



INSTITUTO POLITÉCNICO NACIONAL



Análisis de Algoritmos

Empate de Cadenas. Rabin-Karp

Docente: Roberto Oswaldo Cruz Leija

Alumno: Héctor Mauricio Zamudio Domínguez

Fecha: 08/08/19

Fundamentos

Una cadena es una secuencia de caracteres sobre un alfabeto finito.

El problema de emparejamiento de cadenas es encontrar todas las ocurrencias de una cadena p , llamada patrón, en una cadena más grande T del mismo alfabeto.

Existen varios algoritmos de la búsqueda de subcadenas y por lo tanto su objetivo es buscar la existencia de una su cadena dentro de una cadena. Existe el método de fuerza bruta consiste simplemente en verificar, para cada posición posible del texto en la que el patrón pueda concordar si efectivamente lo hace.

El algoritmo Rabin- Karp, es cual es el implementado, es otra solución al problema de empate de cadenas. Este algoritmo se basa en tratar cada uno de los grupos de m caracteres del texto (siendo m el número de símbolos del patrón) del texto como un índice de una tabla de valores hash de manera que si la función hash de los m caracteres del texto coincide con la del patrón es posible que hayamos encontrado un acierto. para verificarlo hay que comparar el texto con el patrón, una opción puede usar fuerza bruta para corroborar.

La Tabla Hash es una estructura de datos no lineal cuyo propósito final se centra en llevar a cabo las acciones básicas (inserción, eliminación y búsqueda de elementos) en el menor tiempo posible, mejorando las cotas de rendimiento respecto a un gran número de estructuras.

Procedimiento

El código esta realizado en C, donde se implementa una función llamada "rabinKarp" que recibe un texto y patrón a buscar. Lo primero que se crea es una clave Hash que nos ayudara ir desplazarnos por las claves Hash, recorreremos la cadena y calculamos la clave Hash tanto para el patrón (cadena) como para el texto, acumulando el numero entero correspondiente al ASCII del carácter, para generar la clave Hash utilizamos la función módulo (por división) que consiste en tomar el residuo de la división de la clave entre el número de componentes del arreglo.

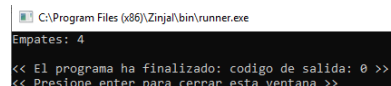
A continuación, verificamos si ambas claves son idénticas, procedemos a verificar si dicha coincidencia es correcta, para esto utilizamos el algoritmo de fuerza bruta que, como anteriormente fue descrito procede a verificar carácter por carácter, si existe una coincidencia aumentamos el contador de coincidencias.

Por último, procedemos a realizar un recalcu de la clave hash del texto, para así seguir recorriendo dicho texto y buscando mas coincidencias, la clave hash del patrón no será necesario recalcular ya que es la será la misma durante todo el transcurso del algoritmo.

Resultados y conclusiones

Prueba 1

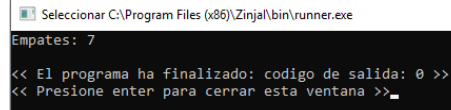
```
int main(int argc, char *argv[]) {
    char text[] = "HolaHolaHolaHolatdrsgjftkufhdjghdxhfdxchgmncbgh";
    char cadena[] = "Hola";
    printf("Empates: %d", rabinKarp(text, cadena));
    return 0;
}
```



```
C:\Program Files (x86)\Zinjal\bin\runner.exe
Empates: 4
<< El programa ha finalizado: codigo de salida: 0 >>
<< Presione enter para cerrar esta ventana >>
```

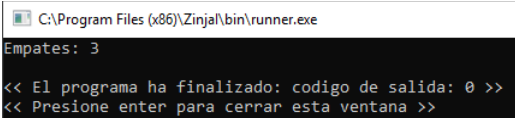
Prueba 2

```
int main(int argc, char *argv[]) {
    char text[] = "HolaHolaHolaHolaAdiosAdiosHola Hola Hola Adios Adios";
    char cadena[] = "Hola";
    printf("Empates: %d", rabinKarp(text, cadena));
    return 0;
}
```



Prueba 3. A diferencia de las pruebas anteriores, esta vez cambiaremos parámetros utilizados por la función Hash utilizada. $P=5$ (anteriormente 3) y se repetirá los mismos parámetros de cadena y texto de la prueba 2.

```
int main(int argc, char *argv[]) {
    char text[] = "HolaHolaHolaHolaAdiosAdiosHola Hola Hola Adios Adios";
    char cadena[] = "Hola";
    printf("Empates: %d", rabinKarp(text, cadena));
    return 0;
}
```



El algoritmo de Rabin-Karp agiliza de manera efectiva (respecto al algoritmo de fuerza bruta) el empate de cadenas, el proceso más lento de este algoritmo es crear la tabla de claves Hash, ya que este proceso crecerá en tiempo conforme crezca el tamaño de la cadena, una vez creada la tabla Hash la búsqueda de coincidencias del patrón dentro del texto se reduce en tiempo considerablemente.

Es importante mencionar que el código implementado es muy básico, realizado únicamente para ejemplificar el funcionamiento del algoritmo, esto se ve ejemplificado en la prueba 3 donde al mover parámetros de la función Hash impacto de manera importante en la eficacia del algoritmo, ya que no se eligió una función de Hash muy compleja y no se tratan colisiones entre claves, entre otros aspectos.

Referencias

Colaboradores de Wikipedia. (2019, 12 noviembre). Algoritmo Karp-Rabin - Wikipedia, la enciclopedia libre. Recuperado 12 noviembre, 2019, de https://es.wikipedia.org/wiki/Algoritmo_Karp-Rabin

EcuRed. (s.f.). Tabla hash - EcuRed. Recuperado 11 noviembre, 2019, de https://www.ecured.cu/Tabla_hash

López, B. Ing. (s.f.). Funciones Hash o Hashing. Recuperado 11 noviembre, 2019, de https://www.ecurhttp://www.itnuevolaredo.edu.mx/takeyas/Apuntes/Administracion_Archivos/Apuntes/Hashing.PDF[ecured.cu/Tabla_hash](https://www.ecured.cu/Tabla_hash)

Franco, E. M. en C.. (s.f.). Análisis de algoritmos. Recuperado 11 noviembre, 19, de <http://www.eafranco.com/docencia/analisisdealgoritmos/files/10/Tema10.pd>