Laboratorio No. 2

Tipos de Algoritmos
(Algoritmos Voraces y Backtracking)

Introducción

En este laboratorio el/la estudiante deberá desarrollar algoritmos utilizando técnicas de diseño y determinando los tipos de algoritmos. A saber: algoritmos voraces, backtracking, programación dinámica, divide y vencerás, algoritmos probabilísticos

Objetivos

Al finalizar este laboratorio, el/la estudiante deberá ser capaz de:

a. Determinar técnicas de diseño para los algoritmos planteados
b. Desarrollar algoritmos voraces
c. Desarrollar algoritmos recursivos utilizando backtracking
d. Aplicar conocimientos discutidos en clase

Contexto

1. Trabaje con un modelo de n capas (domain, controller, test, util).
2. Cree un nuevo proyecto llamado “Laboratory2” utilizando la tecnología javaFX, la cual permitirá trabajar en un entorno gráfico.
3. Defina una clase llamada “Knapsack” e implemente los métodos necesarios para resolver el problema de la mochila (Knapsack Problem) utilizando un tipo de algoritmo voraz. Recuerde que se requiere maximizar las ganancias con los objetos agregados en la mochila sin exceder el peso máximo permitido. **Utilice la media ((valor + peso) / 2), o valor/peso para determinar el orden de selección de cada objeto candidato.** El siguiente fragmento de código muestra parte de la solución del problema:

   ```
   private Item[] list; //lista de objetos candidatos para la mochila
   private double capacity; //capacidad máxima en la mochila
   public Item[] solve(); método que resuelve el problema
   ```

   Nota: Item es una clase con los siguientes atributos: name, value, weight, media, valueWeight

4. Compruebe el funcionamiento de la mochila utilizando una testing class llamada KnapsackTest, de la siguiente forma:
   a. Agregue la siguiente lista a un arreglo de objetos tipo Item:

Nombre	Valor	Peso	Valor/peso
Smart TV 65”	$1000	20 kg	50
PS5	$600	2 kg	300
Libro Java	$20	1 kg	20
Samsung Galaxy	$700	0,5 kg	1400
Huawei	$400	0,5 kg	800
Libro C++	$25	0,5 kg	50
Xbox One	$500	2,2 kg	227,27
Drone	$500	3 kg	166,6

Proyector	$200	3 kg	66,6
Laptop	$800	3 kg	266,6
Impresora 3D	$800	4 kg	200
iPhone	$800	0,5 kg	1600

   b. Muestre la solución al problema de la mochila indicando la lista de elementos que se puede agregar y tomando en cuenta las siguientes capacidades máximas para cada mochila: 14.5, 24.5, 35, 12, 4.5

5. Utilice backtracking y desarrolle algoritmos recursivos que permitan resolver los siguientes problemas:
   a. El problema de las N-reinas (N Queen Problem). Nombre de la clase: NQueenProblem.
   b. El problema de los movimientos del caballo en un tablero de ajedrez (The Knight Tour Problem). Nombre de la clase: KnightTour.
   c. El problema del laberinto (Rat in a Maze). Nombre de la clase: RatInMaze.
   d. El problema del Sudoku (Sudoku Solver). Nombre de la clase: SudokuSolver.

6. Compruebe el funcionamiento de las clases anteriores utilizando una testing class llamada BackTrackingTest, de la siguiente forma:
   a. Para las N-Reinas, muestre la solución para un tablero de 8x8 y otro de 4x4.
   b. Para el Tour del Caballero, muestre la solución para un tablero de 8x8.
   c. Para la Rata en el Laberinto, muestre la solución para los siguientes casos:

```
int[][] maze1 =  {
{1, 0, 0, 0},
{1, 1, 0, 1},
{0, 1, 0, 0},
{1, 1, 1, 1}
};
```

```
int maze2[][] = {
{1, 0, 0, 1, 1, 0, 0, 0},
{1, 0, 1, 1, 0, 1, 1, 1},
{1, 1, 1, 1, 1, 1, 0, 0},
{0, 1, 1, 1, 1, 1, 1, 1},
{1, 0, 1, 0, 1, 0, 1, 1},
{1, 0, 0, 1, 1, 1, 1, 0},
{1, 0, 0, 1, 1, 0, 1, 1},
{1, 0, 0, 1, 1, 0, 0, 1},
};
```

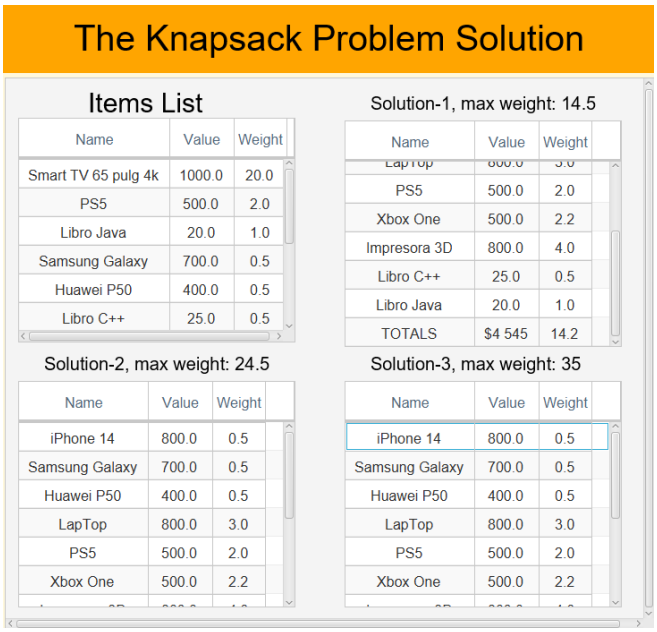
```
int maze3[][] = {
{1, 1, 1, 1, 1, 1, 0, 0},
{0, 0, 0, 1, 1, 1, 1, 1},
{1, 1, 1, 0, 1, 1, 0, 0},
{0, 1, 1, 1, 1, 1, 0, 1},
{1, 0, 1, 0, 1, 1, 1, 1},
{1, 0, 0, 1, 1, 1, 1, 0},
{1, 0, 0, 1, 1, 0, 1, 1},
{1, 0, 0, 1, 1, 0, 0, 1},
};
```

   d. Para el Sudoku, muestre la solución para el siguiente caso:

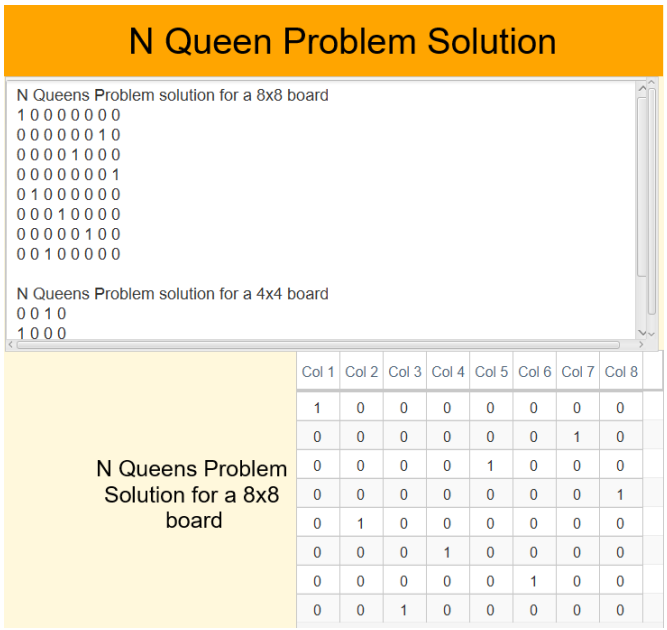
```
int[][] board = {
{5, 3, 0, 0, 7, 0, 0, 0, 0},
{6, 0, 0, 1, 9, 5, 0, 0, 0},
{0, 9, 8, 0, 0, 0, 0, 6, 0},
{8, 0, 0, 0, 6, 0, 0, 0, 3},
{4, 0, 0, 8, 0, 3, 0, 0, 1},
{7, 0, 0, 0, 2, 0, 0, 0, 6},
{0, 6, 0, 0, 0, 0, 2, 8, 0},
{0, 0, 0, 4, 1, 9, 0, 0, 5},
{0, 0, 0, 0, 8, 0, 0, 7, 9}
};
```

7. Utilice la tecnología “javaFX” para crear un entorno gráfico que muestre un menú principal y la solución a los algoritmos anteriores, de la siguiente forma:

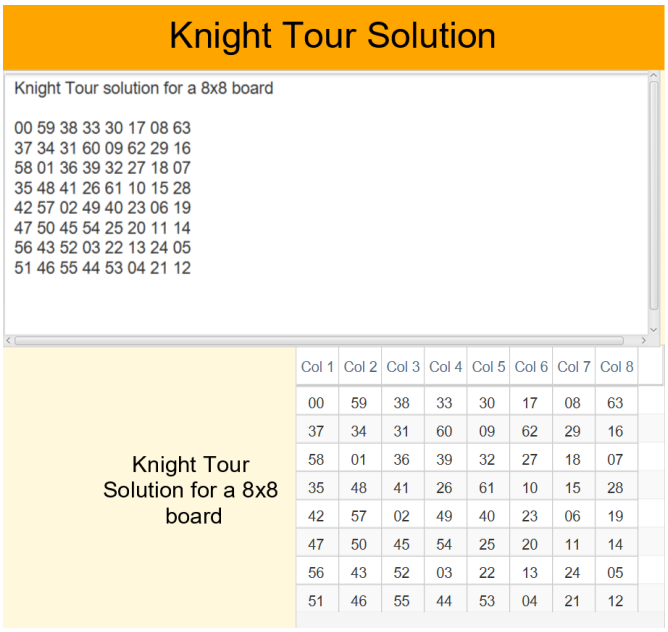
    a. Knapsack Problem: Utilice 4 objetos gráficos tipo TableView para mostrar la solución al problema de la mochila.



    b. N Queen Problem: Utilice un objeto gráfico tipo TextArea para mostrar la solución para un tablero 8x8 y 4x4. Además, utilice un objeto gráfico tipo TableView para mostrar la solución con un tablero 8x8.



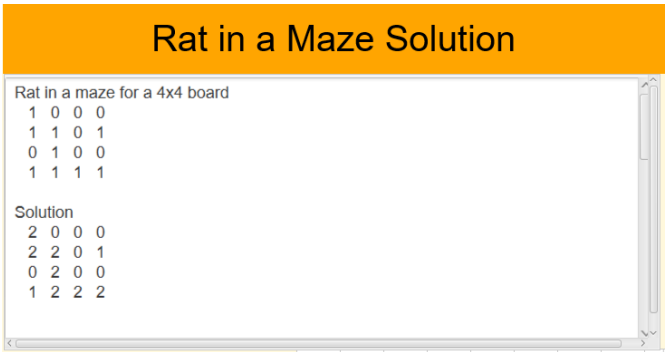
c. The Knight Tour: Similar al caso anterior.



Knight Tour Solution for a 8x8 board

Col 1	Col 2	Col 3	Col 4	Col 5	Col 6	Col 7	Col 8
00	59	38	33	30	17	08	63
37	34	31	60	09	62	29	16
58	01	36	39	32	27	18	07
35	48	41	26	61	10	15	28
42	57	02	49	40	23	06	19
47	50	45	54	25	20	11	14
56	43	52	03	22	13	24	05
51	46	55	44	53	04	21	12

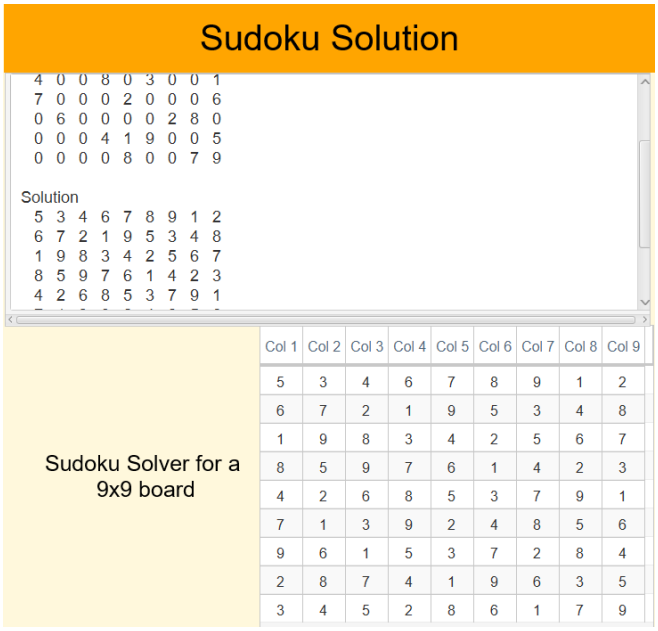
d. Rat in a Maze: Similar a los casos anteriores.



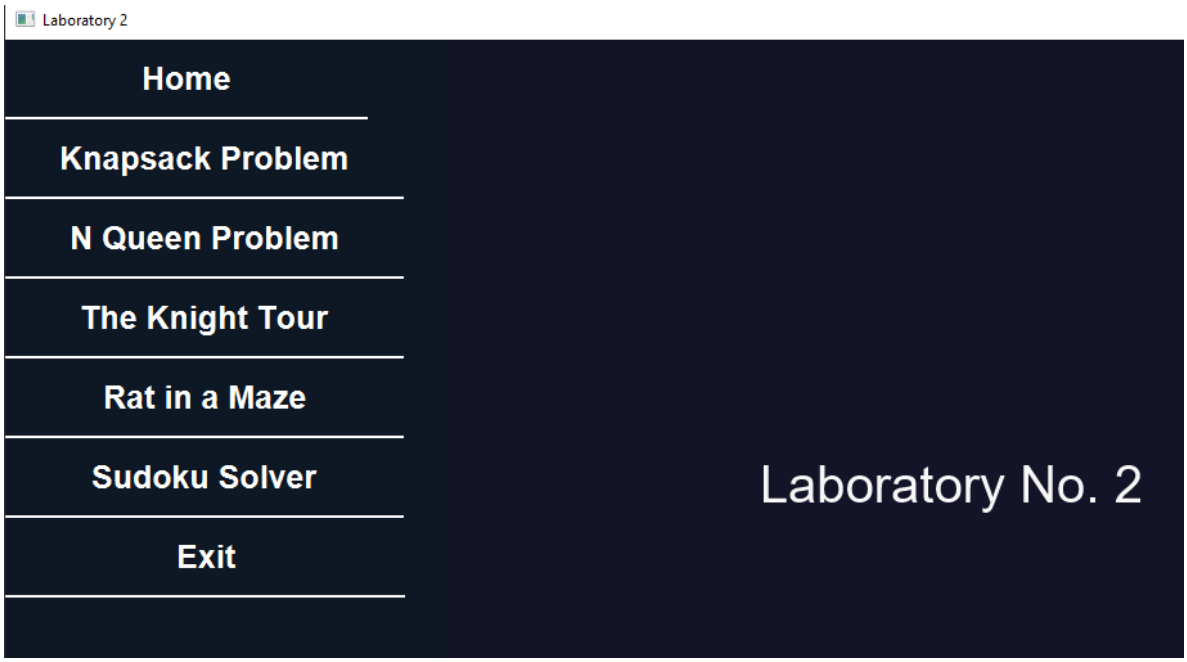
Rat in a Maze Solution for a 8x8 board

Col 1	Col 2	Col 3	Col 4	Col 5	Col 6	Col 7	Col 8
2	2	2	2	2	2	0	0
0	0	0	1	2	2	1	1
1	1	1	0	2	2	0	0
0	1	1	1	2	2	0	1
1	0	1	0	2	2	2	1
1	0	0	1	1	1	2	0
1	0	0	1	1	0	2	2
1	0	0	1	1	0	0	2

e. Sudoku Solver: Similar a los casos anteriores.



Un ejemplo del menú gráfico es el siguiente:



Resuelva y publique el laboratorio en el entorno del curso de la plataforma de mediación virtual (METICS). Verifique la fecha límite para el envío del informe.
URL: https://mv1.mediacionvirtual.ucr.ac.cr/course/view.php?id=7513