Laboratorio N°1: Fuerza bruta

ANDRÉS MUÑOZ

Profesora: Mónica Villanueva

Ayudante: Patricio Vargas

Compiled March 27, 2017

En la siguiente investigación se desarrolla el algoritmo de fuerza bruta, que consiste en generar la mayor cantidad de combinaciones posibles para luego determinar las opciones válidas. El algoritmo es aplicado a un problema particular que consiste en generar contraseñas con ciertas reglas y estará implementado en un lenguaje de programación C para así poder ejemplificar su funcionamiento y posteriormente analizar su tiempo de ejecución entre otras cosas.

1. INTRODUCCIÓN

Desde el comienzo de la existencia humana el hombre se ha visto en la obligación de resolver distintos problemas con el fin de satisfacer sus necesidades básicas, tales como prender fuego, buscar alimento, fabricar los medios para comer este alimento, fabricar sus propios abrigos y hasta construir los lugares en los que iban a vivir, comúnmente llamadas chozas. Para resolver todos estos problemas que se les presentaban y sin tener experiencias previas, supieron que era necesario seguir una serie de pasos, algunos sencillos y otros mucho más complejos pero que finalmente los llevarían a conseguir sus objetivos de manera satisfactoria, ayudándolos en su desarrollo humano y, en esos entonces, en su sobrevivencia.

Con el paso del tiempo el hombre fue evolucionando y así surgió cada vez más la necesidad de crear algoritmos nuevos con el fin de cumplir distintos objetivos. Estos objetivos nuevos se vieron prácticamente forzados por el avance tecnológico y el ingenio del hombre, que hasta el día de hoy ha creado innumerables avances que se subdividen en muchos procesos donde nos vemos en la necesidad de crear y/o usar algoritmos. También nos encontramos con la capacidad de mejorar algoritmos anteriores y, mejor aún, crear algoritmos nuevos, más eficaces y que cumplen la misma función que los anteriores pero esta vez desde puntos de vista diferentes.

En este informe se presentará un problema que se solucionará con un tipo de algoritmo. Este algoritmo será analizado a profundidad para así sacar diferentes conclusiones, tales como ¿el algoritmo termina? ¿Si es así, termina con una solución? ¿Se podría mejorar? ¿Otro algoritmo representaría mejor el problema y su solución?

2. DESCRIPCIÓN DEL PROBLEMA

El problema consta de generar contraseñas de 8 unidades combinando letras y números, estas combinaciones deben cumplir las siguientes reglas:

1. No se puede generar contraseñas en los que se encuentren 3 letras o 3 números consecutivos.

2. Si la contraseña comienza con numero, no puede terminar con numero.

3. La contraseña no puede iniciar con una vocal.

Para generar las contraseñas con las reglas anteriores, se consideran solo letras minúsculas excluyendo la letra 'ñ'. En el caso de los números se consideraran del 0 al 9.

El lenguaje de programación para implementar la solución debe ser C, el cual se basa en el paradigma de programación imperativo procedural.

3. MARCO TEÓRICO

- **Lenguaje de programación:** Es un lenguaje formal diseñado para realizar procesos que pueden ser llevados a cabo por máquinas como las computadoras.

- **Algoritmo:** Conjunto ordenado y finito de operaciones para hallar la solución de un tipo de problemas. Un algoritmo puede ser implementado en diferentes lenguajes de programación.

- **Fuerza bruta:** Fuerza bruta es un tipo de algoritmo, en cual se generan todas las posibles soluciones para posteriormente elegir las soluciones validas.

4. DESCRIPCIÓN DE LA SOLUCIÓN

La solución que se desea implementar es el algoritmo de fuerza bruta. Como se dijo anteriormente genera todas las soluciones posibles para luego elegir cuales serán las validas. Ahora bien, se procederá a explicar como funciona este algoritmo para el caso particular de este problema.

Las soluciones posibles generadas son todas las combinaciones que se pueden formar respecto a los caracteres de entrada. Esto se genera por medio de ocho bucles anidados, en donde cada bucle representará uno de los ocho caracteres que conforman la contraseña. Mientras avanzan los bucles se va cambiando el carácter correspondiente a cada uno de ellos.

A medida que se van generando las contraseñas se va verificando la contraseña generada, esto con el fin de ir guardando la contraseña en un archivo. Cabe recalcar que de todas formas se generan todas las posibles combinaciones entre los caracteres, sea valido o no la contraseña formada, por lo que el programa no terminara hasta revisar todas las combinaciones posibles.

A continuación se muestra una tabla de todas las funciones implementadas para llevar a cabo la solución, en esta tabla se especifica el tiempo de ejecución ademas de del orden para cada función. Cabe recalcar que la función en la cual se implementa el concepto de fuerza bruta tiene el nombre de *generateCombinations*, la cual se encarga de generar las combinaciones. También cabe mencionar que dentro de esta función subyace la función *validWord*, la cual valida una contraseña generada.

Table 1. Funciones, tiempos y orden

Función	Tiempo	Orden
existsFile	$T(n) = c$	$O(1)$
isIn	$T(n) = c + n$	$O(n)$
consecutive	$T(n) = c + n$	$O(n)$
isLetter	$T(n) = c$	$O(1)$
isNumber	$T(n) = c$	$O(1)$
isVocal	$T(n) = c$	$O(1)$
validWord	$T(n) = c + n$	$O(n)$
searchCharacters	$T(n) = c + cn$	$O(n)$
generateCombinations	$T(n) = c + cn^8$	$O(n^8)$

En la siguiente figura se muestra el algoritmo de fuerza bruta implementado en el programa creado:

```
FILE* archivoSalida;
archivoSalida = fopen("salida.txt","w");
int cantidad = strlen(caracteres);
char string[] = { '0', '0', '0', '0', '0','0', '0', '0', '\0' };
int i,j,k,l,m,o,p,q;
for (i=0;i<cantidad;i++){
    for (j=0;j<cantidad;j++){
        for (k=0;k<cantidad;k++){
            for (l=0;l<cantidad;l++){
                for (m=0;m<cantidad;m++){
                    for (o=0;o<cantidad;o++){
                        for (p=0;p<cantidad;p++){
                            for (q=0;q<cantidad;q++){

                                string[0]=caracteres[i];
                                string[1]=caracteres[j];
                                string[2]=caracteres[k];
                                string[3]=caracteres[l];
                                string[4]=caracteres[m];
                                string[5]=caracteres[o];
                                string[6]=caracteres[p];
                                string[7]=caracteres[q];

                                if (validWord(string)){
                                    fputs(string,archivoSalida);
                                    fputs("\n",archivoSalida);
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}
fclose(archivoSalida);
```

Fig. 1. Algoritmo implementado de fuerza bruta

5. ANÁLISIS DE LOS RESULTADOS

Analizando el algoritmo de fuerza bruta implementado, se puede decir que siempre termina y cuando lo hace entrega la solución (contraseñas) de forma correcta. Cabe recalcar que el tiempo para que termine su ejecución depende netamente de la cantidad de caracteres que se entreguen para realizar las combinaciones.

El algoritmo antes de ser mostrado paso por una serie de mejoras, esto con el fin de que fuera mas eficiente a la hora de ejecutarse. Anteriormente el proceso que se realizaba era generar un archivo con todas las combinaciones, para posteriormente recorrer ese archivo e ir validando cada una de las combinaciones contenidas en este. Este proceso muestra claramente un mayor tiempo, ya que tiene dos subprocesos, uno es generar las combinaciones y otro es analizarlas, lo cual hace que todas las contraseñas pasen a través de dos procesos. Luego se juntan estos dos subprocesos para que se realicen al mismo tiempo, por lo que a medida que se generan las combinaciones se verifican si son validas,estas son guardadas en un archivo de salida. En caso de que no sean validas no se realiza ningún procedimiento, aumentando en cierto sentido la eficiencia del algoritmo.

Después de las mejoras mencionadas anteriormente el tiempo del algoritmo es: $T(n) = c + cn^8$. Esto significa que mientras mas caracteres son los que se desean combinar genera un crecimiento muy grande en el tiempo de ejecución. Cabe mencionar también que la potencia de esta ecuación depende del largo de la contraseña ya que en el caso de que fueran 9 los caracteres que la conforman el tiempo del algoritmo seria $T(n) = c + cn^9$.

Existen otros algoritmos para llevar a cabo las mismas soluciones, entre ellos *Retroceso(Backtraking), Ramificación y acotamiento*. Un algoritmo mas eficiente para este problema particular, es el de *ramificación y acotamiento*, ya que este algoritmo evita algunas combinaciones que no son validas.

6. TRAZA DE LA SOLUCIÓN

La siguiente tabla muestra una traza del algoritmo de fuerza bruta, el cual ejemplifica de manera concreta como se van generando las combinaciones. En este ejemplo se utilizan solamente dos caracteres (a,b).

Table 2. Traza de fuerza bruta

Tiempo	1	2	3	4	5	6	7	8	9
string[0]	a	a	a	a	a	a	a	a	a
string[1]	a	a	a	a	a	a	a	a	a
string[2]	a	a	a	a	a	a	a	a	a
string[3]	a	a	a	a	a	a	a	a	a
string[4]	a	a	a	a	a	a	a	a	b
string[5]	a	a	a	a	b	b	b	b	a
string[6]	a	a	b	b	a	a	b	b	a
string[7]	a	b	a	b	a	b	a	b	a

7. CONCLUSIONES

Para concluir este informe se pudo observar que se cumplieron diferentes objetivos.El objetivo principal fue realizar un programa que genere combinaciones validas a partir de un conjunto de caracteres, donde fue implementado el algoritmo denominado *fuerza bruta*.

Posteriormente se analizó este algoritmo y se llego a la conclusión de que no es eficiente para este problema, dado que para una entrada muy grande de caracteres las combinaciones posibles llevan a un numero extremadamente grande, por lo que es muy costoso en términos computacionales.

Para finalizar se encontró se mencionaron otros algoritmos con los cuales se puede llegar a la misma solución, pero se cree que el mejor es el de ramificación y acotamiento ya que evita posibles combinaciones innecesarias a medida que se van formando estas.

8. REFERENCIAS

1. Ataque de fuerza bruta,*https* : *//es.wikipedia.org/wiki/Ataque*$_d$*e*$_f$*uerza*$_b$*ruta*
2. Lenguajes de programación, *http* : *//es.ccm.net/contents/*304 − *lenguajes* − *de* − *programacion*
3. Algoritmo, *http* : *//conceptodefinicion.de/algoritmo/*