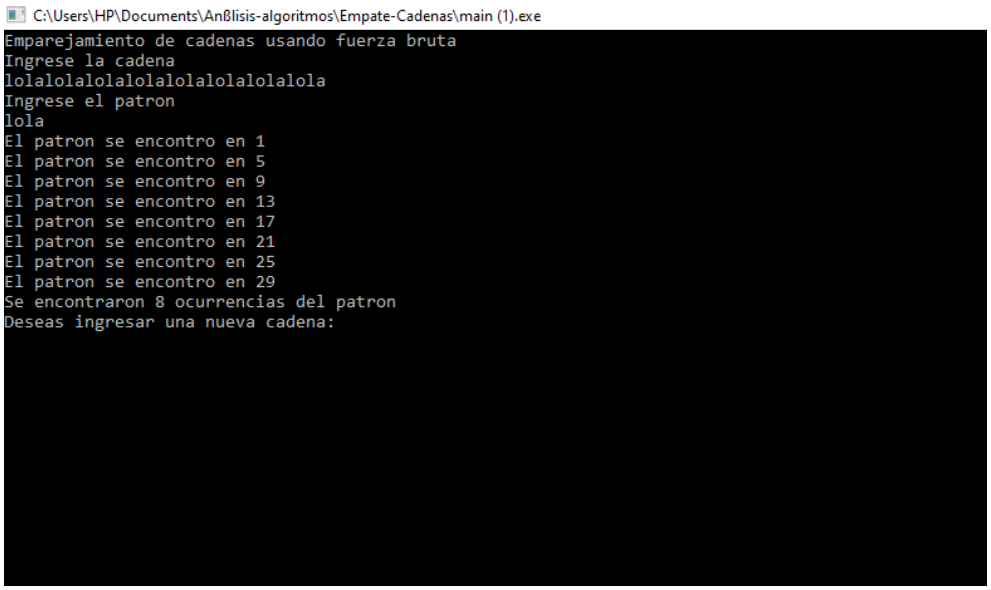
```
/* Following program is a C implementation of Rabin Karp
Algorithm given in the CLRS book */
#include<stdio.h>
#include<string.h>
#define MAX 5000
// d es el numero de carectares que tiene la entrada
#define d 256
void rabinKarp(char pat[MAX], char txt[MAX], int q){
	//Se obtinen los tamanos del texto y del patron
	int M = strlen(pat);
	int N = strlen(txt);
	int i, j;
	int p = 0; //Valor del hash para el patron
	int t = 0; //Valor del hash para la ventana del texto
	int h = 1;

	//para calcular el hash, el valor que representa la cadena, q es
el primo mayor mas proximo al numero de letras
	for (i = 0; i < M-1; i++)
		h = (h*d)%q;
	//Se calcula el hash para el patron y para la primera ventana
del texto
	//Ventana es un sub string del texto al que se compara por hash
	for (i = 0; i < M; i++){
		p = (d*p + pat[i])%q;
		t = (d*t + txt[i])%q;
	}
	// //Recorremos el texto desde el intevalo de [0,N-M] ya que no
se puede hacer comparaciones en la cadena en el intevalo [N-M,N] ya
que se saldria del arreglo
	for (i = 0; i <= N - M; i++){
		//Si el hash de el patron es igual que el hash de el texto
		if ( p == t ){
			//Se hace una comprobacion solo para verificar si
son igules
			for (j = 0; j < M; j++){
				if (txt[i+j] != pat[j])
					break;
			}
			//Si al comprabar caracter a caracter son iguales
tenemos una ocurrencia
			if (j == M)printf("Patron encontrado en %d \n",
i+1);
		}

		//Se calcula el hash para la siguiente ventana
		if ( i < N-M ){
			t = (d*(t - txt[i]*h) + txt[i+M])%q;
			//Si el hash es negativo se hace positivo sumando el
```

```
primo relativo
                    if (t < 0)
                    t = (t + q);
              }
       }
}
int main()
{
       char txt[MAX];
       char pat[MAX];
       printf("Emparejamiento de cadenas usando Rabin Karp \n");
       printf("Ingrese la cadena \n");
       scanf("%s",txt);
       printf("Ingrese el patron \n");
       scanf("%s",pat);
       int q = 101; // A prime number
       rabinKarp(pat, txt, q);
       return 0;
}
```

Resultados

Fuerza bruta





Rabin-Karp

C:\Users\HP\Documents\Anßlisis-algoritmos\Empate-Cadenas\Rabin.exe

```
Emparejamiento de cadenas usando Rabin Karp
Ingrese la cadena
salsalsalsalsalsalsalsalsalsalsalsal
Ingrese el patron
sal
Patron encontrado en 1
Patron encontrado en 4
Patron encontrado en 7
Patron encontrado en 10
Patron encontrado en 13
Patron encontrado en 16
Patron encontrado en 19
Patron encontrado en 22
Patron encontrado en 25
Patron encontrado en 28
Patron encontrado en 31
Patron encontrado en 34
Patron encontrado en 37

--------------------------------
Process exited after 21.36 seconds with return value 0
Presione una tecla para continuar . . .
```

C:\Users\HP\Documents\Anßlisis-algoritmos\Empate-Cadenas\Rabin.exe

```
Emparejamiento de cadenas usando Rabin Karp
Ingrese la cadena
yoyoyoyoyoyoyoyoyoyoyoyoyoyoyoyoyoyoyoyoyoyoyoyo
Ingrese el patron
oyo
Patron encontrado en 2
Patron encontrado en 4
Patron encontrado en 6
Patron encontrado en 8
Patron encontrado en 10
Patron encontrado en 12
Patron encontrado en 14
Patron encontrado en 16
Patron encontrado en 18
Patron encontrado en 20
Patron encontrado en 22
Patron encontrado en 24
Patron encontrado en 26
Patron encontrado en 28
Patron encontrado en 30
Patron encontrado en 32
Patron encontrado en 34
Patron encontrado en 36
Patron encontrado en 38
Patron encontrado en 40
Patron encontrado en 42
Patron encontrado en 44
Patron encontrado en 46

--------------------------------
Process exited after 18.19 seconds with return value 0
Presione una tecla para continuar . . .
```

Conclusiones

En el desarrollo de esta práctica fue interesante implementar ambos algoritmos investigar e indagar en cada uno para saber cómo es que funcionaba o bien tenían su implementación. En ambos algoritmos ejecute pruebas diferentes y a la vez a mano las comprobé siendo así el resultado fue correcto. Respecto al Rabin karp resaltando que este utiliza una función hash para la búsqueda de substrings en el texto. De igual modo dejando claro que el propósito de un hash es tomar una gran cantidad de datos y poder ser representados por una forma más pequeña.

Bibliografía

https://riptutorial.com/es/algorithm/example/24653/introduccion-al-algoritmo-de-rabin-karp

https://slideplayer.es/slide/3173781/

https://www.studocu.com/es-mx/document/universidad-nacional-de-san-agustin/estructura-de-datos-y-algoritmos/ensayos/grupo-3-algoritmo-de-rabin-karp/5375040/view