Assignment 1 Uninformed and informed Search

Aims:

To perform an uninformed and an informed search for a given problem in a search space organized as a tree.

Task:

Specify, design and deploy an application in python that solve your assigned problem using the specified search methods. The applications should follow the following conditions:

1. It must have a nice architecture (for example the following UML diagram - you can add functions and classes as need it)

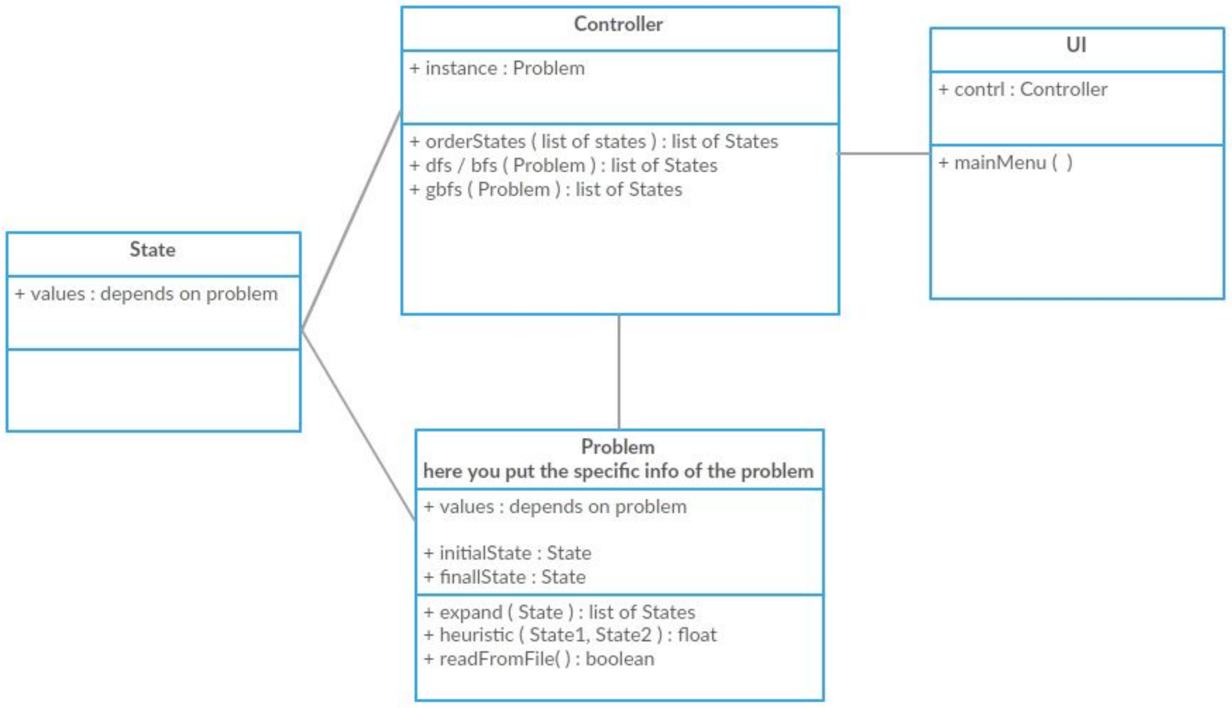

Controller
+ instance : Problem
+ orderStates (list of states) : list of States
+ dfs / bfs (Problem) : list of States
+ gbfs (Problem) : list of States
UI
+ contrl : Controller
+ mainMenu ( )
State
+ values : depends on problem
Problem
here you put the specific info of the problem
+ values : depends on problem
+ initialState : State
+ finallState : State
+ expand (State) : list of States
+ heuristic (State1, State2) : float
+ readFromFile( ) : boolean

Figure 1: The UML diagram

2. the input data will be in a text file
3. the user can choose in a text menu the method that will be used to solve the problem

Each problem must be solved with both two methods!!!
AND NOT WITH OTHER ONES!!!

Points:

- 40 points / method.
- 20 points for the architecture and for the quality of your application.
- A minimum of 50 points must be obtained in order to validate your laboratory.

Time:

Deadline is at the end of the second lab.

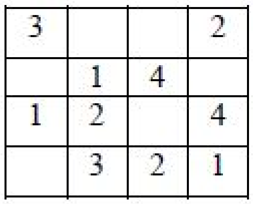
General hints:

- Determine the search tree according to your problem! Will help you A LOT!
- Do not implement functions that you will NEVER use in your application!
- Try to keep the solution simple - these are not difficult problems.
- Ask if you don't know how to solve it! Time is important!
- Do NOT solve the problems with other methods. You will not be granted points if you do this.

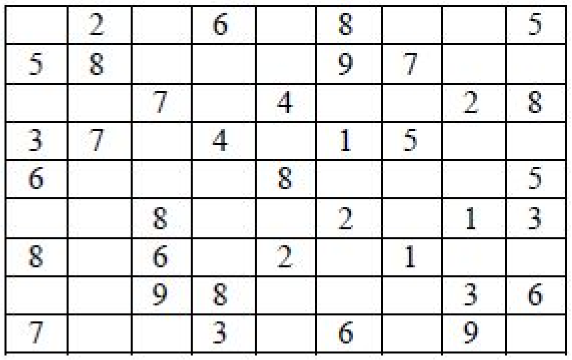
Problems:

1. Sudoku game – solving techniques: BFS, GBFS

Consider a Sudoku game - a logic puzzle represented on a *n* x *n* board; some squares contain already a number, others must be completed with other numbers from {*1,2,…,n*} in such a way that each line, column and square with the edge equal with √n must contain only different numbers. Determine one correct solution for the puzzle.



3			2
	1	4	
1	2		4
	3	2	1



	2		6		8			5
5	8				9	7		
		7		4			2	8
3	7		4		1	5		
6				8				5
		8			2		1	3
8		6		2		1		
		9	8				3	6
7			3		6		9	

Figure 3: a) Sudoku game with 4x4 squares; b) Sudoku game with 9x9 squares

2. Cryptarithmetic game – solving techniques: DFS, GBFS

Implement an algorithm that solves a crypt-arithmetic problem as the ones presented in **Figure 4** knowing that:

- Each letter represent a hexadecimal cipher;

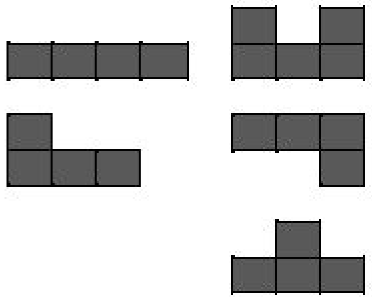
- The result of the arithmetic operation must be correct when the letters are replaced by numbers;
- The numbers can not start with 0;
- Every problem can have only one solution.

```
 SEND+      TAKE+       EAT+     NEVER –
 MORE=          A      THAT=      DRIVE=
------     CAKE=      -----     -------
MONEY     -------     APPLE       RIDE
           KATE
```

Figure 4: Cryptarithmetic problems

3. Geometric forms – solving techniques: DFS, GBFS

Consider the geometric forms from **Figure 5**. Determine an arrangement for this forms on a square board of 5x6 in such a way that the board will be uniform covered and the forms will not overlap.



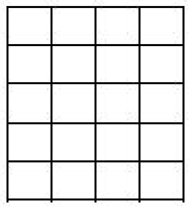
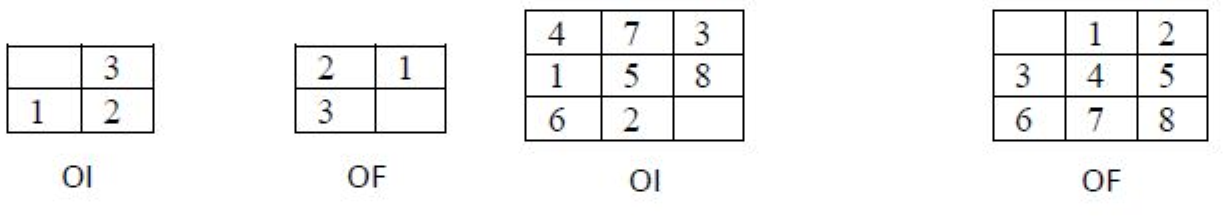


Figure 5: a) the geometric forms. **b) the game board.**

4. The sliding puzzle problem – solving techniques: BFS, GBFS

For a given puzzle of *n* x *n* squares with numbers from 1 to (*n* x *n-1*) (one square is empty) in an initial configuration, find a sequence of movements for the numbers in order to reach a final given configuration, knowing that a number can move (horizontally or vertically) on an adjacent empty square. In **Figure 7** are presented two examples of puzzles (with the initial and final configuration).



OI

	3
1	2

OF

2	1
3	

OI

4	7	3
1	5	8
6	2	

OF

	1	2
3	4	5
6	7	8

Figure 7: a) sliding puzzle with n=2 **b) sliding puzzle with n=3**
(OI – initial order, OF – final order)

5. Any other Game – solving techniques: BFS, GBFS