Task 2: Definition of the problem and Implementation of the Frontier

Escuela Superior de Informatica de Ciudad Real

Universidad de Castilla-La Mancha.

October 2018

Elements of a Problem

The **student** must **define** and implement next **elements**:

- State Space.
- Problem.
- Tree Search.
  - Node of the search tree.
  - Frontier.

Frontier

The frontier can be **implemented** using different **data structures**. So, the **selection** of one of them must be justified **considering** next parameters:

- Insertion **time** (minimum, maximum and average)
- **Maximum** number of nodes.

In the tests the insertion position of the nodes is determined by a **random value "f"**.

State

A **state** is defined by:

- The **current position**, OSM node.
- **List of nodes** I still have to go through. The list is ordered in ascending order depending of the node ID.
- **MD5** representation of the state.

State Space

This **class** takes as input the name of the **graphml file** and must contain next methods:

- **Successors(state):** [(acc1,NewState1,costAct1) ,..., (accM,NewStateM,costActM)] where:
  - **acci** is a string such like: "I am in **ID origin** and I go to **ID destination**"
  - **costActi** is the distance between state and NewStatei.
- **BelongNode(State)**:
  - output: True (if it belongs to the State Space) or False (in other case)

oblem

The **problem** comes determined using:

- State space
- Initial State
- Goal Function (isGoal(State)=> True or False)
  A state satisfies this function if the list of nodes is empty.

The input is a **json** file like this.


```
{
  "graphImfile": "Ciudad Real/data/Anchuras.graphml",
  "IntSt": {
        "node": "4331489739",
        "listNodes": ["4331489528", "4331489668", "4331489711", "4762868815", "4928063625"],
        "id": "f4b616551965fb586e608397c308bf0f"
      }
}
```

Tree Search I

Class TreeNode

This is the node of the tree with next fields:

- Node Information: **Access to the parent** (Structural information)
- Domain information:
  - **State :** current state.
  - **Cost of the path:** from the initial node to the current one.
  - **Action :** from the parent to reach the current state.
  - **d :** depth of the node.
  - **f :** value that determines de insertion order in the frontier.

Tree Search II

Class Frontier

An ordered list containing tree nodes in ascending order ("f") with next methods.

- **CreateFrontier:** It creates the empty frontier and it establishes the criterion order.
- **Insert(TreeNode):** It adds a new node to the frontier.
- **Remove():** It takes the first element of the frontier (lowest "f") and it removes it from the fringe.
- **isEmpty():** True or False.

Deadline for delivery: **October 26th**

- **Source code**. Commit changes to Github repository.
- **Documentation**. The **doc.pdf** document from the previous delivery adding the structures of the created artifacts and a **justification** for them, as well as the name of all **equipment** components and the name of the repository. This document will be **uploaded** to Moodle.