HITO 2

PRACTICA 4 JUEGO DE LAS CIFRAS

Backtracking

ALUMNOS:
Esther Camacho Caro
Diego Dorado Galán

ESCUELA SUPERIOR DE INFORMÁTICA, UCLM
3/5/2021

Tareas a realizar:

Tareas a realizar:

1. Desarrollar un programa Java que implemente un jugador virtual del juego de las cifras que aplique una estrategia de backtracking para encontrar todas las formas posibles mediante las que alcanzar el número objetivo a partir de los 6 números base, y proporcionar al usuario el listado con las mejores combinaciones (las que requieran menos operaciones). Se deberá mostrar lo siguiente:
    a. Número objetivo (generado aleatoriamente)
    b. Números base (generados aleatoriamente, en base a la cantidad seleccionada por el usuario).
    c. Una vez haya concluido el algoritmo: listado de mejores combinaciones, detallando las operaciones que se realizan para llegar al número objetivo.
    d. Opcionalmente, se puede ir mostrando cualquier solución que se vaya encontrando o las soluciones óptimas que se vayan encontrando hasta ese momento durante la ejecución del algoritmo.
2. Elaborar un documento en el que se detalle lo siguiente:
    a. Explicación del enfoque backtracking seguido para encontrar las combinaciones de números que obtengan el valor objetivo (con el menor número de operaciones): cómo se representa la solución que se va construyendo, qué representa la etapa, qué representa cada nodo del árbol conceptual, cómo se van generando los descendientes, test de fracaso (si los hay), test de solución, etc.

b. Determinar y explicar la complejidad teórica del algoritmo principal que habéis implementado (búsqueda de combinaciones que obtengan el valor objetivo).

c. ¿Es el algoritmo óptimo en cuanto al resultado obtenido? Justifica tu respuesta o proporciona contra ejemplos si es necesario.

Cada clase y método del programa que se implemente debe estar debidamente documentado internamente para facilitar su comprensibilidad. Para ello se seguirá el estándar Javadoc para describir propósito, parámetros, tipo de retorno, etc.

El uso de un algoritmo Backtracking es obligatorio.

Se debe realizar un buen diseño orientado a objetos del programa, creando las clases y la organización en métodos que se consideren necesarias.

Explicación del algoritmo

No hemos logrado implementar un algoritmo de backtracking en nuestra práctica. Hemos diseñado la práctica de forma que realice correctamente la primera parte de los objetivos, como son la elección de los números grandes a utilizar, la generación aleatoria del vector de números que se deben usar y el objetivo generado de igual forma, y encontrar una solución a partir de dichos números sin repetir ninguno de estos, pero no hemos logrado aplicar el enfoque deseado y que nos pedían para dicha práctica.

Complejidad de nuestro algoritmo desarrollado

```
public boolean esSolucion(int[] arrayNumeros, int size, int total) {

    for (int i = 0; i < size; i++) {

        if (arrayNumeros[i] == total) {
            return true;
        }

        for (int j = i + 1; j < size; j++) {
            for (int k = 0; k < OPERACIONES.length; k++) {

                int res = OPERACIONES[k].elementos(arrayNumeros[i], arrayNumeros[j]);

                if (res != 0) {
                    int primero = arrayNumeros[i];
                    int segundo = arrayNumeros[j];
                    arrayNumeros[i] = res;
                    arrayNumeros[j] = arrayNumeros[size - 1];

                    if (esSolucion(arrayNumeros, size - 1, total)) {
                        solucion.add(Math.max(primero, segundo) + " " + OPERACIONES[k].simbolo() + " "
                                + Math.min(primero, segundo) + " = " + res);
                        return true;
                    }

                    arrayNumeros[i] = primero;
                    arrayNumeros[j] = segundo;
                }
            }
        }
    }

    return false;
}
```

La estructura básica de nuestro algoritmo se basa en un bucle for, de complejidad O(n). En su interior encontramos un bucle if, de complejidad O(1) para uno de los casos. Seguidamente encontramos dos bucles for anidados, de complejidad O(n).

Por tanto podemos concluir que la complejidad de nuestro algoritmo es $O(n^3)$.