**Course Notes – Search**

* **Introducción**
* **Search**
  + Sort numbers
  + Solve a Maze
  + Driver direction
    - Best way, dependen don traffict
  + Search problems
  + **Terminology**
    - **Agent**

Entity that perceives its environment and acts upon that environment

* + - **State**

A configuration of the agent and its environment

* + - **Initial state**

The stage in which the agent begins

* + - **Action**

Choices that can be made in a state

As a function, Action(s) returns the set of actions that can be executed in state s

* + - **Transition model**

A description of what state results from performing any applicable action in any state

More precise, a function calls return(s,a) return resulting from performing action a in state s.

* + - State space

The set of all starts reachable from the initial state by any sequence of actions. Include all the nodes(states) an path to reach our goal

* + - Goal test

Way to determine whether a given state is a goal state

* + - Past cost

Numerical cost associated with a given path

How expensive it is to get the best option

* + - Solution

A sequence of action s that leads form the initial stat to a goal state

* + - Optimal solution

A solution that has the lowest path cost among all solutions

* + - Summary
      * Main parts of search problems
        + Initial state
        + Actions
        + Transition model
        + Goal test
      * Path cost function problems
        + Initial state
        + Actions
        + Transition model
        + Goal test

Optimal solution

Less path cost

* + - * + Path cost function
  + Solving search problem

Data Structure:

* + - **Node**

A data structure that keeps track of

* + - * A state
      * A parent (node that generated this node)
      * An action (action applied to parent to get node)
      * A path cost (from initial state to node)
    - **Frontier**

Represent all of the things that we could explore next, that we haven’t yet explore or visit.

* + - **Approach**
      * Start with a frontier that contains the initial state
      * Repeat:
        + If the frontier is empty, then no solution
        + Remove a node from the frontier
        + If node contains goal state, return the solution
        + Expand node, add resulting nodes to the frontier
    - **Revised Approach**
      * Start with a frontier that contains the initial state
      * Start with an empty explore set
      * Repeat:
        + If the frontier is empty, then no solution
        + Remove a node from the frontier
        + If node contains goal state, return the solution
        + Add the node to the explored set
        + Expand node, add resulting nodes to the frontier if they aren’t already in the frontier or the explore set.
    - **Depth-first search**
      * Search algorithm that always expands the deepest node in the frontier. It’s a method that use a stack to allocate the frontier, a stack is a FILO data structure, first in, last out.
    - **Breadth-first search**
      * Search algorithm that always expands the shallowest node in the frontier. Dispite of Depth-first search algorithm, this one is a FIFO, first in, first out data structure. Use a queue to allocate the frontier