

## Assignment 1

Instructor: Le Nguyen

Date: Friday, January 29, 2019

Due Date: February 13, 2019

**Total marks: 40.**

**This is an individual assignment.**

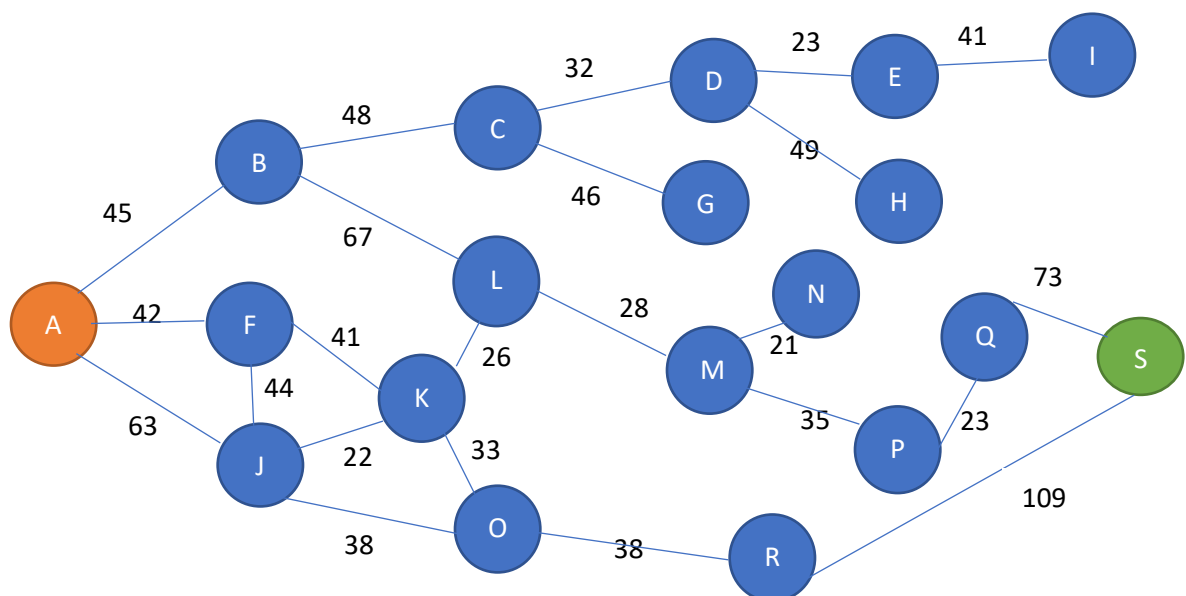
**1. AI approaches. (2 marks)**

- Briefly summarize the four major approaches in AI. You have to provide a discussion on the main focus of each approach.
- What is the difference between weak AI and Strong AI?

**2. Task environment (2 marks)**

- For each of the following activities, give a PEAS description of the task environment and characterize it in term of the properties.
  - Playing soccer
  - Playing tennis match

**3. Informed and Uninformed Search. Draw a search tree and list the order of nodes visited and show to queue information at each depth (12 marks)**



In this question, A to S are cities. The numbers are the distances for one city to another.

Start from A to S.

- a. Breath First search. The expansion of the nodes is from left to right.

**(6 marks)**

- Write down the solution.
- Can it find optimal solution? Explain your answer.

- b. A\* search. **(6 marks)**

Straight line distance is used for heuristic function.

City	SLD to S	City	SLD to S	City	SLD to S
<b>A</b>	184.39	<b>G</b>	107.70	<b>M</b>	100.00
<b>B</b>	161.25	<b>H</b>	101.98	<b>N</b>	89.44
<b>C</b>	145.60	<b>I</b>	100.00	<b>O</b>	111.80
<b>D</b>	141.42	<b>J</b>	144.22	<b>P</b>	78.10
<b>E</b>	140.00	<b>K</b>	128.06	<b>Q</b>	67.08
<b>F</b>	148.66	<b>L</b>	113.14	<b>R</b>	101.98
				<b>S</b>	0.00

- Write down the solution.
- Can it find optimal solution? Explain your answer.

4. Implement 8 Puzzle problem in Java for Search Tree using the following search strategies: (24 marks)

- a. Breadth first search.

- Node expansion should be in order **LEFT, RIGHT, UP, DOWN**.

- b. A\* search with the following **heuristic functions h1 and h2** that discussed in the class.

- Number of misplaced tiles. (h1)
- Manhattan distance (h2)

There are **several java interfaces and classes** provided to help you with the implementation. Note: Please do not alter **these classes**.

- i. **GenericAction**: This interface provides a template to define action associated with 8-puzzle problem.
- ii. **GenericProblem**: This interface provides a template to define the problem.
- iii. **GenericState**: This interface provides a template to construct a state of a problem.
- iv. **Node**: It is a basic from which the search tree is constructed.

You are **to implement and submit** the following classes: (DO NOT USE ANY PACKAGE)

- i. EightPuzzleAction.java
  - a. This class should implement **GenericAction**. It must define the actions: LEFT,RIGHT,UP,DOWN. These can be used to move or change the board state.
- ii. EightPuzzleProblem.java
  - a. This class should implement **GenericProblem**. It must provide the implementation that define in the interface.
  - b. Node expands in order of action (LEFT,RIGHT, UP, DOWN).
- iii. EightPuzzleBoard.java
  - a. This class should implement the **GenericState**. It should provide the implementation that defined in the interface. It should also provide the heuristic functions calculation needed for the A\* search.
- iv. EightPuzzleSearchAgent.java
  - a. This class should implement the treeSearch.
    - i. Breadth First Search.
    - ii. A\* Search with the above heuristic functions
      - 1. **Note**: When the nodes have the same value for evaluation function, they are visited in FIFO order.

It is used to **launch the search agent** that load the initial state and goal state from a file.

A file has a format. Each digit is separated by a space or new line.

**Initial State:**

**1 2 3**

**4 5 6**

**7 8 0**

**Goal State:**

**0 8 7**

**6 5 4**

**3 2 1**

- b. Your program should **display** the following information.
  - i. The nodes that are **generated and its state, g (path cost), h (heuristic value), f (value)**.
  - ii. The number of nodes on the tree.
  - iii. The solution. (Path to the solution)
  - iv. Display the summary of the search strategies.
- c. Discuss the reason why one search strategy is better than another.
  - i. Case 1: Breath First Search vs. A\* Search
  - ii. Case 2: A\* Search with h1 vs. A\* Search with h2.
  - iii. Is h1 is admissible? What happen when it is not admissible? Give an example.

C:\> java **EightPuzzleSearchAgent** StateFile

**Initial State**

**0 1 2**

**3 4 5**

**6 7 8**

**Goal State**

**1 0 2**

**3 4 5**

**6 7 8**

### Breadth First Search:

0 1 2

3 4 5

6 7 8

Action = RIGHT, DOWN //This is the possible actions  
g = 0.0

-----  
1 0 2

3 4 5

6 7 8

Action = LEFT, RIGHT, DOWN //This is the possible actions  
g = 1.0

-----  
..... //other nodes

Number of nodes on the tree = 3.

### Solution:

#### Initial State

0 1 2

3 4 5

6 7 8

Action = RIGHT //An action leads to next node  
g = 0.0

-----  
1 0 2

3 4 5

6 7 8

g = 1.0

Number of nodes on the tree = 3.

### A\* Search with h1:

0 1 2

3 4 5

6 7 8

Action = RIGHT, DOWN      //This is the possible actions  
g = 0.0, h = 1, f = 1

-----  
1 0 2

3 4 5

6 7 8

Action = LEFT, RIGHT, DOWN      //This is the possible actions  
g = 1.0, h = 0, f = 1

-----  
..... //other nodes

Number of nodes on the tree = 3.

### Solution:

#### Initial State

0 1 2

3 4 5

6 7 8

Action = RIGHT      //An action leads to next node  
g = 0.0, h = 1, f = 1

-----  
1 0 2

3 4 5

6 7 8

g = 1.0, h = 0, f = 1

Number of nodes on the tree = 3.

### A\* Search with h2:

.....

After the search is done it should display.

Depth	Search Cost (Number of nodes generated)		
	A*(h1)	A*(h2)	Breadth First Search
1	2	2	2
.....	...	...	...

**Bonus: 3 marks**

Show the branch factor of the search tree and add to the above table

Depth	Search Cost (Number of nodes generated)			Effective branching factor		
	A*(h1)	A*(h2)	Breadth FS	A*(h1)	A*(h2)	Breadth FS
...	...	...	...	...	...	...
.....	...	...	...	...	...	...

### Submission:

Codes that **cannot compile** will receive a zero mark. Please make sure it can compile and it should work on Windows 10 and Linux. Java 8 is used to mark the assignment.

- Please comment your code properly.
- Please don't have any **package statement in the Java classes**.

Please submit the following files:

1. A1.pdf.
2. README File
3. The Code:
  - a. EightPuzzleAction.java
  - b. EightPuzzleSearchAgent.java
  - c. EightPuzzleProblem.java
  - d. EightPuzzleBoard.java